Integrated Justice Exchange Document Methodology, Naming, and Design Rules (MNDR) for GJXDM v3.0.3

Committee Recommendation, April 3, 2007

Document identifier:

wd-ijtc-MNDR-1.1.doc

Location:

Persistent: TBD
This Version: http://www.oasis-open.org/apps/org/workgroup/legalxml-intj-exmndr/
Previous Version http://www.oasis-open.org/apps/org/workgroup/legalxml-intj-exmndr/documents.php

Technical Committee:

OASIS LegalXML Integrated Justice TC

Chairs:

Ellen Perry, MTG Management Consultants
John Ruegg, LA County Information Systems Advisory Body (ISAB)

Editors:

John Ruegg, LA County Information Systems Advisory Body (ISAB)
<jruegg@isab.co.la.ca.us>
Marcus Leon, LA County Information Systems Advisory Body (ISAB)
<mleon@isab.co.la.ca.us>
Marguerite Soto, LA County Internal Services Department
<msoto@isd.lacounty.gov>

Contributors:

Scott Came, Justice Integration Solutions, Inc.
Tom Carlson, National Center for State Courts
Ellen Perry, MTG Management Consultants, L.L.C.
John Ruegg, LA County Information Systems Advisory Body (ISAB)
Nancy Rutter, Maricopa County, Arizona

Abstract:

The purpose of this document is to establish guidance on how to develop GJXDM Information Exchange Package Documentation.

Status:

This document has been drafted by the OASIS LegalXML Integrated Justice MNDR subcommittee and is being submitted for review/revision to the OASIS LegalXML Integrated Justice TC. It will also be given to other groups in the justice community for review/revision. These groups include the GJXDM Training and Technical Assistance Committee (GTTAC), the Global XML Structure Task Force (XSTF), and the IJIS Institute XML Advisory Committee. Other subsequently issued documents will supersede this document

Committee members should send their comments on this specification to the workgroup_mailer@lists.oasis-open.org list. Others should subscribe to and send comments to the mailto:legalxml-intjustice@lists.oasis-open.org list. To subscribe, send an email message to mailto:legalxml-intjustice@lists.oasis-open.org?subject=Subscribe with the word “subscribe” as the body of the message.

Notices

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS Website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification, can be obtained from the OASIS President.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to implement this specification. Please address the information to the OASIS President.

Table of Contents

173

1 Introduction

There is a need to develop standards and best practices for the development of GJXDM conformant information exchange packages. The purpose of this document is to provide:

- A methodology for the construction of GJXDM-conformant information exchange package documentation
- Naming and design rules for the artifacts called for by the methodology
- Guidelines for the customization of GJXDM schema structures

This work will benefit the justice community in the following ways:

- It will improve interoperability by promoting consistent use of GJXDM in constructing schemas
- It will lower risk and improve efficiency of information exchange projects between justice partners by defining, consolidating, and disseminating best practices
- It will provide a formal, normative standard that exchange partners can use to establish quality assurance criteria.

1.1 Audience

This document is intended to provide consistent, practical guidance for technical and business users seeking to create GJXDM-conformant information exchange packages.

Technical Users – to provide technical guidelines, methods, and rules to analysts and developers

Business Users – to provide a concrete framework for business (subject matter) experts to ensure that development work is completed accurately and completely, with appropriate quality assurance features.

1.2 Scope

This document covers the following sections:

Information Exchange Package Documentation Overview – Provides the background and functional uses for the Global Information Exchange Package Documentation (GIEPD); leverages the GJXDM Information Exchange Package Documentation Guidelines **[GJXDM IEPD]** published by the GJXDM XML Structured Task Force.

GIEPD Development Process Guidance provides information about the composition and responsibilities of the development team, including the steps to follow to define an exchange, along with the required artifacts.

Information Exchange Package Documentation (GIEPD) - This section describes the required and optional components which make up a GIEPD including Domain Modeling documentation, Schema(s), XML Instance Documentation, other supporting documents and MNDR standards for Naming and Packaging the GIEPD into a MNDR conformant ZIP file .

Schema Set & Instance Document Naming and Design Rules – Provides detailed instructions about how to code GJXDM conformant XML schemas, including information about XML types, elements, attributes, and documentation, as well as other schema rules and guidelines. Also, describes the rules for constructing instance documents, including requirements for root elements and validation methods.

1.2.1 What this MNDR contains

This document provides step by step guidelines for the analysis, high level object modeling, GJXDM mapping, construction, and customization of all components of GJXDM conformant information exchange packages, including instance document examples.

1.2.2 What this MNDR does not address

This document does not address instructions and requirements regarding messaging and other end-to-end transaction requirements.

1.3 Purpose

This document seeks to achieve and support interoperability beyond what is provided by XML Schema and the GJXDM by defining standard development methodologies, best practices and exchange artifacts that meet the needs of technical and business users. It leverages and extends standards defined elsewhere in the technical and justice communities, including rules defined and endorsed by GTRI, the XSTF, OASIS, and W3C.

1.4 Terminology and Notation

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 Internet Engineering Task Force (IETF) Request for Comments (RFC) 2119. Non-capitalized forms of these words are used in the regular English sense.

[Definition] – A formal definition of a term. Definitions are normative.

[Example] – A representation of a definition or a rule. Examples are informative.

[Note] – Explanatory information. Notes are informative.

[RRRn] – Identification of a rule that requires conformance to ensure that an XML Schema is conformant to OASIS Integrated Justice Exchange Document Methodology, Naming, and Design Rules (MNDR). The value RRR is a prefix to categorize the type of rule where the value of RRR is as defined in Figure 1 and n (1..n) indicates the sequential number of the rule within its category. In order to ensure continuity across versions of the specification, rule numbers that are deleted in future versions will not be re-issued, and any new rules will be assigned the next higher number – regardless of location in the text. Future versions will contain an appendix that lists deleted rules and the reason for their deletion. Only rules and definitions are normative; all other text is explanatory.

261 ***Figure 1 – Rule Prefix Token Value***

262

Attribute Declaration Rules	(ATD)
Code List Rules	(CDL)
ComplexType Definition Rules	(CTD)
ComplexType Naming Rules	(CTN)
Documentation Rules	(DOC)
Element Declaration Rules	(ELD)
General Naming Rules	(GNR)
General Type Definition Rules	(GTD)
General XML Schema Rules	(GXS)
Instance Document Rules	(IND)
Mapping Documentation Rules	(MAP)
Modeling Constraints Rules	(MDC)
Namespace Rules	(NMS)
Root Element Declaration Rules	(RED)
Schema Structure Modularity Rules	(SSM)
Standards Adherence Rules	(STA)
Versioning Rules	(VER)

263

264 1.5 Principles

265 Open Standards – All artifacts will rely on open standards to represent required content.

266

267 Reuse versus Reinvent – This work incorporates proven work developed by other groups,
268 including GJXDM methods and standards and OASIS UBL. For example, this work is patterned
269 after the successful development of similar methodology, naming, and design rules in the OASIS
270 UBL Technical Committee

271

272 Collaboration - In developing these proposed rules and guidelines, the subcommittee will consult
273 and collaborate with other groups in the justice community (including but not limited to the
274 GJXDM Training and Technical Assistance Committee [GTTAC], the Global XML Structure Task
275 Force [XSTF], and the IJIS Institute XML Advisory Committee.) In consulting with these groups,

the subcommittee will seek to incorporate or reference, as appropriate, any existing or emerging work product that they may have

1.6 Terms and Definitions

This document assumes a basic knowledge of XML which allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. A select set of Technical Terms and Definitions can be found in the Appendix C - Technical Terminology.

1.6.1 Software Licensing Definitions

Proprietary

Proprietary refers to software or formats in which an entity retains distribution and modification rights. Proprietary software, and particularly products produced with the software, can be difficult to use without owning a license to the producing software.

In some cases, a proprietary format becomes ubiquitous to the point that it is a de facto standard.

Proprietary may refer to a software application itself, or to a product of a software application. For example, the underlying format of a document produced with a word processor may be proprietary, independent of whether the word processor is proprietary.

Open Source

Open Source refers to a group of licenses that allow entities to distribute software while allowing others to copy and modify it. Typically, Open Source licenses require that, if an entity modifies and distributes their modifications, they must also grant the same redistribution rights. This ensures that improvements to Open Source products are returned to the development community.

Freely Available

Freely Available refers to software or services that are free to use, but are not distributed under an Open Source license.

1.7 Symbols and Abbreviations

CCTS

Core Components Technical Specifications

DOJ

Department of Justice

ebXML

Electronic Business XML

GIEP

GLOBAL Information Exchange Package

GIEPD

GLOBAL Information Exchange Package Documentation

GJXDM

Global Justice XML Data Model

GTRI

Georgia Technical Research Institute

HTML

HyperText Markup Language

IETF

Internet Engineering Task Force

IEP

Information Exchange Package

ISO

International Standards Organization

JXDM

justice xml data model

LCC

lowerCamelCase

MNDR

Methodology, Naming and Design Rules

NDR

Naming and Design Rules

NS

NameSpace

OJP

Office of Justice Planning within the Federal Department of Justice

OASIS

Organization for the Advancement of Structured Information Standards

PDF

Postscript Data Format

RFC

Request For Comment

SSGT

Subset Schema Generation Tool

UBL

Universal Business Language

UCC

UpperCamelCase

UML

Universal Modeling Language

URI

Universal Resource Identifier

URL

Universal Resource Locator

W3C

World Wide Web Consortium

XMI

XML Metadata Interchange

XML

eXtensible Markup Language

XSD

XML Schema Definition

XSTF

XML Structure Task Force Committee sponsored the Federal Department of Justice

1.8 Relationship to other XML Specifications

The MNDR specification leverages other existing, non-proprietary XML specifications wherever possible. In particular, the specification has dependencies on the **[GJXDM]** and the **[UBL NDR]** specifications.

1.8.1 Global Justice XML Data Model (GJXDM)

[GJXDM] conformance, as defined by the GJXDM Implementation Guidelines (**[GJXDM IMPLEMENT]**), is a core objective of this specification. The **[GJXDM]** is an XML standard designed specifically for justice information exchanges, providing law enforcement, public safety agencies, prosecutors, public defenders, and the judicial branch with a tool to effectively share data and information in a timely manner. The **[GJXDM]** provides a library of reusable components that can be combined to automate justice information exchanges. The **[GJXDM]** removes the burden from agencies to independently create exchange standards. Because of its extensibility, there is more flexibility to deal with unique agency requirements and changes. Through the use of a common vocabulary that is understood system to system, **[GJXDM]** enables access from multiple sources and reuse in multiple applications.

The **[GJXDM]** is most useful for describing common objects such as persons and locations and criminal justice-specific processes such as arrest, booking, jail and prosecution. The **[GJXDM]** is not as well developed for describing non-criminal information exchanges and processes. The MNDR assumes the **[GJXDM]** version 3.0.3 structures and definitions. A separate version of the MNDR will be released to support newer version(s) of **[GJXDM]** and **[NIEM]**.

1.8.2 OASIS Universal Business Language

[UBL] is an OASIS Standard that provides a single ubiquitous language for business communication that takes into account the requirements common to all enterprises. **[UBL]** provides a library of reusable components that can be combined to create electronic business schemas. This shared library is essential to interoperability; without a common set of base components, each document format would risk redefining addresses, locations, and other basic information in similar but incompatible ways. Many of the Naming & Design Rules from the **[UBL NDR]** have been incorporated into this **[MNDR]** specification.

1.9 Normative References

[GJXDM] Global Justice XML Data Model 3.0.3, http://www.it.ojp.gov/gjxdm, US DOJ OJP, 2005.

[GJXDM IMPLEMENT] *Global JXDM Implementation Guidelines*, http://it.ojp.gov/topic.jsp?topic_id=138, US DOJ OJP, 2005.

[GJXDM USER] *Catherine Plummer, GJXDM Users Guide,* http://it.ojp.gov/documents/GJXDMUserGuide.pdf , SEARCH, 2005.

[Namespaces] T. Bray, Namespaces in XML, http://www.w3.org/TR/1999/REC-xml-names-19990114 , January 14, 1999.

[UBL] B. Meadows, L. Seaburg (editors), Universal Business Language 1.0, http://docs.oasis-open.org/ubl/cd-UBL-1.0/ , OASIS Standard, September 15 2004

[UBL NDR] M. Crawford (editor), Universal Business Language (UBL) Naming and Design Rules, http://www.oasis-open.org/committees/download.php/10323/cd-UBL-NDR-1.0Rev1c.pdf , OASIS Standard, November 15, 2004

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2110, March 1997.

[Schema Part 1] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, XML Schema Part 1: Structures Second Edition, http://wwww.w3.org/TR/2004/REC-xmlschema-1-20041028/ , W3C Recommendation, October 28, 2004

[Schema Part 2] H.P. Biron, A. Malhotra, XML Schema Part 2:Data Types Second Edition, http://wwww.w3.org/TR/2004/REC-xmlschema-2-20041028/ , W3C Recommendation, October 28, 2004

[UML] Universal Modeling Language, http://www.omg.org/technology/documents/formal/uml.htm , Object Management Group (OMG) UML specifications.

[XML 1.0] T. Bray, Extensible Markup LanguageNamespaces in (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml/REC-XML-20040204, W3C Recommendation, December 2002.

1.10 Non-Normative References

[CONCEPT Maps] Joseph D. Novak, The Theory Underlying Concept Maps and How to Construct Them, http://cmap.coginst.uwf.edu/info/printer.html , Cornell University

[DON NDR] CIO Office of the Department of Navy (DON), Chief Information Officer, Naming and Design Rules, http://xml.coverpages.org/DON-XML-NDR20050127-33942.pdf , Final Version 2.0. , January 2005. 168 pages

[GJXDM HELP] GJXDM Virtual Help Desk, http://it.ojp.gov/gjxdm/helpdesk/

[GJXDM IEPD] M. Hulme, GJXDM Information Exchange Package Documentation Guidelines,http://it.ojp.gov/process_links.jsp?link_id=4581, March 2, 2005.

[GLOBAL] Informational Resource on Federal Department of Justice GLOBAL Organization with links to numerous Justice Information Sharing Standards & Specifications.

http://it.ojp.gov/topic.jsp?topic_id=8

[ISO 11179] Links to general information and specifications covering the International Standards Organization (ISO) 11179 specification.

http://en.wikipedia.org/wiki/ISO/IEC_11179

[LegalXML ECF] R. Winters (editor), LegalXML Electronic Court Filing 3.0, http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v3.0/ecf-v3.0-spec-cd01.zip , November 15, 2005

[MNDR Charter] OASIS Integrated Justice Technical Committee, MNDR Charter, http://www.oasis-open.org/apps/org/workgroup/legalxml-intj-exmndr/description.php

[NIEM] NIEM Concept of Operations, http://www.niem.gov, DOJ/DHS, October 7, 2005.

[SSGT Introduction] Document describing the Schema Subset Generation Tool and its application in developing Subset Schemas for GJXDM Information Exchange Packages.

http://justicexml.gtri.gatech.edu/subset_tool.html

[UBL Customization] Eduardo Gutenberg (editor), Guidelines For The Customization of UBL v1.0 Schemas, http://docs.oasis-open.org/ubl/cd-UBL-1.0/doc/cm/wd-ubl-cmsc-cmguidelines-1.0.html , OASIS Working Draft 1.0, April 22, 2004

[XML TOPICS] XML Coverpages hosted by OASIS provides numerous links on XML topics and standards, http://xml.coverpages.org/xml.html

2 Information Exchange Package Documentation Overview

2.1 Background

This section provides definitions of key terms and organizations, a delineation of GIEPD functional uses, and guidance on the GIEPD development process.

2.1.1 Justice Information Sharing Initiative (Global)

The efforts of the Global Justice Information Sharing Initiative (Global) Advisory Committee (GAC) have direct impact on the work of more than 1.2 million justice professionals. The importance of the organization's mission, however, positions Global to impact citizens of the U.S., Canada, and beyond. Global's mission — the efficient sharing of data among justice entities — is at the very heart of modern public safety and law enforcement.

Global is a “group of groups,” representing more than thirty independent organizations spanning the spectrum of law enforcement, judicial, correctional, and related bodies. Member organizations participate in Global out of shared responsibility and shared belief that, together, they can bring about positive change in inter-organizational communication and data sharing.

2.1.2 Global Justice XML Data Model (GJXDM)

The Global JXDM is an XML standard designed specifically for criminal justice information exchanges, providing law enforcement, prosecution, defense, probation, and the judicial branch with a tool to effectively share data and information in a timely manner. The Global JXDM removes the burden from agencies to independently create exchange standards, and because of its extensibility, there is more flexibility to deal with unique agency requirements and changes. Through the use of a common vocabulary that is understood system to system, Global JXDM enables access from multiple sources and reuse in multiple applications.

2.1.3 GJXDM Information Exchange Package (GIEP)

An “Information Exchange Package” represents a set of data that is transmitted for a specific business purpose. It is the actual XML instance that delivers the payload or information. (The word “package” as used herein refers to a package of the actual data, not a package of artifacts documenting the structure and content of the data.) An Information Exchange Package can be prefixed with “GJXDM” to indicate or highlight that the IEP is GJXDM-conformant, as in “GJXDM Information Exchange Package.” The fact that an IEP is GJXDM-conformant may be readily apparent from the context, so it is not absolutely necessary to use the word “GJXDM” even if the IEP is GJXDM-conformant.

2.1.4 GJXDM Information Exchange Package Documentation (GIEPD)

“Information Exchange Package Documentation” is a collection of artifacts that describe the structure and content of an Information Exchange Package. It does not specify other interface

layers (such as web services). It can optionally be prefixed with “GJXDM” to indicate or highlight that a resulting IEP is GJXDM-conformant.

2.2 GIEPD Functional Uses

Global Information Exchange Package Documentations are not technical specifications, in the usual sense. Rather, they fulfill a number of higher level functions:

2.2.1 Samples / Project starting points

Global Information Exchange Package Documentations may act as starting points for further development. They may be used in part or in whole. The substantive information within a GIEPD may be used apart from other artifacts. Alternately, information about the development process itself may be used as a base for creating additional GIEPDs.

2.2.2 Jurisdictional Standards

Global Information Exchange Package Documentations may define jurisdictional standards. A high-level GIEPD may be further refined for sub-areas within a jurisdiction. Jurisdictions may make a GIEPD a mandatory standard.

2.2.3 Geopolitical Standards

Global Information Exchange Package Documentations may define national or state standards. The GIEPD may be further refined, constrained, or expanded at the state or local level.

2.2.4 Exchange Documents, Reference Documents

Global Information Exchange Package Documentations may be used to define message level and document level exchanges.

2.3 GIEPD Development Process Guidance

The purpose of this section is to provide guidance on how to develop GJXDM information exchange package documentation

2.3.1 Workgroup Composition

GJXDM Exchange Documents are created by a team of people called a *workgroup*. A workgroup is a group of people who are responsible for building the GIEPD. It is important to note that an individual can play more than one role in the workgroup. To be effective, the workgroup needs to contain both technical members and members with expertise in the business area of the exchange. In particular:

The workgroup MUST contain members with business expertise in the business area to be addressed by the exchange.

The workgroup MUST contain members with technical expertise in GJXDM, W3C XML Schema development, and domain modeling.

The workgroup MUST contain a facilitator whose role is:

- to coordinate the efforts of the workgroup
- to ensure that the development process documented here is followed by the workgroup
- to ensure that the other methodology, naming, and design rules documented in this specification are followed by the workgroup

2.3.2 Development Process Steps

The overall goal of the exchange document development process is to produce a package of artifacts that adhere to the methodology, naming and design rules in this specification. The rules in this section are intended to facilitate this goal.

2.3.2.1 Modeling and Mapping Meeting(s)

The workgroup SHOULD meet in person to start the development process.

The agenda for these meetings MUST include the following:

- Appropriate education of workgroup members about the development process to be followed
- Appropriate education of workgroup members about domain modeling and GJXDM mapping techniques and notations to be used
- Agreement among the workgroup as to how decisions will be made (e.g., by vote or consensus)
- Assigning the responsibility for each task to workgroup members
- Domain modeling of the exchange document structure
- Mapping of the domain model to GJXDM

The workgroup facilitator and/or technical workgroup members SHOULD create a candidate domain model prior to the first meeting. The candidate domain model SHOULD be based on existing paper document forms (if available) or input from business experts, to the extent feasible.

If it is not possible to complete the GJXDM mapping in the initial meeting, then the facilitator SHOULD ensure that the workgroup's attention be focused on the most significant domain model elements, as well as any elements that are potentially controversial.

2.3.2.2 Completion of Mapping

If the workgroup is unable to complete the domain-GJXDM mapping in the initial meeting, the facilitator MUST coordinate the efforts of the workgroup's business and technical/GJXDM experts to complete the mapping subsequent to the initial meeting.

The facilitator MUST present the mapping artifact to the workgroup for review, using whatever means is appropriate for the workgroup (e.g., in-person meeting, email, posting to a website).

2.3.2.3 Schema and GIEPD Creation

Following workgroup approval of the domain-GJXDM mapping, the facilitator MUST coordinate the efforts of the workgroup's technical/GJXDM experts to create XML Schemas based on the mapping. These schemas MUST adhere to the naming and design rules documented elsewhere in this specification.

All artifacts developed by the workgroup MUST be packaged according to the guidelines established in Section 3 – Information Exchange Package Documentation (GIEPD).

2.3.2.4 Iterative Process

The process outlined here is designed to be used in an iterative fashion, in which the steps in this section are repeated as often as needed. For example, a jurisdiction may develop an initial

version of an exchange document package, and at some point in the future may discover new requirements (or changes to previous requirements) that compel creation of a new version of the package. In this case, a workgroup would be formed, the workgroup would use the previous domain model and make any necessary changes, which would compel changes to the mapping and then the schema set. Similarly, a national association may develop a reference exchange document to serve as a model for local jurisdictions to use for a particular exchange. When modifying a reference exchange document, it is expected that the local jurisdiction would begin by forming a workgroup, modifying the domain model as necessary, then modifying the mapping and schemas accordingly.

2.3.2.5 Openess of Artifacts

An important goal of the development process is to produce artifacts that are usable by the widest possible audience. Consumers of the artifacts cannot be required to purchase expensive modeling or development tools in order to use the artifacts.

Also, the iterative nature of the development process requires artifacts that can be easily revised. The degree to which artifacts are reusable in new contexts is directly proportional to the openness of the artifacts and the tools used to create them.

See Section 3 for a delineation of artifact documentation requirements. The rules and guidelines included in Section 3 provide for openness of artifacts.

3 Information Exchange Package Documentation (GIEPD)

Mandatory and optional components of a GJXDM information exchange package document are as described in this section of the MNDR. The GIEPD is composed of the following components:

- Description of Development Methodology & Participants
- Domain Model Requirements (Graphical and Textual)
- Schema Set Requirements
- Supporting Documentation Requirements
- GIEPD Packaging Standard

3.1 GIEPD Development Methodology Description

A GIEPD Development Methodology description delineates the process used in creating a particular GIEPD. This documentation MUST include:

- **A Description of the GIEPD Development Process**

  GIEPDs MUST document the steps involved in their development. This helps ensure that best practices were followed in the development of the GIEPD. The process used to create the exchange document package, SHOULD BE recorded in Microsoft Word document(s).

- **Participants in the GIEPD Development Process**

  GIEPDs Development Process documentation MUST list participants in the process. Listing participants, including their affiliation and role (business, technical etc) helps ensure that a broad spectrum of participants was involved. This, in turn, ensures that the resulting GIEPD is applicable to most uses within the realm for which it was developed.

3.2 GIEPD Information Exchange Domain Model Description

Information Exchange Domain Model documentation acts as a guide to aid in the future implementation or modification of the GIEPD.

[Definition] Domain model

A graphical representation of the consensus of a group of business subject-matter experts as to the structure and content of an exchange document.

[Definition] Domain model element

Any symbol in a domain model used to describe a part of the structure of an exchange document. Elements include symbols used to describe individual data

elements, entire named data structures (or collections of data elements), and relationships between data elements or structures.

Domain modeling is an effort to establish agreement among project stakeholders regarding the content and structure of information in the document. It occurs after the business requirements of the exchange have been modeled, and before the exchange document structure is represented in XML Schema.

The output of domain modeling is a domain model. However, the creation of the actual model artifact is only part of the objective. Equally important is that business stakeholders—those who understand the necessary structure of the exchange document—reach consensus on what that structure shall be. Only when this consensus has been reached is the domain modeling step concluded.

A workgroup MUST create a domain model as part of the GIEPD document package. The domain model MUST adhere to the other methodology rules established in this section.

The domain model MUST be drafted before XML Schemas are created. The domain model MAY be edited as new requirements are discovered. Every data structure and data element in an exchange document's XML Schema(s) MUST be associated explicitly with at least one element in the domain model.

The domain model MUST be created in interactive modeling sessions with business subject-matter experts and is a representation of the consensus of the workgroup. Depending on the exchange under review, this could include a review of existing models, documents, vocabulary, a discussion of work flow processes, or other substantive discussions. The domain model MUST at all times reflect the consensus of the business subject-matter experts as to the structure and content of the exchange document.

Workgroups should recognize that the GJXDM already effectively models a wide range of justice domain concepts. Therefore, if GJXDM concepts fit the domain being modeled, the exchange document domain model SHOULD incorporate GJXDM concepts. However, care should be taken that the GJXDM data model does not take precedence over the functional business requirements.

- **Domain Model of business data and relationships.**

  Concept maps - are simple graphical models of relationships between concepts. A concept map diagram contains two basic elements: blocks, circles, or ovals representing concepts (with the concept named by text inside the symbol), and arrows connecting concept symbols to indicate relationships (relationships are named by text on or near the arrow.)

  In modeling exchange documents, the blocks, circles, or ovals represent data structures, and the arrows represent relationships between data structures. Arrows can be used to reflect inheritance (type-of or "is-a") relationships as well as containment, composition, and aggregation relationships. Cardinality of relationships such as one-to-one (1:1) and one-to-many (1:M) can also be included on the arrows.

  In the justice community, concept map diagrams have been used to represent GJXDM concepts and their relationships in developer training workshops and several exchange document development projects.

  The concept map or high level "*business* component model" MAY be used to facilitate the dialog between business and technical staff to determine the elements and relationships

of a data exchange. It is used to assist in the determination of the elements in the GJXDM dictionary that are to be used and provides technical staff building the schema with basic requirements that need to be reflected in the schema. It is a very good starting point easily understood by subject matter experts. A GIEPD MAY include this high-level model. Models SHOULD use diagramming techniques that are readily converted to UML diagramming notation. The concept map is a good starting point for developing the UML or UML equivalent detailed graphical representation of the domain model.

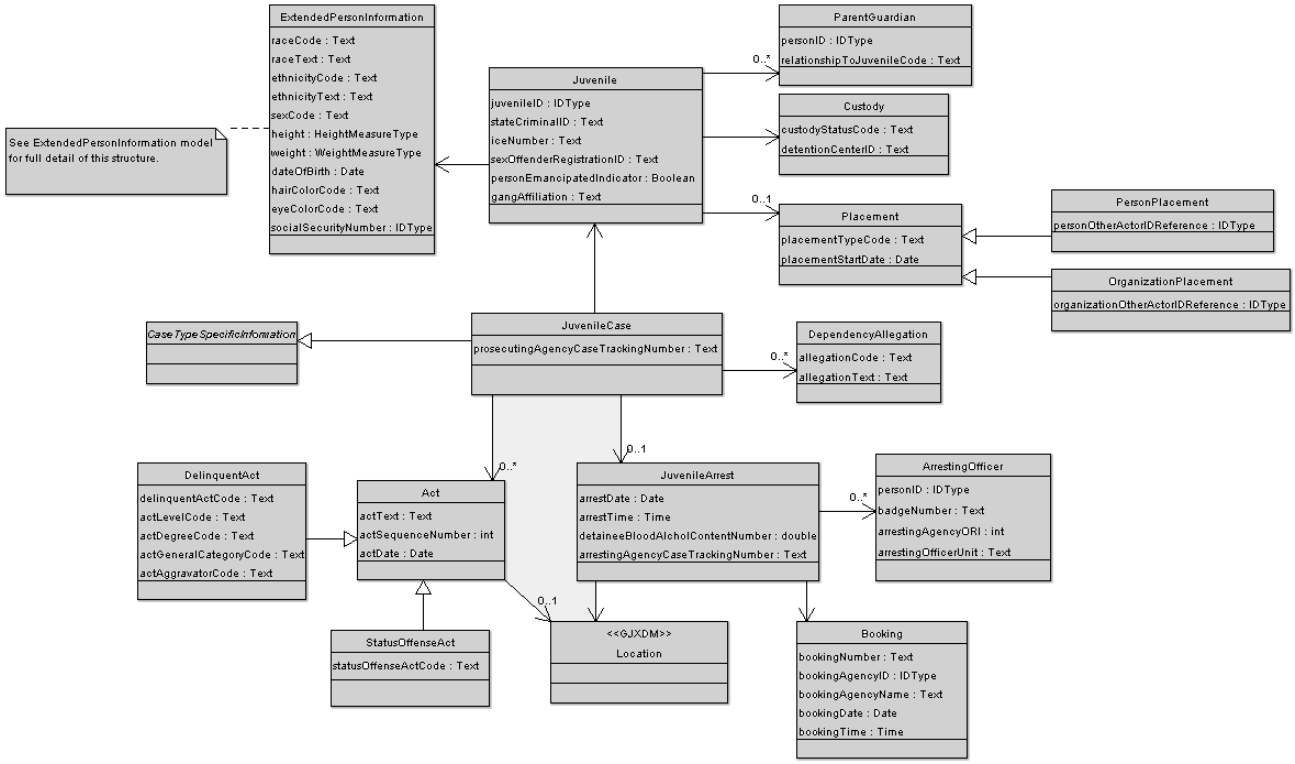
The main advantage of concept map diagrams is that the notation is very simple, with only two kinds of symbols (concepts and relationships.) The main disadvantage of concept map diagrams is that, in their simplicity, they lack some of the descriptive power of more formal notations like UML class diagrams. For instance, concept maps are unable to depict individual data elements within data structures (data elements must be handled as concepts in their own right.) For large and complex exchange documents, diagrams drawn with concept maps can become unwieldy. In addition, there is no normative, formal standard for representing concepts like inheritance and cardinality that are important to object-oriented domain modeling and that can be reflected directly in GJXDM-conformant XML Schemas. Finally, concept map diagrams are generally not interchangeable between modeling tools (there is no equivalent interchange standard for concept maps to correspond to XMI for UML diagrams.) A guide for formulating concept map diagrams can be found in the Non-Normative References **[CONCEPT Maps].**

Class Diagram - The Unified Modeling Language (UML) is a formal industry-standard specification for modeling notations. It is managed by the Object Management Group (OMG), an open industry standards body. It is a broad modeling specification, covering several types of diagrams and other notations. This specification refers only to UML Static Structure diagrams, also called Class diagrams.

Class diagrams are similar to concept maps, but differ in that class diagrams offer more formal notation for representing concepts and relationships between them. Class diagrams describe the types of elements in a data exchange and the various kinds of static relationships that exist among them. Class diagrams also show the attributes of elements and the constraints that apply to the way elements are connected. Class diagrams provide a normative, formal standard for depicting individual data elements within class structures as well as representing concepts like inheritance, cardinality and constraints. Class diagrams provide for interchangeability between modeling tools. Class diagram structure and detail facilitate automation of XML schema generation.

For complete descriptions of UML class diagrams, see **[UML]** in the Normative References section.

A GIEPD MUST include a detailed graphical representation of the domain model of the business data and relationships. Following is an [Example] *Juvenile Case Filing* domain model graphical representation from the OASIS LegalXML Electronic Court Filing 3.0 **[LegalXML ECF]** specification:



768

769 UML Class Diagrams are STRONGLY RECOMMENDED as a consistent method for graphically
770 depicting the domain model. They MAY be a generated artifact created from some other
771 representational source. A UML Class Diagram MAY NOT include non-standard UML symbols
772 and notations. Additionally, the mapping of elements to the GJXDM dictionary MAY NOT rely on
773 any Domain Model concept or element that cannot be represented in standard UML notation.

774 UML Class Diagrams MUST include an XML Metadata Interchange (XMI) formatted file as well as
775 an image file in a common open standard format such as jpeg, gif, png or svg. These files MUST
776 contain only pure UML.

777 Alternate representations MAY be used to show the business data and relationships, provided the
778 domain model is also supplied in one of the above approved formats (optional Concept Map and
779 UML Class Diagram or equivalent). Proprietary concepts MAY be included in this artifact as long
780 as the actual mapping results are not dependent on the concepts. The mapping MUST be
781 producible from the above approved formats alone.

782 If the domain model uses a notation that is different from UML, the domain diagram MUST be
783 semantically equivalent to a UML domain model.

784

785 • **Source documents or requirements specifications.**

786

787 A GIEPD MAY include source documents or supplemental requirement specifications
788 that help describe the data exchange. These materials may be actual paper documents,
789 or some electronic equivalent. As GIEP exchanges are not limited to existing paper
790 exchanges, there may not be existing documentation. If used, this documentation
791 SHOULD be included in the GIEPD to help ensure adequate coverage of existing
792 exchanges.

793

794 • **Spreadsheet of business data to GJXDM component mapping**

795
796 A GIEPD MUST contain a spreadsheet mapping business data requirements to GJXDM
797 components. Note that a spreadsheet may be a generated artifact and not necessarily the
798 instrument used to generate the requirements and their mappings to the GJXDM. The
799 purpose of the spreadsheet it to make the business data requirements and mappings
800 available in a universally readable format.

801

802 • **Subset Schema Generation Tool (SSGT) Want Lists**

803

804 A GIEPD MUST contain the appropriate SSGT Want List used to create its schemas.
805 GJXDM schemas can be created by means other than the SSGT. However, in order to
806 maintain

807

808 ## 3.2.1 Graphical Domain Model Documentation Rules

809 ### 3.2.1.1 Graphical Modeling Notations

810

811 The justice community has learned through experience that there is not a single domain modeling
812 notation that satisfies the needs of all workgroups. The skills and experience of the facilitator and
813 the availability of modeling tools are the most significant factors in determining which notation is
814 appropriate for a workgroup.

815 At the same time, reusability of exchange document packages is improved if the range of
816 modeling notations in use is kept to a minimum. The intent of this section is to balance these
817 factors.

818 The modeling notations identified in this section have been chosen because they satisfy the
819 following key criteria that are important to artifact reusability:

820 They are supported by tools supplied by multiple vendors, some of which are freely
821 available and/or ubiquitous

822 They are familiar to a wide range of facilitator/analysts, and are well-described in publicly
823 available documentation and training materials

824 They are accessible to the range of stakeholders involved in a typical exchange
825 document development project, including technologists and business subject-matter
826 experts (it is recognized that each notation may require a minimal amount of explanation
827 and coaching to be accessible to newcomers)

828

829 ### 3.2.1.2 Graphical Domain Model Artifacts

830

831 A concept map diagram MAY be utilized to initiate the data exchange dialog between
832 workgroup members

833

834 A UML static structure (class) diagram OR alternate diagram that provides functionality
835 equivalent to a UML diagram MUST be used

836

837 Graphical domain model artifacts, such as UML diagrams or informal diagrams, MUST be
838 presented in either JPEG or PNG format.

839

UML models MUST be presented in XML Metadata Interchange (XMI) format, version 1.2 or higher.

Domain model diagrams MAY BE presented in proprietary formats (i.e., formats that require processing by commercial tools), as long as they are also presented in accordance with the other rules in this section.

A domain model SHOULD be a UML static structure (class) diagram.

3.2.2 Textual Domain Model

A textual model is used to describe business data elements and their GJXDM mapping used in the information exchange. The textual model notation involves listing elements in the exchange document in a spreadsheet format. (This specification is neutral on the spreadsheet used, though it is expected that in many cases Microsoft Excel will be used due to its ubiquity.) Data elements are grouped into subject areas (or data structures). A definition is provided for each data element. Common business names for elements are listed with business oriented definitions that are sufficiently detailed to map to the GJXDM Dictionary. As mapping to the GJXDM Dictionary occurs, the GJXDM name, definition and GJXDM path are captured on the spreadsheet.

The business terms and definitions are used to determine if there is a GJXDM Dictionary entry. Terms in the dictionary must exactly match the business definitions for elements. If they do not match (close does not count), then an extension to the GJXDM must be defined.

The model MUST be in a tabular or spreadsheet format (that is, a format with cells, rows, and columns)

The mapping of the Textual Domain Model to GJXDM SHOULD BE recorded in a Microsoft Excel Spreadsheet.

The mapping MAY BE recorded within the a UML domain model , as long as the GJXDM mapping information can be presented along with the domain model structure in XMI format.

The model MUST include grouping of data elements into data structures and the first column of the tabular or spreadsheet format MUST indicate the data structure / group and the second column MUST indicate the data element.

The column immediately to the right of the data element column MUST indicate the business definition of the data element.

The sort order for the mandatory and optional fields included on the tabular or spreadsheet format can use any approach that can be easily understood.

A key advantage of a textual model is that it is a simple notation, involving only textual elements. The chief disadvantage of a textual model is that it does not offer graphical elements for representing important concepts, like relationships between data structures or sharing of common data structures from multiple places. Optionally describing these concepts in a “notes” section MAY be done as a supplement to information contained on the Domain Model.

887

888 The information contained in the Textual Domain Model together with Graphical Domain Model
889 detail provides the GIEPD requirements needed to build XML schema.

890

891 3.2.3 Textual Domain Model Documentation Rules

892 The following Textual Domain Model mapping table describes the name, definition, notes, and
893 required or optional status of each item contained in the mapping documentation table.

894

[MAP1]	All Domain Model and GJXDM Mapping Documentation MUST include the required content identified in the GJXDM Domain Model Mapping table. The table MAY contain the specified optional content.

895

896

GJXDM TEXTUAL DOMAIN MODEL MAPPING STANDARD			
Name	Definition	Notes	Required or Optional
Cardinality	Defines the minimum and maximum allowed occurrences for the element; left blank for class and relationship entries.	Examples of UML syntax follow below: 0..0 (element must not be used) 0..1 (optional element may only be used once) 0..n (optional element may be used up to n number of times) 0..* (optional element that may be used an unlimited number of times) 1..1 (required element that must be used once and only once) 1..n (required element that must be used at least once and may be used up to n number of times) 1..* (required element that must be used at least once may be used an unlimited number of times).	Required
Extension Inheritance	Documents the base type and type extensions of the GJXDM element(s) that were used to build the local extension, if any. .	This field is left blank for GJXDM content.	Required
Functional Element	The definition of the functional		Required

GJXDM TEXTUAL DOMAIN MODEL MAPPING STANDARD			
Name	Definition	Notes	Required or Optional
Description	(business) use and meaning of the data element.		
Functional Element Name	The functional, or business, element name.	Use the name of the related class (as defined by an association, aggregation, or composition relationship) if this element describes a domain model relationship	Required
GJXDM Path	Describes the full GJXDM schema path for the class, property, or relationship.	Namespace prefixes must be used to identify extensions. The path statement may include required attributes. For classes, this is just the type name. For the other content, this is the full path statement.	Required
Class Name	The name of the class or category that contains the element.	Depending on the modeling approach, this could be the UML class as defined in the domain model, or a more general category, such as “arrest information”.	Required
Code Set Source	The source, if any, of the codes assigned to this element.		Optional
Data Type	Defines the kind of information stored in the data element and describes how the field is formatted.		Optional
Examples	A sample value that illustrates how the element would be populated (e.g. “123” or “Accounts Payable”.)		Optional
Exchange Notes	Contains any additional information about exchange-specific mandatory and optional business rules and describes any recent changes made to the element.		Optional
Extension	Indicates that the element requires an extension to the GJXDM.	This field is left blank for GJXDM content	Optional

GJXDM TEXTUAL DOMAIN MODEL MAPPING STANDARD			
Name	Definition	Notes	Required or Optional
Field Length	The maximum field length, if available.		Optional
GJXDM Element Description	The complete GJXDM element description as defined in the GJXDM schemas.	Providing the GJXDM description along with the functional element description allows positive verification of the accuracy of the GJXDM mapping.	Required
GJXDM Element Name	The name of the equivalent GJXDM-conformant element.		Required
GJXDM Mapping Notes	Provides additional information about mapping decisions and explanations about which objects and/or elements were assigned to local extension.	May be used to document required attributes.	Optional
ID	A unique, non-changing number assigned to each row in the spreadsheet when it is added	This information would be useful if the sample instance document includes pointers to the domain model content.	Optional
Relationship	Used to indicate whether or not the element represents a relationship, so that a decision about whether to represent them though inclusion or references can be made later.		Optional
Version	Documents the version of the specification in which this element was incorporated or updated.		Optional

Sample GIEPD spreadsheets can be found in Appendix D – Sample GIEPD(s)

3.3 GIEPD Schemas

XML Schemas are the mechanism used to define GIEPs for the exchange(s) defined by the GIEPD. There are four different Schemas involved in a GIEPD. A supplemental discussion of the role of each type of schema can be found under **[GJXDM IEPD]** in the Non-Normative References section.

3.3.1 Document Schema

The GIEPD MUST include one-and-only-one document schema.

The document schema defines the root element of a GIEP and specifies the schema(s) to be used to validate the structure and contents of an XML Instance Document.

3.3.2 Subset Schema

The GIEPD MUST include one-and-only-one subset schema.

The Subset Schema removes unused GJXDM elements and type definitions. The Subset Schema narrows the focus on what information is actually being exchanged while also reducing the processing load required by validation. The Subset Schema is imported into the exchange via the Document Schema. Any instance document that validates against the Subset Schema MUST also validate against the GJXDM model as a whole.

3.3.3 Constraint Schema

If a constraint schema was developed or used, it MUST be included in the GIEPD.

The Constraint Schema embeds local constraints into GJXDM definitions. Constraint Schemas provide a variety of functionality. They provide a recommended place for enforcing cardinality rules. They also provide a means to enforce business rules that cannot be represented within the limitations of a Subset Schema.

When used solely to enforce cardinality, Constraint Schemas take the place of the Subset Schema for validation purposes. Enforcing cardinality is the more common use of a Constraint Schema.

When used to enforce business rules that go beyond the capacity of a Subset Schema, then Instance Documents are validated against both the Subset Schema (or data model as a whole) and the Constraint Schema in parallel. The Constraint Schema temporarily “stands-in” for the Subset Schema. Validation against such a Constraint Schema will only validate the additional business rules, not GJXDM conformance itself. For an Instance Document to be valid, it MUST validate against both the Subset Schema (or data model as a whole) and the Constraint Schema.

In addition, alternate constraint methods MAY be used, where the schema itself is more general and value constraints are applied using different XML methodologies (such as Schematron).

No new elements are permitted to be defined within the GJXDM constraint schema.

3.3.4 Extension Schema

If an extension schema(s) are developed, they MUST be included in the GIEPD.

The Extension Schema defines local extensions. While large, the data model can not include every element and type that could possibly be used in the context of an exchange of justice information. Some types of information are specific to particular jurisdictions or areas. It would be inappropriate and impractical for these local types of information to be contained in a national-scope data model.

The Extension Schema provides a means for this type of local information to be included in exchanges within the local area. Extension Schemas are imported into the exchange via the Document Schema.

3.4 GIEPD Instance Documents

[Sample instances] should include samples of both simple and complex information exchanges. Realistic data should be used (although data should be “sanitized” to omit actual identifying information that would violate privacy).

Instance documents are XML documents containing actual, although usually fictional, data. They are the payload, or “Package,” in an information exchange. They often show the selected elements of a Subset Schema more clearly than the schema itself, especially to a non-technical audience.

Instance Documents MAY show the minimal number of elements required to validate against the collected Schemas. Instance Documents MAY also show elements that are optional in the collected Schemas. It is RECOMMENDED that a GIEPD contain both minimal Instance Documents and more fully populated examples.

Instance Documents MUST validate though the schemas that are part of the Information Exchange Package Documentation.

3.4.1 Sample Source Documents & XML Instance Documents

Sample real world documents are a valuable tool to help others understand the actual data exchanged. Much of the technical material present in a GIEPD is difficult to read and understand. Sample source documents can clarify the scope and details of a GIEPD.

A GIEPD MAY contain one or more real world business documents representing component(s) of the GIEPD

3.4.1.1 Sample Instance Documents

A GIEPD MUST contain at least one sample XML instance document.

[Sample instances] SHOULD include samples of both simple and complex information exchanges. Realistic data should be used (although data should be “sanitized” to omit actual identifying information that would violate privacy).

Instance documents are XML documents containing actual, although usually fictional, data. They are the payload, or “Package,” in an information exchange. They often show the selected elements of a Subset Schema more clearly than the schema itself, especially to a non-technical audience.

Instance Documents MAY show the minimal number of elements required to validate against the collected Schemas. Instance Documents MAY also show elements that are optional in the collected Schemas. It is RECOMMENDED that a GIEPD contain both minimal Instance Documents and more fully populated examples.

Instance Documents MUST validate though the schemas that are part of the Information Exchange Package Documentation.

3.4.1.2 Sample Style Sheets and Styled Documents

A GIEPD MAY contain sample style sheets and resulting styled documents.

995 Even an XML instance document can be confusing for a layperson to read and understand. If the
996 GIEPD was based on existing paper documents, it may also be confusing as to whether the
997 resulting GIEPD meets the same requirements as the original paper documents.

998 Sample style sheets and the resulting styled documents can clarify the connections between the
999 XML Instance Documents and existing exchanges. If the GIEPD is based on existing exchanges,
1000 then sample style sheets and the resulting styled documents SHOULD be included.

1001 Additionally, style sheets may be used to transform XML instance documents into other variations
1002 of XML or into other data formats entirely.

1003 Where included, style sheets SHOULD use existing styling technologies and produce freely
1004 consumable output formats, such as HTML and PDF.

1005

1006 3.5 Other Supporting Documentation

1007 A wide and growing variety of tools is available for creating GIEPDs. Any supporting
1008 documentation that would be useful to others for further use and refinement of the GIEPD
1009 SHOULD be included in the GIEPD.

1010

1011 3.6 Packaging of GIEPD Artifacts (i.e., zip) and Naming rules

1012 This MNDR specifies File and Folder Naming rules for producing a GIEPD as follows:

1013

[NMS16]	GIEPD artifacts MUST be combined into a ZIP file. The ZIP file MUST have the filename: <IEP name>-<IEP version>.zip Where <IEP name> & <IEP version> = <IEP name> & <IEP version> in the IEP document namespace. For example, if IEP GJXDM xmlns = http://www.myDomainName.com/Citation/1.0/document , THEN the GIEPD ZIP filename would be: Citation-1.0.zip represents an GIEPD ZIP file for version 1.0 of a Citation IEP where <IEP name> = “Citation” and <IEP version> = “1.0”

1014

[NMS17]	The GIEPD ZIP file MUST have the following structure and contents. The root directory in the ZIP archive must contain the following files and directories:

	• The GIEPD Overview document file, in a suitable file format, and named "GIEPD Overview" with a file extension corresponding to the format. The "GIEPD Overview" MUST include the filename of the GIEPD ZIP file. • A directory named "domain model artifacts" • A directory named "mapping artifacts" • A directory named "schemas" • A directory named "sample instances" The directory named "domain model artifacts" MUST contain all artifacts related to the domain model of the IEP, as discussed in section 3 of this specification. The directory named "mapping artifacts" MUST contain all artifacts related to the mapping of the domain model to GJXDM, as discussed in section 3 of this specification. The directory named "sample instances" MUST contain one or more sample XML instances that are valid against the document, extension, constraint, and subset schemas in the IEP. Each sample XML instance: 1) MUST associate referenced IEP namespaces by using the xsi:schemaLocation attribute on the XML Instance root element; 2) MUST use the xsi:schemaLocation attribute 3) MUST use a relative URL, valid within the IEP structure documented here, to locate the schema for each namespace. The directory named "schemas" MUST contain the following: • A document schema file (see rule NMS12 and NMS13 for filename rules) • Extension schema files(s) (see rule NMS10 and NMS11 for filename rules) • A constraint schema file, if the IEP uses a constraint schema (see rule NMS14 and NMS15 for filename rule) • A directory named "subset" that contains the subset schema set; the sub-directory structure underneath the "subset" directory must match the directory structure of the GJXDM distribution version being used. *Note that the subset schema is not a single schema; rather, it is a directory structure that contains many related schemas.*

	To the extent that unzipped schemas in the IEP import each other, the schemaLocation attribute for each schema's xsd:import element(s) MUST use a relative URL to locate the imported schema. The relative URL MUST be valid within the structure of the ZIP file specified above.

1015

1016 A sample GIEPD ZIP file can be found in Appendix D – Sample GIEPD(s)

1017

4 Schema Set and Instance Document Naming and Design Rules

This section provides detailed Naming & Design rules for developing a GJXDM conformant GIEPD. Included in this section are Naming & Design standards for GIEPD XML schemas, including XML types, elements, attributes, permitted and non-permitted XSD methods and documentation standards. This section also describes the rules for constructing XML Instance documents, including requirements for root elements and validation methods.

4.1 General Schema Set Naming and Design Rules

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 subsection 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.

Adherence to these rules will ensure that semantics are unambiguous, enabling the practitioner teams to conduct straightforward comparisons and make recommendations with respect to enterprise reusability across their respective organizations. GJXDM information exchange schema(s) and XML Instances rules are modeled after the same naming and design conventions used to develop the Global Justice XML Data Model (GJXDM).

4.1.1 General Naming Rules

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, DOJ-Global has determined that the best approach for developing a national XML exchange standard is to base its work on W3C XSD.

[STA1]	All GJXDM information exchange schema design rules MUST be based on the W3C XML Schema 1.0 Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

A W3C technical specification holding recommended status represents consensus within the W3C and has the W3C Director's stamp of approval. Recommendations are appropriate for widespread deployment and promote W3C's mission. Before the Director approves a recommendation, it must show an alignment with the W3C architecture. By aligning with W3C specifications holding recommended status, DOJ-Global can ensure that its products and deliverables are well suited for use by the widest possible audience with the best availability of common support tools.

[STA2]	All GJXDM information exchange schema and payloads MUST be based on the W3C suite of technical specifications holding recommended or higher status.

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.

[GNR1]	User-defined information exchange XML elements, attributes and type names MUST be composed in the English language, using the primary English spellings provided in the Webster’s English Dictionary.

The ebXML Core Component Technical Specifications (CCTS) provides a rule set for precisely defining the semantics of a data element in terms of a tripartite naming convention specified by ISO 11179 Part 5 (object class, [qualifiers], property term, and representation term). The three parts are combined to form the names of GJXDM information exchange XML elements, complex types, and attributes.

[GNR2]	User-defined information exchange XML element, attribute and type names MUST be ebXML CCTS ISO 11179 compliant
[GNR3]	User-defined information exchange XML element, attribute and type names MUST NOT include spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names.

Element names and the derivative names MUST be consistent with the GJXDM, even when this would result in conflicts with GNR1 or GNR2.

[GNR9]	GJXDM element names, attributes and type names MUST not be modified, even when GJXDM names conflict with rules GNR1 – GNR3. For example, the use of periods in GJXDM conflicts with GNR3.

Acronyms and abbreviations impact semantic interoperability and are to be avoided to the maximum extent practicable. Since some abbreviations will inevitably be necessary, GJXDM maintains a normative list of authorized acronyms and abbreviations.

Appendix B provides the current list of permissible acronyms, abbreviations and word truncations. The intent of this restriction is to facilitate the use of common semantics and to foster greater understanding. Appendix B is a living document and will be updated by the Global XSTF task force to reflect growing requirements.

[GNR4]	GJXDM information exchange XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix B.
[GNR5]	The acronyms and abbreviations listed in Appendix B MUST always be used.

Generally speaking, the names for GJXDM information exchange XML constructs must always be singular. The only exception permissible is where the concept itself is pluralized.

[GNR6]	GJXDM information exchange XML element, attribute and type names MUST be in singular form unless the concept itself is plural.

Example:

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

XML is case sensitive. Consistency in the use of case for a specific XML component (element, attribute, type) is essential to ensure every occurrence of a component is treated the same. This is especially true in a business-based data-centric environment such as that addressed by GJXDM. Additionally, the use of visualization mechanisms such as capitalization techniques assist in ease of readability and ensure consistency in application and semantic clarity. The ebXML architecture document specifies a standard use of upper and lower case for expressing XML elements and attributes respectively. GJXDM adheres to the ebXML standard. Specifically, GJXDM element and type names will be in UpperCamelCase (UCC).

[GNR7]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.

UpperCamelCase Example:

{Capitalizes the first letter “P” and each sub-word “N”}

```
PersonName
JewelryStone
```

GJXDM information exchange attribute names will be in lowerCamelCase (LCC).

[GNR8]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.

LowerCamelCase Example:

```
{lowercase used for the first letter “a” and upper-case for each sub-
word “C”,”C”,”L” …}
```

```
amountCurrencyCodeListVersionID
characterSetCode
```

GJXDM instance documents are designed to effect data and document electronic exchanges. Including mixed content in exchange documents is undesirable because exchange transactions are based on exchange of discrete pieces of data that must be clearly unambiguous. The white space aspects of mixed content make processing unnecessarily difficult and add a layer of complexity not desirable in information exchanges.

[MDC1]	Mixed content MUST NOT be used except where contained in an xsd:documentation element.

[Definition] Mixed Content

An XML <element> that contains both "string data" and other <element> data is defined as having mixed content.

[Example of Mixed Content]

The XML element <book> has mixed content because it contains string data "The Three Musketeers" and <element> data which is <author>

```
<book>
    The Three Musketeers
    <author> Alexander Dumas </author>
</book>
```

[Example without Mixed Content]

The XML element <book> has only <element> content namely <title> and <author> and therefore is NOT mixed content.

```
<book>
    <title>The Three Musketeers</title>
    <author> Alexander Dumas </author>
</book>
```

The features of W3C XML Schema allow for flexibility of use for many different and varied types of implementation. The GJXDM information exchange MNDR uses the following rules to allow for a more consistent use of these features:

[GXS4]	The root element in all GJXDM information exchange Schema modules MUST contain the following namespace declaration: "xmlns:xsd=http://www.w3.org/2001/XMLSchema."

To avoid overloading implementation systems with unnecessary documentation, developers have an option to create a schema without the documentation. This schema is for run-time and must be a functional equivalent of the documented version.

[GXS5]	GJXDM information exchange schema developers MAY provide a run-time schema devoid of documentation in addition to the fully annotated version.

4.1.1.1 Schema Built-in Simple Types

There are 44 simple types built into XML Schema. They are specified in Part 2 of the XML Schema Recommendation. These built-in types were used in the designing of GJXDM schema. Simple types are the concrete representations of the datatypes defined by ebXML Core Components and specialized dataTypes defined by GJXDM referred to as proxied GJXDM simple types j-xsd:, rather than using the xsd:schema simple types directly. Extensions to simple types must use as their base the set of provided simple types defined in GJXDM.

The GJXDM Schema module incorporates XML Schema built-in types and fundamental CCTS types. The GJXDM Schema module declares the built-in types to be used.

[GXS6]	Any user defined types with simple content MUST be derived via extension or restriction on the proxied GJXDM simpleTypes defined in **http://www.it.ojp.gov/jxdm/3.0/proxy/xsd/1.0 xsd.xsd**

4.1.1.2 XSD:appinfo

The `xsd:appinfo` feature is used by schema to convey processing instructions to a processing application, Stylesheet, or other tool. Some users have determined that this technique poses a security risk and have employed techniques for stripping `xsd:appinfo` from schemas. However, as GJXDM MNDR is committed to ensuring the widest possible target audience, this feature may be used to convey non-normative information. Non-normative information means non-standard.

[GXS14]	GJXDM designed schema MAY use xsd:appinfo. If used, xsd:appinfo MUST only be used to convey non-normative information. Note: appinfo is a recent addition to GJXDM in Version 3.0.2

4.1.2 Namespaces and Schema Locations

[NMS1]	Every GJXDM information exchange schema MUST have a namespace declared using the `xsd:targetNamespace` attribute.

1193

[NMS2]	Every GJXDM information exchange schema version MUST have its own unique namespace.

1194

[NMS4]	GJXDM namespaces MUST only contain GJXDM conformant schema modules.

1195

[NMS5]	GJXDM published namespaces MUST never be changed. The namespace names for GJXDM reference schema releases are of the form:
	
	`http://www.it.ojp.gov/jxdm/{major version . minor release . revision}`
	
	`For example the following namespace http://…/jxdm/3.1.1 would be major-release 3 and minor-release 1.1 of the GJXDM schema.`

1196

1197 ### 4.1.3 External Code List Rules

[CDL1]	All Global JXDM Code Lists MUST be part of a Global JXDM or externally maintained Code List; they MUST NOT be included in a document or extension schema..
[CDL2]	The Global JXDM SHOULD identify and use external standardized code lists whenever practical rather than develop its own Global JXDM-native code lists.
[CDL3]	The Global JXDM information exchange developer, through extension/restriction, MAY design and use a “contextually” defined code list where an existing GJXDM code list needs to be extended, or where no suitable external code list exists.
[CDL4]	All GJXDM maintained or information exchange developer Code Lists MUST be enumerated using the GJXDM Code List Schema Module.
[CDL5]	The name of each GJXDM information exchange Code List Schema MUST be of the form: {`Owning Organization`}_{`Code List Name`}_[version number}].xsd ;
[CDL6]	An `xsd:import` element MUST be declared for every code list required in a GJXDM information exchange schema. Each codelist MUST be in its own namespace; the namespace identifier MUST be consistent with the same rules as extension schemas.
[CDL7]	When creating a local code list, an information exchange developer MUST follow the UBL code list schema and annotation rules. http://docs.oasis-open.org/ubl/cd-UBL-1.0/doc/cl/wd-ublclsc-codelist-20040420.pdf
[CDL8]	Users of the GJXDM MAY identify any subset they wish from an identified code list for their own trading community conformance requirements.
[CDL9]	The `xsd:schemaLocation` MUST include the complete URI used to identify the relevant code list schema.

4.1.4 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.

[GTD1]	All types MUST be named.

Example:

```
<xsd:complexType name="PersonType">
...
</xsd:complexType>
```

4.1.5 Complex Type Definitions

Since even simple datatypes are modeled as property sets in most cases, the XML expression of these models primarily employs `xsd:complexType`. To facilitate reuse, versioning, and customization, all complex types are named. In the GJXDM information exchange model, xsd:complexType(s) with complex content are considered classes(objects) .

[CTD1]	For every class identified in GJXDM extension and document schema, a named xsd:complexType MUST be defined.

Example:

```
<xsd:complexType name="BuildingType">
….
</xsd:complexType>
```

GJXDM classes(objects) are defined in schema as named complexTypes. The sequence of elements contained within the complex type represent the properties of the class(object). These property elements are defined as global elements where each global element.references a corresponding complex or simple type. GJXDM named complexTypes may be customized by creating a user-defined complexType. The user-defined complexType would utilize the `xsd:extension base=GJXDMtype and add additional xsd:sequence elements to extend the GJXDM class(object).`

[CTD2]	Every GJXDM user-defined xsd:complexType definition content model MUST use the

	xsd:sequence element with appropriate global element references to reflect each property of its class. This does not preclude the use of xsd:choice (see *rule name GXS13*)

GJXDM Example:

```
<xsd:complexType name="BailType">
 <xsd:complexContent>
  <xsd:extension base="ActivityType">
   <xsd:sequence>
   <xsd:element ref="BailSetAmountText" minOccurs="0" maxOccurs="unbounded"
/>
  <xsd:element ref="BailSetCourt" minOccurs="0" maxOccurs="unbounded" />
  <xsd:element ref="BailSetCourtReference" minOccurs="0"  maxOccurs =
"unbounded"/>
 …
  </xsd:sequence>
 </xsd:extension>
</xsd:complexContent>
```

There is a direct one-to-one relationship between ebXML CoreComponentTypes and GJXDM PrimaryRepresentationTerms. Additionally, there are several GJXDM SecondaryRepresentationTerms that are subsets of their parent GJXDM PrimaryRepresentationTerm. The total set of ISO 11179 Representation Terms by their nature represent GJXDM Datatypes. Specifically, for each GJXDM PrimaryRepresentationTerm or GJXDM SecondaryRepresentationTerm, an ebXML UnspecializedDatatype exists. In the GJXDM , these ebXML UnspecializedDatatypes are expressed as complex or simple types that correspond to an ebXML CoreComponentType.

The set of valid GJXDM datatypes are based on ebXML Core Component Technical Specification v1.9 and include the following:

1) Amount
2) BinaryObject (secondary: Graphic,Picture,Sound,Video)
3) Code
4) *DateTime (secondary: Date, Time)
5) Identifier (authorized abbreviation: ID)
6) Indicator
7) Measure
8) Numeric (secondary: Value, Rate, Percent)
9) Quantity
10) Text (secondary: Name)

Reference:GTRI May 2004 Developer Workshop

*note: DateTime element not in GJXDM but secondary Date and Time elements are included

[CTD3]	For every user-defined datatype based on the valid set of GJXDM datatypes, a named xsd:complexType or xsd:simpleType MUST be defined.

4.1.6 Complex Type Naming Rules

GJXDM identifies naming rules for types, namely for complex types based on Primary Representation Terms, Secondary Representation Terms and the ebXML Core Component Types. Each of these complex and simple types are a fully qualified type name based on ISO 11179. As such, these names convey explicit semantic clarity with respect to the data being described. Accordingly, these `naming standards` ensure that GJXDM xsd:complexType names are semantically unambiguous, and that there are no duplications of GJXDM type names for different `xsd:type` constructs.

GJXDM `xsd:complexType` names follow general naming rules, and append the suffix "`Type`" to denote a Type Name versus an Element Name.

[CTN1]	A user-defined xsd:complexType ***name*** MUST name the object suffixed by the word "Type". For example <xsd:complexType name="PersonType"> is correct And <xsd:complexType name="Person"> would be incorrect.

4.1.7 Attribute Declarations

Attributes are W3C Schema constructs associated with elements that provide further information regarding elements. While elements can be thought of as containing data, attributes can be thought of as containing metadata. Unlike elements, attributes cannot be nested within each other—there are no "subattributes." Therefore, attributes cannot be extended as elements can. Attribute order is not enforced by XML processors—that is, if the attribute order in an XML instance document is different than the order in which the attributes are declared in the schema to which the XML instance document conforms, no error will result. These limitations dictate that GJXDM MNDR restrict the use of attributes to XSD built-in attributes, or to the GJXDM SuperTypeMetadataAttributeGroup defined by GJXDM.

For a more complete discussion of the application of attributes within GJXDM, the reader should reference the **[GJXDM USER]** link provided in the Non-Normative References Section. Specifically, Part 3 "Metadata in the Global Justice XML Data Model" and Part 5 "XML Schema Elements Versus Attributes" of the GJXDM Users Manual should be reviewed.

[ATD1]	User defined attributes SHOULD NOT be used.

4.1.7.1 Global Attributes

The current GJXDM has attributes that are common to all GJXDM and proxy Code List elements. These common attributes have been declared using the `xsd:globalattributegroup` element and utilizes the following rule. These rules are included to ensure interoperability.

[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. (For an example about how to do this, see the Global JXDM global attribute group named "**SuperTypeMetadata")**

[ATD3]	For each `Global JXDM user-defined element of simpleType and a xsd:restriction` element; an `xsd:base` attribute MUST be declared and set to the appropriate GJXDM datatype. `Note: the set of valid GJXDM datatypes are based on ebXML Core Component Technical Specification v1.9 and include the following 10 simpleTypes:` • Amount • BinaryObject (secondary: Graphic, Picture,Sound,Video) • Code • DateTime (secondary: Date, Time) • Identifier (authorized abbreviation: ID) • Indicator • Measure • Numeric (secondary: Value, Rate, Percent) • Quantity • Text (secondary: Name) Reference:GTRI May 2004 Developer Workshop

4.1.7.2 XSD:nil

You can indicate in a schema that an element may be nil in the instance document. Empty content vs nil:

- Empty: an element with an empty content is constrained to have no content.
- nil: an instance document element may indicate no value is available by setting an attribute - xsi:nil - equal to 'true'

Example:

```
XML Schema:
<xsd:complexType name="PersonNameType">
    <xsd:complexContent>
      <xsd:extension base="j:SuperType">
        <xsd:sequence>
…
          <xsd:element ref="j:PersonGivenName" minOccurs="0"
maxOccurs="unbounded"/>
…
          <xsd:element ref="j:PersonMiddleName" minOccurs="0"
maxOccurs="unbounded"/>
…
          <xsd:element ref="j:PersonSurName" minOccurs="0"
maxOccurs="unbounded"/>
…
        </xsd:sequence>
…
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

<xsd:element name="PersonName" type="j:PersonNameType" nillable="true"/>

<xsd:element name="PersonGivenName" type="j:PersonNameTextType"
nillable="true"/>

<xsd:element name="PersonMiddleName" type="j:PersonNameTextType"
nillable="true"/>

<xsd:element name="PersonSurName" type="j:PersonNameTextType" nillable="true"/>


XML instance document:

<PersonName>
   <PersonGivenName>Hannibal</PersonGivenName>
   <PersonMiddleName xsi:nil=”true”/>
   <PersonSurName>Lecter</PersonSurName>
</PersonName>
```

[ATD5]	For local extensions based on GJXDM version 3.0.2 and above, the xsd built-in nillable attribute MUST be explicitly defined for any Global JXDM user-defined element which has simpleContent . The user-defined element must set nillable to “**true**” or “**false**”; Note: An example from GJXDM v3.0.2 <xsd:element name="**WitnessLocationDescriptionText**" type="**j:TextType**" nillable="**true**">

4.1.7.3

4.1.7.4 Empty Elements

Empty elements may cause application or XML processing difficulty and must be avoided

[ELD5]	Empty elements MUST NOT be declared.

4.1.7.5 XSD:Any Element

GJXDM MNDR disallows the use of `xsd:any`, because this feature permits the introduction of potentially unknown elements into an XML instance. GJXDM MNDR intends that all constructs within the instance be described by the schemas describing that instance - `xsd:any` is seen as working counter to the requirements of interoperability. In consequence, particular attention is given to the need to enable meaningful validation of the document instances.

Were it not for this, `xsd:any` might have been allowed.

[ELD7]	The `xsd:any` element MUST NOT be used.

GJXDM information exchange schema disallows the use of `xsd:anyType`, because this feature permits the introduction of potentially unknown types into an XML instance. The <any> element enables the instance document author to extend his/her document with elements not specified by the schema. GJXDM intends that all constructs within the instance be described by the schemas describing that instance. Per UBL-NDR version 1.0.1 , "`xsd:anyType` is seen as working counter to the requirements of interoperability. In addition, use of xsd:anyType has been identified as a significant security risk. In consequence, particular attention is given to the need to enable meaningful validation of the document instances. Were it not for this, `xsd:anyType` might have been allowed. "

[GTD2]	The xsd:anyType MUST NOT be used.

4.1.7.6

4.1.7.7 XSD:anyAttribute

GJXDM information exchange schema disallows the use of `xsd:anyAttribute`, because this feature permits the introduction of potentially unknown attributes into an XML instance. GJXDM information exchange packages intend that all constructs within the instance be described by the schemas describing that instance - `xsd:anyAttribute` is seen as working counter to the requirements of interoperability. In consequence, particular attention is given to the need to

enable meaningful validation of the GJXDM conformant document instances. (Also see rule GTD2 on the xsd:anyType and ELD7 xsd:any element)

[ATD6]	The xsd:any attribute MUST NOT be used.

4.1.8 Element Declarations and Naming Rules

4.1.8.1 Elements Bound to Complex Types

W3C XSD allows for any globally declared element to be the document root element. To keep consistency in the instance documents and to adhere to the underlying process model that supports each GJXDM information exchange Schema, it is desirable to have one and only one element function as the root element. Since GJXDM follows a global element declaration scheme (See Rule ELD2), each GJXDM Schema will identify one element declaration in each schema as the document root element. This will be accomplished through an `xsd:annotation` child element for that element in accordance with the following rule:

[ELD1]	Each GJXDM `DocumentSchema` MUST identify one and only one global element declaration that defines the exchange document being conveyed in the Schema expression. That global element MUST include an xsd:annotation child element which MUST further contain an xsd:documentation child element that declares "*This element MUST be conveyed as the root element in any instance document based on this `DocumentSchema`.*"

[Definition] Document schema:

The overarching schema within a specific namespace that conveys the business document functionality of that namespace. The document schema declares a target namespace and is likely to pull in additional schema by importing external schema modules. Each namespace will have one, and only one, document schema.

Example:

```
<xsd:element name="Rapsheet" type="RapsheetType">
<xsd:annotation>
<xsd:documentation>This element MUST be conveyed as the root element in
any instance
document based on this Schema expression</xsd:documentation>
</xsd:annotation>
</xsd:element>
```

Global elements are declared as direct children of a root schema element. Global element names, because they are globally reusable, express universally unambiguous semantics in their names. By their nature, global elements can be reused consistently across the Justice & Public Safety enterprise XML domain, wherein every occurrence will have exactly the same meaning and map to exactly the same authoritative source data. Therefore, the GJXDM information exchange schema approach uses global elements in conjunction with global complex types.

Eliminating potential barriers to Justice XML system interoperability is a key driver for choosing the global element approach. The advantages of this approach become more evident when considering inter-agency XML interoperability and data exchange. Because the recommended approach relies on global elements to carry unique semantics, there cannot be a duplicate occurrence of elements with different characteristics in the GJXDM enterprise namespace. This reduces the level of effort to analyze, map, and transform elements that are unique to a particular functional area.

[ELD2]	All element declarations MUST be global.

4.1.8.2 Element Names for complexType(s)

For each class (complexType) defined in an extension, subset or document schema, a global element name MUST be declared with the same name as the name of the corresponding `xsd:complexType` to which it is bound, with the word "`Type`" removed.

Example:

The xsd:complexType named TelephoneNumberType would require a corresponding global element be declared. The the name of this corresponding global element must be TelephoneNumber.

```
<xsd:element name="TelephoneNumber" type="j:TelephoneNumberType"
nillable="true">

<xsd:complexType name="TelephoneNumberType">
…
</xsd:complexType>
```

[ELD3]	For every class defined as a GJXDM user-defined types, a global element bound to the corresponding xsd:complexType MUST be declared. For example, a schema defining a complexType named my:FavoritePersonType would need to declare a global element named "FavoritePerson" of objectType = my:FavoritePersonType to bind a global element name to the complexType.

4.1.8.3 ComplexTypes with SimpleContent

Elements bound to ebXML core-component types and GJXDM j-xsd: proxy built-in xsd dataTypes.

[ELD4]	For every user-defined simpleType, an xsd:restriction element MUST be declared

4.1.8.4 Global Elements with simpleContent

The GJXDM Global elements with simpleContent are reused in multiple contexts. Their reuse in a specific context is typically identified in part through the use of qualifiers. However, these qualifiers do not change the nature of the underlying concept of the GJXDM core component that the element is derived from. As such, qualified global elements with simpleContent are always bound to the same type as that of their unqualified corresponding xsd: built-in datatype, j-xsd or GJXDM (ebXML core component) datatype.

Example:

```
<xsd:element name="StreetNumberText" type="j:TextType"/>
```

[ELD6]	Global simpleType elements declared with Qualified Properties must be of the same type as their corresponding Unqualified Property.

4.1.9 Schema Documentation and Annotations

4.1.9.1 Embedded documentation

The information about each GJXDM user-defined “Type” or “Element” must be documented in schema. Fully annotated Schemas are valuable tools to implementers to assist in understanding the nuances of the information contained therein. GJXDM information exchange schema annotations will consist of a set of metadata elements.

The absence of an optional annotation inside the structured set of annotations in the documentation element implies the use of the default value. For example, there are several annotations relating to context such as `BusinessContext` or `IndustryContext` whose absence implies that their value is "`all contexts`".

The following general documentation rule describes the documentation requirements for GJXDM user-defined “Types” or “Elements”.

[DOC1]	The xsd:documentation element for every GJXDM user-defined Element MUST contain a structured set of annotations in the following sequence and pattern: • Version (optional): An indication of the evolution over time of the Datatype. • Definition(mandatory): The semantic meaning of an Element

	• Cardinality(mandatory): Indication whether the complexType Element (Property) represents a not-applicable, optional, mandatory and/or repetitive characteristic of the parent complexType • AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers describe the 'context' of the relationship with another complexType object. That is, it is the role the contained Element plays within its association with the containing complexType object. • AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at the other end of this association. It represents the Aggregate Business Information Entity contained by the Association Business Information Entity. (For example: the element PersonName within the complexType PersonType has Name as the AssociatedObjectClass contained in the Aggregate Business Information Entity called PersonType.) • AlternativeBusinessTerms (optional): Any synonym terms under which the Element is commonly known and used in the business. • Examples (optional): Examples of possible values for the Element

Example Extension Element Documentation:

```
<xsd:annotation>
  <xsd:documentation>
    <Component>
      <ElementName>ChargeArrestReference</ElementName>
<Version>1.0</Version>
      <Definition> A reference to the Arrest which resulted in the
filing of a charge..</Definition>
      <Cardinality>0..1</Cardinality>
      <ObjectClass>Charge</ObjectClass>
      <PropertyTerm>Arrest</PropertyTerm>
      <RepresentationTerm>Reference</RepresentationTerm>
      <AssociatedObjectClass>Reference</AssociatedObjectClass>
    </Component>
  </xsd:documentation>
</xsd:annotation
```

4.1.10 Schema Version Numbering Rules

GJXDM user-defined namespaces are suffixed with a `IEP name and IEP version.` The GJXDM MNDR has decided to include versioning information to immediately follow the `IEP name` component of the namespace. The version information is divided into `major` and `minor`

fields. The `minor` field has an optional `revision` extension. For example, the namespace URI for an IEP Charging Document schema has this form:

```
http://<IEP owner domain name>/<IEP name>/<IEP version>/<IEP schema
type[-suffix]>

note: where <schema Type> denotes  document or extension
```

The major-version field is "1" for the first release of a namespace. Subsequent major releases increment the value by 1.

For example, the first namespace URI for the first major release of the Charging document has the form:

```
http://myNS.com/Complaint/1.0/document
```

The second major release will have a URI of the form:

```
http://myNS.com/Complaint/2.0/document
```

The distinguished value "0" (zero) is used in the minor-version position when defining a new major version. In general, the namespace URI for every major release of the Complaint domain has the form:

Example:

```
http://myNS.com/<IEP name>/<major-number>.0[.<revision>]/<schema type>
```

[VER1]	Every GJXDM information exchange schema and schema module major version ***draft*** MUST have a version number of the form: <major>.0[.<revision>]

When a major version reaches Standard status the [.<revision>] must not be present.

[VER2]	Every GJXDM Information exchange Schema and schema module major version ***Standard*** MUST have a version number of the form: <major>.0

For each document produced by the TC, the TC will determine the value of the <IEP name> variable. In GJXDM MNDR, the major-version field of a namespace URI must be changed in a release that breaks compatibility with the previous release of that namespace. If a change does

not break compatibility then only the minor version need change. Subsequent minor releases begin with minor-version 1.

Example

The namespace URI for the first minor release of the Complaint document has this form:

```
http://myNS.com/Complaint/1.1/document
```

[VER3]	Every minor version release of a GJXDM Information exchange schema or schema module ***draft*** MUST have a version number of the form: <major >.<non-zero>[.<revision>]

When a minor version reaches Standard status the [.<revision>] must not be present.

[VER4]	Every minor version release of a GJXDM information exchange schema or schema module ***Standard*** MUST have a version number of the form: <major >.<non-zero>

Once a schema version is assigned a namespace, that schema version and that namespace will be associated in perpetuity. Any change to any schema module mandates association with a new namespace**.**

[VER5]	For GJXDM information exchange schema minor version changes, the <IEP name> MUST NOT change.

If a GJXDM schema namespace URI changes then any schema that imports the new version of the namespace must also change (to update the namespace declaration). And since the importing schema changes, its namespace URI in turn must change. The outcome is twofold:

- **There should never be ambiguity at the point of reference in a namespace** declaration or version identification. A dependent schema imports precisely the version of the namespace that is needed. The dependent schema never needs to account for the possibility that the imported namespace can change.

- **When a dependent schema is upgraded to import a new version of a schema,** the dependent schema's version (in its namespace URI) must change.

Version numbers are based on a logical progression. All major and minor version numbers will be based on positive integers. Version numbers always increment positively by one.

[VER6]	For every GJXDM information exchange schema and schema module, the major version number MUST be a sequentially assigned, incremental number greater than zero.

[VER7]	For every GJXDM information exchange schema and schema module, the minor version number MUST be a sequentially assigned, incremental non-negative integer.

In keeping with rules NMS1 and NMS2, each schema minor version will be assigned a separate namespace. A minor revision (of a namespace) imports the schema module for the previous version.

For instance, the document schema defining:

```
http://myNS.com/Complaint/1.2/document
```

will import the namespace:

```
http://myNS.com/Complaint/1.1/document
```

The `version 1.2` revision may define new complex types by extending or restricting `version 1.1` types. It may define brand new complex types and elements. It must not use the XSD redefine element to change the definition of a type or element in the `1.1` version.

The opportunity exists in the `version 1.2` revision to rename derived types. For instance if `version 1.1` defines `Address` and `version 1.2` specializes `Address` it would be possible to give the derived `Address` a new name, e.g. `NewAddress`. This is not required since namespace qualification suffices to distinguish the two distinct types.

The minor revision may give a derived type a new name only if the semantics of the two types are distinct.

For a particular namespace, the minor versions of a major version form a linearly-linked family. The first minor version imports its parent major version. Each successive minor

version imports the schema module of the preceding minor version.

Example

```
http://myNS.com/Complaint/1.2/document
```

imports

```
http://myNS.com/Complaint/1.1/document
```

which imports

```
http://myNS.com/Complaint/1.0/document
```

Ensuring semantic compatibility across minor versions is essential. Semantic compatibility in this sense pertains to preserving the business function.

[VER8]	GJXDM information exchange schema and schema module minor version changes MUST not break semantic compatibility with prior versions; nor may they break existing document instances that are based on any earlier minor version of the last major version. For example, an instance document build on a 1.1 minor version must be able to be processed by any later minor release, for example, a 1.9 version. Minor versions maintain forward compatibility.

Major versions of schema do NOT necessarily have to maintain forward compatibility with the previous major version.

[VER9]	GJXDM information exchange schema and schema module major version changes MAY break semantic and/or structural compatibility with prior versions. No backward compatibility is guaranteed.

4.1.11 Import versus Include

4.1.11.1 Schema Modularity

GJXDM MNDR supports modularity in schema design. The full GJXDM schema may be modularized by creating multiple subset schemas from GJXDM.

[SSM1]	GJXDM Schema MAY be split into a smaller subset schema, but only one GJXDM subset can be created for a given document schema, because the GJXDM schema must reside in one and only one namespace.

GJXDM based document schemas will be developed over time, each of which expresses a separate business function for transaction data or a business document. The GJXDM MNDR schema modularity approach is structured so that users can reuse individual document schemas as is or with modification for local usage.

Additionally, a document schema can import individual schema modules without having to import the entire GJXDM reference schema module. Each document schema will define its own dependencies. The GJXDM MNDR schema modularity model ensures that logical associations exist between document schema and a set of applicable imported schemas (GJXDM subset, extension schema, codelists etc.) . The imported schemas could be reused in defining additional document schemas supporting reuse to the maximum extent possible.

This is accomplished through the use of document schema and external schema modules.

There are two types of schema in the GJXDM MNDR – document schema and schema modules (subset, extension, CodeList(s)…) Document schemas are always in their own namespace. External Schema modules are in separate namespace(s) such as the GJXDM proxy CodeList(s), GJXDM subset(s) and any GJXDM extension schema(s). External schema modules are conformant with W3C XSD.

A namespace is an indivisible grouping of types. A “piece” of a namespace can never be used without all its pieces. GJXDM document schemas may have zero or more external schema modules that they import. The document schema for a namespace then imports those external modules.

[Definition] External schema module –

A schema module that is imported by another schema to expand the scope of schema applicable in validating a document instance.

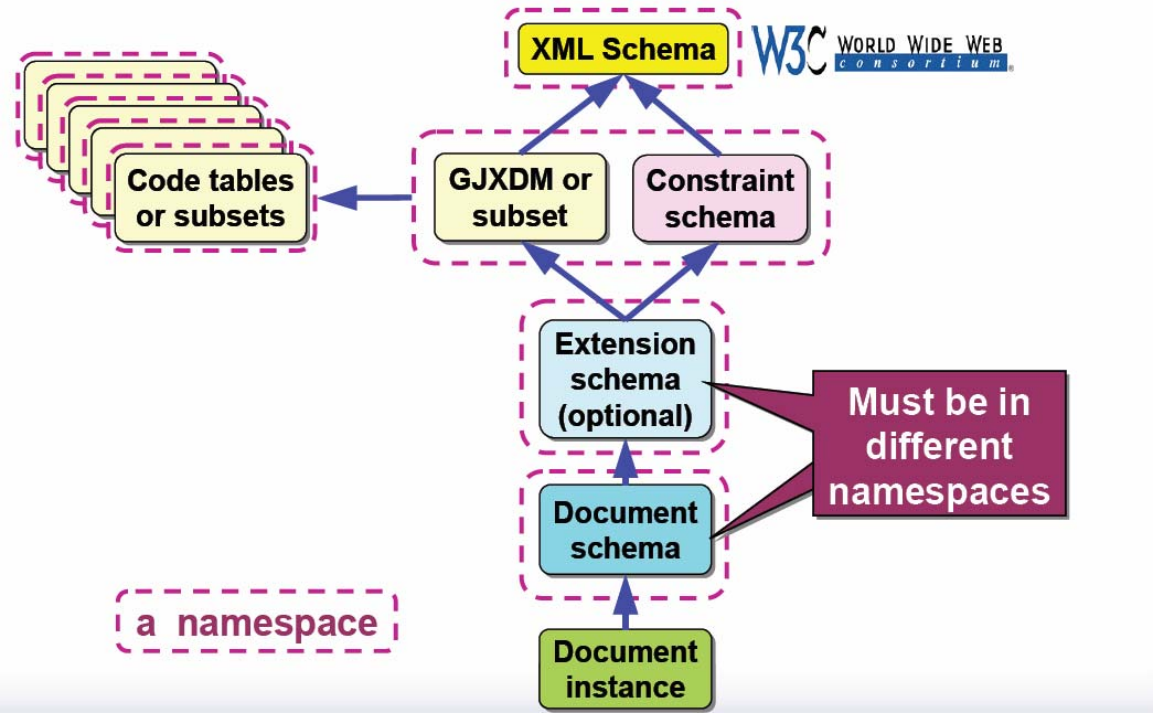


Figure 1.1 shows how the GJXDM MNDR namespace standard for segregating GJXDM schema from locally developed schema.

Any schema module may import other schemas from other namespaces.

4.1.11.2 Module Conformance

GJXDM has defined a set of naming and design rules to ensure maximum interoperability and standardization.

[SSM2]	Imported schema modules MUST be fully conformant with GJXDM information exchange schema naming and design rules.

4.1.11.3 External Schema Modules

Developers will create schema modules which, as illustrated in Figure 1.1 be located in a separate namespace from the corresponding document schema.

[SSM3]	GJXDM schema modules MUST be treated as external schema modules of the document schema.

4.1.11.4 Internal Schema Modules & xsd:include Statement

Internal schema modules do not declare a target namespace, but instead reside in the namespace of their parent schema. All internal schema modules are accessed using xsd:include. The MNDR does not support the use of Internal Schema Modules and therefore IEP's must not use the xsd:Include statements in schema.

[SSM4]	xsd:include MUST NOT be used in development of IEP's because this MNDR does not support use of Internal Schema modules.

4.1.11.5 External Schema Modules

GJXDM is dedicated to maximizing reuse. As the complex types and global element declarations will be reused in multiple schemas, a logical modularity approach is to create GJXDM schema modules based on collections of reusable types and elements.

[SSM5]	GJXDM schema module(s) MAY be created for reusable components.

Developers will create external schema modules. These external schema modules will be based on logical groupings of contents. The set of possible schema modules includes:

- GJXDM Reference Schema
- GJXDM Subset Schema(s)
- GJXDM Constraint Schema(s)
- GJXDM Extension Schema(s)
- GJXDM proxy Code List(s)
- Non-GJXDM External Schema(s) and Code List(s)
- If adopted, CCTS Core Component Parameters (for documentation/annotation of new schema elements)

4.1.12 Character encoding

[IND3]	In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83), all GJXDM XML SHOULD be expressed using UTF-8.

4.1.13 XSD:notation

XSD:notation is used to declare the format of non-XML data. A notation in XML is just like the notation declarations in DTDs. The main difference is that W3C XML Schema notations are namespace-aware and can be imported between schemas. When these declarations are used, the notations are used in xsd:enumeration facets to create simple types.

The notation datatype is used to declare links to external non-XML content (for example, image data) and then associate that content with an external application that handles it.

Notation is a built-in legacy simple type and are very seldom used in production applications, and not optimal for the Justice and Public Safety community.

[GXS10]	xsd:notations MUST NOT be used.

4.1.14 XSD:all

Used within a group, xsd:all has the same meaning as when it is used directly under xsd:complexType, except that there are no minOccurs and maxOccurs attributes and it cannot be marked as optional. The xsd:all compositor requires occurrence indicators of minOccurs=0 and maxOccurs=1. The xsd:all compositor allows for elements to occur in any order. The result is that in an instance document, elements can occur in any order, are always optional, and never occur more then once. Such restrictions are inconsistent with data-centric scenarios such as most of the work in the Justice & Public Safety community.

Another disadvantage of xsd:all is that it cannot be repeated any further. This limits the use of xsd:all to the first occurrence of its set of elements. If a content model requires an element that occurs more than once, then xsd:all cannot be used.

[GXS11]	The xsd:all element MUST NOT be used.

4.1.15 XSD:choice

The `xsd:choice` compositor allows for any element declared inside it to occur in the instance document, but only one. While `xsd:choice` is a very useful construct in situations where customization and extensibility are not a concern, GJXDM MNDR recommends against using `xsd:choice because it` cannot be extended. If extension is not a concern, then xsd:choice may be used.

[GXS13]	The xsd:choice element SHOULD NOT be used where customization and extensibility are a concern.

4.2 Subset Schema naming and Design Rules

GJXDM subset schema(s) provide for limiting the full GJXDM set of object classes and set of element(s) within each class down to a subset which is relevant for a specific Document Schema or specific domain within a local jurisdiction.

4.2.1 Rules for Conformant Subset Schemas

Any instance which validates against a schema subset must be able to validate against the full GJXDM reference schema.

Conformant Subset Schemas MUST NOT:

- Add local components
- Flatten type structures

- Modify namespaces
- Change object types
- Change element or type names
- Change type inheritance
- Make the subset inconsistent with the full reference schema

4.2.2 Subset Namespace and Filename Rules

[NMS3]	A GJXDM schema subset and constraint schema MUST declare the same xsd:targetNamespace as the GJXDM baseline schema. For example: "http://`www.it.ojp.gov/jxdm/3.0.1`" is the required targetNamespace for GJXDM version 3.0.1

[NMS7]	Each schema, for a GIEP, MUST be maintained in a separate namespace but will share a common group path. That GIEP group path MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/". For example, xmlns = "http://www.myDomainName.com/Citation/1.0/" represents a local namespace copy of a GIEP group of schemas that support the exchange or representation of a Citation reference document. The version of 1.0 is assigned to the GIEP namespace by the provider of the reference document. Note: All associated subset, document, constraint and extension schemas must be placed within this path. If any of the associated schemas, within the GIEP group change, the GIEP version MUST change.

[NMS8]	Document and Extension schema's xsd:import element(s) MUST use a relative URL to locate the imported `GJXDM Subset` schema set. The `GJXDM Subset` path MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/subset/jxdm/<GJXDM version>/". For example:

	xmlns="http://www.myDomainName.com/Citation/1.0/subset/jxdm/3.0/" represents the location of a conformant subset of the full GJXDM v3.0 schema for a version 1.0 "Citation" IEP The relative URL MUST be valid within the structure of the GIEPD ZIP file specified in NMS17

4.2.3 Subset Schema File Layout

[GXS1]	GJXDM subset schema or constraint schema MUST conform to the following physical layout as applicable: • XML Declaration • <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): ……. ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: ===== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations== --> • xsd:schema element to include Attribute definitions attributeFormDefault ="unqualified" elementFormDefault="qualified" followed by Namespace Declarations in this order: • Target namespace • Default namespace • <!-- ===== Imports ===== --> • External Codelist Namespaces • xmlns:xsd • External Codelist import schemaLocations and namespaces • <!-- ===== Global Attributes ===== --> • Global Attributes and Attribute Groups • <!-- ===== Complex Types and Simple Types ===== -->

	• <!-- ===== in alphabetized order xsd:TypeDefinitions ===== --> • Complex and Simple Types • <!-- ===== Attribute Declarations SHOULD BE in alphabetized order = --> • <!-- ===== Element Declarations SHOULD BE in alphabetized order == -->

1802

1803 4.2.4 Subset Schema Generation Tool (SSGT)

1804 A "Freely Available" software tool for generating conformant GJXDM subset schema(s) may be
1805 found at:

1806

1807 http://gjxdmtools.gtri.gatech.edu/ssgt/subset

1808

1809 Other GJXDM schema generation tools may be used or developed by other parties, both open
1810 source and proprietary.

1811

[NMS9]	All IEP's utilizing a GJXDM subset schema MUST produce the same files and filenames as the GJXDM Subset Schema Generator Tool (SSGT) For example: The SSGT generates "jxdm.xsd" as the standard filename for any GJXDM subset and produces a set of want-lists with a specified standard file structure. These SSGT artifacts must be produced for the IEP even if another software tool is used to produce them.

1812

1813 4.3 Constraint Schema Naming and Design Rules

1814

1815 4.3.1 Rules for Conformant Constraint Schemas

1816 GJXDM Constraint schemas embed localized constraints into GJXDM definitions.

1817

1818 The GJXDM namespace remains the same – just change the schema location attribute(s).

1819

1820 YOU CAN
1821 1. Change object types
1822 2. Change/drop type inheritance
1823 3. Create differently constrained types based on a single GJXDM type
1824 4. Make local component definitions
1825 5. Add localized constraints

6. Force elements to appear or not appear

YOU CANNOT

1. Change element names
2. Change tag order or hierarchy
3. Define new components to be referenced outside of this schema
4. Leave out components required by instances

```
For example, a component is required if it:


•May appear in a valid instance.
•May be referred to by a schema outside GJXDM.
•Is required by another required component.

You may omit entities which are not required, including:

•xmlns – namespace prefix declarations
•xsd:import
•xsd:complexType and xsd:simpleType elements that are "top-level"
(direct children of the xsd:schema element)
•xsd:element elements that are children of the xsd:schema element
•xsd:attribute elements that are children of the xsd:schema element
•xsd:element elements that are contained in a type definition
•xsd:attribute elements that are contained in a type definition
•xsd:attribute elements that appear in SuperType
MetadataAttributeGroup.
•xsd:enumeration elements for enumerations that are not relevant to the
applications or instances
•xsd:annotation or xsd:documentation elements

Also see Rule GXS1 for Constraint schema file layout.
```

4.3.2 Constraint Namespace and Filename Rules

[NMS14]	The Document and Extension schema's xsd:import element(s) MUST use a relative URL to locate the imported `GJXDM Constraint` schema. The relative URL MUST be valid within the structure of the GIEPD ZIP file specified in NMS17

[NMS15]	An IEP MAY have zero or one Constraint schema defined. The `GJXDM Constraint` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type>`.xsd` where <IEP name> = <IEP name> in the Document or Extension schema

	namespace <IEP schema type> = constraint For example, if the IEP document schema namespace is xmlns = http://www.myDomainName.com/Citation/1.0/document, THEN the constraint schema filename would be: Citation-constraint.xsd represents a constraint schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" and <IEP schema type> = "constraint"

1862

1863 See RULE GXS1 for Constraint Schema File Layout.

1864

1865 4.4 Extension Schema Naming and Design Rules

1866 Extension schemas define common local extensions

1867

[NMS10]	A `GJXDM IEP Extension` schema targetNamespace MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/<IEP schema type[-suffix]>". The "IEP name" and "IEP version" components of this identifier are chosen by the IEP workgroup (or chartering governance body). The IEP version component must follow the namespace versioning rules stated elsewhere in this specification. The "IEP name" component must be declared by the workgroup in the IEP overview document. The "IEP schema type" component of this identifier MUST be "extension" for the IEP's extension namespace. If the IEP has multiple extension namespaces, the IEP workgroup MUST choose appropriate "suffix" namespace name values and document them in the IEP overview document. For example, xmlns:ext = "http://www.myDomainName.com/Citation/1.0/extension" represents the namespace for a version 1.0 Citation extension schema containing user-defined Types derived from extending or restricting GJXDM schema complexTypes and simpleTypes.

1868

[NMS11]	The `GJXDM Extension` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type> [<-suffix>]`.xsd` where <IEP name> = <IEP name> defined in the Extension schema namespace <IEP schema type> = extension [suffix] = the [-suffix] optional name or number in the Extension schema namespace to support multiple extension schemas per IEP. For example, the IEP GJXDM extension schema filename for xmlns = "http://www.myDomainName.com/Citation/1.0/extension-joe" would be: Citation-extension-joe.xsd represents an extension schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" , <IEP schema type> = "extension" and <suffix> = "joe".

1869

[GTD3]	Extension schemas MUST NOT declare elements of type SuperType.

1870

[GTD4]	Extension schemas MUST NOT declare complex types that extend SuperType without adding additional elements.

1871

1872 Also see Rule GXS2 for extension schema layout.

1873

1874 ### 4.4.1 Extension Schema File Layout

[GXS2]	GJXDM extension schema MUST conform to the following physical layout as applicable: • XML Declaration • <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): …… ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: ==== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations== -->

	• xsd:schema element to include Attribute definitions attributeFormDefault =”unqualified” elementFormDefault= “qualified” followed by Namespace Declarations in this order: • Target namespace for Extension schema (http://{my namespace}/…/extension • No Default namespace, a token such as ext: should be used for the Extension schema targetNamespace (eg. xmlns:ext=”http://{my namespace}…/extension”) • Declare the GJXDM schema, subset schema or constraint schema namespace (eg. xmlns:j=”http://www.it.ojp.gov/jxdm/{jxdm version}”) • xmlns:xsd • <!-- ===== Imports ===== --> • External GJXDM reference , subset schema or constraint schema import namespace • <!-- == Extended/Restricted GJXDM Complex Types and Simple Types =--> • <!-- ===== in alphabetized order xsd:TypeDefinitions ===== --> • Complex and Simple Types • <!-- ===== Element Declarations in alphabetized order ===== -->

4.4.2 Extension Patterns

4.4.2.1 Scenarios for mapping business data/documents to GJXDM

Scenario #1

Every property and relationship I need is in GJXDM

Scenario #2

I need additional “property” elements within a currently defined GJXDM Type

Scenario #3

I need to create new relationships between GJXDM types or elements by:

Inclusion (defining a “relational” Element to extend GJXDM type)

Reference (use GJXDM ReferenceType Elements or add ReferenceType Elements)

GJXDM Relationship Element (explicit named referencing)

Scenario #4

I need a new type that doesn't inherit or extend any elements of GJXDM

Scenarios #2 - #3 represent methods for extending/restricting the GJXDM components elements and relationships to define the content of a specific document or data exchange set of elements.

4.4.2.2 Restriction of GJXDM components

An existing GJXDM type can be modified to fit the requirements of customization through XSD derivation. Restriction of ComplexTypes should NOT be done using the xsd: base=restriction construct. The recommended approach is to utilize the GJXDM subset schema generator to define a restricted set of elements for an existing GJXDM ComplexType.

As noted in Rule [ELD4], xsd: base=restriction should only be used for SimpleType components of GJXDM to add constraints and/or other facets to restrict the permissable range of values for the SimpleType.

[GXS15]	Complex Type extension and Simple Type restriction MAY be used where appropriate.

[GTD5]	Extended types MUST be derived from SuperType

4.4.2.3 Extensions to GJXDM components

The following methods may be applied for extending GJXDM components in creating GJXDM compliant Reference Documents and Data exchanges:

Method 1 : Extension using "Type Substitution"

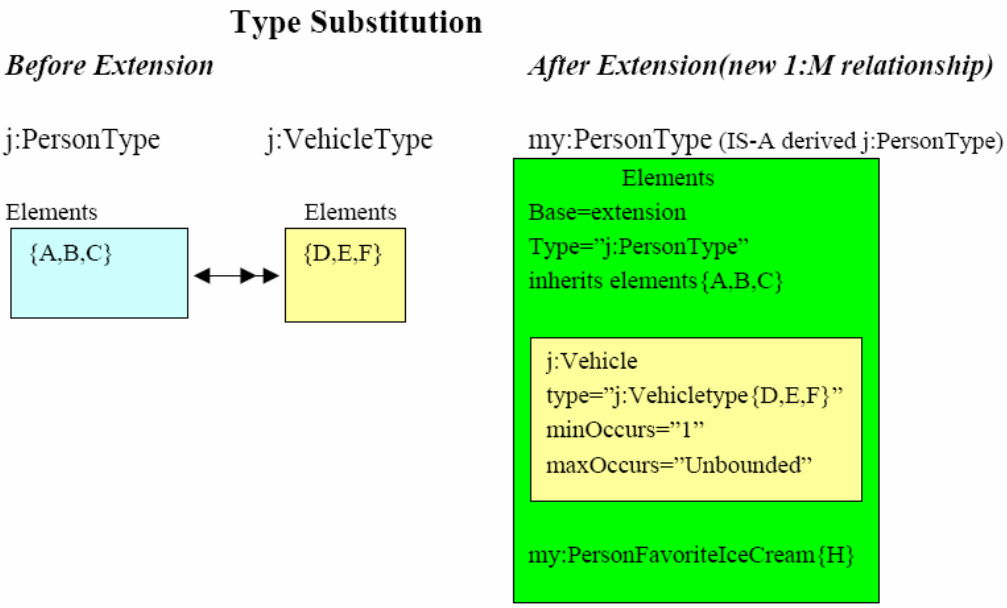
Method 2: Extension using a "Concrete Typing" construction to "Avoid Type Substitution"

Method 3: Extension of relationships using ReferenceType elements

Method 4: Extensions of relationships using GJXDM Relationship Element

Method 1 : Extension using "Type Substitution"

This method and the remaining methods will all assume we are extending PersonType(component) {A,B,C}with a 1:M or M:M "*relationship element*" to the VehicleType (component). Depending on the method we will use j:Vehicle {D,E,F}, j:VehicleReference {F}, my:PersonVehicle{G} and Relationship Qname="xxxx" {R1,R2}to illustrate the extension method. I will also add a "property element" as "my:PersonFavoriteIceCream" {H}



This **Method-1** construction requires a type substitution declaration in the XML instance to dynamically invoke my:PersonType to replace j:PersonType at runtime. Following is an XML snippet of what is required:

```
<my:SampleDocument>
        <j:Person  xsi:type="my:PersonType"  >  ***type substitution***
                <j:PersonElements {A,B,C}>
                <j:Vehicle>
                        <j:Vehicle Elements {D,E,F}>
                </j:Vehicle>
                <my:PersonFavoriteIceCream/ {H}>
        </j:Person>
</my:SampleDocument>
```

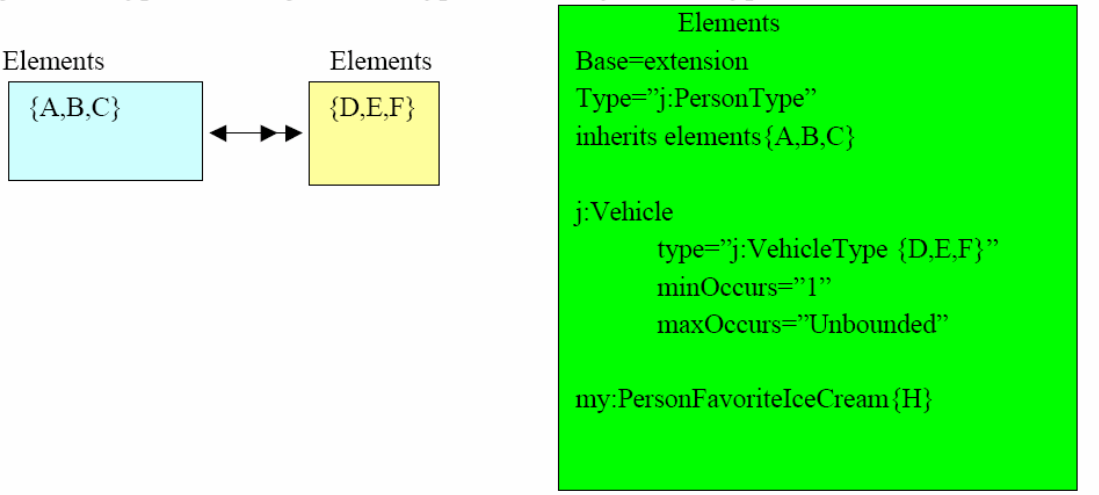
1923 •

Method 2 : Extension using "Concrete Typing" to "Avoid Type Substitution"

This method extends PersonType(component) {A,B,C}with a 1:M "*relationship element*" to the VehicleType (component) {D,E,F} using my:PersonVehicle{G}. I will also add a "property element" as "my:PersonFavoriteIceCream" {H}

Extension Using "Concrete Typing"

Before Extension ***After Extension(new 1:M relationship)***



This **Method-2** construction eliminates the "type substitution" declaration in the method-1 XML instance and avoids requirement to dynamically invoke my:PersonType. Following is an XML snippet of what it looks like:

```
<my:SampleDocument>
        <my:Person  >  *** No type substitution required ***
                <j:PersonElements {A,B,C}>
                <j:Vehicle>
                        <j:Vehicle Elements {D,E,F}>
                </j:Vehicle>
                <my:PersonFavoriteIceCream/ {H}>
        </my:Person>
</my:SampleDocument>
```

note: Cascade of **my:Person** under **my:SampleDocument** instead of dynamically substituting j:Person with my:PersonType as done in Method-1.

1924

Method 3 : Extension using ReferenceType elements

This method extends PersonType(component) {A,B,C}with a M:M "*relationship element*" to the VehicleType (component) {D,E,F} using j:VehicleReference {F}. I will also add a "property element" as "my:PersonFavoriteIceCream" {H}. Both elements will be added using a **Method-2** approach.

Extension Using "Concrete Typing" and Reference Elements

Before Extension

j:PersonType

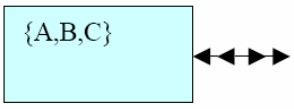
Elements

{A,B,C}

j:VehicleType

Elements

{D,E,F}



After Extension(new M:M relationship)

my:PersonType (IS-A derived j:PersonType)

Elements
Base="extension"
Type="j:PersonType"
inherits elements{A,B,C}

j: VehicleReference
type="j:ReferenceType"
minOccurs="1"
maxOccurs="Unbounded"

my:PersonFavoriteIceCream{H}

my:VehicleType (IS-A derived j:VehicleType)

Elements
Base="extension"
Type="j:VehicleType"
inherits elements{D,E,F}

j: PersonReference
type="j:ReferenceType"
minOccurs="1"
maxOccurs="Unbounded"

1925

This **Method-3** construction includes the j:VehicleReference element in my:PersonType and j:PersonReference element in my:VehicleType for relating one or more persons to one or more vehicles by reference-id's instead of the "Inclusion" method for adding "*relationship elements*" to a component. In the following XML example, this avoids duplicating Vehicle#1 elements for Person_A and Person_B. Instead, Person_A has a VehicleReference j:ref="xx" and Person_B has a VehicleReference j:ref="xx" and Vehicle#1 is only defined once in the XML instance. This would not be the case if we instead used "Inclusion" with the element j:Vehicle as shown in Method-2 above. Following is an XML snippet of what it looks like:

```
<my:SampleDocument>
        <my:Person j:id="yy" > (Person_A)
                <j:PersonElements {A,B,C}>
                <j:VehicleReference j:ref="xx"/ > (pointer to Vehicle#1)
                <my:PersonFavoriteIceCream/ {H}>
        </my:Person>
        <my:Person j:id="zz" > (Person_B)
                <j:PersonElements {A,B,C}>
                <j:VehicleReference j:ref="xx"/ > (pointer to Vehicle#1)
                <my:PersonFavoriteIceCream/ {H}>
        </my:Person>
        <my:Vehicle j:id="xx" > (Vehicle#1)
                <j:Vehicle elements{D,E,F} >
        </my:Vehicle >
        <my:Vehicle j:id="tt" > (Vehicle#2)
                <j:Vehicle elements{D,E,F} >
                <j:PersonReference j:ref="zz"> (pointer to Person B)
        </my:Vehicle >
</my:SampleDocument>

In this example Person A and Person B has-a relationship to Vehicle#1 and
Vehicle#2 has-a relationship to Person B

Note: The (VehicleReference) relationship of Person A and Person B to
Vehicle#1 may not be the same as the (PersonReference) relationship of Vehicle#2
to Person B.

For example Person A and Person B may have the relationship of "photographed"
Vehicle#1 but the Vehicle#2 to Person B relationship may be
Vehicle#2 was "VandalizedBy" Person B.
```

1926

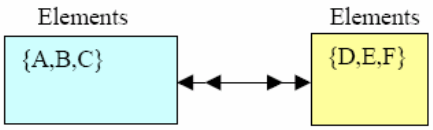
Method4 : Extension using GJXDM Relationship element for Many-to-Many relationships

This method extends PersonType(component) {A,B,C}with a M:M "*relationship element*" to the VehicleType (component) {D,E,F} using j:Relationship Elements {R1,R2}. I will also add a "property element" as "my:PersonFavoriteIceCream" {H}. All elements will be added using a **Method-2** approach.

Extension Using "Concrete Typing" and Relationship Elements

Before Extension ***After Extension(new M:M relationship)***

j:PersonType j:VehicleType my:PersonType (IS-A derived j:PersonType)



Elements
Base="extension"
Type="j:PersonType"
inherits elements{A,B,C}

<j:Relationship {R1} refers to Vehicle#1 {D,E,F}
name = "my:PersonDamagedVehicle "
subject ="*value of my:Person @id=xx*"(Person_A)
object = "*value of Vehicle @id=yy*"/> (Vehicle#1)

OR you could define it this way:

<j:Relationship {R2} refers to Person_B
in my:PersonType
name = "my:VehicleDamagedBy "
subject = "*value of Vehicle @id=yy*"/> (Vehicle#1)
object ="*value of my:Person @id="zz*" (Person_B)

my:PersonFavoriteIceCream{H}

This **Method-4** construction includes two j:Relationship elements in my:PersonType for relating one or more persons to one or more vehicles by reference instead of the "Inclusion" method for adding "*relationship elements*" to a component. This avoids duplicating Vehicle#1 elements for Person_A and Person_B. Instead, Person_A has a Relationship Qname="my:PersonDamaged" with subject="xx" and Person_B with subject="zz" and Vehicle#1. Vehicle#1 is defined once in the

1927

XML instance. This would not be the case if we instead used "Inclusion" with the element j:Vehicle as shown in Method-2 above. Following is an XML snippet of what it looks like:

```
<my:SampleDocument>
        <my:Person j:id="xx" >
                <j:PersonElements {A,B,C}>
                <j:Relationship name=" my:PersonDamagedVehicle " {R1}
                                        subject ="xx"(Person_A)
                                        object  = "yy"/> (Vehicle#1)
                <my:PersonFavoriteIceCream/ {H}>
        </my:Person>
        <my:Person j:id="zz" >
                <j:PersonElements {A,B,C}>
                <j:Relationship name=" my:VehicleDamagedBy " {R2}
                                        subject ="yy"( Vehicle#1)
                                        object  = "zz"/> (Person_B)
                <my:PersonFavoriteIceCream/ {H}>
        </my:Person>
        <j:Vehicle j:id="yy" > (Vehicle#1)
                <j:Vehicle elements{D,E,F}  >
        </j:Vehicle >
</my:SampleDocument>
```

There are many included *"relationship elements"* already defined in GJXDM which should be used before defining a new relation between objects. For example, j:PropertyDisposition, j:CaseWitness, j:CaseCharge etc. all are describing *"relationship elements"* or non-hierarchical relationships between two GJXDM Components (Classes). The ReferenceType and RelationshipType elements need only be used when many-to-many relationships in your data model need to be resolved.

4.4.3 Controlling Type extension/restriction (final)

XSD:final attribute is utilized to stop further restriction or extension on simple or complexTypes

Attribute values for preventing Type extension/restriction:

Final="#all"	no extension or restriction of Type allowed
Final="restriction"	no restriction of Type allowed

Final=”extension” no extension of Type allowed

<xsd:complexType name="Publication" final="#all" …> Publication cannot be extended nor restricted

<xsd:complexType name="Publication" final="restriction" …> Publication cannot be restricted

<xsd:complexType name="Publication" final="extension" …> Publication cannot be extended

[GXS8]	The xsd:final attribute MUST be used when schema developers want to prevent restriction or extension of a user-defined simpleTypes or complexType extensions

4.4.4 Controlling Type and Element Substitution (block)

Schema provides the xsd:attribute named “block” which can be used to block extensions/restrictions and/or substitutions to Components (ComplexTypes & SimpleTypes) and Elements. Following is the block attribute syntax which would be used in Schema at the element or component level as deemed appropriate:

block = “#all” No extension, restriction or substitution allowed

block = “restriction” No restriction allowed

block = “extension” No extension allowed

block = “substitution” No substitution allowed

block = “restriction,extension” No restriction or extention allowed (but substitution okay)

block = “restriction,substitution” No restriction or substitution (but extension okay)

block = “extension,substitution” No extension or substitution (but restriction okay)

For example, the following two snippets of schema will prevent element substitution and type substitution :

<xsd:element name="…" type="…" **block="substitution"**/>

<xsd:complexType name="PublicationType" block=”substitution">

[GXS9]	The xsd:block attribute MUST be used when schema developers want to prevent use of “type substitution” and “element substitution” in XML instance documents.

4.4.4.1 XML Instance Type Substitution

Using method 2 “concrete typing” to include new elements into GJXDM components for extensions versus “type substitution” provides the schema developer with greater control over XML Instances and a greater assurance that XML Instance authors are not inadvertently substituting new types which the developer never intended. Also, “type substitution” at run-time may provide implementation errors for vendor product(s) due to the “dynamic” substitution of validating schema within an Instance XML document.

4.4.4.2 Substitution groups, redefines

XSD:substitution is “a feature of W3C XML Schema, allowing you to define groups of elements that may be used interchangeably in instance documents. They are not declared as element groups, but through the substitutionGroup attribute of xsd:element global definitions.”

The GJXDM information exchange MNDR has made the decision to not allow the use of substitution groups due to the following issues:

- ♦ SubstitutionGroup “may work with some schema processors but relies on a liberal interpretation of the Recommendation, which may lead to interoperability issues.”
- ♦ SubstitutionGroup “introduces multiple names for the same GJXDM element leading to confusion”

[GXS7]	The xsd:SubstitutionGroups feature MUST NOT be used.

<Redefine> enables you to create a local schema which includes all of the components and elements of the full GJXDM and then <redefine> selected GJXDM component(s) or element(s) within your local schema. The redesigned (redefined) GJXDM component(s) and element(s) OVERRIDE the definitions/structures defined in the Official GJXDM dictionary. All your XML instance tags from the redefines will appear to be original GJXDM elements instead of extensions|restrictions to GJXDM elements.

There is concern that the XML instance, when using <redefine>, will be indistinguishable from a schema comprised of GJXDM elements only because the local extension|restriction elements will all have the GJXDM namespace. For that reason, the GJXDM MNDR has formulated the following rule:

[GXS12]	The xsd:redefine element MUST NOT be used.

4.5 Document Schema Naming and Design Rules

[NMS12]	A `GJXDM IEP Document` schema targetNamespace MUST be of the form: “http:// <IEP owner domain name>/<IEP name>/<IEP version>/<IEP schema type[-

	suffix]>". The "IEP name" and "IEP version" components of this identifier are chosen by the IEP workgroup (or chartering governance body). The IEP version component must follow the namespace versioning rules stated elsewhere in this specification. The "IEP name" component must be declared by the workgroup in the IEP overview document. The "IEP schema type" component of this identifier MUST be "document" for the IEP's document namespace. If the IEP has multiple document namespaces, the IEP workgroup MUST choose appropriate "suffix" namespace name values and document them in the IEP overview document. For example, xmlns = "http://www.myDomainName.com/Citation/1.0/document" represents the namespace for a version 1.0 Citation IEP document schema

2009

[NMS13]	The `GJXDM Document` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type> [<-suffix>]`.xsd` where <IEP name> = <IEP name> in the Document schema namespace <IEP schema type> = document [suffix] = the [-suffix] optional name or number in the Document schema namespace to support multiple document schemas per IEP. For example, the IEP GJXDM document schema filename for xmlns = "http://www.myDomainName.com/Citation/1.0/document" would be: Citation-document.xsd represents a document schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" , <IEP schema type> = "document" and <suffix> = option not used.

2010

2011 ### 4.5.1 Document Schema File Layout

2012 The basic GJXDM MNDR document schema template can be summarized in the following steps:

2013

1)Define the target namespace of the document

- Reference/import the GJXDM
- Define the root element and type for the document
- Extend the root type from GJXDM DocumentType (or a derivative)
- Add document content

2) Create a local element and reuse a GJXDM type

- use actual GJXDM type
- extend GJXDM type

[GXS16]	All GJXDM document schema MUST NOT declare new complexTypes or simpleTypes within the document schema. Any complexTypes and simpleTypes needed by the document schema must be imported from external schema.

[GXS3]	GJXDM Document schema MUST conform to the following physical layout as applicable: • XML Declaration • <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): …… ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: ===== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations== --> • xsd:schema element followed by Namespace Declarations in this order: • Target namespace for Document schema • No Default namespace, a token such as doc: or rap: (for a rapsheet) should be used for the Document schema targetNamespace • Declare the "optional" GJXDM extension schema namespace (eg. xmlns:ext="http://{my namespace}…/extension") • Declare the GJXDM schema, subset schema or schema constraint namespace (eg. xmlns:j="http://www.it.ojp.gov/jxdm/{jxdm version}")

	• xmlns:xsd • <!-- ===== Imports ===== --> • External “optional” GJXDM extension schema import namespace • External GJXDM schema, subset schema or constraint schema import namespace • <!-- ===== Root Element ===== --> • Root Element Declaration • Root Element Type Definition • <!-- ===== Type Definition ===== --> • Define Root Type; extend from j:DocumentType; where complexType name=”{Root Element Name}Type” (eg. for Root Element Name “CitationDocument” type=”doc:CitationDocumentType” then the complexType name = “CitationDocumentType” which is the Root Element Name suffixed with the word “Type”).

4.6 Instance Naming and Design Rules

This subsection describes the rules for constructing instance documents, including requirements for root elements and validation methods.

4.6.1 Root Element

In XSD, every global element is eligible to act as a root element in an instance document. Rule ELD1 requires the identification of a single global element in each GJXDM MNDR schema to be carried as the root element in the instance document. GJXDM MNDR exchange documents (XML instances) must have a single root element as defined in the corresponding GJXDM MNDR document schema.

[RED1]	Every GJXDM instance document must have as its root element the single global element defined in its IEPD document schema.

4.6.2 XML Instance validation

Business information exchanges require a high degree of precision to ensure that application processing and corresponding business cycle actions are reflective of the

purpose, intent, and information content agreed to by both trading partners. Schemas provide the necessary mechanism for ensuring that instance documents do in fact support these requirements.

[IND1]	All GJXDM instance documents MUST validate to a corresponding Document schema.

[IND4]	All GJXDM instance documents MUST contain the following namespace declaration in the root element: xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

It is Recommended that Validating parsers SHOULD be able to override the schemaLocation attribute in processing XML Instance Documents. (note: Relying on XML Instance schemaLocation values without verifying before processing is a Security Risk)

4.6.2.1 Schema Location

GJXDM is rapidly becoming a national and potentially international standard that will be used in perpetuity by justice agencies around the globe. It is important that these users have unfettered access to all GJXDM conformant schema.

[ATD4]	Each xsd:schemaLocation attribute declaration MUST contain a system-resolvable URI, referencing the location of the schema or schema module.

4.6.3 Character encoding

[IND2]	All GJXDM instance documents MUST always identify their character encoding with the XML declaration.

[IND3]	In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83), all GJXDM XML SHOULD be expressed using UTF-8.

4.6.4 Empty content

Usage of empty elements within XML instance documents are a source of controversy for a variety of reasons. An empty element does not simply represent data that is missing. It may express data that is not applicable for some reason, trigger the expression of an attribute, denote all possible values instead of just one, mark the end of a series of data, or appear as a result of an error in XML file generation. Conversely, missing data elements can also have meaning - data not provided by a trading partner. In information exchange

environments, different trading partners may allow, require or ban empty elements. GJXDM MNDR has determined that empty elements do not provide the level of assurance necessary for business information exchanges and as such will not be used.

[IND5]	GJXDM conformant instance documents MUST NOT contain an element devoid of content unless explicitly indicated by the xsi:nil="true" attribute

Absence of data should only be represented by using the "nillable" attribute as defined in Rule ATD5.

[IND6]	The absence of a construct or data in a GJXDM instance document MUST NOT carry meaning.

Example:

```
Valid:

<PersonName>
      <PersonGivenName>John</PersonGivenName>
      <PersonMiddleName xsi:nil="true"/>
      <PersonSurName>Doe</PersonSurName>
</PersonName>

Invalid:

<PersonName>
      <PersonGivenName>John</PersonGivenName>
      <PersonMiddleName/> ***EMPTY Content***
      <PersonSurName>Doe</PersonSurName>
</PersonName>
```

Appendix A. GJXDM MNDR Checklist

The following checklist incorporates relevant UBL XML naming and design rules as defined in UBL Naming and Design Rules version 1.0.1, 15 November 2004. UBL rules modified for GJXDM information exchange schema(s) are highlighted in GREEN. Additional rules are included to address GJXDM information exchange specific Naming & Design rules drawn from GTRI GJXDM Training Workshop materials and related documents from IJIS and OASIS LegalXML Integrated Justice Technical Committee documents. The checklist is in alphabetical sequence as follows:

Attribute Declaration Rules	(ATD)
Code List Rules	(CDL)
ComplexType Definition Rules	(CTD)
ComplexType Naming Rules	(CTN)
Documentation Rules	(DOC)
Element Declaration Rules	(ELD)
General Naming Rules	(GNR)
General Type Definition Rules	(GTD)
General XML Schema Rules	(GXS)
Instance Document Rules	(IND)
Mapping Documentation Rules	(MAP)
Modeling Constraints Rules	(MDC)
GIEP Mapping Rules	(MAP)
Namespace Rules	(NMS)
Root Element Declaration Rules	(RED)
Schema Structure Modularity Rules	(SSM)
Standards Adherence Rules	(STA)
Versioning Rules	(VER)

2113
2114

A.1 Attribute Declaration Rules

[ATD1]	User defined attributes SHOULD NOT be used.
[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 MUST be declared as part of a global attribute group. (For example: see the Global JXDM global attribute group named ”**SuperTypeMetadata”)**
[ATD3]	For each `Global JXDM user-defined element of simpleType and a xsd:restriction` element; an `xsd:base` attribute MUST be declared and set to the appropriate GJXDM datatype. `Note: the set of valid GJXDM datatypes are based on ebXML Core Component Technical Specification v1.9 and include the following 10 simpleTypes:` • Amount • BinaryObject (secondary: Graphic, Picture,Sound,Video) • Code • DateTime (secondary: Date, Time) • Identifier (authorized abbreviation: ID) • Indicator • Measure • Numeric (secondary: Value, Rate, Percent) • Quantity • Text (secondary: Name) Reference:GTRI May 2004 Developer Workshop
[ATD4]	Each xsd:schemaLocation attribute declaration MUST contain a system-resolvable URI, referencing the location of the schema or schema module.

[ATD5]	The xsd built in nillable attribute MUST be used and set nillable=”true” for any Global JXDM user-defined element which has simpleContent. Note: An example from GJXDM v3.0.2 : <xsd:element name="**WitnessLocationDescriptionText**" type="**j:TextType**" nillable="**true**">
[ATD6]	The `xsd:any attribute` MUST NOT be used.

2115
2116

A.2 Code List Rules

[CDL1]	All Global JXDM Code Lists MUST be part of a Global JXDM or externally maintained Code List; they MUST NOT be included in a document or extension schema..
[CDL2]	The Global JXDM SHOULD identify and use external standardized code lists whenever practical rather than develop its own Global JXDM-native code lists.
[CDL3]	The Global JXDM information exchange developer, through extension/restriction, MAY design and use a “contextually” defined code list where an existing GJXDM code list needs to be extended, or where no suitable external code list exists.
[CDL4]	All GJXDM maintained or information exchange developer Code Lists MUST be enumerated using the GJXDM Code List Schema Module.
[CDL5]	The name of each GJXDM information exchange Code List Schema MUST be of the form: {`Owning Organization`}_{`Code List Name`}_[version number}].xsd ;
[CDL6]	An `xsd:import` element MUST be declared for every code list required in a GJXDM information exchange schema. Each codelist MUST be in its own namespace; the namespace identifier MUST be consistent with the same rules as extension schemas.
[CDL7]	When creating a local code list, exchange developer MUST follow the UBL code list schema and annotation rules. http://docs.oasis-open.org/ubl/cd-UBL-1.0/doc/cl/wd-ublclsc-codelist-20040420.pdf

2117

A.2 Code List Rules

[CDL8]	Users of the GJXDM MAY identify any subset they wish from an identified code list for their own trading community conformance requirements.
[CDL9]	The `xsd:schemaLocation` MUST include the complete URI used to identify the relevant code list schema.

A.3 ComplexType Definition Rules

[CTD1]	For every class identified in GJXDM extension and document schema, a named `xsd:complexType` MUST be defined.
[CTD2]	Every `GJXDM user-defined xsd:complexType` definition content model MUST use the `xsd:sequence` element with appropriate global element references to reflect each property of its class.
[CTD3]	For every user-defined datatype based on the valid set of GJXDM datatypes, a named `xsd:complexType` or `xsd:simpleType` MUST be defined.

A.4 ComplexType Naming Rules

[CTN1]	A user-defined `xsd:complexType` ***name*** MUST name the object suffixed by the word `"Type"`. For example <xsd:complexType name="PersonType"> is correct And <xsd:complexType name="Person"> would be incorrect.

A.5 Documentation Rules

[DOC1]	The `xsd:documentation` element for every GJXDM user-defined Element MUST contain a structured set of annotations in the following sequence and pattern: • Version (optional): An indication of the evolution over time of the Datatype. • Definition(mandatory): The semantic meaning of an Element • Cardinality(mandatory): Indication whether the complexType Element (Property) represents a not-applicable, optional, mandatory and/or repetitive characteristic of the parent complexType • AssociatedObjectClassQualifier (optional): Associated Object Class Qualifiers describe the 'context' of the relationship with another complexType object. That is, it is the role the contained Element plays within its association with the containing complexType object. • AssociatedObjectClass (mandatory); Associated Object Class is the Object Class at the other end of this association. It represents the Aggregate Business Information Entity contained by the Association Business Information Entity. (For example: the element PersonName within the complexType PersonType has Name as the AssociatedObjectClass contained in the Aggregate Business Information Entity called PersonType.) • AlternativeBusinessTerms (optional): Any synonym terms under which the Element is commonly known and used in the business. • Examples (optional): Examples of possible values for the Element

A.6 Element Declaration Rules

[ELD1]	Each `GJXDM information exchange document schema` MUST identify one and only one global element declaration that defines the exchange document being conveyed in the Schema expression. That global element MUST include an xsd:annotation child element which MUST further contain an `xsd:documentation` child element that declares "*`This element MUST be conveyed as the root element in any instance document based on this Schema expression.`*"
[ELD2]	All element declarations MUST be global
[ELD3]	For every class defined as a GJXDM user-defined types, a global element bound to the corresponding `xsd:complexType` MUST be declared. For e.g. a schema defining a complexType named my:FavoritePersonType would need to declare a global element named "FavoritePerson" of objectType = my:FavoritePersonType to bind a global element name to the complexType.

2137

A.6 Element Declaration Rules

[ELD4]	For every user-defined simpleType, an xsd:restriction element MUST be declared
[ELD5]	Empty elements MUST NOT be declared.
[ELD6]	Global simpleType elements declared with Qualified Properties must be of the same type as their corresponding Unqualified Property.
[ELD7]	The `xsd:any` element MUST NOT be used.

2138
2139
2140
2141
2142
2143
2144

A.7 General Naming Rules

[GNR1]	User-defined information exchange XML elements, attributes and type names MUST be composed in the English language, using the primary English spellings provided in the Webster's English Dictionary.
[GNR2]	GJXDM information exchange XML element, attribute and type names MUST be ebXML CCTS ISO 11179 compliant
[GNR3]	GJXDM information exchange XML element, attribute and type names MUST NOT include spaces, other separators, or characters not allowed by W3C XML 1.0 for XML names. The only **exception** to this rule is the use of periods in GJXDM element names.
[GNR4]	GJXDM information exchange XML element, attribute, and simple and complex type names MUST NOT use acronyms, abbreviations, or other word truncations, except those in the list of exceptions published in Appendix B.
[GNR5]	The acronyms and abbreviations listed in Appendix B MUST always be used.
[GNR6]	GJXDM information exchange XML element, attribute and type names MUST be in singular form unless the concept itself is plural.
[GNR7]	The UpperCamelCase (UCC) convention MUST be used for naming elements and types.
[GNR8]	The lowerCamelCase (LCC) convention MUST be used for naming attributes.
[GNR9]	GJXDM element names, attributes and type names MUST not be modified, even when GJXDM names conflict with rules GNR1 – GNR3. For example, the use of periods in GJXDM conflicts with GNR3.

A.8 General Type Definition Rules

[GTD1]	All types MUST be named.
[GTD2]	The `xsd:anyType` MUST NOT be used.
[GTD3]	Extension schemas MUST NOT declare elements of type SuperType.
[GTD4]	Extension schemas MUST NOT declare complex types that extend SuperType without adding additional elements.

[GTD5]	Extended types MUST be derived from SuperType

2147
2148

A.9 General XML Schema Rules

[GXS1]	GJXDM subset schema or constraint schema MUST conform to the following physical layout as applicable: • XML Declaration • <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): ……. ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: …… ===== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations ===== --> • xsd:schema element to include Attribute definitions attributeFormDefault =”unqualified” elementFormDefault= “qualified” followed by Namespace Declarations in this order: • Target namespace • Default namespace • <!-- ===== Imports ===== --> • External Codelist Namespaces • xmlns:xsd • External Codelist import schemaLocations and namespaces • <!-- ===== Global Attributes ===== --> • Global Attributes and Attribute Groups • <!-- ===== Complex Types and Simple Types ===== --> • <!-- ===== in alphabetized order xsd:TypeDefinitions ===== --> • Complex and Simple Types • <!-- ===== Attribute Declarations SHOULD BE in alphabetized order ===== --> • <!-- ===== Element Declarations SHOULD BE in alphabetized order ===== -->

[GXS2]	GJXDM extension schema MUST conform to the following physical layout as applicable: • XML Declaration • <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): ……. ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: ==== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations ===== --> • xsd:schema element to include Attribute definitions attributeFormDefault =”unqualified” elementFormDefault= “qualified” followed by Namespace Declarations in this order: • Target namespace for Extension schema (http://{my namespace}/…/extension[/name]/version • No Default namespace, a token such as ext: should be used for the Extension schema targetNamespace (eg. xmlns:ext=”http://{my namespace}…/extension/…”) • Declare the GJXDM schema, subset schema or constraint schema namespace (eg. xmlns:j=”http://www.it.ojp.gov/jxdm/{jxdm version}”) • xmlns:xsd • <!-- ===== Imports ===== --> • External GJXDM reference , subset schema or constraint schema import namespace and relevant schemaLocation • <!-- ===== Extended/Restricted GJXDM Complex Types and Simple Types = --> • <!-- ===== in alphabetized order xsd:TypeDefinitions ===== --> • Complex and Simple Types • <!-- ===== Element Declarations in alphabetized order ===== -->
[GXS3]	GJXDM Document schema MUST conform to the following physical layout as applicable: • XML Declaration

	• <!-- ===== Required Documentation Comments Block ===== --> • <!-- ===== Name (common): ……. ===== --> • <!-- ===== Authoring agency/jurisdiction/generation date: …… ===== --> • <!-- ===== Description of business usage: …… ===== --> • <!-- ===== xsd:schema Element With Namespaces Declarations ===== --> • xsd:schema element followed by Namespace Declarations in this order: • Target namespace for Document schema • No Default namespace, a token such as doc: or rap: (for a rapsheet) should be used for the Document schema targetNamespace • Declare the "optional" GJXDM extension schema namespace (eg. xmlns:ext="http://{my namespace}…/extension/…") • Declare the GJXDM schema, subset schema or schema constraint namespace (eg. xmlns:j="http://www.it.ojp.gov/jxdm/{jxdm version}") • xmlns:xsd • <!-- ===== Imports ===== --> • External "optional" GJXDM extension schema import namespace and schemaLocation • External GJXDM schema, subset schema or constraint schema import namespace and schemaLocation • <!-- ===== Root Element ===== --> • Root Element Declaration • Root Element Type Definition • <!-- ===== Type Definition ===== --> • Define Root Type; extend from j:DocumentType; where complexType name="{Root Element Name}Type" (eg. for Root Element Name "CitationDocument" type="doc:CitationDocumentType" then the complexType name = "CitationDocumentType" which is the Root Element Name suffixed with the word "Type").

[GXS4]	The root element in all GJXDM information exchange Schema modules MUST contain the following declaration: "xmlns:xsd=http://www.w3.org/2001/XMLSchema."
[GXS5]	GJXDM information exchange schema developers MAY provide a run-time schema devoid of documentation in addition to the fully annotated version.
[GXS6]	GJXDM defined xsd:simpleTypes SHOULD be used as the base for any user-defined simpleTypes via extension or restriction to the GJXDM simpleType.
[GXS7]	The `xsd:SubstitutionGroups` feature MUST NOT be used.
[GXS8]	The xsd:final attribute MUST be used to control extensions.
[GXS9]	The `xsd:block` attribute SHOULD be used to restrict use of "type substitution" and "element substitution" in XML instance documents.
[GXS10]	`xsd:notations` MUST NOT be used.
[GXS11]	The `xsd:all` element MUST NOT be used.
[GXS12]	The `xsd:redefine` element MUST NOT be used.
[GXS13]	The `xsd:choice` element SHOULD NOT be used where customization and extensibility are a concern.

2149

A.9 General XML Schema Rules

[GXS14]	GJXDM designed schema MAY use xsd:appinfo. If used, xsd:appinfo MUST only be used to convey non-normative information. Note: appinfo is a recent addition to GJXDM in Version 3.0.2
[GXS15]	Complex Type extension and Simple Type restriction MAY be used where appropriate.
[GXS16]	All GJXDM document schema MUST NOT declare new ComplexTypes or SimpleTypes within the document schema. Any ComplexTypes and SimpleTypes needed by the document schema must be imported from external schema.

A.10 Instance Document Rules

[IND1]	All GJXDM instance documents MUST validate to a corresponding Document schema.
[IND2]	All GJXDM instance documents MUST always identify their character encoding with the XML declaration.
[IND3]	In conformance with ISO/IETF/ITU/UNCEFACT Memorandum of Understanding Management Group (MOUMG) Resolution 01/08 (MOU/MG01n83), all GJXDM XML SHOULD be expressed using UTF-8.
[IND4]	All GJXDM instance documents MUST contain the following namespace declaration in the root element: `xmlns:xsi`="http://www.w3.org/2001/XMLSchema-instance"
[IND5]	GJXDM conformant instance documents MUST NOT contain an element devoid of content unless explicitly indicated by the xsi:nil=”true” attribute
[IND6]	The absence of a construct or data in a GJXDM instance document MUST NOT carry meaning.

A.11 Mapping Documentation Rules

[MAP1]	All Domain Model and GJXDM Mapping Documentation MUST include the required content identified in the GJXDM Domain Model Mapping table. The table MAY contain the specified optional content.

A.12 Modeling Constraints Rules

[MDC1]	Mixed content MUST NOT be used except where contained in an `xsd:documentation` element.

A.13 Namespace and Schema Filename Rules	
[NMS1]	Every GJXDM information exchange schema MUST have a namespace declared using the `xsd:targetNamespace` attribute.
[NMS2]	Every GJXDM information exchange schema version MUST have its own unique namespace.
[NMS3]	A GJXDM schema subset and constraint schema MUST declare the same xsd:targetNamespace as the GJXDM baseline schema. For example: "http://`www.it.ojp.gov/jxdm/3.0.1`" is the required targetNamespace for GJXDM version 3.0.1
[NMS4]	GJXDM namespaces MUST only contain GJXDM conformant schema modules.

2157

A.13 Namespace and Schema Filename Rules	

[NMS5]	GJXDM published namespaces MUST never be changed. The namespace names for GJXDM reference schema releases are of the form:
	
	`http://www.it.ojp.gov/jxdm/{major version . minor release . revision}`
	
	`For example the following namespace http://…/jxdm/3.1.1 would be major-release 3 and minor-release 1.1 of the GJXDM schema.`

[NMS6]	Each `GJXDM imported CodeList` schema MUST be maintained in a separate namespace. The proxy Codelist URL MUST be of the form: "http://`www.it.ojp.gov/jxdm/{GJXDM` version}/proxy/{external original codelist name}/{original codelist version}" . For example, xmlns:j-usps = "http://…/jxdm/3.0.2/proxy/usps_states/1.0" represents the United States Postal Services States Code table version 1.0 which is imported into GJXDM schema version 3.0.2 . The proxy namespace "j-usps" is a concatenation of the Justice namespace 'j' and the original source code-list namespace "usps" forming the general rule for Proxy Namespace as "j-xsd"; where xsd is the same as the "original" codelist namespace. Proxy schemas provide an intermediary between external namespaces and the

	GJXDM, and use of Proxies guarantee SuperType metadata on elements based on entities from external namespaces.
[NMS7]	Each schema, for a GIEP, MUST be maintained in a separate namespace but will share a common group path. That GIEP group path MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/". For example, xmlns = "http://www.myDomainName.com/Citation/1.0/" represents a local namespace copy of a GIEP group of schemas that support the exchange or representation of a Citation reference document. The version of 1.0 is assigned to the GIEP namespace by the provider of the reference document. Note: All associated subset, document, constraint and extension schemas must be placed within this path. If any of the associated schemas, within the GIEP group change, the GIEP version MUST change.
[NMS8]	Document and Extension schema's xsd:import element(s) MUST use a relative URL to locate the imported `GJXDM Subset` schema set. The `GJXDM Subset` path MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/subset/jxdm/<GJXDM version>/". For example: xmlns="http://www.myDomainName.com/Citation/1.0/subset/jxdm/3.0/" represents the location of a conformant subset of the full GJXDM v3.0 schema for a version 1.0 "Citation" IEP The relative URL MUST be valid within the structure of the GIEPD ZIP file specified in NMS17
[NMS9]	All IEP's utilizing a GJXDM subset schema MUST produce the same files and filenames as the GJXDM Subset Schema Generator Tool (SSGT) For example: The SSGT generates "jxdm.xsd" as the standard filename for any GJXDM subset and produces a set of want-lists with a specified standard file structure. These

	SSGT artifacts must be produced for the IEP even if another software tool is used to produce them.
[NMS10]	A `GJXDM IEP Extension` schema targetNamespace MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/<IEP schema type[-suffix]>". The "IEP name" and "IEP version" components of this identifier are chosen by the IEP workgroup (or chartering governance body). The IEP version component must follow the namespace versioning rules stated elsewhere in this specification. The "IEP name" component must be declared by the workgroup in the IEP overview document. The "IEP schema type" component of this identifier MUST be "extension" for the IEP's extension namespace. If the IEP has multiple extension namespaces, the IEP workgroup MUST choose appropriate "suffix" namespace name values and document them in the IEP overview document. For example, xmlns:ext = "http://www.myDomainName.com/Citation/1.0/extension" represents the namespace for a version 1.0 Citation extension schema containing user-defined Types derived from extending or restricting GJXDM schema complexTypes and simpleTypes.
[NMS11]	The `GJXDM Extension` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type> [<-suffix>]`.xsd` where <IEP name> = <IEP name> defined in the Extension schema namespace <IEP schema type> = extension [suffix] = the [-suffix] optional name or number in the Extension schema namespace to support multiple extension schemas per IEP. For example, the IEP GJXDM extension schema filename for xmlns = "http://www.myDomainName.com/Citation/1.0/extension-joe" would be: Citation-extension-joe.xsd represents an extension schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" , <IEP schema type> = "extension" and <suffix> = "joe".

[NMS12]	A `GJXDM IEP Document` schema targetNamespace MUST be of the form: "http:// <IEP owner domain name>/<IEP name>/<IEP version>/<IEP schema type[-suffix]>". The "IEP name" and "IEP version" components of this identifier are chosen by the IEP workgroup (or chartering governance body). The IEP version component must follow the namespace versioning rules stated elsewhere in this specification. The "IEP name" component must be declared by the workgroup in the IEP overview document. The "IEP schema type" component of this identifier MUST be "document" for the IEP's document namespace. If the IEP has multiple document namespaces, the IEP workgroup MUST choose appropriate "suffix" namespace name values and document them in the IEP overview document. For example, xmlns = "http://www.myDomainName.com/Citation/1.0/document" represents the namespace for a version 1.0 Citation IEP document schema
[NMS13]	The `GJXDM Document` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type> [<-suffix>]`.xsd` where <IEP name> = <IEP name> in the Document schema namespace <IEP schema type> = document [suffix] = the [-suffix] optional name or number in the Document schema namespace to support multiple document schemas per IEP. For example, the IEP GJXDM document schema filename for xmlns = "http://www.myDomainName.com/Citation/1.0/document" would be: Citation-document.xsd represents a document schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" , <IEP schema type> = "document" and <suffix> = option not used.

[NMS14]	The Document and Extension schema's xsd:import element(s) MUST use a relative URL to locate the imported `GJXDM Constraint` schema. The relative URL MUST be valid within the structure of the GIEPD ZIP file specified in NMS17
[NMS15]	An IEP MAY have zero or one Constraint schema defined. The `GJXDM Constraint` schema module MUST have a schema filename of the following form: <IEP name>-<IEP schema type>`.xsd` where <IEP name> = <IEP name> in the Document or Extension schema namespace <IEP schema type> = constraint For example, if the IEP document schema namespace is xmlns = http://www.myDomainName.com/Citation/1.0/document, THEN the constraint schema filename would be: Citation-constraint.xsd represents a constraint schema for version 1.0 of a Citation IEP where <IEP name> = "Citation" and <IEP schema type> = "constraint"
[NMS16]	GIEPD artifacts MUST be combined into a ZIP file. The ZIP file MUST have the filename: <IEP name>-<IEP version>.zip Where <IEP name> & <IEP version> = <IEP name> & <IEP version> in the IEP document namespace. For example, if IEP GJXDM xmlns = http://www.myDomainName.com/Citation/1.0/document , THEN the GIEPD ZIP filename would be: Citation-1.0.zip represents an GIEPD ZIP file for version 1.0 of a Citation IEP where <IEP name> = "Citation" and <IEP version> = "1.0"

	The GIEPD ZIP file MUST have the following structure and contents. The root directory in the ZIP archive must contain the following files and directories: • The GIEPD Overview document file, in a suitable file format, and named "GIEPD Overview" with a file extension corresponding to the format. The "GIEPD Overview" MUST include the filename of the GIEPD ZIP file. • A directory named "domain model artifacts" • A directory named "mapping artifacts" • A directory named "schemas" • A directory named "sample instances" The directory named "domain model artifacts" MUST contain all artifacts related to the domain model of the IEP, as discussed in section 3 of this specification. The directory named "mapping artifacts" MUST contain all artifacts related to the mapping of the domain model to GJXDM, as discussed in section 3 of this specification.
[NMS17]	The directory named "sample instances" MUST contain one or more sample XML instances that are valid against the document, extension, constraint, and subset schemas in the IEP. Each sample XML instance: 1) MUST associate referenced IEP namespaces by using the xsi:schemaLocation attribute on the XML Instance root element; 2) MUST use the xsi:schemaLocation attribute 3) MUST use a relative URL, valid within the IEP structure documented here, to locate the schema for each namespace. The directory named "schemas" MUST contain the following: • A document schema file (see rule NMS12 and NMS13 for filename rules) • Extension schema files(s) (see rule NMS10 and NMS11 for filename rules) • A constraint schema file, if the IEP uses a constraint schema (see rule NMS14 and NMS15 for filename rule) • A directory named "subset" that contains the subset schema set; the sub-directory structure underneath the "subset" directory must match the directory structure of the GJXDM distribution version being used. *Note that the subset schema is not a single schema; rather, it is a directory structure that contains many related schemas.*

	To the extent that unzipped schemas in the IEP import each other, the schemaLocation attribute for each schema's xsd:import element(s) MUST use a relative URL to locate the imported schema. The relative URL MUST be valid within the structure of the ZIP file specified above.

2158
2159
2160

A.14	Root Element Declaration Rules
[RED1]	Every GJXDM instance document must use the global element defined as the root element in the schema as its root element.

2161
2162

2163

A.15 Schema Structure Modularity Rules	
[SSM1]	GJXDM Schema MAY be split into a smaller subset schema, but only one GJXDM subset can be created for a given document schema, because the GJXDM schema must reside in one and only one namespace.
[SSM2]	Imported schema modules MUST be fully conformant with GJXDM information exchange schema naming and design rules.
[SSM3]	GJXDM schema modules MUST be treated as external schema modules of the document schema.
[SSM4]	xsd:include MUST NOT be used in development of IEP's because this MNDR does not support use of Internal Schema modules.
[SSM5]	GJXDM schema module(s) MAY be created for reusable components.

2164
2165

A.16 Standards Adherence rules	
[STA1]	All Global JXDM information exchange schema design rules MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.
[STA2]	All Global JXDM information exchange schema and messages MUST be based on the W3C suite of technical specifications holding recommendation status.

2166
2167

2168

A.17 Versioning Rules	
[VER1]	Every GJXDM information exchange schema and schema module major version committee ***draft*** MUST have a version number of the form: <major>.0[.<revision>]
[VER2]	Every GJXDM Information exchange Schema and schema module major version OASIS ***Standard*** MUST have a version number of the form: <major>.0
[VER3]	Every minor version release of a GJXDM Information exchange schema or schema module ***draft*** MUST have a version number of the form: <major >.<non-zero>[.<revision>]

2169

A.17 Versioning Rules Continued	
[VER4]	Every minor version release of a GJXDM information exchange schema or schema module ***Standard*** MUST have a version number of the form: <major >.<non-zero>
[VER5]	For GJXDM information exchange schema minor version changes, the <IEP name> MUST NOT change.

[VER6]	For every GJXDM information exchange schema and schema module, the major version number MUST be a sequentially assigned, incremental number greater than zero.
[VER7]	For every GJXDM information exchange schema and schema module, the minor version number MUST be a sequentially assigned, incremental non-negative integer.

2170
2171

[VER8]	GJXDM information exchange schema and schema module minor version changes MUST not break semantic compatibility with prior versions; nor may they break existing document instances that are based on any earlier minor version of the last major version. For example, an instance document build on a 1.1 minor version must be able to be processed by any later minor release, for example, a 1.9 version. Minor versions maintain forward compatibility.
[VER9]	GJXDM information exchange schema and schema module major version changes MAY break semantic and/or structural compatibility with prior versions. No backward compatibility is guaranteed.

Appendix B. Approved Acronyms and Abbreviations

The following Acronyms and Abbreviations have been approved by the GJXDM MNDR Subcommittee for GJXDM use:

"Automated Fingerprint Identification System" *must* appear as "AFIS".

"American National Standards Institute" *must* appear as "ansi".

"Commercial Motor Vehicle" *must* appear as "CMV"

"Deoxyribonucleic Acid" *must* appear as "DNA".

"Federal Bureau of Investigation" *must* appear as "FBI".

"Federal Information Processing Standards" *must* appear as "FIPS".

"Foreign Government Issue" *must* appear as "FGI".

"Hazardous Materials" *must* appear as "HazMat".

"Intelligence Community" *must* appear as "IC"

"International Standards Organization" *must* appear as "ISO"

"Manufacturers Certificate of Origin" *must* appear as "MCO"

"Manufacturers Suggested Retail Price" *must* appear as "MSRP"

"Military Grid Reference System" *must* appear as "MGRS"

"Monthly Arrest and Citation Register" *must* appear as "MACR".

"National Crime Information Center" *must* appear as "NCIC".

"National Incident Based Reporting System" *must* appear as "NIBRS".

"Originating Agency Indicator" *must* appear as "ORI".

"Police Official Standards and Training" *must* appear as "POST".

"Scars, Marks & Tatoos " *must* appear as "SMT".

"Social Security Number" *must* appear as "SSN".

"Uniform Crime Report" *must* appear as "UCR".

"Uniform Resource Identifier" *must* appear as "URI".

"United States" *must* appear as "US".

"Universal Transverse Mercator" *must* appear as "UTM".

This list will henceforth be maintained by the GJXDM XSTF committee, and additions included in current and future versions of the GJXDM standard will be maintained and published separately.

2222 # Appendix C. (Informative) Technical Terminology

Ad hoc schema processing	Doing partial schema processing, but not with official schema validator software; e.g., reading through schema to get the default values out of it.
Aggregate Business Information Entity (ABIE)	A collection of related pieces of business information that together convey a distinct business meaning in a specific Business Context. Expressed in modeling terms, it is the representation of an Object Class, in a specific Business Context.
Application-level validation	Adherence to business requirements, such as valid account numbers.
Assembly	Using parts of the library of reusable GJXDM components to create a new kind of business document type.
Business Context	Defines a context in which a business has chosen to employ an information entity. The formal description of a specific business circumstance as identified by the values of a set of *Context Categories*, allowing different business circumstances to be uniquely distinguished.
class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. A class may use a set of interfaces to specify collections of operations it provides to its environment. See interface.

class diagram	Shows static structure of concepts, types, and classes. Concepts show how users think about the world; types show interfaces of software components; classes show implementation of software components. (OMG Distilled) A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships. (Rational Unified Process)
classification scheme	This is an officially supported scheme to describe a given *Context Category*
Common attribute	An attribute that has identical meaning on the multiple elements on which it appears. A common attribute might or might not correspond to an XSD global attribute.
component	One of the individual entities contributing to a whole.
Concept Map	Simple graphical models of relationships between concepts
context	Defines the circumstances in which a Business Process may be used. This is specified by a set of Context Categories known as Business Context. (See Business Context.)
context category	A group of one or more related values used to express a characteristic of a business circumstance.
Domain model	A graphical representation of the consensus of a group of business subject-matter experts as to the structure and content of an exchange document.
Domain Model Element	Any symbol in a domain model used to describe a part of the structure of an exchange document. Elements include symbols used to describe individual data elements, entire named data structures (or collections of data elements), and relationships between data elements or structures.

Document schema	A schema document corresponding to a single namespace, which is likely to pull in (by importing) schema modules.
instance	An individual entity satisfying the description of a class or type.
Leaf element	An element containing only character data (though it may also have attributes). Note that, because of the XSD mechanisms involved, a leaf element that has attributes must be declared as having a complex type, but a leaf element with no attributes may be declared with either a simple type or a complex type.
Lower-level element	An element that appears inside a business message. Lower-level elements consist of intermediate and leaf level.
Mixed Content	An XML <element> that contains both “string data” and other <element> data is defined as having mixed content.
Object Class	The logical data grouping (in a logical data model) to which a data element belongs (ISO11179).
(XML) Schema	An XML Schema consists of components such as type definitions and element declarations. These can be used to assess the validity of well-formed element and attribute information items (as defined in [XML-Infoset]), and furthermore may specify augmentations to those items and their descendants.
Schema module	A collection of XML constructs that together constitute an XSD conformant schema. Schema modules are intended to be used in combination with other XSD conformant schema.

Schema Processing	Schema validation checking plus provision of default values and provision of new infoset properties.
Schema Validation	Adherence to an XSD schema.
Semantic	Relating to meaning in language; relating to the connotations of words.
Subset Schema Generation Tool (SSGT) Want List	"The wantlist is the xml file that SSGT uses to record and maintain the state of the user's schema subset. When the user generates a schema subset package (and it is assumed that the user is finished using the SSGT), a wantlist is also generated and saved with the user's work. The wantlist is a specification for the user's schema subset and can be reloaded into SSGT to continue editing." In short, the WantList is simply an XML-formatted list of elements and types requested for inclusion in a Subset Schema, plus any elements and types included in the Extension Schema.
Top-level element	An element that encloses a whole business message. Note that GJXDM business messages might be carried by messaging transport protocols that themselves have a higher-level XML structure. Thus, a GJXDM top-level element is not necessarily the root element of the XML document that carries it.
type	Description of a set of entities that share common characteristics, relations, attributes, and semantics. A stereotype of class that is used to specify an area of instances (objects) together with the operations applicable to the objects. A type may not contain any methods.

Appendix D. (Informative) Example GIEPD(s)

A sample IEPD with many of the artifacts conforming to this MNDR can be found at the OASIS LegalXML Court Filing site "http://docs.oasis-open.org/legalxml-courtfiling/specs/ecf/v3.0/ecf-v3.0-spec-cd01.zip" The specific MNDR GIEPD artifacts are described in Appendices B thru D of the parent document named "ecf-v3.0-spec-cd01.doc" contained within this .zip file.

Appendix E. (Informative) Ongoing Work Items

MNDR needs vetting with the full IJTC and Organizations referenced in the Status Section on the cover page of this specification draft.

In the future, the MNDR needs to be evaluated against the new NIEM and “component based version of GJXDM” to determine the scope of changes to this MNDR that would be required to comply with NIEM 0.x and GJXDM post version 3.0.3.

Committee needs to evaluate need for developing and hosting additional workshops for creating new GIEPDs

Committee needs to evaluate whether it wants to commit to an effort to review existing GIEPD’s to identify the degree of common use of GJXDM components across GIEPD’s and to publish findings/recommendations.

Appendix F. (Informative) Acknowledgments

The following individuals were members of the committee during the approval of this draft:

Participants:

Jim Beard, Individual Member
Donald Bergeron, Reed Elsevier
James Cabral, MTG Management Consultants, LLC.
Scott Came, Individual Member
Tom Carlson, National Center for State Courts
Rolly Chambers, American Bar Association
James Bryce Clark, OASIS
Thomas Clarke, National Center for State Courts
Scott Edson, LA County Information Systems Advisory Body
Paul Embley, Chairperson – Global XML Structure Task Force (XSTF)
Robin Gibson, Missouri Office of State Courts Administrator
David Goodwin, Maricopa County
Aaron Gorrell, URL Integration
John Greacen, Individual Member
Jim Harris, National Center for State Courts
Marcus Leon, LA County Information Systems Advisory Body
Rex McElrath, Judicial Council of Georgia
John Messing, American Bar Association
Ellen Perry, MTG Management Consultants, LLC.
Catherine Plummer, Search Group, Inc.
John Ruegg, LA County Information Systems Advisory Body
Nancy Rutter, Maricopa County
Christopher Smith, California Administrative Office of the Courts
Marguerite Soto, LA County Internal Services Department
Eric Tingom, Individual Member
Winfield Wagner, Crossflo Systems Inc.
Sylvia Webb, Individual Member
Roger Winters, Washington State Administrator for the Courts

Appendix G. (Informative) Revision History

[This appendix is optional, but helpful. It should be removed for specifications that are at OASIS Standard level.]

Rev	Date	By Whom	What
Section 2 wd-01	2005-02-21	Tom Carlson	Initial version
Section 2 wd-02	2005-04-10	IJIS TC	Comments
Section 2 wd-03	2005-11-04	Tom Carlson	Incorporation of comments

Rev	Date	By Whom	What
Section 6 wd-.01	2005-02-10	John Ruegg	Initial version
Section 6 wd-.02	2005-03-10	John Ruegg	Incorporates decision to omit UBL component terminology for this first version
Sections 1 thru 7 as separate documents	2005-11-22	John Ruegg Tom Carlson Scott Came Ellen Perry	Prior Version of the MNDR Sections can be found at http://www.oasis-open.org/apps/org/workgroup/legalxml-intj-exmndr/documents.php
wd-ijtc-MNDR-1.0.0	2006-04-20	John Ruegg Marcus Leon Marguerite Soto	Edited and Combined single document version of MNDR draft.
wd-ijtc-MNDR-1.0.1	2007-03-20	John Ruegg	Added reference GIEPD to Appendix D and deleted of out-of-date action items from version 1.0.0 Appendix E and document body.
wd-ijtc-MNDR-1.1	2007-04-03	John Ruegg	Updated title page to reflect ballot approval of Version 1.1 as a Committee Recommendation