UML Profile for NIEM 3 (NIEM-UML-3)

*Initial Submission*

**OMG Document Number:** gov/2014-08-02

**Machine Consumable Files:**

*Note: work-in-progress machine-readable files may be found at* [*https://github.com/NIEM/NIEM-UML/tree/master/Specification*](https://github.com/NIEM/NIEM-UML/tree/master/Specification)

Normative:

Non-normative:

Copyright © 2014 Data Access Technologies (Model Driven Solutions)

Copyright © 2012 Georgia Tech Research Institute (GTRI)

Copyright © 2012 Microsoft

Copyright © 2014 Object Management Group (OMG)

Copyright © 2012 Visumpoint

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty- free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems-- without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING

LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227- 7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 140 Kendrick Street, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , IMMT™, MOF™, OMG Interface Definition Language (IDL) ™, and OMG Systems Modeling Language (OMG SysML) ™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials. Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

Table of Contents

[0 Submission-related material 0-I](#_Toc404681306)

[0.1 Submission Introduction 0-I](#_Toc404681307)

[0.2 Submission Team 0-I](#_Toc404681308)

[0.2.1 Submitters 0-I](#_Toc404681309)

[0.2.2 Government Stakeholders 0-I](#_Toc404681310)

[0.2.3 Contributors 0-I](#_Toc404681311)

[0.3 Resolution of Requirements 0-I](#_Toc404681312)

[0.3.1 Mandatory requirements 0-I](#_Toc404681313)

[0.3.2 Non-mandatory features 0-II](#_Toc404681314)

[0.4 Resolution of Discussion Issues 0-II](#_Toc404681315)

[Preface 3](#_Toc404681316)

[1 Scope 5](#_Toc404681317)

[2 Conformance 6](#_Toc404681318)

[3 Normative References 7](#_Toc404681319)

[4 Terms and Definitions 8](#_Toc404681320)

[5 Symbols 9](#_Toc404681321)

[6 Additional Information 10](#_Toc404681322)

[7 NIEM-UML Modeling Guide 11](#_Toc404681323)

[8 NIEM-UML Profile Reference 12](#_Toc404681324)

[8.1 Profile : Model\_Package\_Description\_Profile 12](#_Toc404681325)

[8.1.1 Overview 12](#_Toc404681326)

[8.1.2 <Stereotype> ApplicationInfo 12](#_Toc404681327)

[8.1.3 <Stereotype> BusinessRulesArtifact 12](#_Toc404681328)

[8.1.4 <Stereotype> ChangeInformationType 12](#_Toc404681329)

[8.1.5 <Stereotype> ChangeLogType 14](#_Toc404681330)

[8.1.6 <Stereotype> ConformanceAssertion 15](#_Toc404681331)

[8.1.7 <Stereotype> ConformanceReport 15](#_Toc404681332)

[8.1.8 <Stereotype> Documentation 15](#_Toc404681333)

[8.1.9 <Stereotype> ExtensionSchemaDocument 15](#_Toc404681334)

[8.1.10 <Stereotype> ExternalSchemaDocument 15](#_Toc404681335)

[8.1.11 <Stereotype> File 15](#_Toc404681336)

[8.1.12 <Stereotype> FileType 16](#_Toc404681337)

[8.1.13 <Stereotype> IEPSampleXMLDocument 17](#_Toc404681338)

[8.1.14 <Stereotype> ModelPackageDescriptionRelationship 18](#_Toc404681339)

[8.1.15 <Stereotype> MPDChangeLog 19](#_Toc404681340)

[8.1.16 <Stereotype> ReadMe 19](#_Toc404681341)

[8.1.17 <Stereotype> ReferenceSchemaDocument 19](#_Toc404681342)

[8.1.18 <Stereotype> RelaxNGSchema 19](#_Toc404681343)

[8.1.19 <Stereotype> RequiredFile 19](#_Toc404681344)

[8.1.20 <Stereotype> SchematronSchema 19](#_Toc404681345)

[8.1.21 <Stereotype> SubsetSchemaDocument 20](#_Toc404681346)

[8.1.22 <Stereotype> Wantlist 20](#_Toc404681347)

[8.1.23 <Stereotype> XMLCatalog 20](#_Toc404681348)

[8.1.24 <Stereotype> XMLSchemaDocument 20](#_Toc404681349)

[8.1.25 <Artifact> ArtifactOrArtifactSet 20](#_Toc404681350)

[8.1.26 <Artifact> ConformanceTargetType 20](#_Toc404681351)

[8.1.27 <Artifact> ConstraintSchemaDocumentSet 21](#_Toc404681352)

[8.1.28 <Artifact> ContactInformationType 21](#_Toc404681353)

[8.1.29 <Artifact> DescribedType 22](#_Toc404681354)

[8.1.30 <Artifact> EntityRepresentation 22](#_Toc404681355)

[8.1.31 <Artifact> EXIXMLSchemaType 23](#_Toc404681356)

[8.1.32 <Artifact> FileSet 23](#_Toc404681357)

[8.1.33 <Artifact> FileSetType 23](#_Toc404681358)

[8.1.34 <Artifact> IEPConformanceTargetType 24](#_Toc404681359)

[8.1.35 <Artifact> ModelPackageDescription 26](#_Toc404681360)

[8.1.36 <Artifact> OrganizationType 63](#_Toc404681361)

[8.1.37 <Artifact> PersonType 63](#_Toc404681362)

[8.1.38 <Artifact> QualifiedNamesType 64](#_Toc404681363)

[8.1.39 <Artifact> RelaxNGValidationType 64](#_Toc404681364)

[8.1.40 <Artifact> SchemaDocumentSet 64](#_Toc404681365)

[8.1.41 <Artifact> SchemaDocumentSetType 64](#_Toc404681366)

[8.1.42 <Artifact> SchematronValidationType 65](#_Toc404681367)

[8.1.43 <Artifact> TextRuleType 65](#_Toc404681368)

[8.1.44 <Artifact> ValidityConstraintType 66](#_Toc404681369)

[8.1.45 <Artifact> ValidityConstraintWithContextType 66](#_Toc404681370)

[8.1.46 <Artifact> ValidityContextType 66](#_Toc404681371)

[8.1.47 <Artifact> XMLSchemaType 66](#_Toc404681372)

[8.1.48 <Artifact> XPathType 66](#_Toc404681373)

[8.1.49 <Enumeration> ChangeCodeSimpleType 67](#_Toc404681374)

[8.1.50 <Enumeration> ModelPackageDescriptionClassCode 67](#_Toc404681375)

[8.1.51 <Enumeration> RelationshipCode 69](#_Toc404681376)

[8.2 Profile : NIEM\_Common\_Profile 70](#_Toc404681377)

[8.2.1 Overview 70](#_Toc404681378)

[8.2.2 <Stereotype> AdapterType 70](#_Toc404681379)

[8.2.3 <Stereotype> AssociationType 71](#_Toc404681380)

[8.2.4 <Stereotype> AugmentationType 72](#_Toc404681381)

[8.2.5 <Stereotype> Choice 75](#_Toc404681382)

[8.2.6 <Stereotype> Deprecated 76](#_Toc404681383)

[8.2.7 <Stereotype> Documentation 77](#_Toc404681384)

[8.2.8 <Stereotype> List 77](#_Toc404681385)

[8.2.9 <Stereotype> LocalTerm 79](#_Toc404681386)

[8.2.10 <Stereotype> LocalVocabulary 81](#_Toc404681387)

[8.2.11 <Stereotype> MetadataApplication 81](#_Toc404681388)

[8.2.12 <Stereotype> MetadataType 82](#_Toc404681389)

[8.2.13 <Stereotype> Namespace 83](#_Toc404681390)

[8.2.14 <Stereotype> NIEMType 103](#_Toc404681391)

[8.2.15 <Stereotype> ObjectType 103](#_Toc404681392)

[8.2.16 <Stereotype> PropertyHolder 111](#_Toc404681393)

[8.2.17 <Stereotype> References 112](#_Toc404681394)

[8.2.18 <Stereotype> Representation 113](#_Toc404681395)

[8.2.19 <Stereotype> Restriction 114](#_Toc404681396)

[8.2.20 <Stereotype> Union 116](#_Toc404681397)

[8.2.21 <Stereotype> UnionOf 119](#_Toc404681398)

[8.2.22 <Stereotype> ValueRestriction 120](#_Toc404681399)

[8.3 Profile : NIEM\_PIM\_Profile 128](#_Toc404681400)

[8.3.1 Overview 128](#_Toc404681401)

[8.3.2 <Stereotype> Augments 128](#_Toc404681402)

[8.3.3 <Stereotype> InformationModel 128](#_Toc404681403)

[8.3.4 <Stereotype> ReferenceName 173](#_Toc404681404)

[8.3.5 <Stereotype> RoleOf 173](#_Toc404681405)

[8.3.6 <Stereotype> RolePlayedBy 174](#_Toc404681406)

[8.3.7 <Stereotype> Subsets 176](#_Toc404681407)

[8.3.8 <Enumeration> DefaultPurposeCode 177](#_Toc404681408)

[8.4 Profile : NIEM\_PSM\_Profile 184](#_Toc404681409)

[8.4.1 Overview 184](#_Toc404681410)

[8.4.2 <Stereotype> XSDAnyProperty 185](#_Toc404681411)

[8.4.3 <Stereotype> XSDDeclaration 186](#_Toc404681412)

[8.4.4 <Stereotype> XSDProperty 186](#_Toc404681413)

[8.4.5 <Stereotype> XSDRepresentationRestriction 214](#_Toc404681414)

[8.4.6 <Stereotype> XSDSimpleContent 215](#_Toc404681415)

[8.4.7 <Enumeration> XSDProcessContentsCode 216](#_Toc404681416)

[8.4.8 <Enumeration> XSDPropertyKindCode 216](#_Toc404681417)

[8.4.9 <Enumeration> XSDWhiteSpaceCode 216](#_Toc404681418)

[8.5 Profile : NIEM\_UML\_Profile 217](#_Toc404681419)

[8.5.1 Overview 217](#_Toc404681420)

[9 NIEM-UML Transformation Reference 218](#_Toc404681421)

# Submission-related material

## Submission Introduction

The NIEM-UML-3 submission team is pleased to present an initial submission to the “UML Profile for NIEM 3” Request for Proposal gov/14-03-01.

The IPR mode for this submission is Non-Assert.

Clause 0 of this document contains information specific to the OMG submission process and is not part of the proposed specification. The proposed specification starts with Clause 1. All clauses are normative unless otherwise specified.

## Submission Team

### Submitters

* Model Driven Solutions, cory-c@modeldriven.com
* Office of the Program Manager for Information Sharing Environment (PM-ISE) [www.ise.gov](http://www.ise.gov)

### Government Stakeholders

* NIEM Program Management Office (PMO), and the NIEM Technical Architecture Committee (NTAC)
* Office of the Secretary of Defense

### Contributors

* Adaptive
* Escape Velocity
* Everware-CBDI
* Georgia Tech Research Institute (GTRI)
* Hidden Symmetry Ltd
* SEARCH
* TethersEnd Consulting

## Resolution of Requirements

### Mandatory requirements

|  |  |
| --- | --- |
| 6.5.1.1 Proposals shall specify a PIM profile constrained to limit the set of representable constructs to those allowed by the NIEM specifications. | The PIM profile is a modified version of the NIEM 1 PIM profile, altered to account for NIEM 3 changes. |
| 6.5.1.2 Proposals shall specify a MPD profile to specify the MPD metadata, content and technology choices required to parameterize the QVT transformations. | The MPD profile is a modified version of the NIEM 1 MPD profile, altered to account for NIEM 3 changes. |
| 6.5.1.3 Proposals shall specify a PSM profile providing stereotypes that enable NIEM technical modelers – or NIEM schema modelers – to model the technical aspect, i.e. XML schema specifics, of an MPD. | The PSM profile is a modified version of the NIEM 1 PSM profile, altered to account for NIEM 3 changes. |
| 6.5.1.4 Proposals shall specify QVT transformations from UML models using the PIM, MPD and PSM profiles to the set of artifacts required in a conformant MPD. | The QVT transformations are a modified versions of the NIEM 1 QVT transformations, altered to account for NIEM 3 changes. |
| 6.5.1.5 Proposals shall utilize the PIM, MPD and PSM profiles to model at least one existing NIEM IEPD and demonstrate the resulting transformation to a NIEM-conformant IEPD. | To be discussed in revised submission. NIEM 3 is not yet finalized. |
| 6.5.2.1 Proposals shall include UML models of the NIEM Version 3 reference schema namespaces [NIEM-RN] (or vocabularies) using the PIM; each UML model shall contain a package representing a NIEM namespace. | To be discussed in revised submission. NIEM 3 is not yet finalized. |
| 6.5.3.1 Proposals shall, wherever practical, reuse elements from the UML Profile for NIEM Version 1.0 PIM profile and provide for forward compatibility of models developed based on the profile. | The version 1.0 profile has been reused to the maximum extent possible. |
| 6.5.3.2 Proposals shall discuss the relationship between the UML Profile for NIEM, Version 1.0 (aligned to NIEM 2.1) and the UML Profile for NIEM Version 3 and the conversion process users would be expected to follow. | To be discussed in revised submission. |
| 6.5.3.3 Requirements outlined in sections 6.5.1 through 6.5.3 shall conform to normative NIEM specifications referenced in section 6.4. | The specification is extensively cross-referenced to the normative NIEM 3 specifications. |

### Non-mandatory features

|  |  |
| --- | --- |
| 6.6.1 Proposals may include a provisioning profile and mapping to other exchange formats such as RDF Schema (http://www.w3.org/standards/techs/rdf#w3c\_all). | May be addressed in revised submission. |
| 6.6.2 Proposals may specify a “reverse engineering” OMG-QVT mapping from a NIEM-conformant MPD to UML models that conform to the NIEM-UML profiles. | May be addressed in revised submission. |
| 6.6.3 Proposals may specify a transformation from the UML Profile for NIEM, Version 1.0 (aligned to NIEM 2.1) and the UML Profile for NIEM Version 3 conformant models. | May be addressed in revised submission. |
|  |  |

## Resolution of Discussion Issues

|  |
| --- |
| 6.7.1 Proposals shall discuss the relationship of NIEM-UML with other on-going and related NIEM standards, the existing NIEM specifications and the NIEM process. |

To be discussed in revised submission.

|  |
| --- |
| 6.7.2 Proposals shall discuss the relationship of NIEM-UML with other on-going exchange standards efforts, including but not limited to the Structured Threat Information Expression (STIX) and EDXL. |

To be discussed in revised submission.

|  |
| --- |
| 6.7.3 Proposals shall discuss their relationship with other relevant standards including but not limited to the Unified Profile for DoDAF/MODAF (UPDM) 2.1. |

To be discussed in revised submission.

# Preface

**OMG**

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia. OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets. More information on the OMG is available at [*http://www.omg.org/*](http://www.omg.org/)*.*

**OMG Specifications**

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. All OMG Specifications are available from this URL: [*http://www.omg.org/spec*](http://www.omg.org/spec)

Specifications are organized by the following categories:

**Business Modeling Specifications**

**Middleware Specifications**

* CORBA/IIOP
* Data Distribution Services
* Specialized CORBA IDL/Language Mapping Specifications

**Modeling and Metadata Specifications**

* UML, MOF, CWM, XMI
* UML Profile  Specifications

**Platform Independent Model (PIM) - Platform Specific Model (PSM) - Interface Specifications**

* CORBAServices
* CORBAFacilities
* OMG Domain Specifications
* CORBA Embedded Intelligence Specifications
* CORBA Security Specifications

All of OMG’s formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at: OMG Headquarters 140 Kendrick Street Building A, Suite 300 Needham, MA 02494 USA Tel: +1- 781-444-0404 Fax: +1-781-444-0320 Email: *pubs@omg.org* Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

**Typographical Conventions**

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

**Helvetica/Arial - 10 pt. Bold:** OMG Interface Definition Language (OMG IDL) and syntax elements.

**Courier - 10 pt. Bold:** Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

**Note –** Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

**Issues**

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/ report\_issue.htm.*

# Scope

# Conformance

# Normative References

# Terms and Definitions

# Symbols

There are no symbols defined in this specification.

# Additional Information

# NIEM-UML Modeling Guide

# NIEM-UML Profile Reference

## Profile : Model\_Package\_Description\_Profile

### Overview

The Model Package Description Profile comprises stereotypes that are used to model NIEM MPDs. The diagram shows all the stereotypes defined in this profile.

### <Stereotype> [ApplicationInfo](#_e5ec7e3e9d1d1d1d3b6e0dfe82c98d0f)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

An MPD artifact that is used by a software tool (e.g., import, export, input, output, etc.).

### <Stereotype> [BusinessRulesArtifact](#_a87115cbcc21d0c435988ee891218e74)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [ChangeInformationType](#_d7f0afacf8acfe2a0ea2d846197421e4)

##### Extends

##### PackageDescription

The ChangeInformationType stereotype applies to a Package that represents one or more detailed change entries. The <<ChangeInformationType>> is a nested UML::Package of <<ChangeLogType>>. It contains descriptive information about one or more detailed change entries. The attributes defined for <<ChangeInformationType>> reflect the required changelog descriptive information for change entries. The change entries themselves, and their relationship with <<ChangeInformationType>> is an implementation detail not constrained by this specification

##### Properties

###### ChangeCodeSimpleType

An enumeration of change codes based on the type of change that is contained in the change log.

|  |  |
| --- | --- |
| **Type** | ChangeCodeSimpleType |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | none |

###### ChangeFullDescriptionText

Descriptive text outlining the details of a specific change contained in a change log.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### ChangeNCCTIssueNumber

Text outlining the NIEMthe NIEM Change Configuration Tool number associated to the specific change contained in the change log.

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | none |

###### ChangeReasonText

Descriptive text providing context to the reason a change noted in the change log was made.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### ChangeSummaryText

Text outlining a summary of a specific change contained in the change log.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

### <Stereotype> [ChangeLogType](#_e3caf14bcc1716f393a7ba2f5642f78d)

##### Extends

##### PackageDescription

The ChangeLogType stereotype applies to a Package that represents the required MPD changelog artifact. The changelog artifact contains descriptive information about the changelog as a whole. The attributes defined for <<ChangeLogType>> reflect the required changelog descriptive information.

##### Properties

###### BaselineModelURL

URL of baseline model the change log applies to.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### ChangeLogApplicationInstructionsText

Descriptive text representing change log applications instructions.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### ChangeLogSubmitterName

A name of the person, or organization submitting the change log.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### ChangeLogSummaryText

Descriptive text providing a summary of the change log.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

### <Stereotype> [ConformanceAssertion](#_3eccbf35ee76685007d7e674d56bc322)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [ConformanceReport](#_73c4286659ba2418e50e7e7b2925c53a)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [Documentation](#_7f60db331c02abe2c1ece306f640a550)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [ExtensionSchemaDocument](#_a359a1e478997f8e27994502878bf6db)

##### Generalization

##### [XMLSchemaDocument](#_b02fa05d43cb5f5cace47c9181b17443)Description

### <Stereotype> [ExternalSchemaDocument](#_2b4a400e6fc31f5a6947a6896141e605)

##### Generalization

##### [XMLSchemaDocument](#_b02fa05d43cb5f5cace47c9181b17443)Description

### <Stereotype> [File](#_101c1dbd4d8fec57fc8c836e27ba2008)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [FileType](#_1adfac21ca9bec3d93546a3277176cea)

##### Extends

##### UsageDescription

##### Properties

###### descriptionText

A description of the file. Implemented as the value of the descriptionText attribute of the File element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### externalURI

An external URI for the file; indicates a same-as relationship to a copy of the file. Implemented as the value of the externalURI attribute of the File element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### mimeMediaTypeText

A classification for an MPD file artifact from the IANA MIME media classes: http://www.iana.org/assignments/media-types.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### pathURI

The relative path name to the file within the MPD directory structure. Implemented as the value of the pathURI attribute of the FileType type in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### RequiredFile

An MPD file artifact that another artifact depends on and should not be separated from.

|  |  |
| --- | --- |
| **Type** |  |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

### <Stereotype> [IEPSampleXMLDocument](#_0b065f020b6b1c58eef43ec5340783b8)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

 This NDR has specific rules about how NIEM content should be used in XML documents. As well as containing rules for XML Schema documents, the NDR contains rules for NIEM-conformant XML content at a finer granularity than the XML document.

[Definition: *conformant instance XML document*]

A **conformant instance XML document** is an XML document that is instance document valid to a conformant schema document set. It is a conformance target of this specification. A conformant instance XML document MUST conform to all rules of this specification that apply to this conformance target.

Characteristics of a conformant instance XML document include:

1. The document element is locally schema-valid.
2. Each element information item within the XML document that has property [namespace name] matching the target namespace of a reference schema document or extension schema document is a conformant element information item.

Schema-validity may be assessed against a single set of schemas or against multiple sets of schemas.

Assessment against schemas may be directed by a Model Package Description (MPD), by other instructions, or by other tools.

Note that this specification does not require the document element of a conformant instance XML document to be a conformant element information item. Other specifications, such as the MPD specification, may add additional constraints to these in order to specify MPD-specific or exchange-specific conformance contraints.

[Definition: *conformant element information item*]

A *conformant element information item* is an element information item that satisfies all of the following criteria:

* 1. Its [namespace name] and [local name] matches an element declared by a reference schema document or extension schema document.
  2. It occurs within a conformant instance XML document.
  3. It is locally schema-valid.
  4. It satisfies all rules specified in this document applicable to an element in a conformant instance XML document.

Because each NIEM-conformant element information item must be locally schema-valid, each element must validate against the schema definition of the element, even if the element information item is allowed within the document because of a wildcard that the {process contents} with a value of skip. As described by XML Schema Structures, Section 3.10.1, [*The Wildcard Schema Component*](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#process_contents), the content of an element introduced by a wildcard with {process contents} set to skip does not have any schema validity constraint; it is only required to be well-formed XML. Within a NIEM-conformant XML document, each element that is from a NIEM namespace conforms to its schema specification.

### <Stereotype> [ModelPackageDescriptionRelationship](#_6631c2d4ec13321e24b2860a929aeed1)

##### Extends

##### DependencyDescription

The ModelPackageDescriptionRelationship stereotype applies to a Dependency that represents a relationship between MPDs or between an MPD and another resource (such as a NIEM specification; as in the case of conforms-to). There are many ways one MPD may relate to another. This makes it extremely difficult to specify a fixed set of values that could objectively define an exact relationship between a pair of MPDs. Therefore, the optional descriptionText attribute is provided to further explain the nature of any of the eight relationshipCode values available (version\_of, specializes, generalizes, deprecates, supersedes, adapts, conforms\_to, updates). In some cases, the value of relationshipCode may be generic enough to require a more detailed explanation in descriptionText (for example, if the value is "adapts").

##### Properties

###### descriptionText

A more detailed or specific textual explanation of the relationship between the MPDs or between an MPD and a resource (such as a specification).

The catalog provides a Relationship element with three attributes (resourceURI, relationshipCode, and descriptionText) to identify the pedigree of an MPD. There are many ways that one MPD may relate to another. This makes it extremely difficult to specify a fixed set of values that could objectively define an exact relationship between a pair of MPDs. Therefore, the optional descriptionText attribute is provided to further explain the nature of any of the eight relationshipCode values available (version\_of, specializes, generalizes, deprecates, supersedes, adapts, conforms\_to, updates). In some cases, the value of relationshipCode may be generic enough to require a more detailed explanation in descriptionText (for example, if the value is "adapts").

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### relationshipCode

A classification or reason for the connectedness between the MPDs or between an MPD and a resource.

|  |  |
| --- | --- |
| **Type** | RelationshipCode |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

### <Stereotype> [MPDChangeLog](#_9e604fe0a2af271aa458b2d5e936b76d)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [ReadMe](#_38627e1c300a22a5b6feea21e20a588f)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [ReferenceSchemaDocument](#_aa336ea37aec2bf19d8fca590d898258)

##### Generalization

##### [XMLSchemaDocument](#_b02fa05d43cb5f5cace47c9181b17443)Description

### <Stereotype> [RelaxNGSchema](#_3e6f2893d6456f759d2d79c2851d9e36)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [RequiredFile](#_eeb8344a90187b9895263f3d68de6591)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [SchematronSchema](#_8f9fefb70f4508be82a701ff1c5d584f)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [SubsetSchemaDocument](#_217d779dd71fb99067a76ecd975a1d3b)

##### Generalization

##### [XMLSchemaDocument](#_b02fa05d43cb5f5cace47c9181b17443)Description

### <Stereotype> [Wantlist](#_3d2e6f85c03cd167477627b5151affef)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [XMLCatalog](#_781c1f78e93c2aecd3d28c5abeddd019)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Stereotype> [XMLSchemaDocument](#_b02fa05d43cb5f5cace47c9181b17443)

##### Generalization

##### [FileType](#_1adfac21ca9bec3d93546a3277176cea)Description

### <Artifact> [ArtifactOrArtifactSet](#_db7585560e9f04d9f4eaefd6a5b723fc)

A data concept for a file or file set in an MPD.

### <Artifact> [ConformanceTargetType](#_a50ba294484866fba55ab0c5578e47e8)

A data type for identifying and describing a conformance target.

##### Properties

###### conformanceTargetURI

A URI for a conformance target.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** | ordered |
| **Composition** | composite |

### <Artifact> [ConstraintSchemaDocumentSet](#_7f9b0d0490e59a1f3dd8335ee3079827)

An MPD artifact set of constraint schema documents and other supporting artifacts.

### <Artifact> [ContactInformationType](#_7dae1480fbe360bd468c61d841d9ded7)

A data type for how to contact a person or an organization.

##### Properties

###### ContactEmailID

An electronic mailing address by which a person or organization may be contacted.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ContactEntity

An entity that may be contacted by using the given contact information.

|  |  |
| --- | --- |
| **Type** | EntityRepresentation |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ContactMailingAddress

A postal address by which a person or organization may be contacted.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ContactResponder

A third party person who answers a call and connects or directs the caller to the intended person.

|  |  |
| --- | --- |
| **Type** | PersonType |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ContactTelephoneNumber

A telephone number for a telecommunication device by which a person or organization may be contacted.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ContactWebsiteURI

A website address by which a person or organization may be contacted.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [DescribedType](#_decd4bb0486b2d193f0fe691f9f3e96f)

Common supertype for NIEM MPD Catalog types which have descriptionText

##### Properties

###### descriptionText

A description of the file. Implemented as the value of the descriptionText attribute of the File element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

### <Artifact> [EntityRepresentation](#_b5992f81868b11b29878271a69332828)

A data concept for a person, organization, or thing capable of bearing legal rights and responsibilities.

##### Properties

###### name

A combination of names and/or titles by which an entity is known.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [EXIXMLSchemaType](#_98143a7f2c08dc26cca7b7eaf80061ac)

An XML Schema to be used for EXI serialization of an IEP Class.

### <Artifact> [FileSet](#_30030dcd7015232d13e92ab474c42ed5)

A generic MPD artifact set; used to group artifacts that are not accounted for by other set classifiers.

### <Artifact> [FileSetType](#_f9ab110d19d406069517dfa76824683e)

A data type for a set of MPD file artifacts.

##### Properties

###### ArtifactOrArtifactSet

A data concept for a file or file set in an MPD.

|  |  |
| --- | --- |
| **Type** | ArtifactOrArtifactSet |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### externalURI

An external URI for the file; indicates a same-as relationship to a copy of the file. Implemented as the value of the externalURI attribute of the File element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### pathURI

The relative path name to the file within the MPD directory structure. Implemented as the value of the pathURI attribute of the FileType type in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

### <Artifact> [IEPConformanceTargetType](#_d9384fc253b8677c36bd42a24a6e670a)

A data type for a class or category of IEP, which has a set of validity constraints and a unique identifier.

##### Properties

###### ArtifactOrArtifactSet

A data concept for a file or file set in an MPD.

|  |  |
| --- | --- |
| **Type** | ArtifactOrArtifactSet |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### descriptionText

A description of the MPD. A statement that provides an explanation or additional detail.

Implemented as the value of the DescriptionText element of the MPDType type within the MPD Catalog document instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### ValidityConstraintWithContext

A data concept for a rule or instructions for validating an IEP candidate (XML document) using some context within that XML document.

|  |  |
| --- | --- |
| **Type** | ValidityConstraintWithContextType |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

##### Constraints

###### MPD3 [Rule 5-44] (IEPD). IEPD Has an IEP Sample for Each c:IEPConformanceTarget

This section discusses sample IEPs in the context of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). However, this is not meant to imply that sample IEPs are not useful in other MPDs.

A sample IEP [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) is a representation of an actual or example exchange data instance. Sample instances can be extremely valuable artifacts in an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). Sample IEPs:

* Help an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) implementer to understand the original intent of the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author.
* Can be used by an implementer as a data point for validation of IEP conformance targets.
* can indicate or imply [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) quality.

For these reasons, the following rule applies to an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation):

Rule 5-44. IEPD Has an IEP Sample for Each c:IEPConformanceTarget

[Rule 5-44] (IEPD) (Constraint)

Within the MPD catalog document of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation), a c:IEPConformanceTarget element MUST contain a c:IEPSampleXMLDocument child element.

The rule above requires that each declared [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) be covered (exemplified or correctly demonstrated) by at least one IEP sample instance XML document. This does not necessarily mandate a different IEP sample for each [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target). It may be possible, and is therefore acceptable, for a given IEP sample to serve as an example of one or more [IEP conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target).

The purpose of this rule is not to provide a test for all possible IEP permutations given the schema definitions and validity constraint declarations; rather, it is to encourage [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors to test their own designs, and to provide implementers with examples for additional understanding, guidance, and testing. To the extent possible, [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors should strive to include sample IEPs that (1) capture real world business cases of data exchanges, and (2) exercise as many [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) and validity constraints as possible. Where it makes sense, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author should strive to provide enough sample IEPs to exercise all the XML document elements (or payload root elements). If a single IEP cannot provide enough example coverage, an author may include multiple IEPs (but is not required to do so).

Each sample IEP usually illustrates a single view of the data based on a chosen set of conditions. Other views based on different conditions likely exist. An implementer will still need to review the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) documentation to ensure understanding of all potential conditions. Therefore, as appropriate, the author should not rely exclusively on sample IEPs to convey implementation understanding, since they will probably not account for all possible permutations.

**context** IEPConformanceTargetType **inv:**

self.oclAsType(InstanceSpecification).clientDependency->exists(d|d.stereotypedBy('IEPSampleXMLDocument'))

###### MPD3 [Rule 5-45] (IEP). Validating an IEP Sample XML Document

The rule above requires that each declared [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) be covered (exemplified or correctly demonstrated) by at least one IEP sample instance XML document. This does not necessarily mandate a different IEP sample for each [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target). It may be possible, and is therefore acceptable, for a given IEP sample to serve as an example of one or more [IEP conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target).

The purpose of this rule is not to provide a test for all possible IEP permutations given the schema definitions and validity constraint declarations; rather, it is to encourage [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors to test their own designs, and to provide implementers with examples for additional understanding, guidance, and testing. To the extent possible, [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors should strive to include sample IEPs that (1) capture real world business cases of data exchanges, and (2) exercise as many [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) and validity constraints as possible. Where it makes sense, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author should strive to provide enough sample IEPs to exercise all the XML document elements (or payload root elements). If a single IEP cannot provide enough example coverage, an author may include multiple IEPs (but is not required to do so).

Each sample IEP usually illustrates a single view of the data based on a chosen set of conditions. Other views based on different conditions likely exist. An implementer will still need to review the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) documentation to ensure understanding of all potential conditions. Therefore, as appropriate, the author should not rely exclusively on sample IEPs to convey implementation understanding, since they will probably not account for all possible permutations.

The following rule relates to validity of an IEP Sample XML Document:

Rule 5-45. Validating an IEP Sample XML Document

[Rule 5-45] (IEP) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) with a c:pathURI attribute owned by a c:IEPSampleXMLDocument, the artifact resolved by the value of c:pathURI MUST be valid for the validity constraints of the c:IEPConformanceTarget parent of c:IEPSampleXMLDocument.

**context** IEPConformanceTargetType **inv:**

The constraint is enforced during provisioning of the Sample XML Document.

### <Artifact> [ModelPackageDescription](#_cb5967a836f94243e179329cb199a401)

A ModelPackageDescription Artifact represents a NIEM Model Package Description (MPD). Specifically, it represents the information in the NIEM-3 MPD catalog, which is defined for target namesapce=http://reference.niem.gov/niem/resource/mpd/catalog/3.0/, in the context of a NIEM-3 subset of the niem-core schema.

An MPD is a logical set of electronic files aggregated and organized to fulfill a specific purpose in NIEM. Directory organization and packaging of an MPD should be designed around major themes in NIEM: reuse, sharing, interoperability, and efficiency. The inclusion of artifacts in an MPD is modeled using a Usage dependency from the InstanceSpecification representing the MPD to the model element representing the artifact (most commonly a Namespace Package).

The attributes of the ModelPackageDescription correspond to components of the MPDType within the MPD Catalog Schema.  The information model fragment corresponding to c:MPDInformation has been partially flattened from the schema containment structure into the ModelPackageDescription Attributes.

In addition to the largely isomorphic representation of the MPD Catalog as parts of the ModelPackageDescription, there are a few convenience mechanisms to simplify the UML Model.   The representation of the MPD relationship between MPDs is modeled as a <<ModelPackageDescriptionRelationship>> Dependency from the client ModelPackageDescription Artifact Instance to the related ModelPackageDescription Artifact Instance as supplier.

Instances of ModelPackageDescription may be the client of a UML Usage to some supplying NIEM concept, such as <<InformationModel>>.  These UML Usages may be stereotyped with a sub-stereotype of <<FileType>>.  When such a Usage is defined, values of some stereotype tags may be derived, such as the pathURI (based on UML Package Structure and/or NIEM packaging structure guidelines).  Note that most <<FileType>>s are implicit anyway, being derived by transitive closure of all <<InformationModel>>s referenced by used <<InformationModel>>s.

##### Properties

###### ArtifactOrArtifactSet

A data concept for a file or file set in an MPD.

|  |  |
| --- | --- |
| **Type** | ArtifactOrArtifactSet |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### AuthoritativeSource

An official sponsoring or authoring organization responsible for an MPD.

|  |  |
| --- | --- |
| **Type** | EntityRepresentation |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### CreationDate

Date this MPD was published or created. Implemented as the value of the CreationDate element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

###### descriptionText

A description of the MPD. A statement that provides an explanation or additional detail.

Implemented as the value of the DescriptionText element of the MPDType type within the MPD Catalog document instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### DomainText

A NIEM Domain applicable to, associated with, or that uses the MPD. Implemented as the value of the DomainText element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ExchangePartnerName

Name of an agency, organization, or entity that uses the MPD (in particular to share or exchange data). Implemented as the value of the ExchangePartnerName element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### ExchangePatternText

A description of a transactional, design, or exchange pattern the MPD uses (generally, applicable to IEPDs only). Implemented as the value of the ExchangePatternText element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### IEPConformanceTarget

A class or category of IEPs which has a set of validity constraints and a unique identifier. Every IEP is an instance of one or more IEP Conformance Targets.

|  |  |
| --- | --- |
| **Type** | IEPConformanceTargetType |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### KeywordText

A keyword associated with the MPD; a common alias, term, or phrase that would help to facilitate search and discovery of this MPD. Implemented as the value of the KeywordText element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### LastRevsionDate

Date the MPD was last revised. Implemented as the value of the LastRevisionDate element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### mpdBaseURI

The left hand substring of an MPD URI that does not include its mpdVersionID. The concatenation of mpdBaseURI and mpdVersionID becomes the value of the mpdURI attribute of the MPD element in the catalog instance.

The last segment of mpdBaseURI becomes the value of the mpdName attribute of the MPD element in the catalog instance.

Note that the relationship between mpdBaseURI, mpdURI, and mpdName are more restrictive than the rules expressed in NIEM 3 MPD, but are consistent with guidelines recommended/implied by the NIEM 3 MPD.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

###### mpdClassCode

The classification code of the MPD. Maps to the value of the mpdClassURIList attribute of the MPDType within the catalog instance. This code designates the classification or kind of the MPD.

Note that NIEM-3 MPD explicitly defines only the iepd classification code, with the other classification codes implied but not formally defined with the MPD specification.

|  |  |
| --- | --- |
| **Type** | ModelPackageDescriptionClassCode |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

###### mpdVersionID

Many published MPDs will be periodically revised and updated; therefore, versioning is required to clearly indicate that changes have occurred. A version number is actually part of the unique identification for an MPD (to be discussed in a subsequent section). All NIEM version numbers adhere to the regular expression: [0-9]+(\.[0-9]+)\*((alpha|beta|rc|rev)[0-9]+)? Where:

* "alpha" indicates early development
* "beta" indicates late development; but changing or incomplete
* "rc" indicates release candidate; complete but not approved as operational
* "rev" indicates very minor revision that does not impact schema validation

Note that the value of mpdVersionID is concatenated with mpdBaseURI to form the mpdURI.  This convention is more restrictive then the NIEM 3 MPD rules, but is consistent with recommended/implied MPD naming conventions.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

###### PurposeText

A description for the purpose, function, intended use of, or reason for the existence of the MPD. Implemented as the value of the PurposeText element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### StatusText

Description of the current state of development or usage of the MPD; may also project future plans for the MPD. Implemented as the value of the StatusText element in the catalog instance.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

##### Constraints

###### MPD [Rule 3-09]

**[Rule 3-9]** A NIEM-conforming IEPD or EIEM MUST contain at least one schema that is either a NIEM reference schema or a subset derived from a NIEM reference schema.

**context** ModelPackageDescription **inv:**

// pseudo code for specifygin constraint in terms of tag values of foreign stereotype instances  
  
  
(  
 (self.mpdClassCode=ModelPackageDescriptionClassCode::iepd)   
 or (self.mpdClassCode=ModelPackageDescriptionClassCode::eiem)  
) implies  
 self.oclAsType(InstanceSpecification).clientDependency->select(d| d.stereotypedBy('XMLSchemaDocument'))  
 ->exists(xmlDoc|  
 xmlDoc.stereotypedBy('SubsetSchemaDocument')  
 or xmlDoc.stereotypedBy('ReferenceSchemaDocument')  
 or xmlDoc.supplier->exists(s|s.isDirectlyOrHasIndirectlyASubsetOrReferenceSchema())  
 or xmlDoc.supplier.getStereotypeApplication('InformationModel').defaultPurpose.name  
 ->exists(purposeURI|  
 (purposeURI='subset')  
 or  
 (purposeURI='reference')  
 )   
 )

###### MPD [Rule 3-10]

**[Rule 3-10]** A NIEM IEPD MUST contain at least one valid sample XML instance (i.e., IEP) artifact for each exchange schema element that can be the root of a corresponding IEP.ding IEP.l>

**context** ModelPackageDescription **inv:**

This constraint is realized by PSM-MPD transformations.

###### MPD [Rule 4-03.1]

**[Rule 4-3.1]** A higher MPD version number within a version series does NOT imply compatibility between versions. Compatibility between or among MPD versions MUST be explicitly stated in documentation.entation.l>

**context** ModelPackageDescription **inv:**

Satisfaction of this constraint requires comparative analysis between versions;  
 can not be expressed easily in OCL.

###### MPD [Rule 4-06]

**[Rule 4-6]** Each file artifact in an MPD MUST have a corresponding File element in the catalog for that MPD.MPD.l>

**context** ModelPackageDescription **inv:**

Constraints for catalog construction are resolved in PSM-MPD transformation.

###### MPD [Rule 4-07]

**[Rule 4-7]** Each file set artifact in an MPD MUST have a corresponding FileSet element in the catalog for that MPD. This FileSet element must identify each file artifact that is a member of that file set artifact.artifact.

**context** ModelPackageDescription **inv:**

Constraints for catalog construction are resolved in PSM-MPD transformations.

###### MPD [Rule 4-08]

**[Rule 4-8]** Each artifact identified in the catalog MUST be assigned an id in the format of an NCName (Non-Colonized Name) as defined by [W3-XML-Namespaces]. This is required for both File and FileSet artifacts.rtifacts.l>

**context** ModelPackageDescription **inv:**

All catalog constraints are resolved in PSM-MPD transformation.

###### MPD [Rule 4-10]

**[Rule 4-10]** NIEM namespaces MUST NOT be used as URIs for MPD artifacts.cts.l>

**context** ModelPackageDescription **inv:**

Constraints on artifact URIs are resolved during PSM-MPD transformations.

###### MPD [Rule 4-11]

**[Rule 4-11]** Every MPD that is a reference schema set (i.e., NIEM releases, core updates, and domain updates) MUST contain an XML change log artifact that:

� Validates with the NIEM change log schemas (mpd-changelog.xsd and niem-model.xsd).  Note: These are the base filenames; the actual filenames also contain a version number. For example: mpd-changelog-1.0.xsd is the current version.

� Records changes to previous reference schemas that this MPD represents.

� Bears the file name "changelog.xml".

� Resides in the root directory of the MPD.

"changelog.xml". � Resides in the root directory of the MPD.

**context** ModelPackageDescription **inv:**

Constraints on changelog are resolved during PSM-MPD transformations.

###### MPD [Rule 4-12]

**[Rule 4-12]** Every MPD that is an IEPD or EIEM MUST contain a change log artifact that:

� Records changes to previous IEPD or EIEM schemas that this MPD represents.

� Begins with the substring "changelog".

� Resides in the root directory of the MPD.

� Resides in the root directory of the MPD.

**context** ModelPackageDescription **inv:**

Constraints on changelogs are resolved by PSM-MPD transformations.

###### MPD [Rule 4-13.1]

**[Rule 4-13.1]** If an IEPD or EIEM contains more than one change log artifact, then each change log artifact MUST:

� Have a file name that begins with the substring "changelog".

� Reside in the MPD root directory .

� Reside in the MPD root directory .

**context** ModelPackageDescription **inv:**

Constraints on changelogs are resolved by PSM-MPD transformations.

###### MPD [Rule 4-13]

**[Rule 4-13]** The initial version of an IEPD or EIEM MUST contain a change log artifact with at least one entry for its creation date.ate.l>

**context** ModelPackageDescription **inv:**

Constraints on changelogs are resolved during PSM-MPD transformation.

###### MPD [Rule 6-3b]

**[Rule 6-3b]** Within an MPD,the and substrings in the file name MUST match exactly the values for attributes mpdName and mpdVersionID within its catalog.xml artifact.artifact.l>

**context** ModelPackageDescription **inv:**

Packaging constraints are resolved by PSM-MPD transformations.

###### MPD [Rule 6-3c]

**[Rule 6-3c]** Within an MPD,the substring in the file name MUST correctly correspond to the value for the attribute mpdClassCode within catalog.xml. Correct correspondence is:

|  |  |
| --- | --- |
| IF file name = | THEN catalog.xml mpdClassCode = |
| rel | release |
| cu | core-update |
| du | domain-update |
| iepd | iepd |
| eiem | eiem |

 cu

**context** ModelPackageDescription **inv:**

Packaging constraints are resolved by PSM-MPD transformations.

###### MPD [Rule 6-7]

**[Rule 6-7]** A published IEPD MUST link (through its catalog) to any EIEM it is based on.on.ml>

**context** ModelPackageDescription **inv:**

An EIEM is an MPD with a packageCode of EIEM. An EIEM is typically bundled as a  
 reusable model library which can be referenced from IEPDs. This relationship between  
 EIEM and IEPD is used by PSM-MPD transformations to construct the catalog entries in  
 resolution of this constraint.

###### MPD [Rule 6-8]

**[Rule 6-8]** Within an MPD archive, if non-NIEM-conforming schemas from other standards are used and referenced within an MPD, then all xsd:import, xsd:include, and xsd:redefine constructs used within those schemas MUST be modified as needed to have a value for the schemaLocation attribute that is a relative path reference that resolves to the correct schema within the sub-tree.ema within the sub-tree.l>

**context** ModelPackageDescription **inv:**

The schemaLocation constraints are resolved during PSM-MPD transformation.

###### MPD3 [Rule 3-2] (MPD). MPD with MPD class of IEPD is an IEPD

[Definition: *information exchange package documentation*]

An [information exchange package documentation](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is a [model package description](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) that:

* has an [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) of http://reference.niem.gov/niem/specification/model-package-description/3.0/#IEPD, and
* conforms to all the rules in this specification for the conformance target [information exchange package documentation](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) (i.e., applicability code IEPD).

This term may be abbreviated IEPD. Rules specifying this conformance target use the applicability code IEPD.

Because it is an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description), an IEPD must also conform to all WF-MPD rules.

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) has one or more c:IEPConformanceTarget elements within its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), to define one or more [information exchange packages](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package), each an [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document).

[Definition: *instance XML document*]

An [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) is an [XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_document) that is valid against an [XML Schema](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_Schema). An instance XML document is said to be an instance of the schema to which it validates.

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) defines one or more data exchanges, each occurring in the form of an XML document known as an [information exchange package](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package) (IEP). A data exchange is specified by an [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target), which defines the class of [information exchange packages](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package) (IEPs) that are exchanged at runtime.

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) contains a NIEM-conformant XML schema document set that may include portions of a NIEM Core schema document (and updates), portions of NIEM Domain schema documents (and updates), and enterprise-specific or IEPD-specific [extension schema documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_extension_schema_document). The [XML schema documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) contained in an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) work together to define one or more classes of [instance XML documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) that consistently encapsulate data for meaningful information exchanges. Any [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) that is valid for the [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) set and that satisfies the conditions of the [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) is a member of that [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) class (or IEP Class).

[XML schema documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) in an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) conform to the [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) and may use or extend data component definitions drawn from NIEM. An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) may also incorporate and use XML schema documents from other standards that do not conform to NIEM. (See [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) for details.) An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) consists of a set of artifacts (XML schema documents, documentation, sample instance XML documents, etc.) that together define and describe one or more implementable data exchanges. An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) should contain all materials necessary to:

* + Understand information exchange context, content, semantics, and structure.
  + Create and validate XML documents defined by the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation), and used for information exchanges.
  + Identify the lineage of the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) itself and optionally its artifacts.

(The terms [information exchange package](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package) (IEP) and [information exchange package documentation](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) (IEPD) first appeared in [FEA Data Reference Model 1.0] and [GJXDM IEPD Guidelines 1.1], respectively.)

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) defines one or more classes of [instance XML documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document). In NIEM, each of these instances is called an [information exchange package](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package) (or IEP) that satisfies all validity constraints for its class as defined by the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). An IEP is an information message payload serialized as XML and transmitted in some way, for example over a communications network. ([[FEA Data Reference Model 1.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#FEA-DRM) and [[GJXDM IEPD Guidelines 1.1]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#GJXDM-IEPD) are the original sources of the terms *information exchange package* and *information exchange package documentation*, respectively).

The following rule specifies an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) as a [conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target):

Rule 3-2. MPD with MPD class of IEPD is an IEPD

[Rule 3-2] (MPD) (Constraint)

A [model package description](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) with an [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) of http://reference.niem.gov/niem/specification/model-package-description/3.0/#IEPD MUST be an [information exchange package documentation](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation).

**context** ModelPackageDescription **inv:**

Rule is definitional

###### MPD3 [Rule 3-3] (IEPD). IEPD Conformance Target Identifier

The following rule is applicable to all IEPDs:

Rule 3-3. IEPD Conformance Target Identifier

[Rule 3-3] (IEPD) (Constraint)

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) MUST have the [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) http://reference.niem.gov/niem/specification/model-package-description/3.0/#IEPD as a value of its c:mpdClassURIList attribute.

How to declare validity constraints for one or more IEP classes within an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) will be covered in more depth in [Section 5.6, *Defining Information Exchange Packages*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#section_5.6).

Note that NIEM conformance does not require that an IEP be native XML on the transmission medium. A NIEM-conformant IEP may be encrypted, compressed (e.g., using [[PKZIP]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#PKZIP), [[EXI Format 1.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-EXI), etc.), or wrapped within an envelope mechanism, as long as its original native XML form can be retrieved by the receiver.

Common to [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) MPDs:

* Requires a readme artifact.
* Its [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) set defines data exchanges ( [information exchange packages](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package) or IEPs).
* Can contain subset, extension, external, or constraint schema documents.
* Must declare at least one or more [IEP conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target).
* Contains sample instance XML documents that validate to XML schema document set.

**context** ModelPackageDescription **inv:**

Constraint realized during provisioning of MPD Catalog.

###### MPD3 [Rule 4-1] (Schema-subset). Fundamental NIEM Subset Rule

A NIEM *schema document subset* is a set of XML schema documents that constitutes a reduced set of components derived from a NIEM reference schema document or document set associated with a given numbered release or domain update.

[Definition: *schema document subset*]

An XML schema document set based on a reference schema document set intended to ensure that any [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) valid to the schema document subset is also valid to the reference schema document set.

The primary purpose for a schema document subset is to reduce and constrain the scope and size of a full NIEM reference schema document set for use within an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). A schema document subset is derived from a reference schema document set (such as a NIEM release) by applying subset operations (See [Section 4.2.2, *Constructing a Schema Document Subset*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#section_4.2.2)). Also, note that employing a subset of a reference schema document set within an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is optional; it is completely valid to reuse NIEM reference schema documents as-is within IEPDs.

The fundamental rule for a valid NIEM schema document subset is formally stated follows:

Rule 4-1. Fundamental NIEM Subset Rule

[Rule 4-1] (Schema-subset) (Constraint)

A schema document subset ($SUBSET) for a given reference schema document set ($REFERENCE) MUST be defined such that for all instance XML documents ($XML), where $XML is valid to $SUBSET, $XML is valid to $REFERENCE.

A [schema document subset](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_schema_document_subset) is composed of [XML schema documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document). A [schema document subset](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_schema_document_subset) can essentially be a [reference schema document set](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_reference_schema_document_set) (i.e., a NIEM release) that has been modified by applying subset operations to support business requirements represented in an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). A subset derived from a reference schema document set may differ from that reference such that its content has been reduced and/or constrained.

[Definition: *subset schema document*]

An XML schema document that meets all of the following criteria:

* It is built from a reference schema document set where one or more reference schema documents have been substituted by corresponding subset schema documents.
* It is built from a reference schema document by applying subset operations to the XML schema statements in a reference schema document.
* It is explicitly designated as a subset schema document. This is accomplished by declaration in the relevant MPD catalog or by a tool-specific mechanism outside the subset schema document.
* It has a target namespace previously defined by a reference schema document. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema document.
* It does not alter the business semantics of components in its namespace. The reference schema document defines these business semantics.
* It is intended to express the limited vocabulary necessary for an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) and to support XML Schema validation for an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation).

**context** ModelPackageDescription **inv:**

Rule is definitional.

###### MPD3 [Rule 5-10] (WF-MPD). MPD Version Number Syntax

Published MPDs may be periodically revised and updated; therefore, versioning is required to clearly indicate changes have occurred. In order to maintain some consistency while allowing reasonable flexibility to authors, this specification establishes a simple version numbering scheme that is consistent with most common practices. This is the same version numbering scheme that is used for NIEM releases.

An MPD version number is the value of the c:mpdVersionID attribute owned by the c:MPD element within its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document). A consistent version number syntax is enforced by the MPD catalog schema in [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A). The syntax rule is as follows:

Rule 5-10. MPD Version Number Syntax

[Rule 5-10] (WF-MPD) (Constraint)

An MPD MUST be assigned a version number that adheres to the regular expression:

version ::= digit+ ('.' digit+)\* (status digit+)?

Where:

digit ::= [0-9]

status ::= 'alpha' | 'beta' | 'rc' | 'rev'

The meaning of the status values are as follows:

* alpha indicates early development; changing significantly.
* beta indicates late development; but changing or incomplete.
* rc indicates release candidate; complete but not approved as operational.
* rev indicates very minor revision that does not impact schema validation.

The regular expression notation used above is from [[W3C XML 1.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XML) [#sec-notation](http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation).

Note that the absence of a status string in the version number indicates that the version has been baselined and published.

The following examples are valid MPD version numbers:

* + 1
  + 1.2
  + 1.3.1.0
  + 1.2alpha13
  + 199.88.15rev6

There are two implications in [Rule 5-10, *MPD Version Number Syntax*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_5-10). The first is that in some cases this version scheme implies and confirms a chronology of releases. For example, a given product labeled version 2.3 must have been released before the same product labeled 2.3.1. Therefore, version 2.3.1 is more current than version 2.3.

However, this is a multi-series version scheme, and chronological relationships exist only within a given series. So, for example, nothing can be said about a chronological relationship between versions 2.2.4 and 2.3. This is because version 2.2.4 is in a different series (i.e., 2.2) and could actually have been released after 2.3.

**context** ModelPackageDescription **inv:**

self.mpdVersionID.match('[0-9]+(\\.[0-9]+)\*((alpha|beta|rc|rev)[0-9]+)?')

###### MPD3 [Rule 5-11] (WF-MPD). MPD URI Is Absolute

o facilitate MPD sharing and reuse, the assignment of a URI (Uniform Resource Identifier) to an MPD is essential. This is enforced by the MPD catalog schema document [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A). It is also important to ensure that an MPD URI is absolute.

Rule 5-11. MPD URI Is Absolute

[Rule 5-11] (WF-MPD) (Constraint)

In an MPD catalog document, the value of a c:mpdURI attribute of type xs:anyURI MUST match the production <absolute-URI> as defined by [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI), [§4.3, Absolute URI](http://tools.ietf.org/html/rfc3986#section-4.3).

This rule implies that a URI assigned to an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) must be valid. Furthermore, the entity (person or organization) assigning the MPD URI either (1) *is* the registrant of the domain name or namespace identifier, or (2) *has* authority from the registrant to assign this URI.

Examples of valid MPD URIs:

* http://example.gov/niem-iepd/prescription-monitoring-info-exchange/3.0/
* http://example.gov/niem-iepd/pmix/3.0/
* http://release.niem.gov/niem/niem-core/3.0/
* http://niem.gov/niem/domains/cyfs/2.1/1

This specification does not mandate that basic MPD catalog metadata be designed into an MPD URI. However, including such can obviously provide convenient visual recognition. That said, an author should ensure any metadata embedded in the URI accurately reflect the MPD catalog metadata (in particular, the values of c:mpdURI, c:mpdName, c:mpdVersionID, and c:mpdClassURIList defined in the MPD catalog document).

**context** ModelPackageDescription **inv:**

Expressing constraint in OCL is deferred.

###### MPD3 [Rule 5-12] (WF-MPD). MPD URI Supports Fragment

Artifacts in other MPDs can be referenced from within an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) to identify equivalence (signify reuse, one aspect of lineage). To support this concept, the following MPD URI rules are necessary:

Rule 5-12. MPD URI Supports Fragment

[Rule 5-12] (WF-MPD) (Constraint)

A valid MPD URI MUST support the inclusion of a *fragment identifier* (as a suffix) [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI).

This rule ensures that an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) can always uniquely identify and refer to each artifact within another MPD. This MPD specification follows [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI) which forbids a URI to contain more than a single fragment identifier. To construct an MPD artifact URI, add a fragment (that locally identifies the artifact) to an MPD URI, and therefore, an MPD URI cannot already contain a fragment.

**context** ModelPackageDescription **inv:**

Constraint is definitional.

###### MPD3 [Rule 5-13] (WF-MPD). MPD URI Has No Fragment [Rule 5-13] (WF-MPD) (Constraint)

Rule 5-13. MPD URI Has No Fragment

[Rule 5-13] (WF-MPD) (Constraint)

A valid MPD URI MUST NOT contain a *fragment identifier* [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI).

Rationale: If a URI for an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) (do NOT confuse this with a URI for an MPD artifact) already contains a fragment identifier, then that URI cannot be employed as an MPD artifact URI, because [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI) only allows a single fragment identifier.

**context** ModelPackageDescription **inv:**

Constraint is definitional.

###### MPD3 [Rule 5-14] (WF-MPD). MPD Artifact URI Syntax

By the following rule, each file artifact or artifact set is uniquely identified by its [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name) relative to the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory).

Rule 5-14. MPD Artifact URI Syntax

[Rule 5-14] (WF-MPD) (Interpretation)

Within an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) a URI reference to an artifact in another external MPD (i.e., an MPD artifact URI) is the concatenation of:

* The URI of the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) that contains the artifact.
* A pound-sign character ("#" — also known as a hashtag character).
* An identifier that is the artifact’s locally unique [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name) relative to the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory).

An artifact set has a locally unique [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name). An artifact has a path name that terminates with its file name which is unique to the directory it resides in.

The following are examples of valid MPD artifact URIs:

* + http://example.gov/niem-iepd/pmix/3.0/#subset/niem-core.xsd (a file artifact)
  + http://example.gov/niem-iepd/pmix/3.0beta2/#extension/ext-1.1.xsd (a file artifact)
  + http://example.gov/niem-iepd/pmix/3.0/#application-info (a set artifact)
  + http://example.gov/niem-iepd/pmix/3.0/#iep-sample/query (a set artifact)

Since MPD URIs require the support of fragment identifiers (by [Rule 5-12, *MPD URI Supports Fragment*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_5-12)), it does not appear that the urn URI scheme may be used as an MPD URI. Fragments use the # charater, and the specification for the urn scheme ([[RFC 2141 URN Syntax]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC2141-URN-Syntax)) indicates that they are not valid in URNs, when it states:

RFC 1630 reserves the characters /, ?, and # for particular purposes. The URN-WG has not yet debated the applicability and precise semantics of those purposes as applied to URNs. Therefore, these characters are RESERVED for future developments.

Artifact URIs are used as values for the c:externalURI attribute in the MPD catalog XML document to declare equivalence relationships between artifacts (See [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A)). A simple scenario follows. Consider two different IEPDs with the following URIs:

* + 1. http://example.gov/niem-iepd/pmix/3.0/
    2. http://www.abc.org/niem-iepd/order/2.1.2rev3/

The author of [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) (2) has decided to reuse the base-xsd/extension/req1.xsd artifact in [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) (1) as-is. He/she can optionally create an MPD catalog c:ExtensionSchemaDocument entry for this artifact (assuming it is an extension schema document), and add the attribute:

c:externalURI="http://example.org/niem-iepd/pmix/3.0/#base-xsd/extension/req1.xsd"

Additional c:externalURI attributes may be added to this entry if the author knows of other uses of this same artifact in other MPDs and wishes to acknowledge them.

A URI does not have the same meaning as namespace. NIEM namespaces cannot be used as MPD artifact URIs. Recall that the target namespace used in a subset schema document derived from a NIEM release schema document is identical to the target namespace of that release schema document. Furthermore, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) may contain multiple subsets. NIEM namespaces are not necessarily unique to an artifact within an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description). Later, [Section 5.5, *XML Catalogs*, below,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#section_5.5) will describe the use of [[XML Catalogs 1.1]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#OASIS-XML-Catalogs) to correlate namespaces to local URIs in order to [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) them to local resources.

The value of c:externalURI is an identifier for a remote resource that is not necessarily accessible online. For this reason, even though such URIs should be correct (i.e. a resource with that URI should exist), their verification is not within the scope of this specification.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-15] (WF-MPD). c:pathURI Resolves to a Resource

An MPD uses the file directory system of path names and file names to identify local artifacts and artifact sets. All local URIs are relative to the location of the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), and therefore, they are also relative to the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory) since the MPD catalog document resides in the MPD root directory.

In general, every value of attribute c:pathURI in an MPD catalog document will be a relative [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name) to a directory (i.e., an artifact set), or to a file (i.e., an artifact). The following are typical examples of each:

Artifact Set: c:pathURI="base-xsd/niem/niem-core/3.0"

Artifact:      c:pathURI="base-xsd/niem/niem-core/3.0/niem-core.xsd"

Note that per [Table 5-1, *Summary of*](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#table_5-1) [*RFC 3986 URI: Generic Syntax*](http://tools.ietf.org/html/rfc3986), below, and [Table 5-2, *Summary of MPD URI attributes*, below,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#table_5-2) a local URI may contain an optional fragment. Although c:pathURI has no use for a URI with a fragment, MPD documentation artifacts could reference a subpart within a local artifact by using a relative URI with a fragment.

Despite its simplicity, c:pathURI comes with over a dozen rules that help to define a [model package description](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description). These rules ensure that every c:pathURI attribute value in a well-formed [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) resolves to a correct local resource:

Rule 5-15. c:pathURI Resolves to a Resource

[Rule 5-15] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a resource.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog. c:pathURI is either set implicitly to a location, or if a value is provided, the resource is moved to specified location.

###### MPD3 [Rule 5-16] (WF-MPD). c:pathURI for c:XMLCatalog

Rule 5-16. c:pathURI for c:XMLCatalog

[Rule 5-16] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:XMLCatalog element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to an [XML catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_catalog_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-17] (WF-MPD). c:pathURI for c:MPDChangeLog

Rule 5-17. c:pathURI for c:MPDChangeLog

[Rule 5-17] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:MPDChangeLog element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a [change log](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_change_log).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog. Provisioning set values of c:pathURI based on relative location of changelog package.

###### MPD3 [Rule 5-18] (WF-MPD). c:pathURI for c:ReadMe

Rule 5-18. c:pathURI for c:ReadMe

[Rule 5-18] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:ReadMe element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-19] (WF-MPD). c:pathURI for c:IEPSampleXMLDocument

Rule 5-19. c:pathURI for c:IEPSampleXMLDocument

[Rule 5-19] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:IEPSampleXMLDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to an [XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-1] (WF-MPD). MPD Has an mpd-catalog.xml in its Root Directory

XML schema documents (and the schemas that result from them) are the essence of a NIEM [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description). However, a variety of documentation files may be incorporated into a NIEM [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description). However, in addition to XML schema documents, there is one mandatory documentation artifact required by every [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description): the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document). An mpd-catalog.xml contains basic metadata, relationship and lineage data, [conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target) specifications, and validation information.

A [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) (formerly known as a *master document*) is mandatory for IEPDs. This [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) may be built by different developers, and may be registered into a repository for reuse by many other users, developers, and implementers; therefore, a minimal form of documentation is absolutely necessary. An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) readme file is the primary source and starting point for human readable documentation, and should reference (and describe) any other separate documentation artifacts. This requirement ensures that baseline documentation is consistently rooted in a clearly visible artifact within each [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description).

The following subsections address these documentation artifacts and the concepts, metadata, and content each supports.

5.1. NIEM MPD Catalog

[Definition: *MPD catalog document*]

An [instance XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_instance_XML_document) that:

* conforms to all the rules in this specification for the conformance target [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) (i.e., applicability code MPD-catalog), and
* contains metadata describing:
  + MPD unique identification
  + [Conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target)
  + Basic characteristics and properties
  + Key artifacts and directory structure
  + Relationships to other MPDs and their artifacts

This term may be abbreviated MPD-catalog. Rules specifying this conformance target use the applicability code MPD-catalog.

Each [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) may have somewhat different catalog requirements. The catalog metadata are formally defined by the XML Schema document in [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A). MPD catalog metadata are designed to be the minimal needed to facilitate human understanding, tool support, and machine processing. The metadata can support a number of [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) uses and functions including (but not limited to):

* + - Identification of key artifacts
    - Generation of a hyperlinked content display using XSLT
    - Browsing and understanding of artifacts and their content
    - Automatic registration into a registry/repository
    - Search, discovery, retrieval of MPDs (through metadata and relationships)
    - Reuse of MPDs and their artifacts
    - Tracing and analysis of MPD lineage
    - General conformance and validation of the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) itself
    - Definition, identification, and validation of IEP conformance targets

Rule 5-1. MPD Has an mpd-catalog.xml in its Root Directory

[Rule 5-1] (WF-MPD) (Constraint)

Within its [root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory), an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) MUST contain an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) artifact with name mpd-catalog.xml.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-20] (WF-MPD). c:pathURI for c:BusinessRulesArtifact

Rule 5-20. c:pathURI for c:BusinessRulesArtifact

[Rule 5-20] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:BusinessRulesArtifact element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a [business rule schema](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_business_rule_schema) or [business rules](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_business_rules) artifact.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-21] (WF-MPD). c:pathURI for c:XMLSchemaDocument

Rule 5-21. c:pathURI for c:XMLSchemaDocument

[Rule 5-21] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:XMLSchemaDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-22] (WF-MPD). c:pathURI for c:ExternalSchemaDocument

Rule 5-22. c:pathURI for c:ExternalSchemaDocument

[Rule 5-22] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:ExternalSchemaDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-23] (WF-MPD). c:pathURI for c:ReferenceSchemaDocument

Rule 5-23. c:pathURI for c:ReferenceSchemaDocument

[Rule 5-23] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:ReferenceSchemaDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a NIEM [reference schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_reference_schema_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-24] (WF-MPD). c:pathURI for c:ExtensionSchemaDocument

Rule 5-24. c:pathURI for c:ExtensionSchemaDocument

[Rule 5-24] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:ExtensionSchemaDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a NIEM [extension schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_extension_schema_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-25] (WF-MPD). c:pathURI for c:SubsetSchemaDocument

Rule 5-25. c:pathURI for c:SubsetSchemaDocument

[Rule 5-25] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:SubsetSchemaDocument element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a NIEM [subset schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_subset_schema_document).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-26] (WF-MPD). c:pathURI for c:Wantlist

Rule 5-26. c:pathURI for c:Wantlist

[Rule 5-26] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:Wantlist element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a [NIEM wantlist](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_NIEM_wantlist) XML document.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-27] (WF-MPD). c:pathURI for c:SchematronSchema

Rule 5-27. c:pathURI for c:SchematronSchema

[Rule 5-27] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:SchematronSchema element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a [Schematron schema](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_Schematron_schema).

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-28] (WF-MPD). c:pathURI for c:RelaxNGSchema

Rule 5-28. c:pathURI for c:RelaxNGSchema

[Rule 5-28] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:RelaxNGSchema element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a RelaxNG schema.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-29] (WF-MPD). c:pathURI for c:SchemaDocumentSet

Rule 5-29. c:pathURI for c:SchemaDocumentSet

[Rule 5-29] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:SchemaDocumentSet element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) set.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-2] (MPD-catalog). MPD Catalog Document Valid to mpd-catalog-3.0.xsd

Rule 5-2. MPD Catalog Document Valid to mpd-catalog-3.0.xsd

[Rule 5-2] (MPD-catalog) (Constraint)

An [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) MUST be valid to mpd-catalog-3.0.xsd [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A).

This rule requires validation with mpd-catalog-3.0.xsd, which also imports a NIEM schema subset. So, validation of the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) must be done in the context of the catalog schema document, its associated NIEM subset, and mpd-catalog.xml. This does not require the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) to contain copies of the catalog schema document or the schema subset (since these are standard for all MPDs). However, a validation tool must have access to all three XML documents.

The XML schema documents required to validate an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) are available in the [[NIEM MPD Toolkit]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-MPD-Toolkit). Note that validators often require references to schemas and their imports. This may be done through a command line instruction or by adding a schemaLocation attribute to xs:import statements.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-30] (WF-MPD). c:pathURI for c:ConstraintSchemaDocumentSet

Rule 5-30. c:pathURI for c:ConstraintSchemaDocumentSet

[Rule 5-30] (WF-MPD) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document), the value of a c:pathURI attribute owned by a c:ConstraintSchemaDocumentSet element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a NIEM [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) set.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-31] (WF-MPD).

Rule 5-31.

[Rule 5-31] (WF-MPD) (Interpretation)

Any [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) set whose c:pathURI attribute resolves to a [constraint schema document set](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_constraint_schema_document_set) MUST be interpreted to be a [constraint schema document set](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_constraint_schema_document_set).

**context** ModelPackageDescription **inv:**

Constraint is definitional.

###### MPD3 [Rule 5-32] (WF-MPD). Resolve MPD URI with Fragment

Rule 5-32. Resolve MPD URI with Fragment

[Rule 5-32] (WF-MPD) (Interpretation)

Given an absolute MPD URI [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI), [§4.3, Absolute URI](http://tools.ietf.org/html/rfc3986#section-4.3) with a fragment, resolve this URI as follows:

1. Resolve the base URI (per [[RFC 3986 URI]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#RFC3986-URI)) to retrieve the resource MPD. If the resource MPD does not exist, then fail (existence error).
2. Apply the fragment (without "#") to the MPD resource:
   1. Locate a structures:id attribute value that matches the fragment string. If more than one exist, then fail (ambiguity error). If none exists, then continue.
   2. Locate a [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name) (for a directory or file) that matches the fragment string. If more than one exist, then fail (ambiguity error). If none exists, then fail (existence error).
3. Return the element, directory, or file found.

In the presence of NIEM [reference elements](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_reference_element), URI resolution may require an additional step to account for indirect references.

**context** ModelPackageDescription **inv:**

Constraint is definitional.

###### MPD3 [Rule 5-33] (XML-catalog). XML Catalog uri Value Resolves to Resource

An [XML catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_catalog_document) conforms to [[XML Catalogs 1.1]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#OASIS-XML-Catalogs). For the purpose of MPD validation, the following rules ensure that an XML catalog document contains URIs that correctly resolve.

Rule 5-33. XML Catalog uri Value Resolves to Resource

[Rule 5-33] (XML-catalog) (Constraint)

Within an [XML catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_catalog_document), the value of a uri attribute owned by a er:uri element MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a resource.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of XML Catalog.

###### MPD3 [Rule 5-34] (XML-catalog). XML Catalog uri Value Resolves to Resource with Correct Target Namespace

Rule 5-34. XML Catalog uri Value Resolves to Resource with Correct Target Namespace

[Rule 5-34] (XML-catalog) (Constraint)

Within an [XML catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_catalog_document), given an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) resolved by the value of a uri attribute owned by a er:uri element, the [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) target namespace MUST equal the value of the name (a namespace string) attribute owned by the er:uri element.

**context** ModelPackageDescription **inv:**

Constraint is realized during provisioning of MPD Catalog.

###### MPD3 [Rule 5-35] (IEPD). IEPD Has a Change Log

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) change log is not required to conform to any particular XML schema or other format specification. However, a change log is still required for an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation).

Rule 5-35. IEPD Has a Change Log

[Rule 5-35] (IEPD) (Constraint)

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) MUST contain a [change log](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_change_log) artifact that is identified by a c:MPDChangeLog element in its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document).

The [change log](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_change_log) for the first version release of an IEPD simply contains its release date.

The format of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) change log is left to the discretion of the author. A flexible [change log](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_change_log) format encourages and facilitates easier and more rapid development. IEPDs are developed by a variety of NIEM domains, organizations, and users; and they are intended to specify implementable exchanges. As a result, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) may contain both documentation artifacts and machine readable application artifacts in a large variety of formats. As a result, a consistent standard [change log](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_change_log) would be very difficult to specify.

**context** ModelPackageDescription **inv:**

self.oclAsType(InstanceSpecification).clientDependency->select(d|d.stereotypedBy('MPDChangeLog'))->notEmpty()

###### MPD3 [Rule 5-36] (WF-MPD). Readme Describes Purpose, Scope, Business Value, etc.

[Definition: *readme artifact*]

An informal documentation artifact contained in a [model package description](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) that serves as the initial general source of human readable descriptive or instructional information. A *readme* artifact or file (formerly known as a *master document*) may index or reference other more specific documentation or other explanatory materials within the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description).

A [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) is only required for IEPDs since these MPDs are allowed the greatest design flexibility, can be developed and implemented different ways, and are not centrally managed. On the other hand, releases and domain updates have restrictive rules, standard documentation for using them, and central management.

Rule 5-36. Readme Describes Purpose, Scope, Business Value, etc.

[Rule 5-36] (WF-MPD) (Interpretation)

A [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) SHOULD (at a minimum) describe the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) purpose, scope, business value, exchange information, typical senders/receivers, interactions, and references to other documentation.

**context** ModelPackageDescription **inv:**

The readme artifact provisioned includes purpose, scope, business value, exchange information, typical senders/receivers, interactions, and references to other documentation. This information is obtained from certain modeling conventions in the model.

###### MPD3 [Rule 5-37] (IEPD). IEPD Has a ReadMe Artifact

Rule 5-37. IEPD Has a ReadMe Artifact

[Rule 5-37] (IEPD) (Constraint)

An IEPD MUST contain a [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) that is identified by a c:ReadMe element in its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document).

The [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) may replicate some of the metadata in the MPD catalog. However, the MPD catalog is intentionally designed to be efficient, easy to parse, and minimal. It is intended for search, discovery, registration, and Web page generation, and not to support various types of detailed technical prose often required for human understanding.

The primary purposes of the [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) include:

* To help facilitate understanding and reuse of IEPDs.
* To ensure that fundamental and detailed business-level information about an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) are documented for human understanding.
* To ensure the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author has considered and conveys such fundamental information.
* To provide an initial source within an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) for human consumable documentation and/or references to other business or technical documentation needed for understanding.

The [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact) is not intended to be the only source of written documentation for an MPD (though it can be). It is expected to be the initial resource that references and coordinates all others whether physically present in the MPD or linked by reference. Many organizations have their own customized formats and operating procedures for documenting their work and products. This specification does not attempt to standardize readme file name, location, format, or layout; only that it be identified in the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) of an IEPD. The following section will generally describe minimal content that should be in the [readme artifact](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_readme_artifact). This guidance is non-normative, so adherence is a subjective judgment by the author.

**context** ModelPackageDescription **inv:**

self.oclAsType(InstanceSpecification).clientDependency->select(d|d.stereotypedBy('ReadMe'))->notEmpty()

###### MPD3 [Rule 5-38] (MPD-catalog). Conformance Target Identifier

An MPD may declare one or more *IEP conformance targets* within its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document).

[Definition: *IEP conformance target*]

A [conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target) that is a class or category of IEP which has a set of one or more validity constraints and a [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier). Every IEP is an instance of one or more *IEP conformance targets*.

This definition requires that an IEP conformance target be assigned a [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) that distinguishes it from all other [IEP conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target). Construct a [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) using a fragment identifier (similar to an MPD artifact URI) per this rule:

Rule 5-38. Conformance Target Identifier

[Rule 5-38] (MPD-catalog) (Interpretation)

A [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) for an [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) declared in an MPD is formed by concatenating in sequence:

1. the IEPD URI, and
2. the pound sign character (#). and
3. a locally unique NCName (i.e., a non-colonized name, as defined by [[W3C XML Schema Part 2 Datatypes]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XML-Schema-Datatypes), [§3.3.7, NCName](http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#NCName)).

This rule requires that an [IEP conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_IEP_conformance_target) has a URI, i.e., its [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier).

**context** ModelPackageDescription **inv:**

Rule enforced during provisioning. The target identifier is synthesized.

###### MPD3 [Rule 5-39] (MPD-catalog). IEP Conformance Target Has a structures:id

The following rule is required for an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document). It supplements the rule above.

Rule 5-39. IEP Conformance Target Has a structures:id

[Rule 5-39] (MPD-catalog) (Constraint)

A c:IEPConformanceTarget element MUST own a structures:id attribute.

This rule ensures that a [conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target) can be referenced between MPDs (not just within an MPD). The value of the structures:id attribute is the NCName in [Rule 5-38, *Conformance Target Identifier*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_5-38).

**context** ModelPackageDescription **inv:**

Rule enforced during provisioning. The target structure:id attribute is synthesized.

###### MPD3 [Rule 5-3] (MPD-catalog). MPD Catalog Extension XML Catalog Document in Root Directory

An MPD catalog may be extended to accommodate new or additional metadata, artifact classifiers, or validity constraints that are not already defined in [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A).

To extend the MPD catalog, an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) author must provide both an XML catalog extension document (XML) and one or more MPD extension schema documents (XSD). The XML catalog extension identifies that one or more MPD catalog extensions are present, and resolves their namespaces to local URIs. The MPD catalog extension is a schema that defines and declares the new [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) for metadata, classifiers, and/or constraints. Both general [[NIEM Conformance 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-Conformance) and specific [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) conformance rules apply to these components. The XML catalog extension document must reside in the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory). The MPD extension schema documents may bear any file name and reside anywhere in the [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description). This is because the XML catalog is expected to [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) all local URIs. MPD processing tools are expected to look for and recognize the XML catalog (that identifies MPD catalog extensions exist) by its file name.

The following rules specify the requirements for an MPD catalog extension XML catalog document:

Rule 5-3. MPD Catalog Extension XML Catalog Document in Root Directory

[Rule 5-3] (MPD-catalog) (Constraint)

An MPD catalog extension XML catalog document MUST reside in the same relative directory as the mpd-catalog.xml artifact (normally in the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory))

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-40] (IEPD). IEPD Declares One or More IEP Conformance Targets

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) defines IEP conformance targets by explicitly declaring them within its MPD catalog per the rules above.

Rule 5-40. [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) Declares One or More IEP Conformance Targets

[Rule 5-40] (IEPD) (Constraint)

The [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) MUST contain one or more c:IEPConformanceTarget elements.

**context** ModelPackageDescription **inv:**

Rule enforced during provisioning. Multiple c:IEPConformanceTarget elements are produced based on MPD specification.

###### MPD3 [Rule 5-41] (MPD-catalog).

[Validity constraint context](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_validity_constraint_context) is explicitly declared by an XPath expression that is the value of c:xPathText. c:ValidityContext can contain any of the specific validity constraints that are substitutable for c:ValidityConstraint.

Rule 5-41.

[Rule 5-41] (MPD-catalog) (Interpretation)

Given a c:xPathText attribute owned by c:ValidityContext, the [validity constraint context](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_validity_constraint_context) for the descendant’s validity constraint SHALL be the value of c:xPathText evaluated against the IEP’s document information item (See [[W3-XML-InfoSet]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XML-InfoSet), [§2.1, The Document Information Item](http://www.w3.org/TR/2004/REC-xml-infoset-20040204/#infoitem.document)).

**context** ModelPackageDescription **inv:**

Rule is definitional.

###### MPD3 [Rule 5-42] (IEP). Identifying the Document Element of an IEP

c:HasDocumentElement is a validity constraint that identifies all intended XML document elements for an IEP conformance target, and it is directly substitutable for c:ValidityConstraintWithContext. This constraint ensures that an IEP artifact is rooted by one XML document element that is a member of the list of elements in its c:qualifiedNameList attribute. This is a common validity constraint employed by simple IEPDs that declare one or more intended XML document elements.

Note that [validity constraint context](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_validity_constraint_context) of c:HasDocumentElement is always on the IEP’s *document information item* as defined in [[W3-XML-InfoSet]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XML-InfoSet), [§2.1, The Document Information Item](http://www.w3.org/TR/2004/REC-xml-infoset-20040204/#infoitem.document). This is because it can only declare XML document elements. So, if an IEP defines a payload that may be included in some XML envelope, then c:HasDocumentElement should not be used. Instead, use c:ValidityContext with another specific validity constraint and c:xPathText to explicitly declare [validity constraint context](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_validity_constraint_context).

When employing c:HasDocumentElement the following rule applies:

Rule 5-42. Identifying the Document Element of an IEP

[Rule 5-42] (IEP) (Constraint)

Within an MPD catalog document, if an c:IEPConformanceTarget element for an IEP has a c:HasDocumentElement child element owning a c:qualifiedNameList attribute with a value of $LIST, then the document element of the IEP MUST have a *QName* that is a member of $LIST.

In UML, the QualifiedNamesType represents the c:HasDocumentElement....

**context** ModelPackageDescription **inv:**

Probably some model adjustment here to make the concept of documentElement more clear.

###### MPD3 [Rule 5-43] (IEP). Validating an XPath Expression

c:ValidToXPath is a specific validity constraint whose purpose is to ensure that a condition is satisfied within an IEP. The condition is defined by an XPath expression contained in the c:xPathText attribute. If the XPath expression applied to a target instance [XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_document) returns a Boolean value of TRUE, then the condition is satisfied by that XML document.

This validity constraint is useful for a variety of purposes. For example, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author may require that a given c:IEPConformanceTarget must contain a particular element with a particular attribute whose value is an integer greater than some required minimum. An XPath expression can validate this.

c:ValidToXPath can also employ a simple XPath expression to validate that an IEP is rooted with an intended XML document element. However, other validity constraints can do this as well; the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author may choose the constraint representation.

Note that if c:ValidToXPath is used (substituted) within c:ValidityContext there will be two XPath expressions — the expression within c:ValidToXPath is the condition to validate, the other is the context (where the condition will be validated). For example, the context provided by c:ValidityContext might be //my:speedingTicket, while the c:ValidToXPath might require that a test for exists(nc:DriverPerson) be true.

This specific validity constraint as well as those that follow below can either be substituted for the c:ValidityConstraint or used within the c:ValidityContext element (i.e., substituted for its c:ValidityConstraint child).

Note that if c:ValidToXPath is substituted for c:ValidityConstraint within the c:ValidityContext element, then the explicit context, the c:xPathText value, can imply that multiple items must be checked and each must return "true" in order for an IEP to pass the c:ValidToXPath constraint.

When employing c:ValidToXPath the following rule applies:

Rule 5-43. Validating an XPath Expression

[Rule 5-43] (IEP) (Constraint)

Within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) with a c:xPathText attribute owned by a c:ValidToXPath element, a candidate IEP is a valid IEP, ONLY IF the value of c:ValidToXPath applied to the candidate IEP (an [XML document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_document)) has an effective Boolean value (EBV) equal to true. EBV is defined by [[W3C XPath 2.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XPath), [§2.4.3, Effective Boolean Value](http://www.w3.org/TR/2010/REC-xpath20-20101214/#id-ebv).

**context** ModelPackageDescription **inv:**

Probably some model adjustment here to make the concept of validToXPath more clear.

###### MPD3 [Rule 5-46] (IEPD). IEPD Has Conformance Assertion

This section discusses a [conformance assertion](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_assertion) in the context of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). However, this artifact may also be useful to other classes of MPDs.

Independent authors build NIEM IEPDs from NIEM [reference schema document sets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_reference_schema_document_set). Presently, a formal NIEM conformance certification process for IEPDs does not exist. Therefore, this specification requires that an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) contain an artifact that asserts NIEM conformance and provides a small amount of information to support such.

[Definition: *conformance assertion*]

An artifact that provides a declaration that an MPD conforms to relevant NIEM specifications and associated rules, including [[NIEM Conformance 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-Conformance), [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR), [[NIEM Conformance Targets Attribute Specification 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-CTAS) and [[NIEM MPD Specification 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-MPD) (this NIEM MPD Specification).

Rule 5-46. IEPD Has Conformance Assertion

[Rule 5-46] (IEPD) (Constraint)

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) MUST contain a [conformance assertion](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_assertion) artifact that is identified by a c:ConformanceAssertion element in its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document).

A [conformance assertion](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_assertion) provides information to increase the level of confidence that an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) was checked for NIEM conformance and quality. It does NOT constitute a guarantee or contract. In fact, a conformance assertion can be self-asserted.

In the absence of a formal NIEM certification process, both weak and strong conformance assertions will exist. An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) user or implementer (who is not the author) must decide his/her level of confidence in the assertion. A self-signed artifact that simply claims an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is NIEM-conformant may be considered weak. On the other hand, a stronger self-assertion could provide information that may include (but is not limited to):

* Date of assertion
* URI of the MPD claiming NIEM conformance
* Assertion of NIEM conformance
* Author (name and/or organization, or sponsoring entity; indication of NIEM and XML background or experience)
* Certifier (may be the author or another person/organization)
* Details of conformance verification:
  + How, what, and/or who? (e.g., automatic checks, manual checks, other reviews?)
  + Tool(s) employed? (e.g., tool, version, how used, on what, etc.)
  + Results? (e.g., issues, pass/fails, warnings, confirmations, etc.)

Inclusion of a [conformance assertion](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_assertion) made by a reputable, independent, trusted entity (person or organization) would likely increase confidence in conformance. Another strong case can be made by supplementing a conformance assertion with a formal conformance test report or similar artifact. The MPD catalog schema document provides a c:ConformanceReport element to identify a conformance report if one is present.

In the future, as NIEM procedures and tools advance, a conformance or quality report and a corresponding certificate may become required artifacts. A tool might check conformance and issue the report and certificate together as a digitally signed and hashed artifact that reports conformance, and proves both author and [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) identity (i.e., that the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is an unaltered copy of the original). For now, inclusion of an informal [conformance assertion](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_assertion) artifact in an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is the only requirement.

**context** ModelPackageDescription **inv:**

self.clientDependency->select(d|d.oclIsKindOf(ConformanceAssertion))->notEmpty()

###### MPD3 [Rule 5-4] (MPD-catalog). MPD Catalog Extension XML Catalog Document Name Is mpd-catalog-extension-xml-catalog.xml

Rule 5-4. MPD Catalog Extension XML Catalog Document Name Is mpd-catalog-extension-xml-catalog.xml

[Rule 5-4] (MPD-catalog) (Constraint)

An MPD catalog extension XML catalog document MUST bear the file name (and type) mpd-catalog-extension-xml-catalog.xml.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-5] (MPD-catalog). MPD Catalog Extension XML Catalog Document Resolves Namespaces to URIs

Rule 5-5. MPD Catalog Extension XML Catalog Document Resolves Namespaces to URIs

[Rule 5-5] (MPD-catalog) (Constraint)

An MPD catalog extension XML catalog document MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) all MPD catalog schema extension document namespaces to the correct corresponding local URIs in the MPD.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-6] (MPD-catalog). MPD Catalog Extension Schema Document Conforms to NDR Extension Rules

Rule 5-6. MPD Catalog Extension Schema Document Conforms to NDR Extension Rules

[Rule 5-6] (MPD-catalog) (Constraint)

An MPD catalog extension schema document MUST conform to the [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) extension schema conformance target rules.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-7] (MPD-catalog). MPD Catalog Schema and Its Extensions Conform to NDR Schema Set Rules

Rule 5-7. MPD Catalog Schema and Its Extensions Conform to NDR Schema Set Rules

[Rule 5-7] (MPD-catalog) (Constraint)

Within an MPD, the schema set formed by mpd-catalog-3.0.xsd and all MPD catalog extension schema documents MUST conform to the [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) schema set conformance target rules.

Whether extending an MPD catalog with new metadata elements, artifact classifier elements, or validity constraint elements, [Appendix A, *MPD Catalog XML Schema Document*, below,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A) provides an abstract element as a substitution group head in each case. The user simply derives a new type (through extension or restriction), or reuses an existing type, then declares a new element (of that type), and identifies it with the appropriate substitution group. Whenever possible, the user should reuse types, elements, and attributes that are already defined/declared within the [Appendix A, *MPD Catalog XML Schema Document*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_A).

If an MPD catalog schema document extension uses NIEM [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) that are not already contained in the NIEM Core subset provided with [[NIEM MPD Toolkit]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-MPD-Toolkit), then the additional components must be additive.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-8] (MPD-catalog). MPD Schema Document Extension Support Schemas Are Supersets of Spec Subsets

Rule 5-8. MPD Schema Document Extension Support Schemas Are Supersets of Spec Subsets

[Rule 5-8] (MPD-catalog) (Constraint)

Subset schema documents provided to support an MPD schema document extension MUST be a superset of the subset schema documents provided with this specification to support the MPD catalog schema document.

**context** ModelPackageDescription **inv:**

Concept and constraint not implemented in NIEM-UML.

###### MPD3 [Rule 5-9] (WF-MPD). MPD Class Determined by Conformance Target Identifier in c:mpdClassURIList

An [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) is a [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) to which the given [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) claims to conform. The MPD catalog c:mpdClassURIList attribute declares a list of [conformance target identifiers](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier), identifying the [conformance targets](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target) to which the MPD claims to conform. The following rule establishes the [class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) of an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description):

Rule 5-9. MPD Class Determined by Conformance Target Identifier in c:mpdClassURIList

[Rule 5-9] (WF-MPD) (Interpretation)

An MPD MUST have an [MPD class](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_class) of a [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) if and only if that [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) appears in the c:mpdClassURIList attribute within its [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document).

It should be clear that an [MPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) that is an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) should have a value for its c:mpdClassURIList attribute that contains both [conformance target identifiers](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) above. In the future, additional conformance target identifiers will be assigned by other appropriate NIEM specifications that specialize MPDs (for example, releases, domain updates, core updates, etc.).

The c:mpdClassURIList attribute is an XML list type that may declare that an MPD conforms to multiple conformance targets. An MPD developer can establish a new MPD [conformance target](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target) identifier in addition to those provided by this and other NIEM specifications. The identifier represents the new conformance target which should be associated with one or more rules or constraints to which an MPD must conform if it is assigned that identifier.

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authoring organization might use another classification system for its IEPDs. For example, the organization *ABC* might establish the [conformance target identifier](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_conformance_target_identifier) http://example.org/niem-iepd/1.0/#abc-org to indicate its IEPDs also conform to its own stricter set of [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) conformance rules.

**context** ModelPackageDescription **inv:**

Constraint realized during provisioning of MPD Catalog.

###### MPD3 [Rule 6-1] (WF-MPD). Wantlist Location

A NIEM schema document subset is often associated with a NIEM *wantlist*. A *wantlist* is an abbreviated XML representation of a NIEM schema document subset, and identifies only the [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) a user selected (as requirements) to build a schema document subset. To reconstruct the complete schema document subset there are usually a number of additional [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) that the user selections depend upon. These must be computed from the appropriate NIEM reference model and added to reconstruct the complete schema document subset. For example, a user may select nc:Person for the subset. In this case, the wantlist will only contain that component, but the associated full subset must contain both nc:Person and nc:PersonType. A software tool that understands how to process NIEM wantlists and schema document subsets (such as the NIEM Schema Subset Generator Tool [[NIEM SSGT]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-SSGT)) can rebuild an accurate schema document subset from a wantlist (and the reverse).

[Definition: *NIEM wantlist*]

An XML document that represents a complete NIEM schema document subset.

A NIEM wantlist identifies the [data component](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) requirements declared by the subset author; it does not identify the [data component](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component) dependencies required to reconstitute the complete subset. The complete subset can be computed with the reference schema document set from which the subset was derived.

A wantlist is always associated with a schema document subset. A wantlist may also be associated with a [constraint schema document set](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_constraint_schema_document_set), because constraint schema documents are often built from a [schema document subset](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_schema_document_subset). For a simple [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation), it can sometimes be trivial to identify a single schema document subset. However, this MPD Specification does not prohibit building complex IEPDs that contain schema document sets supported by multiple schema document subsets and associated wantlists. As with other complex cases, the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author is responsible to clearly document the associations between wantlists and schema document sets. In order to maintain a minimal degree of consistency for placement of a wantlist within an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) the following rule applies.

Rule 6-1. Wantlist Location

[Rule 6-1] (WF-MPD) (Constraint)

If present, a NIEM wantlist MUST reside within the root of the MPD subdirectory that groups and defines its corresponding subset schema document set (e.g., niem).

**context** ModelPackageDescription **inv:**

self.oclAsType(InstanceSpecification).clientDependency->select(d|d.stereotypedBy('WantList'))->notEmpty()

###### MPD3 [Rule 7-1] (WF-MPD). MPD Is a ZIP File

An MPD is a logical set of electronic files aggregated and organized to fulfill a specific purpose in NIEM. Directory organization and packaging of an MPD should be designed around major themes in NIEM: reuse, sharing, interoperability, and efficiency.

Rule 7-1. MPD Is a ZIP File

[Rule 7-1] (WF-MPD) (Constraint)

An MPD is packaged as a single [ZIP file](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_ZIP_file) that represents a sub-tree of a file system. This archive MUST preserve and store the logical directory structure intended by its author.

**context** ModelPackageDescription **inv:**

Rule enforcement is provided by provisioning, which produces the zip file.

###### MPD3 [Rule 7-2] (WF-MPD). XSD and XML Documents Conform to Applicable NDR Conformance Targets

NIEM XSD and XML artifacts in an MPD must be valid for both XML Schema and NIEM. This also implies these artifacts must adhere to applicable [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) conformance target rules.

Rule 7-2. XSD and XML Documents Conform to Applicable NDR Conformance Targets

[Rule 7-2] (WF-MPD) (Constraint)

Within an MPD archive, each XML schema document (XSD) or instance XML document (XML) artifact that uses a conformance targets attribute (as defined by [[NIEM Conformance Targets Attribute Specification 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-CTAS)) MUST satisfy the [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) rules for the conformance targets it declares.

**context** ModelPackageDescription **inv:**

Rule enforcement is provided by provisioning and NDR/MPD rules expressed in OCL and applied to the NIEM-UML model.

###### MPD3 [Rule 7-3] (WF-MPD). MPD Archive Uncompresses to a Single Root Directory

NIEM releases, core updates, and domain updates maintain a relatively consistent directory organization [[NIEM Domain Update Specification 1.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-DomainUpdate). But there are many ways to organize [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) directories that may depend on a number of factors including (not limited to) business purpose and complexity. For this reason, strict rules for [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) directory structure are difficult to establish. Therefore, [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors may create their own logical directory structures subject to the rules of this section.

[Definition: *MPD root directory*]

The top level file directory relative to all MPD artifacts and subdirectories.

Rule 7-3. MPD Archive Uncompresses to a Single Root Directory

[Rule 7-3] (WF-MPD) (Constraint)

An MPD archive MUST uncompress (unzip) to one and only one [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory).

The foregoing rule ensures that:

* Unpacking an MPD archive will not scatter its contents on a storage device.
* A common starting point always exists to explore or use any MPD.
* mpd-catalog and change log artifacts will always be found in the [MPD root directory](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_root_directory).

**context** ModelPackageDescription **inv:**

Rule enforcement is provided by provisioning.

###### MPD3 [Rule 7-5] (IEPD). IEPD File Name Syntax

Additional non-normative for directory naming and organization for IEPDs is in [Appendix E, *Guidance for IEPD Directories (non-normative)*, below](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#appendix_E).

It is important to understand that this section does not apply to the syntax for the c:mpdName attribute in the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document). Refer to [Section 5.2.1, *MPD Name Syntax (c:mpdName)*, above,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#section_5.2.1) for details regarding the c:mpdName metadata attribute.

The MPD Specification is intended to help facilitate tool support for processing MPDs. Providing tools and search mechanisms the basic information about an MPD as early as possible will help reduce processing time and complexity. So, if an MPD name, version, and class can be identified from its file name, then a tool would not have to open the archive and parse the MPD catalog to determine such. Of course, to do anything useful, a tool will eventually have to open the MPD archive. However, standard file name syntax allows a tool to search through a set of MPDs to find a particular MPD name, version, or class without having to open each. File name consistency can also make it easier to scan and identify MPDs in a long list sorted by file name.

Rule 7-5. IEPD File Name Syntax

[Rule 7-5] (IEPD) (Constraint)

The file name of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) (iepd-filename) SHOULD adhere to the syntax defined by the regular expression:

iepd-filename ::= name '-' version '.iepd.zip'

Where:

name ::= alphanum ((alphanum | special)\* alphanum)?

alphanum ::= [a-z0-9]

special ::= '.' | '-' | '\_'

version ::= digit+ ('.' digit+)\* (status digit+)?

digit ::= [0-9]

status ::= 'alpha' | 'beta' | 'rc' | 'rev'

The status values are as defined in [Rule 5-10, *MPD Version Number Syntax*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_5-10).

Regular expression notation in the rule above is from [[W3C XML 1.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#W3-XML) [#sec-notation](http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation).

Alphabetic characters are lower case to reduce complications across various file systems.

An example of an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) file name that follows this rule is: abc-query-2.0beta1.iepd.zip

File names can easily be changed by a person or process that executes a download on the Internet. Nonetheless, [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) authors and publishers should ensure that their application of [Rule 7-5, *IEPD File Name Syntax*, above,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_7-5) is consistent with an IEPD’s catalog. The basic metadata in the [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) (e.g., IEPD name, version, class, URI, etc.) should match any such information incorporated into the file name.

**context** ModelPackageDescription **inv:**

Packaging constraints are resolved by PSM-MPD transformations.

###### MPD3 [Rule 7-6] (WF-MPD). MPD Reference to Resource Uses Common URI Scheme

It is important to understand that the URI scheme defined in [Section 5.2.4.2, *URI Scheme for MPD Artifacts (c:externalURI)*, above,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#section_5.2.4.2) can only be used to identify relationships among and provide source links to external schemas being reused. It is not sufficient to allow references or links to such schemas stand in for a physical copy. Thus, all schema artifacts necessary to define, validate, and use an MPD must be physically present within that MPD. In accordance with the [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR), if MPD schemas are moved to an operational environment for implementation, validation, or other purposes, then absolute references may replace relative [path name](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_path_name) references when needed. The following rule applies when absolute references to Internet resources are required.

Rule 7-6. MPD Reference to Resource Uses Common URI Scheme

[Rule 7-6] (WF-MPD) (Constraint)

An absolute reference to an Internet resource MUST use a well-known URI scheme (e.g., http, https, ftp, ftps) and MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI). If applicable, documentation SHOULD describe how to [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) with security, account, and/or password information.

**context** ModelPackageDescription **inv:**

Constraints on URIs are partially satisfied by specific URI Constraints  
 expressed elsewhere in the NDR and MPD. For URI references embedded elsewhere in the  
 model, it would be difficult to express the constraint in OCL. This constraint must  
 be manually resolved by the modeler.

###### MPD3 [Rule 7-7] (WF-MPD). IEPD Completeness

Since an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) defines an information exchange and is often implemented by persons other than the original author, it is important to ensure that they are relatively complete and provide all artifacts needed to use the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation).

Rule 7-7. IEPD Completeness

[Rule 7-7] (IEPD) (Constraint)

An [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) SHOULD contain all artifacts needed to understand it and facilitate its correct implementation.

The rule above means that an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) implementer should not be forced to search for or track down specialized schema documents, documentation, or other artifacts required to validate and implement exchanges defined by an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). Specialized artifacts refer to those designed and built by an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) author, not artifacts that are standards and publicly available to all implementers. For example, this rule does not imply that an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) should contain a schema document that defines the XML schema component vocabulary identified by the namespace name http://www.w3.org/2001/XMLSchema (i.e., XS), or http://www.w3.org/2001/XMLSchema-instance (i.e., XSI). All schema processors have appropriate declarations for these built in. Likewise, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is not required to contain mpd-catalog-3.0.xsd or the standard NIEM subset that supports it.

On the other hand, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) whose author has extended the MPD catalog schema is clearly required to contain the catalog extension schema document, since this is a specialized customization created by the author. If a different NIEM schema subset is also used, then the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) must also contain its superset (i.e., a complete subset that incorporates both the original subset with additional NIEM components used to extend the catalog schema document; see [Rule 5-8, *MPD Schema Document Extension Support Schemas Are Supersets of Spec Subsets*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_5-8).)

The rationale for "SHOULD" in [Rule 7-7, *IEPD Completeness*, above,](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_7-7) relates to issues of security. Although NIEM is generally public, some IEPDs (and even other MPDs) may contain XML tags that provide more semantics or structure than a domain is willing to expose. In such cases, it may be necessary to simply refer to schema documents that are required for validation and implementation, instead of circulating them within a public [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). Implementers would then be expected to know how and where to obtain the required documents.

The [[NIEM NDR 3.0]](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#NIEM-NDR) explains how NIEM employs adapter types to encapsulate and use other standards (e.g., geospatial and emergency management standards) in their native forms that are not NIEM-conformant. Other standards may use xs:import without requiring schemaLocation attributes (instead, relying only on the namespace value). These standards may also use xs:include. This XML Schema construct is disallowed by NIEM.

**context** ModelPackageDescription **inv:**

This constraint is resolved by PSM-MPD transformations.

###### MPD3 [Rule 7-8] (WF-MPD). MPD External Schema Documents Are Local Resources

When standards external to NIEM are required within MPDs, the following rule applies:

Rule 7-8. MPD External Schema Documents Are Local Resources

[Rule 7-8] (WF-MPD) (Constraint)

Within an MPD, a non-NIEM-conformant external schema document reference to another schema document and/or namespace MUST [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to a local resource. schemaLocation attributes or XML catalogs can be used to ensure resolution.

For the case of non-NIEM-conformant schemas, this rule ensures that all schemas (or corresponding artifacts and namespaces) from external standards required for definition, validation, and use of the MPD are present within the archive.

**context** ModelPackageDescription **inv:**

This constraint is resolved by the NIEM-UML Model, which requires all InformationModels to be defined, and PSM-MPD transformations which enforce schemaLocations to be local.

###### MPD3 [Rule 7-9] (WF-MPD). Key MPD Resources Are Local Resources

XML schemas are the heart of MPDs since they formally specify normative structure and semantics for [data components](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_data_component). However, in general, an MPD is a closed set of artifacts. This means that all hyperlink references within artifacts should [resolve](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_resolve_URI) to the appropriate artifact.

Rule 7-9. Key MPD Resources Are Local Resources

[Rule 7-9] (WF-MPD) (Constraint)

Within any artifact of an MPD archive, any direct reference to another resource (i.e., another artifact such as an image, schema, stylesheet, etc.) that is required to process or display an artifact SHOULD exist within the archive at the location specified by that reference.

This means that MPD artifacts, including documentation artifacts, should be complete. For example, if an HTML document within an MPD contains a hyperlink reference (href) to an artifact that is part of or used by the MPD, then the file associated with that hyperlink should be present in the MPD; likewise for a sourced (src) image. Authors should exercise good judgment with this rule. For example, it does not require an MPD to contain copies of all cited documents from a table of references if it contains hyperlinks to those documents. The key operating words in this rule are: "another resource is required to process or display an artifact SHOULD exist within the archive."

In some cases, it may not be possible to include all artifacts, even schemas, in an MPD without violating laws, regulations, or policies. For example, an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) may require use of a schema document that is not publicly accessible; it might be classified or controlled unclassified information (CUI). This is a valid reason for exception to [Rule 7-9, *Key MPD Resources Are Local Resources*, above](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#rule_7-9). If the [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation) is placed in the public domain, the author should omit the non-public schema document, and if appropriate, document the omission, and explain where and/or how the missing schema document can be obtained.

**context** ModelPackageDescription **inv:**

This constraint is resolved by provisioning; all generated artifacts are local to the MPD and all references between them are relative.

### <Artifact> [OrganizationType](#_9e5e3792b05eaa89037072cf187ea5a3)

A data type for a body of people organized for a particular purpose.

##### Properties

###### OrganizationPrimaryContactInformation

A preferred means of contacting an organization.

|  |  |
| --- | --- |
| **Type** | ContactInformationType |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [PersonType](#_c13d446160fd08efc7b471ace1b401f2)

Represents an AuthoritativeSource for the MPD corresponding to a niem-core:PersonType.   In this case, an InstanceSpecification of PersonType is mapped to the MPD Catalog element c:MPDInformationType/c:AuhoritativeSource/niem-core:EntityPerson (whose type is niem-core:PersonType).

An InstanceSpecification of PersonType may also represent a ContactEntity within a ContactInformatiionType.  In this case, the instance of PersonType is mapped to the MPD Catalog element .../niem-core:EntityOrganization/niem-core:OrganizationPrimaryContactInformation/niem-core:ContactEntity/niem-core:EntityPerson (whose type is niem-core:PersonType).

An InstanceSpecification of PersonType may also represent a ContactResponder within a ContactInformatiionType.  In this case, the instance of PersonType is mapped to the MPD Catalog element .../niem-core:EntityOrganization/niem-core:OrganizationPrimaryContactInformation/niem-core:ContactResponder/niem-core:EntityPerson (whose type is niem-core:PersonType).

### <Artifact> [QualifiedNamesType](#_791eceae6b47d8ad81d2fe6c4dfccc60)

A data type for a set of qualified names.

##### Properties

###### qualifiedNameList

A list of qualified names.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [RelaxNGValidationType](#_f27689b94a45f8fa6e44dbf83830f403)

A data type for a RelaxNG validation constraint, indicating a RelaxNG schema document against which an artifact may be validated, as well as a description of the validation roots for assessment of validity.

### <Artifact> [SchemaDocumentSet](#_655c98e4012da39f545c761c140df3a5)

A conformant schema document set is a set of schema documents that are capable of validating XML docments.

[Definition: *conformant schema document set*]

A **conformant schema document set** is a collection of schema documents that together are capable of validating a conformant instance XML document. It is a conformance target of this specification. A conformant schema document set MUST conform to all rules of this specification that apply to this conformance target.

A conformance schema document set has strong dependencies on reference schema documents and extension schema documents. Without the guarantees provided by those conformance targets, the rules for a conformant schema document set would not be helpful. Those schemas in a schema set that are marked as reference or extension schemas are required to conform to the appropriate conformance targets.

An MPD artifact set that may include subset schema documents, extension and external schema documents, and other supporting artifacts.

### <Artifact> [SchemaDocumentSetType](#_03c988919690710762e6cd5069e6c488)

A data type for an MPD artifact set that may include subset schema documents, extension schema documents, and external schema documents or constraint schema documents.

##### Constraints

###### MPD3 [Rule 7-4] (MPD-catalog). Constraint on Elements of Type c:SchemaDocumentSetType

In order to accommodate the [model package description](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_model_package_description) concept, the design of the MPD catalog schema does not enforce a rule that is required to ensure a c:SchemaDocumentSet within an [MPD catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_MPD_catalog_document) is used correctly for an [IEPD](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_information_exchange_package_documentation). This rule assumes that within the IEPD’s MPD catalog any c:SchemaDocumentSet element identifies [XML schema documents](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) to be assembled into an [XML Schema](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_Schema).

Rule 7-4. Constraint on Elements of Type c:SchemaDocumentSetType

[Rule 7-4] (MPD-catalog) (Constraint)

An element information item with a type definition validly derived from c:SchemaDocumentSetType MUST have a child element with an element declaration that is in the substitution group of c:XMLCatalog or c:XMLSchemaDocument.

This rule ensures that a c:SchemaDocumentSet element always has at least one child element that is an [XML catalog document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_catalog_document) (which itself defines an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document)) set, or an [XML schema document](http://reference.niem.gov/niem/specification/model-package-description/3.0/model-package-description-3.0.html#definition_XML_schema_document) (which constitutes a set of at least one schema document). This rule cannot be enforced within the MPD catalog schema without introducing a UPA error, but it could be enforced by a Schematron rule.

**context** SchemaDocumentSetType **inv:**

self.oclAsType(InstanceSpecification).clientDependency->select(d|d.stereotypedBy('XMLSchemaDocument')or d.stereotypedBy('XMLCatalog'))->notEmpty()

### <Artifact> [SchematronValidationType](#_872765b097ba96ae7fbfa6a1033c18f7)

A data type for a Schematron validation constraint, indicating a Schematron schema document against which an artifact may be validated as well as a description of the validation roots for assessment of validity.

### <Artifact> [TextRuleType](#_cba3cc9d607874f9cf56eb2d996e7440)

A data type for a rule drafted in a human language.

##### Properties

###### RuleText

A rule written in a human language.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [ValidityConstraintType](#_9863680455d9ae6e5de2db10c1a2edbc)

A data concept for a rule or instructions for validating an IEP candidate.

### <Artifact> [ValidityConstraintWithContextType](#_dc6a1ac72032832f049a1273df5c4b46)

A data concept for a rule or instructions for validating an IEP candidate (XML document) using some context within that XML document.

### <Artifact> [ValidityContextType](#_7edeea2403c320db2be8b8f5239f6642)

A data type for a rule or instructions for validating an IEP candidate within context defined by an XPath expression.

##### Properties

###### ValidityConstraint

A data concept for a rule or instructions for validating an IEP candidate.

|  |  |
| --- | --- |
| **Type** | ValidityConstraintType |
| **Multiplicity** | 1..\* |
| **Ordering** |  |
| **Composition** | composite |

###### xPathText

An XPath expression.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

### <Artifact> [XMLSchemaType](#_5e48a7ce22c2dd0be5cc1ed17c6e3f38)

A data type for a validity constraint that indicating an XML Schema against which an artifact may be validated, or which can be used for other purposes. c:XMLSchemaDocument identifies the root or starting XML schema document.

### <Artifact> [XPathType](#_d9cef304ec90b0b2215e77012b9a3829)

A data type for an XPath expression.

##### Properties

###### xPathText

An XPath expression.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | composite |

### <Enumeration> [ChangeCodeSimpleType](#_4da536014dc3338ea6d5abb73de4cfd6)

Purpose of change.

##### Literals

###### new\_requirement

###### bug\_fix

###### refactoring

###### harmonization

###### general\_improvement

### <Enumeration> [ModelPackageDescriptionClassCode](#_ed6920419309a280640605a48ca910a1)

A specified classification (type or kind) of the MPD.

The MPD specification applies to all NIEM model package descriptions (MPDs).   Currently, MPDs include the following:

* A NIEM information exchange package documentation (IEPD) that defines a NIEM data exchange.
* A NIEM release (including a major, minor, or micro release) [as defined in the NIEM High-Level Version Architecture 1.0].
* A NIEM domain update [as described in NIEM Domain Update Specification 1.0]. (Note these are NOT the same as a NIEM domain schema document that is a part of a NIEM release).
* A NIEM core update to a NIEM release.
* A NIEM Enterprise Information Exchange Model (EIEM) on which one or more IEPDs can be based.

Of these kinds of MPDs, the only kind which is formally specified in NIEM-3 is an IEPD.  The NIEM-3 UML Models all kinds of MPD, and the kind is defined as the EnumerationLiterals of this ModelPackageDescriptionClassCode Enumeration.

The kind of MPD is reflected in the MPD Catalog c:mpdClassURIList attribute.  That attribute will be provisioned with the appropriate list of URIs based on the value of this ModelPackageDescriptionClassCode Enumeration.

##### Literals

###### eiem

An Enterprise Information Exchange Model (EIEM) is an MPD that incorporates BIECs that meet enterprise business needs for exchanging data using NIEM [NIEM-BIEC]. An EIEM is an adaptation of NIEM schemas, tailored and constrained for and by an enterprise. An EIEM will contain the following schemas that are commonly used or expected to be used by the authoring enterprise: one standard NIEM schema subset and one or more NIEM extension schemas that extend existing NIEM data components or establish new data components.

###### iepd

NIEM Information Exchange Package Documentation (IEPD) is an MPD that defines a recurring XML data exchange. An NIEM IEPD is a set of valid XML schemas that may include portions of NIEM Core schemas, portions of NIEM Domain schemas, enterprise-specific or IEPD-specific extension schemas, and at least one exchange schema that defines a document element (as defined in [W3-XML-InfoSet]). The schemas contained in an IEPD work together to define a class of XML instances that consistently encapsulate data for information exchanges. Each XML instance in this class validates against the set of XML schemas contained within the IEPD.

gainst the set of XML schemas contained within the IEPD.

###### core\_update

When necessary, the NIEM PMO can publish a core update. This is essentially identical to a domain update in terms of structure and use, with two important exceptions. First, a core update records changes that apply to a particular NIEM core version or another core update. This also means it is applicable to all NIEM releases using that same core version. Second, a core update is never published to replace a NIEM core. It is intended to add new schemas, new data components, new code values, etc. to a core without waiting for the next major release. In some cases, minor modifications to existing data components are possible.

modifications to existing data components are possible.

###### release

A NIEM release is an MPD containing a full set of harmonized reference schemas that coherently define all content within a single version of NIEM. NIEM releases include major, minor, and micro releases (as defined in the NIEM High Level Version Architecture (HLVA)).

tecture (HLVA)).

###### domain\_update

A domain update is an MPD containing reference schemas that represent changes to NIEM domains. The [NIEM-HLVA] defines a domain update as both a process and a NIEM product. Through use and analysis of NIEM releases and published content, domain users will identify issues and new data requirements for the domain and sometimes Core. NIEM domains use these issues as the basis for incremental improvements, extensions, and proposed changes to future NIEM releases. Both the process and product of the process are referred to as domain update.

of the process are referred to as domain update.

### <Enumeration> [RelationshipCode](#_9fa33413ed68ab4977db181d70e0d689)

The possible reasons for the connectedness between the MPDs or between an MPD and a resource. This enumeration defines the possible values for the relationshipCode attribute of the ModelPackageDescriptionRelationship stereotype. [Note these NIEM-UML enumeration literals differ from the NIEM MPD Specification v1.0 in that they use underscore ("\_") instead of dash ("-"). This is due to issues with dashes in some UML tools.] Reference Section 4.2.5 and Appendix B of NIEM MPD Specification v1.0 (http://reference.niem.gov/niem/specification/model-package-description/1.0/).

##### Literals

###### updates

A relationshipCode value for indicating that this MPD is an incremental update to the resource referenced in resourceURI. Used by a core or domain update to identify the domain schema in a NIEM release being incrementally updated (not replaced).

###### conforms\_to

A relationshipCode value for indicating that this MPD conforms to the specification or standard referenced in resourceURI.URI.l>

###### version\_of

A relationshipCode value for indicating that this MPD is a different version of the MPD referenced in resourceURI. This code value is only needed in cases where significant name changes might obscure the relationship to the previous version. For example, NIEM Justice 4.1 is a version of GJXDM 3.0.3.ion of GJXDM 3.0.3.l>

###### specializes

A relationshipCode value for indicating that this MPD is a specialization of the MPD referenced in resourceURI. This value is the inverse of generalizes.eralizes.l>

###### generalizes

A relationshipCode value for indicating that this MPD is a generalization of the MPD referenced in resourceURI. This value is the inverse of specializes.cializes.l>

###### supersedes

A relationshipCode value for indicating that this MPD replaces the MPD referenced in resourceURI.URI.l>

###### deprecates

A relationshipCode value for indicating that content in this MPD is preferred over content in the MPD referenced in resourceURI; and at some time in the future will supersede the MPD referenced in resourceURIsourceURIl>

###### adapts

A relationshipCode value for indicating that this MPD is an adaptation of the MPD referenced in resourceURI.URI.l>

###### derives\_from

A relationshipCode value for indicating that this MPD has been derived from another; used to indicate an IEPD is derived from an EIEM (may have other uses as well)

A relationshipCode value for indicating that this MPD has been derived from another; used to indicate an IEPD is derived from an EIEM (may have other uses as well)

## Profile : NIEM\_Common\_Profile

### Overview

The NIEM Common Profile comprises stereotypes that are used in both the NIEM PIM  
 Profile and the NIEM PSM Profile. In addition, the UML metamodel subset covered by the  
 NIEM Common Profile also includes the metaclasses PrimitiveType, Enumeration,  
 EnumerationLiteral, Property and Generalization, even though they are not specifically  
 extended by any stereotypes in the profile.

### <Stereotype> [AdapterType](#_f1dc99cf7a92fbe72e77380ed5c138c2)

##### Generalization

##### [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)Description

An AdapterType is a NIEMType Class that represents a NIEM adapter type. A NIEM adapter type is a NIEM object type that adapts external components for use within NIEM. External components are not NIEM-conforming (e.g., data components from other standards, e.g. GML, ISO, etc.). An adapter type creates a new class of object that embodies a single concept composed of external components.

AdapterType is implemented in XML Schema as a complex type definition with complex content. Section 3.4 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses complex type definitions in XML Schema; Section 7.7 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) addresses adapter types in NIEM-conformant XML Schema.

NIEM NDR v1.3 addresses adapter types in NIEM-conformant XML Schema.body>

##### Constraints

###### NDR3 [Rule 10-11] (REF,EXT). External adapter type not a base type

[[NDR] Rule 10-11](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-11), External adapter type not a base type (REF, EXT): Section 10.2.3.2, External adapter types

**context** AdapterType **inv:**

self.base\_Class.\_directedRelationshipOfTarget->select(t|t.oclIsKindOf(Generalization) or t.stereotypedBy('Restriction'))->size()=0

###### NDR3 [Rule 10-12] (SET). External adapter type not a base type

[[NDR] Rule 10-12](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-12), External adapter type not a base type (SET): Section 10.2.3.2, External adapter types

**context** AdapterType **inv:**

self.base\_Class.\_directedRelationshipOfTarget->select(t|t.oclIsKindOf(Generalization) or t.stereotypedBy('Restriction'))->size()=0

###### NDR3 [Rule 10-69] (REF). External adapter type indicator annotates complex type

[[NDR] Rule 10-69](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-69). External adapter type indicator annotates complex type

**context** AdapterType **inv:**

This constraint realized by provisioning:  
A Class stereotyped as AdapterType will result in production of appinfo:externalAdapterTypeIndicator attribute on the xs:complexType representing the AdapterType.

###### NDR3 [Rule 10-8] (REF,EXT). External adapter type has indicator

[[NDR] Rule 10-8](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-8), External adapter type has indicator (REF, EXT): Section 10.2.3.2, External adapter types.

**context** AdapterType **inv:**

The constraint is resolved during provisioning:  
An AdapterType and only an AdapterType has the appinfo:externalAdapterTypeIndicator set to a value of true.

###### NDR3 [Rule 10-9] (REF,EXT). Structure of external adapter type definition follows pattern

[[NDR] Rule 10-9](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-9), Structure of external adapter type definition follows pattern (REF, EXT): Section 10.2.3.2, External adapter types.

**context** AdapterType **inv:**

self.base\_Class.general->isEmpty()  
and  
self.base\_Class.clientDependency->select(d|d.stereotypedBy('Restriction'))->isEmpty()

### <Stereotype> [AssociationType](#_837c9f834aece85a107a3143b6b5a0bc)

##### Generalization

##### [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)Description

AssociationType is a NIEMType class that represents a NIEM association type. A NIEM association type establishes a relationship between objects, along with the properties of that relationship. A NIEM association is an instance of an association type. Associations are used when a simple NIEM property is insufficient to model the relationship clearly and when properties of a UML Association or AssociationClass may not necessarily be sufficient to reflect the variability of a NIEM association. Consequently, the AssociationType Stereotype is applied to a UML Class. Since an AssociationClass is also a Class, the AssociationType Stereotype may be applied to a UML AssociationClass where appropriate.  Note that a UML AssociationClass specializing another AssociationClass must have the same number of ends as the other AssociationClass and must have at least two ends. This UML constraint prevents the usage of AssociationClass to model abstract NIEM association types that are intended to be extended by subtypes with additional ends. A UML AssociationClass can specialize an abstract UML Class. AssociationType is implemented in XML Schema as a complex type definition with complex content. Section 3.4 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses complex type definitions in XML Schema; [Section 10.3.1](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#section_10.3.1) of [NIEM-NDR] addresses association types in NIEM-conformant XML Schema.

##### Constraints

###### NDR3 [Rule 10-19](REF,EXT). Association types is derived from association type

[[NDR] Rule 10-19](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-19), Association types is derived from association type (REF, EXT): Section 10.3.1, Association types

**context** AssociationType **inv:**

(  
 (  
 self.stereotypedBy('AssociationType') or self.oclIsKindOf(AssociationClass)  
 )  
 implies   
 (  
 self.name.endsWith('AssociationType')  
 and  
 self.general  
 ->union(self.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->forAll(c|c.stereotypedBy('AssociationType') or c.oclIsKindOf(AssociationClass))   
 )  
)  
and  
(  
 self.name.endsWith('AssociationType')  
 implies  
 self.stereotypedBy('AssociationType') or self.oclIsKindOf(AssociationClass)  
)

### <Stereotype> [AugmentationType](#_21c871b89ae2da4f5e1f1de3d639ddce)

##### Generalization

##### [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)Description

AugmentationType is a NIEMType Class that represents a NIEM augmentation type. A NIEM augmentation type is a complex type that provides a reusable block of data that may be added to object types or association types. An augmentation of an object type is a block of additional data that is an instance of an augmentation type, added to an object type to carry additional data beyond that of the original object definition. The applicability of an augmentation may be restricted using an AugmentationApplication Dependency or an Augments Generalization.  A Class that is the specific Classifier of an Augments Generalization shall be inferred to be an AugmentationType.

AugmentationType is implemented in XML Schema as a complex type definition with complex content. Section 3.4 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses complex type definitions in XML Schema; Section 7.4.5 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) addresses augmentation types in NIEM-conformant XML Schema.

##### Constraints

###### NDR [Rule 7-14]

**[Rule 7-14] (REF, EXT)** A component definition SHALL begin with a standard opening phrase that depends on the class of the component per Table 7-1 in NDR: Standard Opening Phrases:

**Rationale** A standard opening phrase based on component class helps to ensure consistent definitions that appropriate for the type of component item being defined. These opening phrases also provide a cue that facilitates recognition of the particular kind of component.

s recognition of the particular kind of component.

**context** AugmentationType **inv:**

self.base\_Class.ownedComment->exists(documentation|  
 documentation.stereotypedBy('Documentation') and documentation.\_'body'.startsWith('A data type that supplements ')  
 )

###### NDR [Rule 7-47]

**NDR [Rule 7-47] (REF, EXT)** Essentially states that the general of the >, if it exists, must be an >.

Due to schema and other NDR rules, this also requires

* a maximum of one general for the given >
* any subtypes of the given > must also be >

**context** AugmentationType **inv:**

(self.base\_Class.general->size()<=1)   
and self.base\_Class.general->forAll(g|g.stereotypedBy('AugmentationType'))   
and (  
 self.base\_Class.\_directedRelationshipOfTarget->forAll(r|r.oclIsKindOf(Generalization) implies r.oclAsType(Generalization).specific.stereotypedBy('AugmentationType'))  
)

###### NDR [Rule 7-48]

**[Rule 7-48] (REF, SUB, EXT)** Within the schema, an augmentation element definition:

1. SHALL have a type that is an augmentation type.

2. SHALL use the substitutionGroup attribute such that it is transitively substitutable for the element structures:Augmentation.

An element that is not an augmentation element SHALL NOT meet either of the above criteria.

**Rationale** An augmentation is trivially identifiable as such. The use of the common structures:Augmentation element allows message builders to optionally delay specifying augmentations to be applied to a type until runtime.

**context** AugmentationType **inv:**

The constraint is enforced by the transformation from PSM to XSD Schema  
 artifact. A property whose type is an  
 <<AugmentationType>> is an augmentation element. The  
 property may directly or indirectly use the UML subsettedProperty mechanism to  
 identify a substitutionGroup, which will be transitively substitutable for the  
 element structures:Augmentation.

###### NDR3 [Rule 10-30] (INS). Element within instance of augmentation type modifies base

[[NDR] Rule 10-30](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-30), Element within instance of augmentation type modifies base (INS): Section 10.4.4, Augmentation types

**context** AugmentationType **inv:**

The instance rule is outside the scope of the NIEM-UML model.

###### NDR3 [Rule 10-31] (REF,EXT). Only an augmentation type name ends in "AugmentationType"

[[NDR] Rule 10-31](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-31), Only an augmentation type name ends in AugmentationType (REF, EXT): Section 10.4.4, Augmentation types

**context** AugmentationType **inv:**

self.name.endsWith('AugmentationType') = self.stereotypedBy('AugmentationType')

###### NDR3 [Rule 10-32] (REF,EXT). Schema component with name ending in "AugmentationType" is an augmentation type

[[NDR] Rule 10-32](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-32), Schema component with name ending in AugmentationType is an augmentation type (REF, EXT): Section 10.4.4, Augmentation types

**context** AugmentationType **inv:**

self.general  
->union(self.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
->forAll(g|g.stereotypedBy('AugmentationType'))

###### NDR3 [Rule 10-33] (REF,EXT). Type derived from augmentation type is an augmentation type

[[NDR] Rule 10-33](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-33), Type derived from augmentation type is an augmentation type (REF, EXT): Section 10.4.4, Augmentation types

**context** AugmentationType **inv:**

self.\_directedRelationshipOfTarget->select(d|d.oclIsKindOf(Generalization)).oclAsType(Generalization).specific  
->union(self.supplierDependency->select(d|d.stereotypedBy('Restriction')).client->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
->forAll(g|g.stereotypedBy('AugmentationType'))

### <Stereotype> [Choice](#_83d824155dc5d9ef7bf45a97ee8a5f7d)

##### Extends

##### ClassDescription

A Choice Class groups a set of attributes whose values are mutually exclusive. That is, in any instance of a Choice Class, at most one of its attributes may be non-empty.

Choice represents the use of a choice model group in XML Schema. Section 3.8 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses choice model groups in XML Schema. Sections 6.1.8.1 and 6.1.8.2 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) address choice model groups in NIEM-conformant XML Schema.

##### Constraints

###### No Generalizations or subtypes

A Choice Class shall not participate in any Generalizations, either as the general or the special Classifier.

**context** Choice **inv:**

self.base\_Class.generalization->isEmpty()   
and self.base\_Class.\_directedRelationshipOfTarget->select(d|d.oclIsKindOf(Generalization))->isEmpty()

###### ownedAttributes have multiplicity 0..1.

The ownedAttributes of a Choice class shall have multiplicity 0..1.

**context** Choice **inv:**

self.base\_Class.attribute->forAll(a|(a.lower=0) and  
 (a.upper=1))

### <Stereotype> [Deprecated](#_6bcaa350df5d4ba5a748782050a035c2)

##### Extends

##### NamedElementDescription

A deprecated component is one whose use is not recommended. A deprecated component may be kept in a schema for support of older versions but should not be used in new efforts. A deprecated component may be removed, replaced, or renamed in a later version of a namespace.

[Definition: *deprecated component*]

A **deprecated component** is one that developers are discouraged from using, typically because a better alternative exists, yet which is maintained in the schema for compatibility with previous versions of the namespace.

##### Constraints

###### NDR3 [Rule 10-66] (REF,EXT). Component marked as deprecated is deprecated component

The appinfo schema provides a construct for indicating that a construct is deprecated. A deprecated component is one whose use is not recommended. A deprecated component may be kept in a schema for support of older versions but should not be used in new efforts. A deprecated component may be removed, replaced, or renamed in a later version of a namespace.

[Definition: deprecated component]

The appinfo schema provides a construct for indicating that a construct is deprecated. A deprecated component is one whose use is not recommended. A deprecated component may be kept in a schema for support of older versions but should not be used in new efforts. A deprecated component may be removed, replaced, or renamed in a later version of a namespace.

[Definition: *deprecated component*]

A **deprecated component** is one that developers are discouraged from using, typically because a better alternative exists, yet which is maintained in the schema for compatibility with previous versions of the namespace.

Rule 10-66. Component marked as deprecated is deprecated component

[Rule 10-66] (REF, EXT) (Interpretation)

A schema component that has an attribute appinfo:deprecated with a value of true MUST be a deprecated component.

Deprecation can allow version management to be more consistent; versions of schema may be incrementally improved without introducing validation problems and incompatibility. As XML Schema lacks a deprecation mechanism, NIEM defines such a mechanism.

**context** Deprecated **inv:**

Rule is informative. Provisioning ensure that an appinfo:deprecated maps to a model element stereotyped by Deprecated.

### <Stereotype> [Documentation](#_82bb1c940042ed82646c146a0dd34770)

##### Extends

##### CommentDescription

A Documentation Comment is the data definition of the Element that owns it.  For an Element owning only one Comment, that Comment will be inferred to be a Documentation Comment.

A Documentation Comment owned by an Element representing a NIEM type or property is implemented as a documentation element of the annotation for the corresponding type definition or property declaration.

ng type definition or property declaration.

##### Constraints

###### Max One <<Documentation>> per Element

The owner of a Documentation Comment must have no other Documentation Comments.

**context** Documentation **inv:**

self.base\_Comment.annotatedElement->notEmpty() and  
 self.base\_Comment.annotatedElement->forAll(e|e=self.base\_Comment.owningElement) and  
 (self.base\_Comment.owningElement.ownedComment->select(c|c.stereotypedBy('Documentation'))->size()=1)

### <Stereotype> [List](#_e3b514eabd36c708392ea55009da0bb2)

##### Extends

##### DataTypeDescription

A List is a DataType whose values consist of a finite length (possibly empty) sequence of values of another DataType, which is the item type of the List. A List DataType must have a single Property with multiplicity 0..\* whose type is the item type. The name of this element is not material.

A List DataType is implemented in XML schema as a list simple type definition. List represents a relationship between two simple type definitions: the first is a list simple type definition whose item type definition is the second. This relationship is implemented in XML Schema through the itemType attribute on the xsd:list element of the list simple type definition, the actual value of which resolves to the second type definition. Section 3.14 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses list simple type definitions in XML Schema; Section 7.3 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) addresses list simple type definitions in NIEM-conformant XML Schema.

##### Constraints

###### single ownedAttribute with multiplicity 0..\* typed <DataType>>

A List DataType shall have a single ownedAttribute with multiplicity 0..\* whose type is also a DataType.

**context** List **inv:**

(self.base\_DataType.attribute->size()=1)   
and  
 self.base\_DataType.attribute ->forAll(a|(a.lower=0) and (a.upper=-1))

###### NDR3 [Rule 11-6](REF,EXT). Use lists only when data is uniform

[[NDR] Rule 11-6](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-6), Use lists only when data is uniform (REF, EXT): Section 11.1.2.1, Derivation by list

**context** List **inv:**

Not currently expressed in OCL.

###### NDR3 [Rule 11-7] (REF,EXT). List item type defined by conformant schemas

[[NDR] Rule 11-7](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-7), List item type defined by conformant schemas (REF, EXT): Section 11.1.2.1, Derivation by list

**context** List **inv:**

self.base\_DataType.attribute.type  
->forEach(t|t.\_'package'->forAll(p|p.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant or( p.name='XMLPrimitiveTypes'))

###### NDR3 [Rule 9-13](REF,EXT). No list item type of xs:ID

[[NDR] Rule 9-13](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-13), No list item type of xs:ID (REF, EXT): Section 9.1.2.1, Simple types prohibited as list item types

**context** List **inv:**

self.base\_DataType.attribute.type->exists(t|not((t.name='ID')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-14] (REF,EXT). No list item type of xs:IDREF

[[NDR] Rule 9-14](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-14), No list item type of xs:IDREF (REF, EXT): Section 9.1.2.1, Simple types prohibited as list item types

**context** List **inv:**

self.base\_DataType.attribute.type->exists(t|not((t.name='IDREF')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-15] (REF,EXT). No list item type of xs:anySimpleType

[[NDR] Rule 9-15](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-15), No list item type of xs:anySimpleType (REF, EXT): Section 9.1.2.1, Simple types prohibited as list item types

**context** List **inv:**

self.base\_DataType.attribute.type->exists(t|not((t.name='anySimpleType')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-16] (REF,EXT). No list item type of xs:ENTITY

[[NDR] Rule 9-16](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-16), No list item type of xs:ENTITY (REF, EXT): Section 9.1.2.1, Simple types prohibited as list item types

**context** List **inv:**

self.base\_DataType.attribute.type->exists(t|not((t.name='ENTITY')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### no generalizations

A List shall not have any generalizations.

**context** List **inv:**

self.base\_DataType.generalization->isEmpty()

### <Stereotype> [LocalTerm](#_2a022f1bd464d869186dc4bf692bd4fb)

##### Extends

##### EnumerationLiteralDescription

The Source stereotype allows an external definition or authority to be referenced by a NIEM element. The sourceURI tag may be used to reference the external definition. When used with an enumeration literal in a <<LocalVocabulary>> the sourceURI will correspond with the NIEM sourceURI.

##### Properties

###### definition

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### literal

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | composite |

###### sourceURIs

The sourceURI tag may be used to reference the external definition. When used with an enumeration literal in a <<LocalVocabulary>> the sourceURI will correspond with the NIEM sourceURI.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### NDR3 [Rule 10-74] (REF,EXT). term:LocalTerm annotates schema

[[NDR] Rule 10-74](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-74), term:LocalTerm annotates schema (REF, EXT): Section 10.9.2, The NIEM local terminology namespace

**context** LocalTerm **inv:**

The constraint is realized through provisioning:  
A provisioned term:LocalTerm will be owned by an xs:appinfo which is owned by an xs:annotation which is owned by an xs:schema

###### NDR3 [Rule 10-75] (REF,EXT). term:LocalTerm has literal or definition

[[NDR] Rule 10-75](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-75), term:LocalTerm has literal or definition (REF, EXT): Section 10.9.2, The NIEM local terminology namespace

**context** LocalTerm **inv:**

self.literal.oclIsUndefined()<>self.definition.oclIsUndefined()

### <Stereotype> [LocalVocabulary](#_9202bf13120d24a7ef936bfc93532ad2)

##### Extends

##### EnumerationDescription

Local vocabulary defines a set of domain specific terms or abbreviations that then may be used in NIEM names and definitions. The local vocabulary is defined as stereotype of enumeration where each enumeration literal is a vocabulary term. The enumeration literal’s UML name corresponds with the domain specific abbreviation, the “UML Specification” corresponds with the expansion of the abbreviation and the UML Documentation corresponds to the NIEM definition. The NIEM sourceURI may also be set by applying the “Source” stereotype to the enumeration literal.  
  
Clarification of mapping:  
SourceText[0..\*] is represented as UML ownedComment.body  
term[1..1] is represented as UML name   
literal[0..1] is represented as UML specification.name  
definition[0..1] is represented as UML specification.oclAsType(UML::StringLiteral).value  
sourceURIs[1..\*] is represented as the tag sourceURIs on the applied Stereotype <<Source>>

### <Stereotype> [MetadataApplication](#_4473e2b97744c5193c47a5dc388acabd)

##### Extends

##### UsageDescription

The <<MetadataApplication>> stereotype is a specialization of Application that is between a <<MetadataType>> Class and either another <<MetadataType>> Class or a Property.

It represents a constraint on a NIEM <<MetadataType>> that limits the application of the NIEM <<MetadataType>> to specific schema types or schema elements.

If a <<MetadataType>> Class is the client of a <<MetadataApplication>> Usage, then any Property with the <<MetadataType>> Class as its type must be for a Class that is a (direct or indirect) subclass of the supplier Class of the <<MetadataApplication>>. A <<MetadataType>> Class may be the client of multiple <<MetadataApplication>> Usages, in which case a Property for it may be in a Class that is a subclass of a supplier Class of any of the <<MetadataApplication>>s. If a <<MetadataType>> is not a client of any <<MetadataApplication>>, then it applies to any type.

If a Property is the supplier of a <<MetadataApplication>> Usage, then the allowable elements referencing the <<MetadataType>> are restricted to the indicator supplier Property and any of its (transitive) substitutions.

A <<MetadataApplication>> Usage with a Class supplier is implemented in XML schema as a NIEM appinfo:appliesToTypes.  A <<MetadataApplication>> Usage with a Property supplier is implemented in XML schema as a NIEM appinfo:appliesToElements.

##### Constraints

###### NDR3 [Rule 10-70] (REF,EXT). appinfo:appliesToTypes annotates metadata element

[[NDR] Rule 10-70](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-70), appinfo:appliesToTypes annotates metadata element (REF, EXT): Section 10.9.1.2, appinfo:appliesToTypes annotation

**context** MetadataApplication **inv:**

self.base\_Usage.client->forAll(g|g.oclIsKindOf(Property) and g.oclAsType(Property).type.stereotypedBy('MetadataType'))

###### NDR3 [Rule 10-71] (SET). appinfo:appliesToTypes references types

[[NDR] Rule 10-71](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-71), appinfo:appliesToTypes references types (SET): Section 10.9.1.2, appinfo:appliesToTypes annotation

**context** MetadataApplication **inv:**

self.base\_Usage.supplier->forAll(g|g.oclIsKindOf(Classifier)or g.oclIsKindOf(Property))

###### NDR3 [Rule 10-72] (REF,EXT). appinfo:appliesToElements annotates metadata element

[[NDR] Rule 10-72](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-72), appinfo:appliesToElements annotates metadata element (REF, EXT): Section 10.9.1.3, appinfo:appliesToElements annotation

**context** MetadataApplication **inv:**

self.base\_Usage.client->forAll(g|g.oclIsKindOf(Property) and g.oclAsType(Property).type.stereotypedBy('MetadataType'))

###### NDR3 [Rule 10-73] (SET). appinfo:appliesToElements references elements

[[NDR] Rule 10-73](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-73), appinfo:appliesToElements references elements (SET): Section 10.9.1.3, appinfo:appliesToElements annotation

**context** MetadataApplication **inv:**

self.base\_Usage.supplier->forAll(g|g.oclIsKindOf(Classifier)or g.oclIsKindOf(Property))

### <Stereotype> [MetadataType](#_db312bed3254e2172249a43917fd4ab0)

##### Generalization

##### [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)Description

A MetadataType is a NIEMType Class that represents a NIEM metadata type. A NIEM metadata type describes data about data, that is, information that is not descriptive of objects and their relationships, but is descriptive of the data itself. Metadata is specified as an instance of a metadata type and may include information such as the security of a piece of data or the source of the data. The applicability of such metadata may be modeled using MetadataApplication dependencies to one or more classes representing the applicable types.

MetadataType is implemented in XML Schema as a complex type definition with complex content. Section 3.4 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses complex type definitions in XML Schema; Section 7.4.4 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) addresses metadata types in NIEM-conformant XML Schema.

rules/1.3/">NIEM NDR v1.3 addresses metadata types in NIEM-conformant XML Schema.body>

##### Constraints

###### NDR3 [Rule 10-36] (REF,EXT). Metadata type has data about data

[[NDR] Rule 10-36](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-36), Metadata type has data about data (REF, EXT): Section 10.5.1, Metadata types

**context** MetadataType **inv:**

Rule is definitional.

###### NDR3 [Rule 10-37] (REF,EXT). Metadata type derived from structures:MetadataType

[[NDR] Rule 10-37](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-37), Metadata type derived from structures:MetadataType (REF, EXT): Section 10.5.1, Metadata types

**context** MetadataType **inv:**

self.base\_Class.general->isEmpty()  
and  
self.base\_Class.clientDependency->select(d|d.stereotypedBy('Restriction'))->isEmpty()

###### NDR3 [Rule 10-38] (REF,EXT). Metadata types are derived from metadata types

[[NDR] Rule 10-38](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-38), Metadata types are derived from metadata types (REF, EXT): Section 10.5.1, Metadata types

**context** MetadataType **inv:**

self.base\_Class.general->isEmpty()  
and  
self.base\_Class.clientDependency->select(d|d.stereotypedBy('Restriction'))->isEmpty()

### <Stereotype> [Namespace](#_a3b43d75feafe90b105d1a836eb3d6a2)

##### Extends

##### PackageDescription

A Namespace Package represents a NIEM namespace identified by a target namespace URI. All UML model elements contained, directly or indirectly within the Package, that represents NIEM types and properties, are considered to be in this target namespace.

A Namespace Package is implemented in XML Schema as an XML schema document.

ML Schema as an XML schema document.body>

##### Properties

###### conformanceTargets

The Conformance Targets Attribute Specification defines an attribute that, when it appears in an XML document, claims the document conforms to one or more conformance targets. This pattern and specification was developed to overcome shortcomings in the NIEM 2 ConformantIndicator element, and to provide needed capabilities in future specifications.

The attribute is a claim of conformance, and not a statement that should be trusted by a validating system. A validator would use this claim to identify to which conformance rules a document should be validated.

The attribute's value is a list of internationalized resource identifiers (IRIs). A later specification may define an IRI for its conformance target, and when an XML document has that IRI in its conformance target attribute, the document is claiming to conform to that conformance target.

The *effective conformance targets attribute* of a conformant document is the first occurrence of the attribute {http://release.niem.gov/niem/conformanceTargets/3.0/}conformanceTargets, in document order.

The only conformance target explicitly defined in NIEM 3 are

<http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument>

and

<http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument>

For NIEM-3 UML, the above formally defined conformance targets are implicitly associated with a NIEM target schema based on the defaultPurpose of an <<InformationModel>>.  Thus, the "conformanceTargets" tag need be populated only with domain-specific conformance target values.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | composite |

###### defaultPrefix

The default prefix for the namespace, used to represent common NIEM prefixes. This prefix should beis used on all XML and/or XML Schema serializations using thatthe namespace, unless it conflicts with another XML and/or XML Schema serialization. If there is a conflict, the actual prefix used is the given default prefix with a number appended in order to make it unique.mapping should append a number to the default prefix to make it unique.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### isConformant

Indicates whether the namespace is NIEM-conformant.

If the namespace is NIEM-conformant, it is implemented in XML Schema as the content of the i:ConformantIndicator application information on the xsd:schema document element. Per Rule 7-1 of NIEM NDR v1.3, the content must be "true". If the namespace is not NIEM-conformant, it is implemented in XML Schema as the content of the i:ConformantIndicator application information on the xsd:import element. Per Rule 7-61 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/), the content must be "false".

NDR v1.3, the content must be "false".dy>

|  |  |
| --- | --- |
| **Type** | Boolean |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### targetNamespace

The target namespace URI for this NIEM namespace.

It is implemented in XML Schema as the value of the targetNamespace attribute on the xsd:schema document element. Per Rules 6-35 and 6-36 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/), the value of the targetNamespace attribute must be present and must be an absolute URI.

and must be an absolute URI.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### version

The version of the NIEM namespace.

It is implemented in XML Schema as the value of the version attribute on the xsd:schema document element. Per Rules 6-37 and 6-38 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/), the value of the version attribute must be present and must not be the empty string. Default is "1".

ty string. Default is "1".

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### NDR3 [Rule 10-7] (REF,EXT). Import of external namespace has data definition

[[NDR] Rule 10-7](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-7), Import of external namespace has data definition (REF, EXT): Section 10.2.3.1, Import of external namespace.

**context** Namespace **inv:**

This constraint resolved during provisioning;   
in UML, all InformationModels must be documented;   
an xs:import is generated if content of the InformationModel refers to another InformationModel;  
generated xs:import elements from a external namespace will include xs:documentation obtained from their documented "external" InformationModels.

###### NDR3 [Rule 7-2](REF,EXT,INS). Document uses XML namespaces properly

Rule 7-2. Document uses XML namespaces properly

[Rule 7-2] (REF, EXT, INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:report test="true()">The document MUST be namespace-well-formed and namespace-valid.</sch:report>

</sch:rule>

</sch:pattern>

The terms *namespace-well-formed* and *namespace-valid* are normatively defined by XML Namespaces and XML Namespaces Errata.

**context** Namespace **inv:**

self.targetNamespace is namespace-well-formed and namespace-valid.  
  
namespace-valid conformance is enforced during provisioning by ensuring that the production of names is conformant with NDR naming rules and XML Namespaces Specification.  
namespace-well-formed conformance is enforced during provisioning by ensuring that the target xml document is well formed with respected to the XML Namespaces Specification.

###### NDR3 [Rule 7-3] (REF,EXT). Document is a schema document

Rule 7-2. Document uses XML namespaces properly

[Rule 7-2] (REF, EXT, INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:report test="true()">The document MUST be namespace-well-formed and namespace-valid.</sch:report>

</sch:rule>

</sch:pattern>

The terms *namespace-well-formed* and *namespace-valid* are normatively defined by XML Namespaces and XML Namespaces Errata.

**context** Namespace **inv:**

Enforced during provisioning to Schema from Namespace Package.

###### NDR3 [Rule 7-4] (REF,EXT). Document element is xs:schema

Rule 7-2. Document uses XML namespaces properly

[Rule 7-2] (REF, EXT, INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:report test="true()">The document MUST be namespace-well-formed and namespace-valid.</sch:report>

</sch:rule>

</sch:pattern>

The terms *namespace-well-formed* and *namespace-valid* are normatively defined by XML Namespaces and XML Namespaces Errata.

**context** Namespace **inv:**

Enforced during provisioning to Schema from Namespace Package.

###### NDR3 [Rule 7-5] (REF,EXT). Component name follows ISO 11179 Part 5 Annex A

[[NDR] Rule 7-5](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_7-5), Component name follows ISO 11179 Part 5 Annex A (REF, EXT): Section 7.5, ISO 11179 Part 5

**context** Namespace **inv:**

The default normative naming rules based on ISO 11179-5 are not easily computable, so are not represented as an executable OCL Constraint.

###### NDR3 [Rule 9-10] (REF,EXT). Simple type definition is top-level

Rule 9-10. Simple type definition is top-level

[Rule 9-10] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleType">

<sch:assert test="exists(parent::xs:schema)"

>A simple type definition MUST be top-level.</sch:assert>

</sch:rule>

</sch:pattern>

NIEM does not support anonymous types in NIEM-conformant schemas. All XML Schema top-level types (children of the document element) are required by XML Schema to be named. By requiring NIEM type definitions to be top level, they are forced to be named and are globally reusable.

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Class)).oclAsType(Class).nestedClassifier->isEmpty()

###### NDR3 [Rule 9-1] (REF,EXT). No base type in the XML namespace

Although the XML namespace is to be imported as if it is conformant, types from that namespace may not be the [base type definition](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_base_type_definition) of any type.

[Rule 9-1] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="namespace-uri-from-QName(resolve-QName(@base, .)) != xs:anyURI('http://www.w3.org/XML/1998/namespace')"

>A schema component must not have a base type definition with a {target namespace} that is the XML namespace.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->forAll(g|g.\_'package'.name<>'xml')  
)

###### NDR3 [Rule 9-1] (REF,EXT). No base type in the XML namespace1

Rule 9-40. Element type not in the XML namespace

The XML namespace may be imported into an conformant schema document as if it were conformant. This specification does not enable a reference to any types that may be defined by any implementation of a schema for that namespace.

[Rule 9-40] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)]">

<sch:assert test="namespace-uri-from-QName(resolve-QName(@type, .)) != 'http://www.w3.org/XML/1998/namespace'"

>An element type MUST NOT have a namespace name that is in the XML namespace.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute.type->forAll(c|  
 c.\_'package'.name<>'xml'  
)

###### NDR3 [Rule 9-23] (REF,EXT). Enumeration has data definition

Rule 9-23. Enumeration has data definition

[Rule 9-23] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:enumeration">

<sch:assert test="some $definition in xs:annotation/xs:documentation[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>An enumeration facet MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Enumeration)).oclAsType(Enumeration).ownedComment  
->exists(c|not(c.\_'body'.oclIsUndefined()) and (c.\_'body'<>''))

###### NDR3 [Rule 9-24] (REF,EXT). Complex type definitions is top-level

Rule 9-24. Complex type definitions is top-level

[Rule 9-24] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:assert test="exists(parent::xs:schema)"

>A complex type definition MUST be top-level.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Class)).oclAsType(Class).nestedClassifier->isEmpty()

###### NDR3 [Rule 9-25] (REF,EXT). Complex type has data definition

Rule 9-25. Complex type has data definition

[Rule 9-25] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:assert test="some $definition in xs:annotation/xs:documentation[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>A complex type MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).ownedComment  
->exists(c|not(c.\_'body'.oclIsUndefined()) and (c.\_'body'<>''))

###### NDR3 [Rule 9-26] (REF,EXT). No mixed content on complex type

Rule 9-26. No mixed content on complex type

[Rule 9-26] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType[exists(@mixed)]">

<sch:assert test="xs:boolean(@mixed) = false()"

>A complex type definition MUST NOT have mixed content.</sch:assert>

</sch:rule>

</sch:pattern>

Mixed content allows the mixing of data tags with text. Languages such as XHTML use this syntax for markup of text. NIEM-conformant schemas define XML that is for data exchange, not text markup. Mixed content creates complexity in processing, defining, and constraining content.

Well-defined markup languages exist outside NIEM and may be used with NIEM data. External schema documents may include mixed content and may be used with NIEM.

**context** Namespace **inv:**

There is no option in NIEM-UML to specify mixed content, consequently there is no mixed content produced during provisioning of target schemas.

###### NDR3 [Rule 9-27] (REF,EXT). No mixed content on complex content

Rule 9-27. No mixed content on complex content

[Rule 9-27] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexContent[exists(@mixed)]">

<sch:assert test="xs:boolean(@mixed) = false()"

>A complex type definition with complex content MUST NOT have mixed content.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

There is no option in NIEM-UML to specify mixed content, consequently there is no mixed content produced during provisioning of target schemas.

###### NDR3 [Rule 9-28] (REF,EXT). Complex type content is explicitly simple or complex

Rule 9-28. Complex type content is explicitly simple or complex

[Rule 9-28] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:assert test="exists(xs:simpleContent) or exists(xs:complexContent)"

>An element xs:complexType MUST have a child element xs:simpleContent or xs:complexContent.</sch:assert>

</sch:rule>

</sch:pattern>

XML Schema provides shorthand to defining complex content of a complex type, which is to define the complex type with immediate children that specify elements, or other groups, and attributes. In the desire to normalize schema representation of types and to be explicit, NIEM forbids the use of that shorthand.

**context** Namespace **inv:**

Complex type content is always enforced to be simple or complex based on defined provisioning.

###### NDR3 [Rule 9-2](REF,EXT). No base type of xs:ID

Rule 9-2. No base type of xs:ID

[Rule 9-2] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:ID')"

>A schema component MUST NOT have an attribute {}base with a value of xs:ID.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='ID') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-3] (REF,EXT). No base type of xs:IDREF

Rule 9-3. No base type of xs:IDREF

[Rule 9-3] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:IDREF')"

>A schema component MUST NOT have an attribute {}base with a value of xs:IDREF.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='IDREF') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-44] (REF,EXT). No element default value

Rule 9-44. No element default value

[Rule 9-44] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element">

<sch:assert test="empty(@default)"

>An element xs:element MUST NOT have an attribute {}default.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

Constraint is realized via Provisioning, which does not create any @default attributes.

###### NDR3 [Rule 9-45] (REF,EXT). No element fixed value

Rule 9-45. No element fixed value

[Rule 9-45] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element">

<sch:assert test="empty(@fixed)"

>An element xs:element MUST NOT have an attribute {}fixed.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

Constraint is realized via Provisioning, which does not create any @fixed attributes.

###### NDR3 [Rule 9-46] (REF). Element declaration is nillable

Rule 9-46. Element declaration is nillable

All elements declared by reference schemas allow a nil value. This enables the ID/IDREF mechanism linking structures:ref and structures:id, as described by [Section 12.2, *Reference elements*, below](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#section_12.2).

A developer may constrain the use of nil in an instance by setting nillable to false in subset schemas, or by use of non-XML Schema mechanisms, such as Schematron.

[Rule 9-46] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@name and (empty(@abstract)

or xs:boolean(@abstract) = false())]">

<sch:assert test="exists(@nillable)

and xs:boolean(@nillable) = true()"

>An element declaration MUST have the {nillable} property with a value of true.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

Specification as OCL Constraint is deferred.

###### NDR3 [Rule 9-4] (REF,EXT). No base type of xs:IDREFS

Rule 9-4. No base type of xs:IDREFS

[Rule 9-4] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:IDREFS')"

>A schema component MUST NOT have an attribute {}base with a value of xs:IDREFS.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='IDREFS') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-58] (REF,EXT). No use of element xs:notation

Rule 9-58. No use of element xs:notation

[Rule 9-58] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:notation">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:notation.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint enforced by provisioning, which does not produce any xs:notation schema components.

###### NDR3 [Rule 9-59] (EXT). Model group does not affect meaning

Complex content definitions in XML Schema use model group schema components. These schema components, xs:all, xs:choice and xs:sequence, also called compositors, provide for ordering and selection of particles within a model group.

XML Schema defines a **particle** as an occurrence of xs:element, xs:sequence, xs:choice, xs:any (wildcard) and xs:group (model group) within a content model. For example, an xs:sequence within an XML Schema complex type is a particle. An xs:element occurring within an xs:sequence is also a particle.

Rule 9-59. Model group does not affect meaning

[Rule 9-59] (EXT) (Interpretation)

The use of model groups xs:all, xs:sequence, and xs:choice MUST NOT define the semantics of an instance. The meaning of an element occurrance within an element occurrence MUST be identical, regardless of the model group used to define a schema component.

**context** Namespace **inv:**

This constraint is not computable.

###### NDR3 [Rule 9-5] (REF,EXT). No base type of xs:anyType

Rule 9-5. No base type of xs:anyType

[Rule 9-5] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:anyType')"

>A schema component MUST NOT have an attribute {}base with a value of xs:anyType.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='anyType') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-60] (REF,EXT). No xs:all

Rule 9-60. No xs:all

[Rule 9-60] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:all">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:all</sch:assert>

</sch:rule>

</sch:pattern>

The element xs:all provides a set of particles (e.g., elements) that may be included in an instance, in no particular order. This compositor does not support a variety of cardinalities, has shown to be confusing in practice, and has functionality that overlaps with NIEM’s use of substitution groups. Use of substitution groups is the preferred method in NIEM schemas for obtaining flexible content models.

**context** Namespace **inv:**

This constraint enforced by provisioning, there is not model representation for xs:all and no production of an xs:all model group.

###### NDR3 [Rule 9-6] (REF,EXT). No base type of xs:anySimpleType

XML Schema has the concept of the ur-type, a type that is the root of all other types. This type is realized in schemas as xs:anyType.

NIEM-conformant schemas must not use xs:anyType, because this feature permits the introduction of arbitrary content (i.e., untyped and unconstrained data) into an XML instance. NIEM intends that the schemas describing that instance describe all constructs within the instance.

Rule 9-6. No base type of xs:anySimpleType

[Rule 9-6] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:anySimpleType')"

>A schema component MUST NOT have an attribute {}base with a value of xs:anySimpleType.</sch:assert>

</sch:rule>

</sch:pattern>

XML Schema provides a restriction of the ur-type that contains only simple content. This provides a wildcard for arbitrary text. It is realized in XML Schema as xs:anySimpleType.

NIEM-conformant schemas must not use xs:anySimpleType because this feature is insufficiently constrained to provide a meaningful starting point for content definitions. Instead, content should be based on one of the more specifically defined simple types defined by XML Schema

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='anySimpleType') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-7] (REF,EXT). No base type of xs:NOTATION

Rule 9-7. No base type of xs:NOTATION

[Rule 9-7] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:NOTATION')"

>A schema component MUST NOT have an attribute {}base with a value of xs:NOTATION.</sch:assert>

</sch:rule>

</sch:pattern>

XML Schema notations allow the attachment of system and public identifiers on fields of data. The notation mechanism does not play a part in validation of instances and is not supported by NIEM.

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='NOTATION') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-82] (REF,EXT). Schema document defines target namespace

[[NDR] Rule 9-82](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-82), Schema document defines target namespace (REF, EXT): Section 9.7, Schema as a whole

**context** Namespace **inv:**

not(self.targetNamespace.oclIsUndefined()) and  
 (self.targetNamespace<>'')

###### NDR3 [Rule 9-83] (REF,EXT). Target namespace is absolute URI

[[NDR] Rule 9-83](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-83), Target namespace is absolute URI (REF, EXT): Section 9.7, Schema as a whole

**context** Namespace **inv:**

Specification of this constraint in OCL has been deferred.

###### NDR3 [Rule 9-84] (REF,EXT). Schema has version

[[NDR] Rule 9-84](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-84), Schema has version (REF, EXT): Section 9.7, Schema as a whole

**context** Namespace **inv:**

not(self.version.oclIsUndefined())   
and  
self.version<>''

###### NDR3 [Rule 9-89] (REF,EXT). xs:import must have namespace

I

Rule 9-89. xs:import must have namespace

[Rule 9-89] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:import">

<sch:assert test="exists(@namespace)"

>An element xs:import MUST have an attribute {}namespace.</sch:assert>

</sch:rule>

</sch:pattern>

An import that does not specify a namespace is enabling references to non-namespaced components. NIEM requires that all components have a defined namespace. It is important that the namespace declared by a schema be universally defined and unambiguous.

**context** Namespace **inv:**

This constraint resolved during provisioning; all xs:import declarations are implicitly specified by the relationships between Information Model elements and will include the namespace specified in the targetNamespace of the referenced Information Model.

###### NDR3 [Rule 9-8] (REF,EXT). No base type of xs:ENTITY

Rule 9-8. No base type of xs:ENTITY

[Rule 9-8] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:ENTITY')"

>A schema component MUST NOT have an attribute {}base with a value of xs:ENTITY.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='ENTITY') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

###### NDR3 [Rule 9-90] (SET). XML Schema document set must be complete

Rule 9-90. XML Schema document set must be complete

An XML Schema document set defines an XML Schema that may be used to validate an XML document. This rule ensures that a schema document set under consideration contains definitions for everything that it references; it has everything necessary to do a complete validation of XML documents, without any unresolved references. Note that some tools may allow validation of documents using partial schemas, when components that are not present are not exercised by the XML document under validation. Such a schema does not meet qualify as a conformant schema document set.

[Rule 9-90] (SET) (Constraint)

The schema document set must constitute a complete XML Schema; it must contain the definition of every schema component referenced by any component defined by the schema set.

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-91] (REF,EXT). Namespace referenced by attribute type is imported

The [XML Schema definition language](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_XML_Schema_definition_language) requires that, when a [schema document](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_schema_document) references a component in some other namespace, it must use xs:import to import the namespace of the referenced component. The use of xs:import is described by [[XML Schema Structures]](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#XMLSchema-1) [Section 4.2.3, *References to schema components across namespaces*](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#nsi-schema_components).

Some tools do not enforce this constraint; one such tool carries imports from a [schema document](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_schema_document) into [schema documents](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_schema_document) that it imports. This has the potential to introduce incompatibility into schema documents and schema document sets that exercise this bug. To maintain compatibility across tool sets, this requirement is an explicit rule for NIEM-conformant schemas.

Rule 9-91. Namespace referenced by attribute type is imported

[Rule 9-91] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@type]">

<sch:assert test="every $namespace in namespace-uri-from-QName(resolve-QName(@type, .)) satisfies (

$namespace = nf:get-target-namespace(.)

or $namespace = xs:anyURI('http://www.w3.org/2001/XMLSchema')

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace])"

>The namespace of a type referenced by @type MUST be the target namespace, the XML Schema namespace, or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-92] (REF,EXT). Namespace referenced by attribute base is imported

Rule 9-92. Namespace referenced by attribute base is imported

[Rule 9-92] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@base]">

<sch:assert test="every $namespace in namespace-uri-from-QName(resolve-QName(@base, .)) satisfies (

$namespace = nf:get-target-namespace(.)

or $namespace = xs:anyURI('http://www.w3.org/2001/XMLSchema')

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace])"

>The namespace of a type referenced by @base MUST be the target namespace, the XML Schema namespace, or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-93] (REF,EXT). Namespace referenced by attribute itemType is imported

Rule 9-93. Namespace referenced by attribute itemType is imported

[Rule 9-93] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@itemType]">

<sch:assert test="every $namespace in namespace-uri-from-QName(resolve-QName(@itemType, .)) satisfies (

$namespace = nf:get-target-namespace(.)

or $namespace = xs:anyURI('http://www.w3.org/2001/XMLSchema')

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace])"

>The namespace of a type referenced by @itemType MUST be the target namespace, the XML Schema namespace, or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-94] (REF,EXT). Namespaces referenced by attribute memberTypes is imported

Rule 9-94. Namespaces referenced by attribute memberTypes is imported

[Rule 9-94] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@memberTypes]">

<sch:assert test="every $type in tokenize(normalize-space(@memberTypes), ' '),

$namespace in namespace-uri-from-QName(resolve-QName($type, .)) satisfies (

$namespace = nf:get-target-namespace(.)

or $namespace = xs:anyURI('http://www.w3.org/2001/XMLSchema')

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace])"

>The namespace of a type referenced by @memberTypes MUST be the target namespace, the XML Schema namespace, or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-95] (REF,EXT). Namespace referenced by attribute ref is imported

Rule 9-95. Namespace referenced by attribute ref is imported

[Rule 9-95] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@ref]">

<sch:assert test="every $namespace in namespace-uri-from-QName(resolve-QName(@ref, .)) satisfies

$namespace = nf:get-target-namespace(.)

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace]"

>The namespace of a component referenced by @ref MUST be the target namespace or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-96] (REF,EXT). Namespace referenced by attribute substitutionGroup is imported

Rule 9-96. Namespace referenced by attribute substitutionGroup is imported

[Rule 9-96] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[@substitutionGroup]">

<sch:assert test="every $namespace in namespace-uri-from-QName(resolve-QName(@substitutionGroup, .)) satisfies

$namespace = nf:get-target-namespace(.)

or nf:get-document-element(.)/xs:import[xs:anyURI(@namespace) = $namespace]"

>The namespace of a component referenced by @substitutionGroup MUST be the target namespace or be imported.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

This constraint resolved during provisioning; the transitive closure of all schemas referenced will be in the schema document set for an MPD.

###### NDR3 [Rule 9-9] (REF,EXT). No base type of xs:ENTITIES

Rule 9-9. No base type of xs:ENTITIES

[Rule 9-9] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@base)]">

<sch:assert test="resolve-QName(@base, .) != xs:QName('xs:ENTITIES')"

>A schema component MUST NOT have an attribute {}base with a value of xs:ENTITIES.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Namespace **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
 ->select(t|(t.name='ENTITIES') and (t.\_'package'.name='XMLPrimitiveTypes'))->size()=0  
)

### <Stereotype> [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)

##### Extends

##### ClassDescription

A NIEMType is an Class that represents one of the specific semantic kinds of NIEM complex types (i.e., types that may have attributive structure). NIEMType is abstract.

A NIEMType Class is implemented in XML Schema as a complex type definition with complex content.

### <Stereotype> [ObjectType](#_30b3ae28677385d6d198eae570b7a23a)

##### Generalization

##### [NIEMType](#_cddcf0aa38f9fb92183a65a83b2b548f)Description

ObjectType is a NIEMType Class that represents a NIEM object type. A NIEM object type represents some kind of object: a thing with its own lifespan that has some existence. The object may or may not be a physical object. It may be a conceptual object.

ObjectType is implemented in XML Schema as a complex type definition. Section 3.4 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses complex type definitions in XML Schema; Section 7.4.1 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) addresses object types in NIEM-conformant XML Schema.

R v1.3 addresses object types in NIEM-conformant XML Schema.

##### Constraints

###### NDR [Rule 7-39] (REF, EXT)

Within the schema, an object type SHALL be a complex type definition that either constitutes a NIEM-conformant component or for which there exists a NIEM-conformant component of one of the following forms:

1. Has simple content, is based on a simple type, and contains the attribute group structures:SimpleObjectAttributeGroup, and has application information appinfo:Base of structures:Object.
2. Has complex content, and is based on complex type structures:ComplexObjectType, and has application information appinfo:Base of structures:Object.
3. Is a complex type that is derived from an object type, which is defined according to this rule.

**context** ObjectType **inv:**

###### NDR3 [Rule 10-18] (REF,EXT). Proxy type has designated structure

The NIEM 3.0 release schema niem/proxy/xsd/3.0/xs.xsd provides complex type bases for some of the simple types in the XML Schema namespace. The complex types in this schema reuse the local names of the XML Schema simple types they extend, even though those names don’t follow the naming structure of most conformant complex types. There is a special exception to naming rules to allow the reuse of the XML Schema simple type names in conformant schemas. This is done to make conformant schemas more understandable to people that are familiar with the names of the XML Schema namespace simple types.

A complex type that is a direct extension of a simple type from the XML Schema namespace (e.g., xs:string, xs:integer, xs:boolean) is allowed to have the same local name as the XML Schema simple type, if and only if the extension adds no content other than the attribute group structures:SimpleObjectAttributeGroup. This allows for an intuitive name when using an XML Schema simple type in a conformant schema.

Rule 10-18. Proxy type has designated structure

[Rule 10-18] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType[some $name in @name,

$extension in xs:simpleContent/xs:extension,

$base-qname in resolve-QName($extension/@base, $extension) satisfies

$base-qname = QName('http://www.w3.org/2001/XMLSchema', @name)]">

<sch:assert test="xs:simpleContent[

xs:extension[

empty(xs:attribute)

and count(xs:attributeGroup) = 1

and xs:attributeGroup[

resolve-QName(@ref, .) = xs:QName('structures:SimpleObjectAttributeGroup')]]]"

>A proxy type MUST have the designated structure. It MUST use xs:extension. It MUST NOT use xs:attribute. It MUST include exactly one xs:attributeGroup reference, which must be to structures:SimpleObjectAttributeGroup.</sch:assert>

</sch:rule>

</sch:pattern>

**context** ObjectType **inv:**

(  
 self.general->forAll(g|(g.\_'package'.name='XMLPrimitiveTypes')and(g.name=self.name))  
 and  
 self.clientDependency->select(d|d.stereotypedBy('XSDSimpleContent')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(g|(g.\_'package'.name='XMLPrimitiveTypes')and(g.name=self.name))  
)  
implies  
self.attribute->isEmpty()

###### NDR3 [Rule 10-21] (REF). Augmentable type has augmentation point element

[[NDR] Rule 10-21](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-21), Augmentable type has augmentation point element (REF): Section 10.4.1, Augmentable types

**context** ObjectType **inv:**

The constraint is enforced by provisioning; if an AugmentationPoint is not defined in the model, then it is created

###### NDR3 [Rule 10-22] (REF,EXT). Augmentable type has at most one augmentation point element

[[NDR] Rule 10-22](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-22), Augmentable type has at most one augmentation point element (REF, EXT): Section 10.4.1, Augmentable types

**context** ObjectType **inv:**

self.attribute->select(a|self.name.replace('Type','AugmentationPoint')=a.name)->size()<=1

###### NDR3 [Rule 10-23] (REF,EXT). Augmentation point corresponds to augmentable type

[[NDR] Rule 10-23](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-23), Augmentation point corresponds to augmentable type (REF, EXT): Section 10.4.2, Augmentation point element declarations

**context** ObjectType **inv:**

self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|self.\_'package'.ownedType.name->exists(n|n.replace('Type','AugmentationPoint')=a.name))

###### NDR3 [Rule 10-24] (REF,EXT). An augmentation point has no type

[[NDR] Rule 10-24](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-24), An augmentation point has no type (REF, EXT): Section 10.4.2, Augmentation point element declarations

**context** ObjectType **inv:**

self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|a.type.oclIsUndefined())

###### NDR3 [Rule 10-25] (REF,EXT). An augmentation point has no substitution group

[[NDR] Rule 10-25](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-25), An augmentation point has no substitution group (REF, EXT): Section 10.4.2, Augmentation point element declarations

**context** ObjectType **inv:**

self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|a.subsettedProperty->isEmpty())

###### NDR3 [Rule 10-26] (REF,EXT). Augmentation point element may only be referenced by its type

[[NDR] Rule 10-26](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-26), Augmentation point element may only be referenced by its type (REF, EXT): Section 10.4.3, Augmentation point element use

**context** ObjectType **inv:**

not(self.stereotypedBy('PropertyHolder'))  
implies  
self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|a.name.replace('AugmentationPoint','Type')=self.name)

###### NDR3 [Rule 10-27] (REF). Augmentation point reference is optional

[[NDR] Rule 10-27](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-27), Augmentation point reference is optional (REF): Section 10.4.3, Augmentation point element use

**context** ObjectType **inv:**

(  
 not(self.\_'package'.oclIsUndefined())  
 and  
 self.\_'package'.stereotypedBy('InformationModel')  
 and  
 self.\_'package'.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(im|(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.subset)or(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.reference))  
)   
implies  
self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|a.lower=0)

###### NDR3 [Rule 10-28] (REF). Augmentation point reference is unbounded

[[NDR] Rule 10-28](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-28), Augmentation point reference is unbounded (REF): Section 10.4.3, Augmentation point element use

**context** ObjectType **inv:**

(  
 not(self.\_'package'.oclIsUndefined())  
 and  
 self.\_'package'.stereotypedBy('InformationModel')  
 and  
 self.\_'package'.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(im|(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.subset)or(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.reference))  
)   
implies  
self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|a.upper<0)

###### NDR3 [Rule 10-29] (REF). Augmentation point reference must be last particle

[[NDR] Rule 10-29](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-29), Augmentation point reference must be last particle (REF, EXT): Section 10.4.3, Augmentation point element use

**context** ObjectType **inv:**

(  
 not(self.\_'package'.oclIsUndefined())  
 and  
 self.\_'package'.stereptypedBy('informationModel')  
 and  
 self.\_'package'.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(im|(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.subset)or(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode.reference))  
)   
implies  
self.attribute->select(a|a.name.endsWith('AugmentationPoint'))  
->forAll(a|self.attribute->last()=a)

###### NDR3 [Rule 11-11] (REF,EXT). Complex type with simple content has structures:SimpleObjectAttributeGroup

[[NDR] Rule 11-11](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-11), Complex type with simple content has structures:SimpleObjectAttributeGroup (REF, EXT): Section 11.1.3, Complex type definition

**context** ObjectType **inv:**

This constraint realized during provisioning;  
The structures:SimpleObjectAttributeGroup is not in the UML-NIEM model, it is produced as required during construction of a complex type with simple content

###### NDR3 [Rule 11-1] (REF,EXT). Name of type ends in "Type"

[[NDR] Rule 11-1](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-1), Name of type ends in Type (REF, EXT): Section 11.1, Type definition components

**context** ObjectType **inv:**

(  
 self.\_'package'.stereotypedBy('InformationModel')  
 and  
 not(self.stereotypedBy('PropertyHolder'))  
 and  
 not(self.stereotypedBy('LocalVocabulary')   
 and  
 not(self.general->forAll(g|(g.name=self.name)and(g.\_'package'.name='XMLPrimitiveTypes'))  
)  
implies  
self.name.endsWith('Type')

###### NDR3 [Rule 11-2] (REF,EXT). Name of type other than proxy type is in upper camel case

[[NDR] Rule 11-2](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-2), Name of type other than proxy type is in upper camel case (REF, EXT): Section 11.1, Type definition components

**context** ObjectType **inv:**

(  
 self.\_'package'.stereotypedBy('InformationModel')  
 and  
 not(self.stereotypedBy('PropertyHolder'))  
 and  
 not(self.stereotypedBy('LocalVocabulary')   
 and  
 not(self.general->forAll(g|(g.name=self.name)and(g.\_'package'.name='XMLPrimitiveTypes'))  
)  
implies  
self.name.match('^([A-Z][A-Za-z0-9\\-]\*)+$')

###### NDR3 [Rule 11-32] (REF,EXT). Standard opening phrase for complex type

Rule 11-32. Standard opening phrase for complex type

[Rule 11-32] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType[ends-with(@name, 'AssociationType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type for (a relationship|an association)'))"

>The data definition for an association type SHOULD begin with the standard opening phrase "a datatype for (a relationship|an association)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType[ends-with(@name, 'AugmentationType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type (that supplements|for additional information about)'))"

>The data definition for an augmentation type SHOULD begin with the standard opening phrase "a data type (that supplements|for additional information about)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType[ends-with(@name, 'MetadataType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type for (metadata about|information that further qualifies)'))"

>The data definition for a metadata type SHOULD begin with the standard opening phrase "a data type for (metdata about|information that further qualifies)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type'))"

>The data definition for a type SHOULD begin with the standard opening phrase "a data type...".</sch:report>

</sch:rule>

</sch:pattern>

**context** ObjectType **inv:**

(  
 not(self.stereotypedBy('LocalVocabulary'))   
 and  
 not(self.stereotypedBy('PropertyHolder'))   
 and  
 self.namespace.stereotypedBy('InformationModel')   
 )   
implies   
if (self.name.endsWith('AssociationType')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^a data type for (a relationship|an association)'))   
 else if (self.name.endsWith('AugmentationType')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^a data type (that supplements|for additional information about)'))   
 else if (self.name.endsWith('MetadataType')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^a data type for (metadata about|information that further qualifies)'))   
 else self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^a data type'))   
 endif   
 endif   
endif

###### NDR3 [Rule 11-3] REF,EXT. Base type definition defined by conformant schema

[[NDR] Rule 11-3](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-3), Base type definition defined by conformant schema (REF, EXT): Section 11.1.1, Type definition hierarchy

**context** ObjectType **inv:**

self.general->select(g|not(g.\_'package'.name='XMLPrimitiveTypes'))  
->forAll(g|g.appliedStereotype(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant)

###### NDR3 [Rule 11-4] (REF,EXT). Name of simple type ends in "SimpleType"

[[NDR] Rule 11-4](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-4), Name of simple type ends in SimpleType (REF, EXT): Section 11.1.2, Simple type definition

**context** ObjectType **inv:**

(  
 (self.oclIsKindOf(Enumeration)and self.oclAsType (Enumeration).ownedLiteral->notEmpty())  
 or  
 self.stereotypedBy('List')  
 or  
 self.stereotypedBy('Union')  
 or  
 self.stereotypedBy('ValueRestriction')  
 or  
 self.stereotypedBy('XSDRepresentationRestriction')   
 or  
 self.supplierDependency->exists(c|c.stereotypedBy('XSDSimpleContent'))  
)  
implies  
self.name.endsWith('SimpleType')

###### NDR3 [Rule 11-5](REF,EXT). Name of simple type is upper camel case

[[NDR] Rule 11-5](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-5), Name of simple type is upper camel case (REF, EXT): Section 11.1.2, Simple type definition

**context** ObjectType **inv:**

(  
 (self.oclIsKindOf(Enumeration)and self.oclAsType (Enumeration).ownedLiteral->notEmpty())  
 or  
 self.stereotypedBy('List')  
 or  
 self.stereotypedBy('Union')  
 or  
 self.stereotypedBy('ValueRestriction')  
 or  
 self.stereotypedBy('XSDRepresentationRestriction')   
 or  
 self.supplierDependency->exists(c|c.stereotypedBy('XSDSimpleContent'))  
)  
implies  
self.name.match('^([A-Z][A-Za-z0-9\\-]\*)+$')

### <Stereotype> [PropertyHolder](#_1b6d271c15434199db0d11d815b96d97)

##### Extends

##### ClassDescription

PropertyHolder is a Class holding global Properties that are not the subject of any specific NIEM type. A Property of a NIEM type may then be defined by reference to a Property of a PropertyHolder by using a References realization with the Property in the PropertyHolder as the supplier. Note that the multiplicity of Properties in a PropertyHolder is immaterial -- the multiplicities are established by Properties in the corresponding References client. The target namespace of Properties in a PropertyHolder is the target namespace of the Namespace Package that contains the PropertyHolder (which may be different than the target namespace of NIEM types that use the Properties in the PropertyHolder).

PropertyHolder does not represent any NIEM concept; it exists to permit the user to define a NIEM property that is not the subject of any NIEM type. There are significant differences between the UML representation and XML Schema implementation of a NIEM property. Sections 6.1.6.2 and 6.1.6.3 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/), Rule 6-14 and Rule 6-15, require that an attribute or element declaration be a top-level declaration, but NIEM NDR v1.3 does not require a corresponding attribute use or element particle; however, Section 7.3.44 of [UML] requires that a Property be the ownedAttribute of a Classifier. Thus in the UML representation, the declaration and use of a Property are not distinct, and the declaration of a Property requires its use. In the XML Schema implementation, the declaration and use are distinct, and the declaration does not require a corresponding use. To resolve this difference, any Property within a PropertyHolder shall represent an attribute or element declaration without a corresponding attribute use or element particle.

PropertyHolders may be used to hold the properties of a substitution group.  Where a PropertyHolder is used to define a substitution group an extension of that substitution group shall be a subclass of the substitution group PropertyHolder.

ies of a substitution group.  Where a PropertyHolder is used to define a substitution group an extension of that substitution group shall be a subclass of the substitution group PropertyHolder.

### <Stereotype> [References](#_c7b8a68ef50d3d361f495647dd4876ec)

##### Extends

##### RealizationDescription

The References Stereotype applies to a Realization between Properties, Classes or Packages. It allows for Properties in one Class to be defined by reference to Properties in another class. A References Realization between two classes is defined to be equivalent to having References Realizations between matching Properties of the Classes where matching is determined by identical NIEM names. A References Realization between two packages is defined to be equivalent to having References Realizations between matching Classes contained in the Packages where matching is determined by having identical NIEM names. Matching is based on the NIEMName of the elements, either as derived implicitly or as set explicitly using the ReferenceName stereotype.

If a Property is the client of a References Realization, then it represents a NIEM property defined by reference to the NIEM property declaration represented by the supplier of the Realization. It is implemented in XSD schema as an attribute use or element particle that references the attribute or element declaration that implements the supplier of the Realization. Note that the supplier Property may be in a different Namespace than the client property, in which case the attribute or element declaration represented by the supplier will be in a different target namespace than the use represented by the client.

##### Constraints

###### MDR [Rule 3-02]

**[Rule 3-2]** NIEM subsets may omit elements with zero cardinality and adjust the cardinality of elements in reference schemas from which they are derived, as long as the subset property is maintained.intained.l>

**context** References **inv:**

(  
 ( self.base\_Realization.client->size()=1)   
 and ( self.base\_Realization.supplier->size()=1)   
 and self.base\_Realization.client->forAll(client|client.oclIsKindOf(Classifier))  
 and self.base\_Realization.supplier->forAll(supplier|supplier.oclIsKindOf(Classifier)and not(supplier.stereotypedBy('PropertyHolder')) )  
 ) implies (   
 (  
 self.base\_Realization.client.oclAsType(Classifier).attribute  
 ->forAll(clientAttribute| self.base\_Realization.supplier.oclAsType(Classifier).attribute  
 ->forAll(supplierAttribute|  
 (clientAttribute.name=supplierAttribute.name)  
 implies   
 (   
 (clientAttribute.lower>=supplierAttribute.lower)   
 and   
 ( (supplierAttribute.upper=-1) or (clientAttribute.upper<=supplierAttribute.upper) )   
 and ( (clientAttribute.upper=-1) or (clientAttribute.lower<=clientAttribute.upper) )  
 )  
 )  
 )  
 )   
 and  
 ( self.base\_Realization.supplier.oclAsType(Classifier).attribute->select(a|a.lower>0)  
 ->forAll(supplierAttribute| self.base\_Realization.client.oclAsType(Classifier).attribute->exists(clientAttribute|clientAttribute.name=supplierAttribute.name) )  
 )   
)

###### References must be between like metaclasses

References may only be between packages, classifiers or properties and the metatype of the client must be the same as the metatype of the supplier.

**context** References **inv:**

### <Stereotype> [Representation](#_d6f498351d650cdf2cbfc200d54cec79)

##### Extends

##### PropertyDescription

One need frequently faced by schema developers is for multiple representations for a single concept. To handle this need, NIEM has adopted the Representation pattern, in which a type may contain a representation element, and the various representations for that element type are in the substitution group for that representation element.

In NIEM-3 UML, the Representation concept may be expressed as an abstract type-less Property whose name has a suffix of "Representation".  Alternatively, an abstract type-less Property Stereotyped by <<Representation>> may be used to represent the Representation concept, in which case the NIEM naming rule for Representation elements will be implicitly applied during transformation to the target schema element.

##### Constraints

###### NDR3 [Rule 10-41] (REF,EXT). Name of element that ends in "Representation" is abstract

[[NDR] Rule 10-41](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-41), Name of element that ends in Representation is abstract (REF, EXT): Section 10.7, The Representation pattern

**context** Representation **inv:**

(   
 (   
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 not(self.name.oclIsUndefined())   
 and   
 self.name.endsWith('Representation')  
 )   
 implies   
 self.stereotypedBy('Representation'))  
and  
(  
 self.stereotypedBy('Representation')  
 implies  
 self.type.oclIsUndefined()  
)

###### NDR3 [Rule 10-42] (REF,EXT). A substitution for a representation element declaration is a value for a type

[[NDR] Rule 10-42](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-42), A substitution for a representation element declaration is a value for a type (REF, EXT): Section 10, Rules for NIEM modeling, by NIEM concept

**context** Representation **inv:**

Rule is definitional.

### <Stereotype> [Restriction](#_5168057aef57b6169e0750b82ef3bdf8)

##### Extends

##### RealizationDescription

A Restriction Realization represents a relationship between two type definitions: the first is derived by restriction from the second. The two types must either both be NIEMType Classes or both be DataTypes. If the two types are Classes, then the attributes of the client class must be a subset of the attributes of the supplier class and omitted attributes must have a multiplicity lower bound of zero. if the two classes are DataTypes, then the client type is considered to have a value space that is a subset of that of the supplier, as may be further specified using a ValueRestriction stereotype on the client.

This relationship is implemented in XML Schema through the base attribute on the xsd:restriction element of the first type definition, the actual value of which resolves to the second type definition. If a type is a ValueRestriction the generalization owned by that type is implicitly an XSDRestriction. Sections 3.4 and 3.14 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) address the use of restriction in XML Schema; Sections 6.5.2 and 6.5.3 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) address the use of restriction in NIEM-conformant XML Schema.

##### Constraints

###### XSDRestrictionComplexTypeComplexContent

If the general Classifier is a NIEMType that is not the client of a NIEMSimpleContent Realization, the specific Classifier must be a NIEMType that is not the client of a NIEMSimpleContent Realization.

**context** Restriction **inv:**

(  
 self.base\_Realization.supplier.namespace->forAll(o|o.stereotypedBy('InformationModel'))   
 and  
 self.base\_Realization.supplier.clientDependency->select(d|d.stereotypedBy('NIEMSimpleContent'))->isEmpty()  
)   
implies  
 self.base\_Realization.client.clientDependency->select(d|d.stereotypedBy('NIEMSimpleContent'))->isEmpty()

###### XSDRestrictionComplexTypeSimpleContent

If the general Classifier is a NIEMType that is the client of a NIEMSimpleContent Realization, the specific Classifier must be a NIEMType that is the client of a NIEMSimpleContent Realization.

**context** Restriction **inv:**

(self.base\_Realization.supplier.namespace->forAll(o|o.stereotypedBy('InformationModel')))   
and(  
 self.base\_Realization.supplier.clientDependency->select(d|d.stereotypedBy('XSDSimpleContent'))->notEmpty()  
 implies   
 self.base\_Realization.client.namespace->forAll(o|o.stereotypedBy('InformationModel'))   
 and self.base\_Realization.client.clientDependency->select(d|d.stereotypedBy('XSDSimpleContent'))->notEmpty()  
)

###### XSDRestrictionSimpleType

If the general Classifier is a DataType, the specific Classifier must be a DataType.

**context** Restriction **inv:**

self.base\_Realization.supplier->forAll(o|o.oclIsKindOf(DataType)) implies  
 self.base\_Realization.client->forAll(o|o.oclIsKindOf(DataType))

### <Stereotype> [Union](#_f70f8892e25db199be6f3858b52a6adf)

##### Extends

##### DataTypeDescription

A Union is a DataType whose value space is the union of one or more other DataTypes, which are the member types of the Union. The member types are specified using UnionOf Usage dependencies.

A Union DataType is implemented in XML Schema as a union simple type definition. Each UnionOf dependency of which the Union is the client represents a relationship between two type definitions: the first is a union simple type definition whose member type definition is the second. This relationship is implemented in XML Schema through the memberTypes attribute on the xsd:union element of the union simple type definition, the actual value of which resolves to the second type definition. Section 3.14 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses union simple type definitions in XML Schema.

##### Constraints

###### NDR3 [Rule 11-8] (REF,EXT). Union member types defined by conformant schemas

[[NDR] Rule 11-8](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-8), Union member types defined by conformant schemas (REF, EXT): Section 11.1.2.2, Derivation by union

**context** Union **inv:**

self.base\_DataType.clientDependency->select(d|d.stereotypedBy('UnionOf')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
 ->forAll(t|(t.\_'package'.name='XMLPrimitiveTypes')or(t.\_'package'.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant))

###### NDR3 [Rule 9-17] (REF,EXT). No union member types of xs:ID

Rule 9-17. No union member types of xs:ID

[Rule 9-17] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:ID')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:ID.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
 ->forAll(t|(t.name<>'ID')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 9-18] (REF,EXT). No union member types of xs:IDREF

Rule 9-18. No union member types of xs:IDREF

[Rule 9-18] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:IDREF')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:IDREF.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
->forAll(t|(t.name<>'IDREF')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 9-19] (REF,EXT). No union member types of xs:IDREFS

Rule 9-19. No union member types of xs:IDREFS

[Rule 9-19] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:IDREFS')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:IDREFS.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
->forAll(t|(t.name<>'IDREFS')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 9-20] (REF,EXT). No union member types of xs:anySimpleType

Rule 9-20. No union member types of xs:anySimpleType

[Rule 9-20] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:anySimpleType')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:anySimpleType.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
->forAll(t|(t.name<>'anySimpleType')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 9-21] (REF,EXT). No union member types of xs:ENTITY

Rule 9-21. No union member types of xs:ENTITY

[Rule 9-21] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:ENTITY')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:ENTITY.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
->forAll(t|(t.name<>'ENTITY')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 9-22] (REF,EXT). No union member types of xs:ENTITIES

Rule 9-22. No union member types of xs:ENTITIES

[Rule 9-22] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:\*[exists(@memberTypes)]">

<sch:assert test="every $type-qname

in tokenize(normalize-space(@memberTypes), ' ')

satisfies resolve-QName($type-qname, .) != xs:QName('xs:ENTITIES')"

>A schema component MUST NOT have an attribute {}memberTypes that includes a value of xs:ENTITIES.</sch:assert>

</sch:rule>

</sch:pattern>

**context** Union **inv:**

self.base\_DataType.clientDependency  
 ->select(d|d.stereotypedBy('UnionOf')).supplier  
 ->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier)  
->forAll(t|(t.name<>'ENTITIES')and(t.\_'package'.name<>'XMLPrimitiveTypes'))

###### no generalizations

A Union shall not have any generalizations.

**context** Union **inv:**

self.base\_DataType.generalization->isEmpty()

###### no owned attributes

A Union shall not have any ownedAttributes.

**context** Union **inv:**

self.base\_DataType.attribute->isEmpty()

### <Stereotype> [UnionOf](#_13919d689991d7b275167f98a0cd7bdd)

##### Extends

##### UsageDescription

The UnionOf stereotype is applied to a Usage dependency, the client of which must  
 be a Union DataType and the supplier of which must be a DataType that represents a  
 legal union member type. A UnionOf dependency specifies that the supplier DataType is  
 a member type of the client Union.

##### Constraints

###### client must be union

**context** UnionOf **inv:**

self.base\_Usage.client->forAll(c|c.stereotypedBy('Union'))

###### supplier must be data type

**context** UnionOf **inv:**

self.base\_Usage.supplier->forAll(s|s.oclIsKindOf(DataType))

### <Stereotype> [ValueRestriction](#_669b6c02099ad44a2386a5fbd8510a0b)

##### Extends

##### DataTypeDescription

The ValueRestriction stereotype applies to a DataType (Enumeration or Primitive type) that is a specialization of a general DataType. It defines restrictions on which values of the general DataType that are allowed as values of the specialized DataType. A ValueRestriction DataType is implemented in XML Schema as a simple type definition that is a restriction of the simple type that implements the general DataType. The attributes of the ValueRestriction are implemented as restriction facets.

ValueRestriction represents a NIEM type which is implemented in XML Schema as a simple type definition. Section 3.14 of [XML Schema Structures](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/) addresses simple type definitions in XML Schema; Sections 6.1.6.1, 7.2.1, 7.3, and 9.12.2 of [NIEM NDR v1.3](http://reference.niem.gov/niem/specification/naming-and-design-rules/1.3/) address simple type definitions in NIEM-conformant XML Schema. The variety of the simple type definition may be union, list, or atomic. As the ValueRestriction stereotype is a specialization of DataType, it may be applied to Enumeration. In this case, the ValueRestriction represents a NIEM code type, which is implemented in XML Schema as a simple type definition that contains multiple xsd:enumeration facets.

##### Properties

###### fractionDigits

A restriction on the value space of a numeric data type that places an upper  
 limit on the arithmetic precision of decimal values. The value space is restricted  
 to those values that can be represented lexically in decimal notation using at most  
 fractionDigits to the right of the decimal point. fractionDigits is implemented in  
 XML Schema as the value of the value attribute on the xsd:fractionDigits element,  
 the child of the xsd:restriction element which is the immediate child of the  
 xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### length

A restriction on the value space of a data type to values with a specific  
 length, where the units of length depends on the base type being restricted. For  
 String and URI values, the units are characters. For Binary values, the units are  
 octets. For lists, the length is the number of items in the list. length is  
 implemented in XML Schema as the value of the value attribute on the xsd:length  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### maxExclusive

The exclusive upper bound of the value space for a data type with ordered  
 values. The value of maxExclusive must be equal to some value in the value space of  
 the base data type or to the maxExclusive restriction of the base type (if it has  
 one). maxExclusive is implemented in XML Schema as the value of the value attribute  
 on the xsd:maxExclusive element, the child of the xsd:restriction element which is  
 the immediate child of the xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### maxInclusive

The inclusive upper bound of the value space for a data type with ordered  
 values. The value of maxInclusive must be equal to some value in the value space of  
 the base data type. maxInclusive is implemented in XML Schema as the value of the  
 value attribute on the xsd:maxInclusive element, the child of the xsd:restriction  
 element which is the immediate child of the xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### maxLength

A restriction on the value space of a data type to values with a specific  
 maximum length, where the units of length depends on the base type being restricted.  
 For String and URI values, the units are characters. For Binary values, the units  
 are octets. For lists, the length is the number of items in the list. maxLength is  
 implemented in XML Schema as the value of the value attribute on the xsd:maxLength  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element.

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### minExclusive

The exclusive lower bound of the value space for a data type with ordered  
 values. The value of minExclusive must be equal to some value in the value space of  
 the base data type or to the minExclusive restriction of the base type (if it has  
 one). minExclusive is implemented in XML Schema as the value of the value attribute  
 on the xsd:minExclusive element, the child of the xsd:restriction element which is  
 the immediate child of the xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### minInclusive

The inclusive lower bound of the value space for a data type with ordered  
 values. The value of minInclusive must be equal to some value in the value space of  
 the base data type. minInclusive is implemented in XML Schema as the value of the  
 value attribute on the xsd:minInclusive element, the child of the xsd:restriction  
 element which is the immediate child of the xsd:simpleType element..

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### minLength

A restriction on the value space of a data type to values with a specific  
 minimum length, where the units of length depends on the base type being restricted.  
 For String and URI values, the units are characters. For Binary values, the units  
 are octets. For lists, the length is the number of items in the list. minLength is  
 implemented in XML Schema as the value of the value attribute on the xsd:minLength  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element.

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### pattern

A constraint on the value space of a data type achieved by constraining the  
 value space to those values represented by literals that match each member of a set  
 of regular expressions. Each pattern must be a valid regular expression. pattern is  
 implemented in XML Schema as the value of the value attribute on the xsd:pattern  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..\* |
| **Ordering** |  |
| **Composition** | none |

###### totalDigits

Restricts the magnitude and arithmetic precision of values in the value space of  
 a numeric data type. The value space is restricted to those values that can be  
 represented lexically using at most totalDigits digits in decimal notation or at  
 most totalDigits digits for the coefficient, in scientific notation. totalDigits is  
 implemented in XML Schema as the value of the value attribute on the xsd:totalDigits  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element.

|  |  |
| --- | --- |
| **Type** | Integer |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### NDR [Rule 9-25]

Rule 10-17. Name of code type ends in CodeType

[Rule 10-17] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:let name="has-code-type-name" value="ends-with(@name, 'CodeType')"/>

<sch:let name="has-code-type-base" value="

exists(xs:simpleContent[

exists(xs:\*[local-name() = ('extension', 'restriction')

and (ends-with(@base, 'CodeSimpleType')

or ends-with(@base, 'CodeType'))])])"/>

<sch:assert test="$has-code-type-name = $has-code-type-base"

>A complex type definition MUST have a {name} that ends in 'CodeType' if and only if it has a {base type definition} of a code type or code simple type.</sch:assert>

</sch:rule>

</sch:pattern>

Using the qualifier Code (e.g. CodeType, CodeSimpleType) immediately identifies a type as representing a fixed list of codes. These types may be handled in specific ways, as lists of codes are expected to have their own lifecycles, including versions and periodic updates. Codes may also have responsible authorities behind them who provide concrete semantic bindings for the code values.

**context** ValueRestriction **inv:**

(  
 not(self.name.oclIsUndefined())  
 and  
 self.name.endsWith('CodeType')  
)   
implies  
self.general->union(self.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).oclAsType(Classifier))  
->forAll(c|c.name.endsWith('CodeType')or c.name.endsWith('CodeSimpleType'))

###### NDR3 [Rule 11-10] (REF,EXT). Code simple type has enumerations

Rule 11-10. Code simple type has enumerations

[Rule 11-10] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleType[exists(@name) and ends-with(@name, 'CodeSimpleType')]">

<sch:assert test="xs:restriction[ends-with(local-name-from-QName(resolve-QName(@base, .)), 'CodeSimpleType')]

or xs:restriction/xs:enumeration

or (for $union in xs:union,

$member-types in $union/@memberTypes return

some $member-type in tokenize(normalize-space($member-types), ' ') satisfies

ends-with(local-name-from-QName(resolve-QName($member-type, $union)), 'CodeSimpleType'))"

>A code simple type MUST be derived from a code simple type or have an enumeration facet.</sch:assert>

</sch:rule>

</sch:pattern>

Using the qualifier Code (e.g. CodeType, CodeSimpleType) immediately identifies a type as representing a fixed list of codes. These types may be handled in specific ways, as lists of codes are expected to have their own lifecycles, including versions and periodic updates. Codes may also have responsible authorities behind them who provide concrete semantic bindings for the code values.

**context** ValueRestriction **inv:**

(  
 self.\_'package'.stereotypedBy('InformationModel')  
 and   
 not(self.name.oclIsUndefined())  
 and   
 self.name.endsWith('CodeSimpleType')  
)   
implies  
 (  
 (self.oclIsKindOf(Enumeration) and self.oclAsType(Enumeration).ownedLiteral->notEmpty())  
 or  
 self.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->forAll(s|s.name.endsWith('CodeSimpleType'))  
 or  
 (self.stereotypedBy('Union') and self.clientDependency->select(d|d.stereotypedBy('UnionOf')).supplier->exists(s|s.name.endsWith('CodeSimpleType'))   
 )

###### NDR3 [Rule 11-32] (REF,EXT). Standard opening phrase for complex type

Rule 11-32. Standard opening phrase for complex type

[Rule 11-32] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType[ends-with(@name, 'AssociationType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type for (a relationship|an association)'))"

>The data definition for an association type SHOULD begin with the standard opening phrase "a datatype for (a relationship|an association)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType[ends-with(@name, 'AugmentationType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type (that supplements|for additional information about)'))"

>The data definition for an augmentation type SHOULD begin with the standard opening phrase "a data type (that supplements|for additional information about)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType[ends-with(@name, 'MetadataType')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type for (metadata about|information that further qualifies)'))"

>The data definition for a metadata type SHOULD begin with the standard opening phrase "a data type for (metdata about|information that further qualifies)...".</sch:report>

</sch:rule>

<sch:rule context="xs:complexType/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^a data type'))"

>The data definition for a type SHOULD begin with the standard opening phrase "a data type...".</sch:report>

</sch:rule>

</sch:pattern>

**context** ValueRestriction **inv:**

(  
 not(self.stereotypedBy('LocalVocabulary'))   
 and  
 self.namespace.stereotypedBy('InformationModel')   
 )   
implies   
self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^a data type'))

###### NDR3 [Rule 11-9] (REF,EXT). Name of a code simple type has standard suffix

Rule 11-9. Name of a code simple type has standard suffix

[Rule 11-9] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleType[exists(@name)

and (xs:restriction/xs:enumeration

or xs:restriction[ends-with(local-name-from-QName(resolve-QName(@base, .)), 'CodeSimpleType')])]">

<sch:assert test="ends-with(@name, 'CodeSimpleType')"

>A simple type definition schema component that has an enumeration facet or that is derived from a code type MUST have a name that ends in "CodeSimpleType".</sch:assert>

</sch:rule>

</sch:pattern>

**context** ValueRestriction **inv:**

(  
 self.\_'package'.stereotypedBy('InformationModel')  
 and  
 (self.ownedLiteral->notEmpty() or self.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->forAll(s|s.name.endsWith('CodeSimpleType')))  
)   
implies  
self.name.endsWith('CodeSimpleType')

###### ValueRestriction Generalization

A valuerestriction that is not an enumeration or list may only generalize the  
 same metatype. A valuerestriction that is an enumeration may not generalize a list.  
 A valuerestriction that is a list may not generalize an enumeration..

**context** ValueRestriction **inv:**

###### ValueRestrictionGeneralization

A ValueRestriction DataType must be the special classifier in a single Generalization whose general classifier is also a DataType.

**context** ValueRestriction **inv:**

self.base\_DataType.generalization.general->size()=1

## Profile : NIEM\_PIM\_Profile

### Overview

The NIEM PIM Profile comprises stereotypes that are used in NIEM PIMs but not NIEM  
 PSMs. Further, the NIEM PIM Profile imports the NIEM Common Profile and, therefore,  
 includes all the stereotypes and metaclasses covered by that profile. In addition, the  
 UML metamodel subset covered by the NIEM PIM Profile also includes the metaclasses  
 Association and AssociationClass, even though they are not specifically extended by any  
 stereotypes in the profile.

### <Stereotype> [Augments](#_2937b2c47cb1edf3656df071be47a7bc)

##### Extends

##### GeneralizationDescription

An Augments Generalization specifies that the special Class is an augmentation  
 type that is restricted to apply to instances of the general Class.

### <Stereotype> [InformationModel](#_1498be86b30971330c90f909c07d3938)

##### Generalization

##### [Namespace](#_a3b43d75feafe90b105d1a836eb3d6a2)Description

The contents of an InformationModel Package provide a platform-independent  
 perspective on the structure of information to be exchanged in NIEM messages. Such a  
 model is always taken to represent a NIEM namespace, but it may also be given a  
 default purpose as modeled, independent of the implementation of that namespace. This  
 allows a modeler to identify the intended purposes (e.g., reference, subset, exchange,  
 etc.) of various information models within a set, without having to create a complete  
 MPD model for the set.

##### Properties

###### defaultPurpose

The default purpose for which an information model is intended. If an  
 InformationModel Package is modeled as being included as an artifact in an MPD,  
 then, unless otherwise specified, the purpose of the artifact is by default taken to  
 be the schema purpose code corresponding to the value of the defaultPurpose  
 attribute.

|  |  |
| --- | --- |
| **Type** | DefaultPurposeCode |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### NDR [Rule 6-55]

**[Rule 6-55] (REF)** Within the schema, the element xsd:complexContent MUST have as an immediate child the element xsd:extension.

**Rationale** NIEM does not support, as conformant, the use of complex type restriction. NIEM defines a language, in which specific content is allowed. It does not specify messages that forbid content. Such restrictions may be performed in nonconformant schemas or within constraint schemas or other artifacts of constraint.

Note that XML Schema requires use of the attribute base on xsd:extension.

Note also that the applicability allows for the use of restriction in subset schemas, extension schemas, exchange schemas, and constraint schemas.

**context** InformationModel **inv:**

(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).supplierDependency->select(d|d.stereotypedBy('Restriction'))->isEmpty()

###### NDR [Rule 6-57]

**[Rule 6-57] (EXT)** Within the schema, given an element xsd:complexContent with a child xsd:restriction owning an attribute base, the attribute base MUST have a value that resolves to the name of a complex type that is a NIEM-conformant component.

**[Rationale]** This ensures that a CCC defined through restriction has well-defined semantics.

**context** InformationModel **inv:**

not(self.isConformant)  
implies  
   
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).supplierDependency->select(d|d.stereotypedBy('Restriction'))->isEmpty()

###### NDR [Rule 9-26]

**[Rule 9-26] (REF, SUB, EXT)** Within the schema, any type definition which has a base type definition of a code type or which is transitively based on a code type SHALL have a name that uses the representation term qualifier Code.

**Rationale** This expands the use of the representation term qualifier Code to any type based on a code list.

fier Code to any type based on a code list.

**context** InformationModel **inv:**

self.base\_Package.ownedType   
 ->select(t|  
 t.clientDependency->select(d|d.stereotypedBy('XSDSimpleContent')).supplier->exists(s|s.oclIsKindOf(Enumeration))  
 or (  
 t.oclIsKindOf(Classifier)   
 and t.oclAsType(Classifier).allParents().clientDependency->select(d|d.stereotypedBy('XSDSimpleContent')).supplier->exists(s|s.oclIsKindOf(Enumeration))  
 )  
 )  
 ->forAll(enumerationDerivedType|enumerationDerivedType.name.match('.\*Code.\*'))

###### NDR [Rule 9-32]

**[Rule 9-32] (REF, SUB, EXT)** Within the schema, the name of an association element SHALL use the representation term qualifier Association.

**Rationale** Using the qualifier Association immediately identifies an element as representing an association.

ment as representing an association.

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute->select(a|not(a.type.oclIsUndefined()))  
 ->forAll(a|a.type.stereotypedBy('AssociationType') implies a.name.match('.\*Association.\*'))

###### NDR [Rule 9-33]

**[Rule 9-33] (REF, SUB, EXT)** Within the schema, the name of an augmentation element SHALL use the representation term Augmentation.

**Rationale** Using the qualifier Augmentation immediately identifies an element as representing an augmentation.

representing an augmentation.

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute->select(a|not(a.type.oclIsUndefined()))  
 ->forAll(a|a.type.stereotypedBy('AugmentationType') implies a.name.match('.\*Augmentation.\*'))

###### NDR [Rule 9-34]

**[Rule 9-34] (REF, SUB, EXT)** Within the schema, the name of a metadata element SHALL use the representation term Metadata.

**Rationale** Using the qualifier Metadata immediately identifies an element as representing metadata.

ent as representing metadata.

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute  
->forAll(a|(not(a.type.oclIsUndefined()) and a.type.stereotypedBy('MetadataType')) implies a.name.match('.\*Metadata.\*'))

###### NDR3 [Rule 10-1] (REF,EXT). Complex type has a category

[[NDR] Rule 10-1](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-1), Complex type has a category (REF, EXT): Section 10.1, Categories of NIEM type definitions

**context** InformationModel **inv:**

The constraint is satisfied during provisioning, which produce one of the NDR defined complex type categories based on explicit or implicit model specifications.

###### NDR3 [Rule 10-2] (REF,EXT). Object type with complex content is derived from object type

[[NDR] Rule 10-2](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-2), Object type with complex content is derived from object type (REF, EXT): Section 10.2.1.1, Object types with complex content

**context** InformationModel **inv:**

The constraint is satisfied during provisioning, which produce derivation of each Object Type from another Object Type or, if not modeled explicitly, from structures:ObjectType.

###### NDR3 [Rule 10-43] (REF,EXT). Schema component name composed of English words

[[NDR] Rule 10-43](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-43), Schema component name composed of English words (REF, EXT): Section 10.8, Naming rules

**context** InformationModel **inv:**

This rule is not readily computational.

###### NDR3 [Rule 10-44] (REF,EXT). Schema component names have only specific characters

[[NDR] Rule 10-44](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-44), Schema component names have only specific characters (REF, EXT): Section 10.8, Naming rules

**context** InformationModel **inv:**

self.base\_Package.ownedType ->select(t|not(t.stereotypedBy('PropertyHolder')))  
 ->forAll(schemaComponent|schemaComponent.name.match('[\\w|\\-]\*'))   
and   
self.base\_Package.ownedType ->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute  
 ->forAll(schemaComponent|schemaComponent.name.match('[\\w|\\-]\*'))

###### NDR3 [Rule 10-45] (REF,EXT). Hyphen in component name is a separator

[[NDR] Rule 10-45](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-45), Hyphen in component name is a separator (REF, EXT): Section 10.8, Naming rules

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-46](REF,EXT). Names use camel case

[[NDR] Rule 10-46](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-46), Names use camel case (REF, EXT): Section 10.8.1, Character case

**context** InformationModel **inv:**

Rule is not reliably computational.

###### NDR3 [Rule 10-48] (REF,EXT). Name of schema component other than attribute begins with upper case letter

[[NDR] Rule 10-48](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-48), Name of schema component other than attribute begins with upper case letter (REF, EXT): Section 10.8.1, Character case

**context** InformationModel **inv:**

self.base\_Package.ownedType  
 ->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier )  
 ->forAll(c|  
 c.oclAsType(NamedElement)->asSet()  
 ->union(c.attribute  
 ->select(a|   
 not(a.stereotypedBy('XSDProperty'))  
 or   
 not(a.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)  
 )  
 .oclAsType(NamedElement)->asSet()  
 )  
 ->forAll(n|not(n.name.oclIsUndefined()or (n.name='')) implies (n.name.firstToUpper()=n.name))  
)

###### NDR3 [Rule 10-49] (REF,EXT). Names use common abbreviations

[[NDR] Rule 10-49](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-49), Names use common abbreviations (REF, EXT): Section 10.8.2, Use of acronyms and abbreviations

**context** InformationModel **inv:**

self.base\_Package.ownedType  
 ->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)  
 ->forAll(c|  
 c.oclAsType(NamedElement)->asSequence()  
 ->union(c.attribute.oclAsType(NamedElement))  
 ->forAll(n|not(n.name.oclIsUndefined()or (n.name=''))   
 implies   
 (n.name.match('.\*Identifier.\*')or n.name.match('.\*UniformResourceIdentifier.\*'))  
 )  
)

###### NDR3 [Rule 10-4] (REF,EXT). Only object type has RoleOf element

[[NDR] Rule 10-4](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-4), Only object type has RoleOf element (REF, EXT): Section 10.2.2, Role types and roles

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)  
 ->select(t|t.stereotypedBy('MetadataType')or t.stereotypedBy('AssociationType')or t.stereotypedBy('AugmentationType') or t.oclIsKindOf(AssociationClass)).attribute  
 ->forAll(a|not(a.stereotypedBy('RoleOf') or a.name.startsWith('RoleOf')))

###### NDR3 [Rule 10-50] (REF,EXT). Local term declaration is local to its schema document

[[NDR] Rule 10-50](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-50), Local term declaration is local to its schema document (REF, EXT): Section 10.8.2.1, Use of Acronyms, Initialisms, Abbreviations, and Jargon

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-51] (REF,EXT). Local terminology interpretation

[[NDR] Rule 10-51](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-51), Local terminology interpretation (REF, EXT): Section 10.8.2.1, Use of Acronyms, Initialisms, Abbreviations, and Jargon

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-52] (REF,EXT). Singular form is preferred in name

[[NDR] Rule 10-52](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-52), Singular form is preferred in name (REF, EXT): Section 10.8.3, Word forms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-53] (REF,EXT). Present tense is preferred in name

[[NDR] Rule 10-53](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-53), Present tense is preferred in name (REF, EXT): Section 10.8.3, Word forms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-54] (REF,EXT). Name does not have nonessential words

[[NDR] Rule 10-54](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-54), Name does not have nonessential words (REF, EXT): Section 10.8.3, Word forms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-55] (REF,EXT). Component name follows pattern

[[NDR] Rule 10-55](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-55), Component name follows pattern (REF, EXT): Section 10.8, Naming rules

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-56] (REF,EXT). Object-class term identifies concrete category

[[NDR] Rule 10-56](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-56), Object-class term identifies concrete category (REF, EXT): Section 10.8.4, Object-class term

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-57] (REF,EXT). Property term describes characteristic or subpart

[[NDR] Rule 10-57](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-57), Property term describes characteristic or subpart (REF, EXT): Section 10.8.5, Property term

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-58] (REF,EXT). Name may have multiple qualifier terms

[[NDR] Rule 10-58](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-58), Name may have multiple qualifier terms (REF, EXT): Section 10.8.6, Qualifier terms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-59] (REF,EXT). Name has minimum necessary number of qualifier terms

[[NDR] Rule 10-59](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-59), Name has minimum necessary number of qualifier terms (REF, EXT): Section 10.8.6, Qualifier terms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-5] (REF,EXT,INS). RoleOf elements indicate the base types of a role type

[[NDR] Rule 10-5](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-5), RoleOf elements indicate the base types of a role type (REF, EXT, INS): Section 10.2.2, Role types and roles

**context** InformationModel **inv:**

This rule is definitional.

###### NDR3 [Rule 10-60] (REF,EXT). Order of qualifies is not significant

[[NDR] Rule 10-60](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-60), Order of qualifies is not significant (REF, EXT): Section 10.8.6, Qualifier terms

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 10-61] (REF,EXT). Redundant term in name is omitted

[[NDR] Rule 10-61](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-61), Redundant term in name is omitted (REF, EXT): Section 10.8.7, Representation terms

**context** InformationModel **inv:**

The constraint can not be expressed easily in OCL.

###### NDR3 [Rule 10-65](REF,EXT). Machine-readable annotations are valid

XML Schema provides *application information* schema components to provide for automatic processing and machine-readable content for schemas, as described by XML Schema Structures, Section 3.13.2, XML Representation of Annotation Schema Components. NIEM utilizes application information to convey information that is outside schema definition and outside human-readable text definitions. NIEM uses application information to convey high-level data model concepts and additional syntax to support the NIEM conceptual model and validation of NIEM-conformant XML instances.

XML elements, attributes, and text content may appear as machine-readable annotations within an XML Schema document. The methods provided by XML Schema for machine-readable annotations are:

1. An element in the XML Schema namespace (e.g., xs:schema, xs:complexType, …) may carry attributes from namespaces other than the XML Schema namespace. By the rules of XML Schema, any XML Schema element may have attributes that are from other namespaces. These attributes do not participate in XML Schema validation, but may carry information useful to tools that process schemas. In XML Schema Structure, these attributes are described in the XML Representation summary of XML Schema elements as {any attributes with non-schema namespace . . .}.
2. XML Schemas may include xs:appinfo elements, which may include arbitrary XML content. This XML does not participate in XML Schema validation, but may communicate useful information to schema readers or processors. These are described by XML Schema Structure in Section 3.13.1, The Annotation Schema Component and Section 3.13.2, XML Representation of Annotation Schema Components.

This document defines the term marchine-readable annotation to normatively refer to such annotations within XML Schema documents:

[Definition: *machine-readable annotation*]

An information item within a schema is defined to be a machine-readable annotation when all of the following are true:

* 1. It is one of the following:
     1. It is an element information item that is a child of an xs:appinfo element.
     2. It is an attribute information item that satisfies the {any attributes with non- schema namespace . . .} clause within the declaration of an XML Schema element.
  2. The namespace name property of the item is not the XML namespace, XML Schema namespace, or the XSI namespace.

Attributes from the XML namespace, the XML Schema namespace, and the XML Schema instance namespace have special meanings within XML Schema, and may have effects on validation, and so are not considered machine-readable annotations.

[Definition: *application information*]

A component is said to have **application information** of some element *$element* when the XML Schema element that defines the component has an immediate child element xs:annotation, which has an immediate child element xs:appinfo, which has as an immediate child the element *$element*.

If a component is described as having some application information, this means that the elements in question appear in an xs:appinfo annotation of the element that defines the component.

The majority of uses of application information from the appinfo namespace are described in the modeling rules for the specific component.

Rule 10-65. Machine-readable annotations are valid

[Rule 10-65] (REF, EXT) (Constraint)

Every element information item or attribute information item that appears as a machine-readable annotation in a schema MUST be a valid instance, according to its specification.

The specification for an element or attribute may be via an XML Schema, a Schematron schema, via a DTD, by some other specification, or by other means. This rule is intended to allow NIEM schema developers to leverage relevant vocabularies without being limited by the vocabulary’s method of specification, while ensuring that developers do not subvert or misuse those vocabularies.

**context** InformationModel **inv:**

The constraint is realized through provisioning:  
 there are no NIEM-UML constructs related specifically to machine-readable annotations;  
the production of machine-readable annotations is based on the mapping of specific NDR rules to target schema annotations.

###### NDR3 [Rule 10-67] (REF,EXT). Deprecated annotates schema component

Rule 10-67. Deprecated annotates schema component

[Rule 10-67] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="\*[exists(@appinfo:deprecated)]">

<sch:assert test="namespace-uri-from-QName(node-name(.)) = xs:anyURI('http://www.w3.org/2001/XMLSchema')"

>The attribute appinfo:deprecated MUST be owned by an element with a namespace name <namespace-uri-for-prefix>xs</namespace-uri-for-prefix>.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The constraint is realized through provisioning:  
A NamedElement with applied Stereotype Deprecated will create the appinfo:deprecated on the target schema component.

###### NDR3 [Rule 10-68] (REF,EXT). External import indicator annotates import

Rule 10-68. External import indicator annotates import

[Rule 10-68] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="\*[exists(@appinfo:externalImportIndicator)]">

<sch:assert test="exists(self::xs:import)"

>The attribute {http://release.niem.gov/niem/appinfo/3.0/}externalImportIndicator MUST be owned by an element xs:import.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The constraint is realized through provisioning:  
A provisioned xs:import will own an appinfo:externalImportIndicator if the import InformationModel has a defaultPurpose of external.

###### NDR3 [Rule 10-6] (INS). Instance of RoleOf element indicates a role object

[[NDR] Rule 10-6](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-6), Instance of RoleOf element indicates a role object (INS): Section 10.2.2, Role types and roles

**context** InformationModel **inv:**

This rule is definitional.

###### NDR3 [Rule 10-76](REF,EXT,INS). Use structures as specified

NIEM provides the structures schema that contains base types for types defined in NIEM- conformant schemas. It provides base elements to act as heads for substitution groups. It also provides attributes that provide facilities not otherwise provided by XML Schema. These structures should be used to augment XML data. The structures provided are not meant to replace fundamental XML organization methods; they are intended to assist them.

[Definition: *structures namespace*]

The **structures namespace** is the namespace represented by the URI http://release.niem.gov/niem/structures/3.0/.

The structures namespace is a single namespace, separate from namespaces that define NIEM-conformant data. This document refers to this content via the prefix structures.

Rule 10-76. Use structures as specified

[Rule 10-76] (REF, EXT, INS) (Constraint)

The schema or instance MUST use content within the NIEM structures namespace as specified in this document and ONLY as specified by this document.

This rule further enforces uniformity and consistency by mandating use of the NIEM structures namespace as is, without modification. Users are not allowed to insert types, attributes, etc. that are not specified by this document.

**context** InformationModel **inv:**

The constraint is realized through provisioning:  
The structures namespace is not part of the NIEM-UML model, all usages of the namespace are provisioned according to the NDR rules governing that namespace.

###### NDR3 [Rule 11-24] (REF,EXT). Schema uses only known attribute groups

Rule 11-24. Schema uses only known attribute groups

In conformant schemas, use of attribute groups is restricted. The only attribute group defined by NIEM for use in conformant schemas is structures:SimpleObjectAttributeGroup. This attribute group provides the attributes necessary for IDs, references, metadata, and relationship metadata. In addition, there are attributes defined by ISM and NTK namespaces, which may be used in conformant schemas.

[Rule 11-24] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attributeGroup[@ref]">

<sch:assert test="some $ref in resolve-QName(@ref, .) satisfies (

$ref = xs:QName('structures:SimpleObjectAttributeGroup')

or namespace-uri-from-QName($ref) = (xs:anyURI('urn:us:gov:ic:ism'),

xs:anyURI('urn:us:gov:ic:ntk')))"

>An attribute group reference MUST be structures:SimpleObjectAttributeGroup or have the IC-ISM or IC-NTK namespace.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Expression of this constraint as OCL has been deferred.

###### NDR3 [Rule 11-25] (REF,EXT). Data definition does not introduce ambiguity

Rule 11-25. Data definition does not introduce ambiguity

[Rule 11-25] (REF, EXT) (Constraint)

Words or synonyms for the words within a data definition MUST NOT be reused as terms in the corresponding component name if those words dilute the semantics and understanding of, or impart ambiguity to, the entity or concept that the component represents.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-26] (REF,EXT). Object class has only one meaning

Rule 11-26. Object class has only one meaning

[Rule 11-26] (REF, EXT) (Constraint)

An object class MUST have one and only one associated semantic meaning (i.e., a single word sense) as described in the definition of the component that represents that object class.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-27] (REF,EXT). Data definition of a part does not redefine the whole

Rule 11-27. Data definition of a part does not redefine the whole

[Rule 11-27] (REF, EXT) (Constraint)

An object class MUST NOT be redefined within the definitions of the components that represent properties or subparts of that entity or class.

Data definitions should be concise, precise, and unambiguous without embedding additional definitions of data elements that have already been defined once elsewhere (such as object classes).  ISO 11179-4 says that definitions should not be nested inside other definitions. Furthermore, a data dictionary is not a language dictionary. It is acceptable to reuse terms (object class, property term, and qualifier terms) from a component name within its corresponding definition to enhance clarity, as long as the requirements and recommendations of ISO 11179-4 are not violated. This further enhances brevity and precision.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-28] (REF,EXT). Do not leak representation into data definition

Rule 11-28. Do not leak representation into data definition

[Rule 11-28] (REF, EXT) (Constraint)

A data definition MUST NOT contain explicit representational or data typing information such as number of characters, classes of characters, range of mathematical values, etc., unless the very nature of the component can be described only by such information.

A component definition is intended to describe semantic meaning only, not representation or structure. How a component with simple content is represented is indicated through the representation term, but the primary source of representational information should come from the XML Schema definition of the types themselves. A developer should try to keep a component’s data definition decoupled from its representation.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-29] (REF,EXT). Data definition follows 11179-4 requirements

Rule 11-29. Data definition follows 11179-4 requirements

[Rule 11-29] (REF, EXT) (Constraint)

Each [data definition](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_data_definition) MUST conform to the requirements for data definitions provided by ISO 11179-4 Section 5.2, *Requirements*.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-30] (REF,EXT). Data definition follows 11179-4 recommendations

Rule 11-30. Data definition follows 11179-4 recommendations

[Rule 11-30] (REF, EXT) (Constraint)

Each [data definition](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_data_definition) SHOULD conform to the recommendations for data definitions provided by ISO 11179-4 Section 5.3, *Recommendations*.

**context** InformationModel **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-34] (REF,EXT). Same namespace means same components

[[NDR] Rule 11-34](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-34), Same namespace means same components (REF, EXT): Section 11.7.1, xs:schema document element restrictions

**context** InformationModel **inv:**

Constraint expression as OCL is deferred.

###### NDR3 [Rule 11-35] (REF,EXT). Different version means different view

[[NDR] Rule 11-35](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-35), Different version means different view (REF, EXT): Section 11.7.1, xs:schema document element restrictions

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 11-36] (SET). Reference schema imports reference schema

Rule 11-36. Reference schema imports reference schema

[Rule 11-36] (SET) (Constraint)

<sch:pattern>

<sch:rule context="xs:import[

nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))

and exists(@namespace)

and empty(@appinfo:externalImportIndicator)

and not(xs:anyURI(@namespace) = (xs:anyURI('http://release.niem.gov/niem/structures/3.0/'),

xs:anyURI('http://www.w3.org/XML/1998/namespace')))]">

<sch:assert test="some $schema in nf:resolve-namespace(., @namespace) satisfies

nf:has-effective-conformance-target-identifier($schema, xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))"

>A namespace imported as conformant from a reference schema document MUST identify a namespace defined by a reference schema document.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

(  
 (self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)  
 or  
 (self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset)  
)   
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general.namespace.oclAsType(NamedElement)->asSet()  
 ->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).namespace.oclAsType(NamedElement)->asSet())  
 ->union(c.attribute->select(a|not(a.type.oclIsUndefined())).type.namespace.oclAsType(NamedElement)->asSet())  
 ->union(c.attribute.clientDependency->select(d|d.stereotypedBy('References')).supplier->select(s|s.oclIsKindOf(Property)).namespace.namespace.oclAsType(NamedElement)->asSet())  
 .appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel).defaultPurpose->forAll(p|(p=DefaultPurposeCode::reference)or(p=DefaultPurposeCode::subset))  
)

###### NDR3 [Rule 11-37] (SET). Extension schema document imports reference or extension schema

Rule 11-37. Extension schema document imports reference or extension schema

[Rule 11-37] (SET) (Constraint)

<sch:pattern>

<sch:rule context="xs:import[

nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument'))

and exists(@namespace)

and empty(@appinfo:externalImportIndicator)

and not(xs:anyURI(@namespace) = (xs:anyURI('http://release.niem.gov/niem/structures/3.0/'),

xs:anyURI('http://www.w3.org/XML/1998/namespace')))]">

<sch:assert test="some $schema in nf:resolve-namespace(., @namespace) satisfies (

nf:has-effective-conformance-target-identifier($schema, xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))

or nf:has-effective-conformance-target-identifier($schema, xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument')))"

>A namespace imported as conformant from an extension schema document MUST identify a namespace defined by a reference schema document or an extension schema document.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::extension)  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(c|  
 c.general.namespace.oclAsType(NamedElement)->asSet()  
 ->union(c.clientDependency->select(d|d.stereotypedBy('Restriction')).supplier->select(s|s.oclIsKindOf(Classifier)).namespace.oclAsType(NamedElement)->asSet())  
 ->union(c.attribute->select(a|not(a.type.oclIsUndefined())).type.namespace.oclAsType(NamedElement)->asSet())  
 ->union(c.attribute.clientDependency->select(d|d.stereotypedBy('References')).supplier->select(s|s.oclIsKindOf(Property)).namespace.namespace.oclAsType(NamedElement)->asSet())  
 ->select(p|p.stereotypedBy('InformationModel'))  
 .appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel).defaultPurpose->forAll(p|  
 (p=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)  
 or(p=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset)  
 or(p=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::extension))  
)

###### NDR3 [Rule 11-38] (REF,EXT). Structures imported as conformant

Rule 11-38. Structures imported as conformant

[Rule 11-38] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:import[exists(@namespace)

and xs:anyURI(@namespace) = xs:anyURI('http://release.niem.gov/niem/structures/3.0/')]">

<sch:assert test="empty(@appinfo:externalImportIndicator)"

>An import of the structures namespace MUST NOT be labeled as an external import.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint realized by provisioning;  
there is no explicit modeling of xs:import in NIEM-UML; xs:import is produced as required, according to this and other NDR rules

###### NDR3 [Rule 11-39] (REF,EXT). XML namespace imported as conformant

Rule 11-39. XML namespace imported as conformant

[Rule 11-39] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:import[exists(@namespace)

and xs:anyURI(@namespace) = xs:anyURI('http://www.w3.org/XML/1998/namespace')]">

<sch:assert test="empty(@appinfo:externalImportIndicator)"

>An import of the XML namespace MUST NOT be labeld as an external import.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint realized by provisioning;  
there is no explicit modeling of xs:import in NIEM-UML; xs:import is produced as required, according to this and other NDR rules

###### NDR3 [Rule 11-40] (SET). Each namespace may have only a single root schema in a schema set

[[NDR] Rule 11-40](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-40), Each namespace may have only a single root schema in a schema set (SET): Section 11.8, Schema assembly

**context** InformationModel **inv:**

Expressing Constraint in OCL has been deferred

###### NDR3 [Rule 11-41] (REF,EXT). Consistently marked namespace imports

[[NDR] Rule 11-41](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-41), Consistently marked namespace imports (REF, EXT): Section 11.8, Schema assembly

**context** InformationModel **inv:**

Constraint ensured by provisioning:  
xs:import is not in the NIEM-UML Model, it is created during provisioning, which consistently constructs the externalImportIndicator based on the tag values in the referenced InformationModel.

###### NDR3 [Rule 12-10] (INS). Values of structures:metadata refer to values of structures:id

Rule 12-10. Values of structures:metadata refer to values of structures:id

[Rule 12-10] (INS) (Constraint)

Given that each IDREF in the value of an attribute structures:metadata must match the value of an ID attribute on some element in the XML document, that ID attribute MUST be an occurrence of the attribute structures:id.

**context** InformationModel **inv:**

Constraint is realized during provisioning of XML instance documents.

###### NDR3 [Rule 12-11] (INS). Value of structures:relationshipMetadata refers to value of structures:id

Rule 12-11. Value of structures:relationshipMetadata refers to value of structures:id

[Rule 12-11] (INS) (Constraint)

Given that each IDREF in the value of an attribute structures:relationshipMetadata must match the value of an ID attribute on some element in the XML document, that ID attribute MUST be an occurrence of the attribute structures:id.

**context** InformationModel **inv:**

Constraint realized during provisioning of XML instance documents.

###### NDR3 [Rule 12-12] (INS). structures:metadata and structures:relationshipMetadata refer to metadata elements

Rule 12-12. structures:metadata and structures:relationshipMetadata refer to metadata elements

[Rule 12-12] (INS) (Constraint)

Each element referenced by an attribute structures:metadata or an attribute structures:relationshipMetadata MUST have [element declaration] that is a [metadata element declaration](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_metadata_element_declaration).

**context** InformationModel **inv:**

Constraint realized by provisioning of the XML Instance Documents.

###### NDR3 [Rule 12-13] (INS). Attribute structures:metadata references metadata element

Rule 12-13. Attribute structures:metadata references metadata element

[Rule 12-13] (INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[exists(@structures:metadata)]">

<sch:assert test="every $metadata-ref in tokenize(normalize-space(@structures:metadata), ' ') satisfies

exists(//\*[exists(@structures:id[. = $metadata-ref])

and ends-with(local-name(), 'Metadata')])"

>Each item in the value of an attribute structures:metadata MUST appear as the value of an attribute structures:id with an owner element that is a metadata element.</sch:assert>

</sch:rule>

</sch:pattern>

Note that this will NOT diagnose a scenario in which the element with a name ending in Metadata is an external element; additional tests would be required to catch that.

**context** InformationModel **inv:**

Constraint realized during provisioning of XML instance documents.

###### NDR3 [Rule 12-14] (INS). Attribute structures:relationshipMetadata references metadata element

Rule 12-14. Attribute structures:relationshipMetadata references metadata element

[Rule 12-14] (INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[exists(@structures:relationshipMetadata)]">

<sch:assert test="every $metadata-ref in tokenize(normalize-space(@structures:relationshipMetadata), ' ') satisfies

exists(//\*[exists(@structures:id[. = $metadata-ref])

and ends-with(local-name(), 'Metadata')])"

>Each item in the value of an attribute structures:relationshipMetadata MUST appear as the value of an attribute structures:id with an owner element that is a metadata element.</sch:assert>

</sch:rule>

</sch:pattern>

Note that this will NOT diagnose a scenario in which the element with a name ending in Metadata is an external element; additional tests would be required to catch that.

**context** InformationModel **inv:**

Constraint realized by provisioning of XML Instance Documents.

###### NDR3 [Rule 12-15] (INS). Metadata is applicable to element

Rule 12-15. Metadata is applicable to element

[Rule 12-15] (INS) (Constraint)

Given that an element *$SUBJECT-ELEMENT* uses a metadata element *$METADATA-ELEMENT* through a value in either an attribute structures:metadata or an attribute structures:relationshipMetadata, the element *$SUBJECT-ELEMENT* MUST be an applicable element for *$METADATA-ELEMENT*.

**context** InformationModel **inv:**

Constraint realized when provisioning XML Instance Document.

###### NDR3 [Rule 12-1] (INS). Instance must be schema-valid

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

NIEM does not require a specific encoding or specific requirements for the XML prologue, except as specified by XML.

Rule 12-1. Instance must be schema-valid

[Rule 12-1] (INS) (Constraint)

The XML document MUST be schema-valid, as assessed against a [conformant schema document set](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_conformant_schema_document_set), composed of authoritative, comprehensive schema documents for the relevant namespaces.

The schemas that define the exchange must be authoritative. Each is the reference schema or extension schema for the namespace it defines. Application developers may use other schemas for various purposes, but for the purposes of determining conformance, the authoritative schemas are relevant.

This rule should not be construed to mean that XML validation must be performed on all XML instances as they are served or consumed; only that the XML instances validate if XML validation is performed. The XML Schema component definitions specify XML documents and element information items, and the instances should follow the rules given by the schemas, even when validation is not performed.

NIEM embraces the use of XML Schema instance attributes, including xsi:type, xsi:nil, and xsi:schemaLocation, as specified by XML Schema Structures.

**context** InformationModel **inv:**

Constraint can not be easily expressed in OCL, the constraint must be enforced by an XML Schema Document Validation tool.

###### NDR3 [Rule 12-2] (INS). Element with structures:ref does not have content

Content elements are sufficient to represent data that takes the form of a tree. However, use of content elements has limitations; expression of all relationships via element containment is not always possible. Situations that cause problems include:

* Cycles: the relationships transitively held by an object include a relationship to itself.
* For example, suppose that object 1 has a relationship to object 2 and object 2 has a relationship to object 1. This is not a tree, and so needs some representation other than a simple tree.
* Reuse: multiple objects have a relationshp to a common object.
* For example, suppose object 1 has a relationship to object 2 and object 3 has a relationship to object 2. Expressed via containment, this would result in a duplicate of object 2.

A method that solves this problem is the use of references. In a C or assembler, you could use a pointer. In C++, a reference might be used. In Java, a reference value might be used. The method defined by the XML standard is the use of ID and IDREF. An IDREF refers to an ID. NIEM uses this method and assigns to it specific semantics.

Naive solutions to these problems that use only content elements require techniques such as repeating data and identifying and excluding duplicate data; these operation entail the use of excess storage and processing time.

It is good to avoid these problems; in order to avoid them, NIEM allows [reference elements](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#definition_reference_element). A reference element expresses a relationship to another object by using a reference attribute, structures:ref.

NIEM XML instances use IDREF attributes to establish links between XML elements.

[Definition: *reference element*]

A **reference element** is an element information item that has an attribute structures:ref. A reference element refers to its value by reference, instead of carrying it as content.

Rule 12-2. Element with structures:ref does not have content

[Rule 12-2] (INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[@structures:ref]">

<sch:assert test="empty(element() | text())"

>An element that has attribute structures:ref MUST NOT have element or text content.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint is realized during provisioning of instance documents, if any. Provisioning of an element with @structures:ref attribute will not have element content.

###### NDR3 [Rule 12-3] (INS). Attribute structures:ref must reference structures:id

Rule 12-3. Attribute structures:ref must reference structures:id

[Rule 12-3] (INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[@structures:ref]">

<sch:let name="ref" value="@structures:ref"/>

<sch:assert test="exists(//\*[@structures:id = $ref])"

>The value of an attribute structures:ref MUST match the value of an attribute structures:id of some element in the XML document.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint is realized during provisioning of instance documents, if any. Any @structures:ref will reference an element with the same value in an @structures:id.

###### NDR3 [Rule 12-3] (INS). Attribute structures:ref must reference structures:id2

Rule 12-3. Attribute structures:ref must reference structures:id

[Rule 12-3] (INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[@structures:ref]">

<sch:let name="ref" value="@structures:ref"/>

<sch:assert test="exists(//\*[@structures:id = $ref])"

>The value of an attribute structures:ref MUST match the value of an attribute structures:id of some element in the XML document.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint is realized during provisioning of instance documents, if any. Any @structures:ref will reference an element with the same value in an @structures:id.

###### NDR3 [Rule 12-4] (INS). Linked elements have same validation root

Rule 12-4. Linked elements have same validation root

[Rule 12-4] (INS) (Constraint)

Given that:

* *$element* is an element information item
* *$element* has attribute structures:ref with value *$ref*
* *$element* has property validation context with value called the *referencing element validation root*
* *$target* is an element information item
* *$target* has attribute structures:id with value *$ref*
* *$target* has property validation context with value called the *referenced element validation root*

Every element that has an attribute structures:ref MUST have a referencing element validation root that is equal to the referenced element validation root.

**context** InformationModel **inv:**

Constraint is realized during provisioning of instance documents, if any.

###### NDR3 [Rule 12-5] (INS). Attribute structures:ref references element of correct type

Rule 12-5. Attribute structures:ref references element of correct type

[Rule 12-5] (INS) (Constraint)

Given that:

* *$element* is an element information item
* *$element* has attribute structures:ref with value *$ref*
* *$element* has property element declaration with value *$element-declaration*
* *$element-declaration* has property type definition with value called the *referencing element type definition*
* *$target* is an element information item
* *$target* has attribute structures:id with value *$ref*
* *$target* has property type definition with value called the *referenced element type definition*

Every element that has an attribute structures:ref MUST have a referencing element type definition that is validly derived from the referenced element type definition.

**context** InformationModel **inv:**

Constraint is realized during provisioning of instance documents, if any.

###### NDR3 [Rule 12-6] (INS). Reference and content elements have the same meaning

An important aspect of the use of NIEM reference and content elements is that they have the same meaning. The use of a content element versus a reference element is merely for convenience and ease of serialization. There is no change in meaning or semantics between content and reference elements.

Any claim that content elements represent composition while reference elements represent aggregation is incorrect. Any life cycle dependency is not expressed through the use of content and reference elements, and is not an explicit part of the NIEM model.

Rule 12-6. Reference and content elements have the same meaning

[Rule 12-6] (INS) (Interpretation)

There MUST NOT be any difference in meaning between a relationship established via an element declaration instantiated as a content element and that element declaration instantiated as a reference element.

There is no difference in meaning between relationships established by content elements and those established by reference elements. They are simply two mechanisms for expressing connections between objects. Neither mechanism implies that properties are intrinsic or extrinsic; such characteristics must be explicitly stated in property definitions.

Being of type xs:ID and xs:IDREF, validating schema parsers will perform certain checks on the values of structures:id and structures:ref. Specifically, no two IDs may have the same value. This includes structures:id and other IDs that may be used within an XML document. Also, any value of structures:ref must also appear as the value of an ID.

NIEM-conformant data instances may use content elements and reference elements as needed, to represent the meaning of the fundamental data. There is no difference in meaning between reference and content data representations. The two different methods are available for ease of representation. No difference in *meaning* should be implied by the use of one method or the other.

Assertions that indicate included data is intrinsic, while referenced data is extrinsic, are not valid and are not applicable to NIEM-conformant data instances and data definitions.

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 12-7] (INS). Empty content has no meaning

Rule 12-7. Empty content has no meaning

[Rule 12-7] (INS) (Interpretation)

Within the instance, the meaning of an element with no content is that additional properties are not asserted. There MUST NOT be additional meaning interpreted for an element with no content.

Elements without content only show a lack of asserted information. That is, all that is asserted is what is explicitly stated, through a combination of XML instance data and its schema. Data that is not present makes no claims. It may be absent due to lack of availability, lack of knowledge, or deliberate withholding of information. These cases should be modeled explicitly, if they are required.

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 12-8] (INS). Metadata applies to referring entity

NIEM provides the metadata mechanism for giving information about object assertions. An object may have an attribute that refers to one or more metadata objects. A structures:metadata attribute indicates that a data item has the given metadata. A structures:relationshipMetadata attribute asserts that the link (or relationship) established by an element has the given metadata.

haracteristics of metadata:

* Metadata objects may appear outside the data they describe.
* Metadata objects may be reused.
* Data may refer to more than one metadata object.
* Metadata pertains to an object or simple content, while link metadata pertains to the relationship between objects.

An instance would not be valid XML if the structures:metadata or structures:relationshipMetadata attributes contained references for which there were no defined IDs. The instance would not be NIEM-conformant if the references were not to IDs defined with the structures:id attribute.

Application of metadata to a type or element to which it is not applicable is not NIEM-conformant. A metadata element may be labeled as applicable to multiple elements via attribute appinfo:appliesToElements, or to multiple types via attribute appinfo:appliesToTypes. In either case it may apply to an instance of any of the listed elements or types. A metadata element with neither attribute appinfo:appliesToElements nor attribute appinfo:appliesToTypes may be applied to any element of any type.

Rule 12-8. Metadata applies to referring entity

[Rule 12-8] (INS) (Interpretation)

Within an element instance, when an object *$O* links to a metadata object via an attribute structures:metadata, the information in the metadata object MUST be applied to the object *$O*.

structures:metadata applies metadata to an object.

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 12-9] (INS). Referent of structures:relationshipMetadata annotates relationship

Rule 12-9. Referent of structures:relationshipMetadata annotates relationship

[Rule 12-9] (INS) (Interpretation)

Within an element instance, when an object *$O1* contains an element *$E*, with content object *$O2* or with a reference to object *$O2*, and *$O2* links to a metadata object via an attribute structures:relationshipMetadata, the information in the metadata object MUST be applied to the relationship *$E* between *$O1* and *$O2*.

structures:relationshipMetadata applies metadata to a relationship between two objects.

**context** InformationModel **inv:**

Rule is definitional.

###### NDR3 [Rule 4-1] Schema marked as reference schema document must conform

Rule 4-1. Schema marked as reference schema document must conform

[Rule 4-1] (SET) (Constraint)

Any schema document with an effective conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument MUST be a reference schema document.

**context** InformationModel **inv:**

This constraint realized by the aggregate of constraints targeting REF schema documents.

###### NDR3 [Rule 4-2] Schema marked as extension schema document must conform

Rule 4-2. Schema marked as extension schema document must conform

[Rule 4-2] (SET) (Constraint)

Any schema document with an effective conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument MUST be an extension schema document.

**context** InformationModel **inv:**

This constraint realized by the aggregate of constraints targeting EXT schema documents.

###### NDR3 [Rule 4-3] Schema is CTAS-conformant

Rule 4-3. Schema is CTAS-conformant

[Rule 4-3] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:report test="true()">The document MUST be a conformant document as defined by the NIEM Conformance Targets Attribute Specification.</sch:report>

</sch:rule>

</sch:pattern>

The term conformant document is defined by CTAS Section 3.2, [*Conformance to this Specification*](http://reference.niem.gov/niem/specification/conformance-targets-attribute/3.0/NIEM-CTAS-3.0-2014-07-31.html#section_3.2).

**context** InformationModel **inv:**

This constraint realized by the aggregate of constraints targeting REF and EXT schema documents.

###### NDR3 [Rule 4-4] (REF,EXT). Document element has attribute ct:conformanceTargets

Rule 4-4. Document element has attribute ct:conformanceTargets

[Rule 4-4] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)

or exists(@ct:conformanceTargets)]">

<sch:assert test="(. is nf:get-document-element(.)) = exists(@ct:conformanceTargets)"

>The [document element] of the XML document, and only the [document element], MUST own an attribute {http://release.niem.gov/niem/conformanceTargets/3.0/}conformanceTargets.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

This constraint realized during provisioning of the schema associated with the InformationModel

###### NDR3 [Rule 4-5](REF). Schema claims reference schema conformance target

Rule 4-5. Schema claims reference schema conformance target

[Rule 4-5] (REF) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:assert test="nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))"

>The document MUST have an effective conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

This constraint realized during provisioning of the schema associated with the InformationModel

###### NDR3 [Rule 4-6](EXT). Schema claims extension conformance target

Rule 4-6. Schema claims extension conformance target

[Rule 4-6] (EXT) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:assert test="nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument'))"

>The document MUST have an effective conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

This constraint realized during provisioning of the schema associated with the InformationModel

###### NDR3 [Rule 7-1](REF,EXT,INS). Document is an XML document

Rule 7-1. Document is an XML document

[Rule 7-1] (REF, EXT, INS) (Constraint)

<sch:pattern>

<sch:rule context="\*[. is nf:get-document-element(.)]">

<sch:report test="true()">The document MUST be an XML document.</sch:report>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

This constraint realized during provisioning of the schema associated with the InformationModel

###### NDR3 [Rule 9-11] (REF). No simple type disallowed derivation

Rule 9-11. No simple type disallowed derivation

[Rule 9-11] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleType">

<sch:assert test="empty(@final)"

>An element xs:simpleType MUST NOT have an attribute {}final.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset))  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(DataType)).oclAsType(DataType)->forAll(dt|not(dt.isFinalSpecialization))

###### NDR3 [Rule 9-12] (REF,EXT). Simple type has data definition

Rule 9-12. Simple type has data definition

[Rule 9-12] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleType">

<sch:assert test="some $definition in xs:annotation/xs:documentation[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>A simple type MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(DataType)).oclAsType(DataType)  
 ->forAll(dt|dt.ownedComment->exists(c|not(c.\_'body'.oclIsUndefined())and(c.\_'body'<>'')))

###### NDR3 [Rule 9-29] (REF). Complex content uses extension

Rule 9-29. Complex content uses extension

[Rule 9-29] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexContent">

<sch:assert test="exists(xs:extension)"

>An element xs:complexContent MUST have a child xs:extension.</sch:assert>

</sch:rule>

</sch:pattern>

NIEM does not support the use of complex type restriction in reference schemas. The use of restriction in a reference schema would reduce the ability for that schema to be reused. Restriction may be used in extension schemas.

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset))  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).clientDependency  
 ->select(d|d.stereotypedBy('Restriction'))->isEmpty()

###### NDR3 [Rule 9-30] (REF,EXT). Base type of complex type with complex content must have complex content

Rule 9-30. Base type of complex type with complex content must have complex content

[Rule 9-30] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType/xs:complexContent/xs:\*[

(self::xs:extension or self::xs:restriction)

and (some $base-qname in resolve-QName(@base, .) satisfies

namespace-uri-from-QName($base-qname) = nf:get-target-namespace(.))]">

<sch:assert test="some $base-type in nf:resolve-type(., resolve-QName(@base, .)) satisfies

empty($base-type/self::xs:complexType/xs:simpleContent)"

>The base type of complex type that has complex content MUST be a complex type with complex content.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Provisioning to target schemas ensures the base type of Complex types with complex content will have complex content.

###### NDR3 [Rule 9-31] (SET). Base type of complex type with complex content must have complex content

Rule 9-31. Base type of complex type with complex content must have complex content

[Rule 9-31] (SET) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType[

nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))

or nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument'))

]/xs:complexContent">

<sch:assert test="some $derivation in xs:\*[self::xs:extension or self::xs:restriction],

$base-qname in resolve-QName($derivation/@base, $derivation),

$base-type in nf:resolve-type($derivation, $base-qname) satisfies

empty($base-type/self::xs:complexType/xs:simpleContent)"

>The base type of complex type that has complex content MUST have complex content.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Provisioning to target schemas ensures the base type of Complex types with complex content will have complex content.

###### NDR3 [Rule 9-32] (REF). Simple content uses extension

Rule 9-32. Simple content uses extension

[Rule 9-32] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:simpleContent">

<sch:assert test="exists(xs:extension)"

>A complex type definition with simple content schema component MUST have a derivation method of extension.</sch:assert>

</sch:rule>

</sch:pattern>

This rule ensures that the definition of a complex type with simple content will use XML Schema extension. This allows for the above cases while disallowing more complicated syntactic options available in XML Schema. The above rule allows for use of xs:restriction within xs:simpleContent in extension schemas.

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset))  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).clientDependency  
 ->select(d|d.stereotypedBy('Restriction'))->isEmpty()

###### NDR3 [Rule 9-33] (REF). No complex type disallowed substitutions

Rule 9-33. No complex type disallowed substitutions

[Rule 9-33] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:assert test="empty(@block)"

>An element xs:complexType MUST NOT have an attribute {}block.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The concept of disallowed substitutions is currently not supported by the NIEM UML Profile. Currently, there will be no "block" provisioned for a complex type.

###### NDR3 [Rule 9-34] (REF). No complex type disallowed derivation

Rule 9-34. No complex type disallowed derivation

[Rule 9-34] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:complexType">

<sch:assert test="empty(@final)"

>An element xs:complexType MUST NOT have an attribute {}final.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset))  
implies  
self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)->forAll(dt|not(dt.isFinalSpecialization))

###### NDR3 [Rule 9-35] (REF,EXT). Element declaration is top-level

[[NDR] Rule 9-35](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-35), Element declaration is top-level (REF, EXT): Section 9.2.1, Element declaration

**context** InformationModel **inv:**

Constraint is enforced during provisioning, top level elements are always created and referenced by non top level elements.

###### NDR3 [Rule 9-36] (REF,EXT). Element declaration has data definition

Rule 9-36. Element declaration has data definition

[Rule 9-36] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@name)]">

<sch:assert test="some $definition in xs:annotation/xs:documentation[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>An element declaration MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute  
 ->forAll(a|a.ownedComment->exists(c|not(c.\_'body'.oclIsUndefined()) and (c.\_'body'<>'')))

###### NDR3 [Rule 9-39] (REF,EXT). Element type not in the XML Schema namespace

Rule 9-39. Element type not in the XML Schema namespace

[Rule 9-39] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)]">

<sch:assert test="for $type-qname in resolve-QName(@type, .) return

$type-qname = xs:QName('xs:anySimpleType')

or namespace-uri-from-QName($type-qname) != xs:anyURI('http://www.w3.org/2001/XMLSchema')"

>An element type that is not xs:anySimpleType MUST NOT have a namespace name <namespace-uri-for-prefix>xs</namespace-uri-for-prefix>.</sch:assert>

</sch:rule>

</sch:pattern>

The prohibition of element types having the XML Schema namespace subsumes a prohibition of the type xs:anyType.

Note that we express this as a constraint that none of the information models in an IEPD can have a targetNamespace of a schema.

**context** InformationModel **inv:**

self.targetNamespace<>'http://www.w3.org/2001/XMLSchema'

###### NDR3 [Rule 9-40] (REF,EXT). Element type not in the XML namespace

Rule 9-40. Element type not in the XML namespace

The XML namespace may be imported into an conformant schema document as if it were conformant. This specification does not enable a reference to any types that may be defined by any implementation of a schema for that namespace.

[Rule 9-40] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)]">

<sch:assert test="namespace-uri-from-QName(resolve-QName(@type, .)) != 'http://www.w3.org/XML/1998/namespace'"

>An element type MUST NOT have a namespace name that is in the XML namespace.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute.type->forAll(t| t.\_'package'.name<>'xml')

###### NDR3 [Rule 9-41] (REF,EXT). Element type is not simple type

Rule 9-41. Element type is not simple type

[Rule 9-41] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@type]">

<sch:assert test="every $type-qname in resolve-QName(@type, .),

$type-ns in namespace-uri-from-QName($type-qname),

$type-local-name in local-name-from-QName($type-qname) satisfies (

$type-qname = xs:QName('xs:anySimpleType')

or (($type-ns = nf:get-target-namespace(.)

or exists(nf:get-document-element(.)/xs:import[

xs:anyURI(@namespace) = $type-ns

and empty(@appinfo:externalImportIndicator)]))

and not(ends-with($type-local-name, 'SimpleType'))))"

>An element type that is not xs:anySimpleType MUST NOT be a simple type.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

OCL representation of this constraint is deferred.

###### NDR3 [Rule 9-42](REF). No element disallowed substitutions

Rule 9-42. No element disallowed substitutions

[Rule 9-42] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:element">

<sch:assert test="empty(@block)"

>An element xs:element MUST NOT have an attribute {}block.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The concept of disallowed substitutions (@block) is currently not supported by NIEM-UML. There will be no provisioning of the @block attribute.

###### NDR3 [Rule 9-47] (REF,EXT). Attribute declaration is top-level

[[NDR] Rule 9-47](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-47), Attribute declaration is top-level (REF, EXT): Section 9.2.3, Attribute declaration

**context** InformationModel **inv:**

Constraint is enforced during provisioning, top level attributes are always created and referenced by non top level elements.

###### NDR3 [Rule 9-48] (REF,EXT). Attribute declaration has data definition

Rule 9-48. Attribute declaration has data definition

[Rule 9-48] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@name)]">

<sch:assert test="some $definition in xs:annotation/xs:documentation[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>An attribute declaration MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

self.base\_Package.ownedType->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier).attribute  
 ->forAll(a|a.ownedComment->exists(c|not(c.\_'body'.oclIsUndefined()) and (c.\_'body'<>'')))

###### NDR3 [Rule 9-49] (REF,EXT). Attribute declaration has type

Rule 9-49. Attribute declaration has type

[Rule 9-49] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@name)]">

<sch:assert test="exists(@type)"

>A top-level attribute declaration MUST have a type.</sch:assert>

</sch:rule>

</sch:pattern>

Untyped XML Schema attributes allow arbitrary content, with no semantics. Attributes must have a type so that specific syntax and semantics will be provided.

**context** InformationModel **inv:**

Specification as OCL Constraint is deferred.

###### NDR3 [Rule 9-61] (REF). xs:sequence must be child of xs:extension

Rule 9-61. xs:sequence must be child of xs:extension

[Rule 9-61] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:sequence">

<sch:assert test="exists(parent::xs:extension)"

>An element xs:sequence MUST be a child of element xs:extension.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:sequence is always produced as a child of an xs:extension.

###### NDR3 [Rule 9-62] (EXT). xs:sequence must be child of xs:extension or xs:restriction

Rule 9-62. xs:sequence must be child of xs:extension or xs:restriction

[Rule 9-62] (EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:sequence">

<sch:assert test="exists(parent::xs:extension) or exists(parent::xs:restriction)"

>An element xs:sequence MUST be a child of element xs:extension or xs:restriction.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:sequence is always produced as a child of an xs:extension or xs:restriction.

###### NDR3 [Rule 9-63] (REF). No xs:choice

[[NDR] Rule 9-63](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-63), No xs:choice (REF): Section 9.3.1.2, Choice

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference))  
implies  
self.base\_Package.ownedType  
->select(t|t.oclIsKindOf(Classifier)).oclAsType(Classifier)   
->union(self.base\_Package.ownedType->select(t|t.oclIsKindOf(Class)).oclAsType(Class).nestedClassifier)  
->select(t|t.stereotypedBy('Choice'))->isEmpty()

###### NDR3 [Rule 9-64] (EXT). xs:choice must be child of xs:sequence

[[NDR] Rule 9-64](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-64), xs:choice must be child of xs:sequence (EXT): Section 9.3.1.2, Choice

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:choice is always produced as a child of an xs:sequence.

###### NDR3 [Rule 9-65] (REF,EXT). Sequence has minimum cardinality 1

XML Schema allows each particle to specify cardinality (how many times the particle may appear in an instance). NIEM restricts the cardinality of xs:sequence particles to exactly one, to ensure that content model definitions are defined in as straightforward a manner as possible.

A schema developer who requires the instance syntax that would be obtained from the use of specific cardinality on sequences should define cardinality on individual element uses.

Rule 9-65. Sequence has minimum cardinality 1

[Rule 9-65] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:sequence">

<sch:assert test="empty(@minOccurs) or xs:integer(@minOccurs) = 1"

>An element xs:sequence MUST either not have the attribute {}minOccurs, or that attribute MUST have a value of 1.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:sequence is always produced with @minOccurs=1.

###### NDR3 [Rule 9-66] (REF,EXT). Sequence has maximum cardinality 1

Rule 9-66. Sequence has maximum cardinality 1

[Rule 9-66] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:sequence">

<sch:assert test="empty(@maxOccurs) or (@maxOccurs instance of xs:integer

and 1 = xs:integer(@maxOccurs))"

>An element xs:sequence MUST either not have the attribute {}maxOccurs, or that attribute MUST have a value of 1.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:sequence is always produced with @maxOccurs=1.

###### NDR3 [Rule 9-67] (EXT). Choice has minimum cardinality 1

[[NDR] Rule 9-67](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-67), Choice has minimum cardinality 1 (EXT): Section 9.3.2.2, Choice cardinality

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:choice is always produced with @minOccurs=1.

###### NDR3 [Rule 9-68] (EXT). Choice has maximum cardinality 1

[[NDR] Rule 9-68](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-68), Choice has maximum cardinality 1 (EXT): Section 9.3.2.2, Choice cardinality

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:choice is always produced with @maxOccurs=1.

###### NDR3 [Rule 9-69] (REF). No use of xs:any

Rule 9-69. No use of xs:any

[Rule 9-69] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:any">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:any.</sch:assert>

</sch:rule>

</sch:pattern>

The xs:any particle (see Model Group Restrictions for an informative definition of particle) provides a wildcard that may carry arbitrary content. The particle xs:any may appear within an extension schema document.

**context** InformationModel **inv:**

((self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)or(self.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset))  
implies  
self.base\_Package.ownedType->forAll(dt|not(dt.stereotypedBy('XSDAnyProperty')))

###### NDR3 [Rule 9-70] (REF). No use of xs:anyAttribute

Rule 9-70. No use of xs:anyAttribute

[Rule 9-70] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:anyAttribute">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:anyAttribute.</sch:assert>

</sch:rule>

</sch:pattern>

The xs:anyAttribute element provides a wildcard, where arbitrary attributes may appear.

**context** InformationModel **inv:**

Specification of this constraint as OCL has been deferred.

###### NDR3 [Rule 9-71] (REF,EXT). No use of xs:unique

XML Schema provides NIEM with the ability to apply uniqueness constraints to schema-validated content. These mechanisms, however, establish relationships in a way that is very difficult to understand, extend, and keep consisent through schema reuse.

Rule 9-71. No use of xs:unique

[Rule 9-71] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:unique">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:unique.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:unique can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-72] (REF,EXT). No use of xs:key

Rule 9-72. No use of xs:key

[Rule 9-72] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:key">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:key.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:key can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-73] (REF,EXT). No use of xs:keyref

Rule 9-73. No use of xs:keyref

[Rule 9-73] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:keyref">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:keyref.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:keyref can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-74] (REF,EXT). No use of xs:group

Rule 9-74. No use of xs:group

[Rule 9-74] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:group">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:group.</sch:assert>

</sch:rule>

</sch:pattern>

NIEM does not allow groups of elements to be named other than as named complex types. A group in XML Schema creates a named entity that may be included in multiple types, and which consists of a sequence of or choice between element particles. NIEM has not developed a semantic model for these components, and they are not integrated into NIEM’s design.

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:group can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-75] (REF,EXT). No definition of attribute groups

Rule 9-75. No definition of attribute groups

Per Rule 11-24[, *Schema uses only known attribute groups*](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_11-24) *(REF,EXT)*, the only attribute groups used in NIEM-conformant schemas are structures:SimpleObjectAttributeGroup and attribute groups defined by the IC-ISM and IC-NTK schemas. Therefore, NIEM-conformant schemas do not define additional attribute groups.

[Rule 9-75] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attributeGroup[@name]">

<sch:assert test="false()"

>The schema MUST NOT contain an attribute group definition schema component.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

Constraint enforced by provisioning, an xs:attributeGroup can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-76] (REF,EXT). Comment is not recommended

[[NDR] Rule 9-76](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-76), Comment is not recommended (REF, EXT): Section 9.6, Annotation components

**context** InformationModel **inv:**

Constraint enforced by provisioning, an XML comment can not be modeled nor is it produced in a target schema.

###### NDR3 [Rule 9-77] (REF,EXT). Documentation element has no element children

[[NDR] Rule 9-77](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-77), Documentation element has no element children (REF, EXT): Section 9.6, Annotation components

**context** InformationModel **inv:**

Constraint enforced by provisioning, the xs:documentation is populated by a UML Comment body, which is a String (possibly escaped to ensure no nested xml elements are present).

###### NDR3 [Rule 9-78] (REF,EXT). xs:appinfo children are comments, elements, or whitespace

XML Schema provides special annotations for support of automatic processing. The XML Schema specification provides the element xs:appinfo to carry such content and does not specify what style of content they should carry. In NIEM, xs:appinfo elements carry structured XML content.

Rule 9-78. xs:appinfo children are comments, elements, or whitespace

[Rule 9-78] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:appinfo/node()">

<sch:assert test="self::comment()

or self::element()

or self::text()[string-length(normalize-space(.)) = 0]"

>A child of element xs:appinfo MUST be an element, a comment, or whitespace text.</sch:assert>

</sch:rule>

</sch:pattern>

Application information elements are intended for automatic processing; the meaning of an appinfo annotation is provided via elements.

**context** InformationModel **inv:**

Constraint enforced by provisioning; the xs:appinfo is not directly modeled, and is provisioned in accordance with NDR-specified rules associated with specific NIEM concepts. Thus, an XML element is the child of an xs:appinfo.

###### NDR3 [Rule 9-79] (REF,EXT). Appinfo child elements have namespaces

Rule 9-79. Appinfo child elements have namespaces

[Rule 9-79] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:appinfo/\*">

<sch:assert test="namespace-uri() != xs:anyURI('')"

>An element that is a child of xs:appinfo MUST have a namespace name.</sch:assert>

</sch:rule>

</sch:pattern>

The XML namespaces specification includes the concept of content not in a namespace. Use of elements without namespaces can lead to conflicting data definitions, and makes it difficult to identify relevant data definitions.

**context** InformationModel **inv:**

Constraint enforced by provisioning; the xs:appinfo is not directly modeled, and is provisioned in accordance with NDR-specified rules associated with specific NIEM concepts. Thus, an XML element is the child of an xs:appinfo and will have a namespace name.

###### NDR3 [Rule 9-80] (REF,EXT). Appinfo descendants are not XML Schema elements

Rule 9-80. Appinfo descendants are not XML Schema elements

[Rule 9-80] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:appinfo//xs:\*">

<sch:assert test="false()"

>An element with a namespace name of xs: MUST NOT have an ancestor element xs:appinfo.</sch:assert>

</sch:rule>

</sch:pattern>

NIEM-conformant schemas are designed to be very easily processed. Although uses of XML Schema elements as content of xs:appinfo elements could be contrived, it is not current practice and complicates the processing of XML elements by their namespaces and names. Forbidding the use of XML Schema elements outside valid uses of schema simplifies such processing.

**context** InformationModel **inv:**

Constraint enforced by provisioning; the xs:appinfo is not directly modeled, and is provisioned in accordance with NDR-specified rules associated with specific NIEM concepts. Thus, an XML element is the child of an xs:appinfo and will not contain elements with the schema namespace.

###### NDR3 [Rule 9-81] (REF,EXT). Schema has data definition

The XML Schema language defines that the document element xs:schema may contain the optional attributes attributeFormDefault and elementFormDefault. The values of these attributes are not material to a conformant schema, as each attribute and element defined by a conformant schema is defined as a top-level component, and so each is qualified by its target namespace.

Rule 9-81. Schema has data definition

[Rule 9-81] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:schema">

<sch:assert test="some $definition in (xs:annotation/xs:documentation)[1] satisfies

string-length(normalize-space(string($definition))) > 0"

>An element xs:schema MUST have a data definition.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

self.base\_Package.ownedComment.\_'body'->exists(doc|not(doc.oclIsUndefined())and(doc<>''))

###### NDR3 [Rule 9-85] (REF). No disallowed substitutions

Rule 9-85. No disallowed substitutions

[Rule 9-85] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:schema">

<sch:assert test="empty(@blockDefault)"

>An element xs:schema MUST NOT have an attribute {}blockDefault.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The concept of disallowed substitutions (@blockDefault) is currently not supported by NIEM-UML. There will be no provisioning of the @blockDefault attribute.

###### NDR3 [Rule 9-86] (REF). No disallowed derivations

Rule 9-86. No disallowed derivations

[Rule 9-86] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:schema">

<sch:assert test="empty(@finalDefault)"

>An element xs:schema MUST NOT have an attribute {}finalDefault.</sch:assert>

</sch:rule>

</sch:pattern>

**context** InformationModel **inv:**

The concept of disallowed derivations is currently not in the NIEM-UML model; the attribute @finalDefault will not be produced for any InformationModel schema.

###### NDR3 [Rule 9-87] (REF,EXT). No use of xs:redefine

Rule 9-87. No use of xs:redefine

[Rule 9-87] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:redefine">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:redefine.</sch:assert>

</sch:rule>

</sch:pattern>

The xs:redefine element allows an XML Schema document to restrict and extend components from a namespace, in a separate schema document from the one that initially defined that namespace. Such redefinition introduces duplication of definitions, allowing multiple definitions to exist for components from a single namespace.

**context** InformationModel **inv:**

The concept of xs:redefine is not in the NIEM-UML model; the schema construct xs:redefine can not be modeled and will not be produced for any InformationModel schema.

###### NDR3 [Rule 9-87] (REF,EXT). No use of xs:redefine2

Rule 9-87. No use of xs:redefine

[Rule 9-87] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:redefine">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:redefine.</sch:assert>

</sch:rule>

</sch:pattern>

The xs:redefine element allows an XML Schema document to restrict and extend components from a namespace, in a separate schema document from the one that initially defined that namespace. Such redefinition introduces duplication of definitions, allowing multiple definitions to exist for components from a single namespace.

**context** InformationModel **inv:**

The concept of xs:redefine is not in the NIEM-UML model; the schema construct xs:redefine can not be modeled and will not be produced for any InformationModel schema.

###### NDR3 [Rule 9-88] (REF,EXT). No use of xs:include

Rule 9-88. No use of xs:include

[Rule 9-88] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:include">

<sch:assert test="false()"

>The schema MUST NOT contain the element xs:include.</sch:assert>

</sch:rule>

</sch:pattern>

Element xs:include brings schemas defined in separate files into the current namespace. It breaks a namespace up into arbitrary partial schemas, which needlessly complicates the schema structure, making it harder to reuse and process, and also increases the likelihood of conflicting definitions.

Inclusion of schemas that do not have namespaces also complicates schema understanding. This inclusion makes it difficult to find the realization of a specific schema artifact and create aliases for schema components that should be reused.

**context** InformationModel **inv:**

The concept of xs:include is not in the NIEM-UML model; the schema construct xs:include can not be modeled and will not be produced for any InformationModel schema.

### <Stereotype> [ReferenceName](#_2da26d153c0374af43f1186eb7b8bc65)

##### Extends

##### ElementDescription

The ReferenceName stereotype is used on an Element that has a name that does not  
 conform to the naming conventions required by the NIEM NDR or is otherwise not the  
 desired NIEM name. The NIEMName attribute must provide a name for the Element that  
 conforms to the relevant NDR naming rules for the specific kind of Element to which  
 the stereotype is applied.

##### Properties

###### NIEMName

A NIEM NDR-conformant name to be applied to an Element. The NIEMName will  
 override any name generated from the UML name..

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

### <Stereotype> [RoleOf](#_648c0d453d1db791aba631928aa50005)

##### Extends

##### PropertyDescription

The RoleOf stereotype is applied to a Property of a Class representing a NIEM role type, whose type identifies the base type of that role type. A RoleOf Property must be a reference (i.e., have aggregation=none). A NIEM role type is a type that represents a particular function, purpose, usage, or role of an object.

##### Constraints

###### NDR [Rule 7-40]

**[Rule 7-40] (REF, SUB, EXT)** Within the schema, any element with a name beginning with the string RoleOf SHALL represent a base type, of which the containing type represents a role.

**Rationale** A RoleOf element references its corresponding base element. The RoleOf label on the reference element ensures that a role object is distinguishable from other objects and its link to the associated base is also distinguishable from the additional properties that are characteristic of this role or that add information.

**context** RoleOf **inv:**

This constraint is implemented by the PIM/PSM transformation. Identifying a  
 <Property> as a <<RoleOf>> corresponds to  
 the NIEM naming convention used to identify the roleOf...reference and furthermore  
 establishes the owning <Classifier> as a NIEM Role.

###### NDR [Rule 9-35]

**[Rule 9-35] (REF, SUB, EXT)** Within the schema, the name of a role SHALL use the property term RoleOf.

**Rationale** Using the property term RoleOf immediately identifies an element as representing a role.

**context** RoleOf **inv:**

This constraint is enforced by the PIM/PSM transformation. The Transformation  
 ensures that the "RoleOf" property term becomes part of the target PSM property  
 name.

###### NDR3 [Rule 10-3] (REF,EXT). RoleOf element type is an object type

[[NDR] Rule 10-3](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-3), RoleOf element type is an object type (REF, EXT): Section 10.2.2, Role types and roles

**context** RoleOf **inv:**

(  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and   
 not(self.type.oclIsUndefined())  
 and   
 not(self.name.oclIsUndefined())  
 and  
 (self.name.startsWith('RoleOf') or self.stereotypedBy('RolePlayedBy'))  
)   
implies  
self.type->forAll(t|not(t.stereotypedBy('MetadataType')or t.stereotypedBy('AssociationType')or t.stereotypedBy('AugmentationType') or t.oclIsKindOf(AssociationClass)))

### <Stereotype> [RolePlayedBy](#_8793a0f4acd9a4ae54c85323629be721)

##### Extends

##### GeneralizationDescription

RolePlayedBy Generalization specifies that the special class is to be considered  
 the type of a role that is played by instances of the general class. In the PSM this  
 will map to a property with the "RoleOf" prefix..

##### Constraints

###### NDR3 [Rule 10-3] (REF,EXT). RoleOf element type is an object type

NIEM differentiates between an object and a role of the object. The term role is used here to mean a function or part played by some object. The simplest way to represent a role of an object is to use an element.

 In many cases, there is a further need to represent characteristics and additional information associated with a role of an object. In such cases, the above element is insufficient. When a role must be modeled with additional information, a role type is called for.

The term role type has a normative definition:

[Definition: *role type*]

A **role type** is an object type that represents a particular function, purpose, usage, or role of one or more objects of its base type.

A role type describes a role of a thing. A role is a function or position played by something in a particular situation or context. A role type holds information that is specific to the role, but that is not specific to the context, and is not specific to thing that plays the role.

Developers of NIEM-conformant schemas and exchanges should create and use role types whenever they have information specific to a base object’s function. Such information is a characteristic of the rule, and not of the base object. Information that is a characteristic of a base object probably does not belong in a role type.

[Definition: *RoleOf element*]

A **RoleOf element** is an element declaration that

* is defined by a reference schema document or an extension schema docuement, and
* has a {name} that begins with RoleOf.

A RoleOf element represents a base type for a role type.

Rule 10-3. RoleOf element type is an object type

[Rule 10-3] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@name[starts-with(., 'RoleOf')]]">

<sch:assert test="every $type in @type,

$type-local-name in local-name-from-QName(resolve-QName($type, .)) satisfies

not(ends-with($type-local-name, 'AssociationType')

or ends-with($type-local-name, 'MetadataType')

or ends-with($type-local-name, 'AugmentationType'))"

>The type definition of a RoleOf element MUST be an object type.</sch:assert>

</sch:rule>

</sch:pattern>

**context** RolePlayedBy **inv:**

self.base\_Generalization.general->  
 forAll(t|not(t.stereotypedBy('MetadataType')or t.stereotypedBy('AssociationType')or t.stereotypedBy('AugmentationType') or t.oclIsKindOf(AssociationClass)))

### <Stereotype> [Subsets](#_9c82075d00d9b39547d9afa6c913ef7f)

##### Generalization

##### [References](#_c7b8a68ef50d3d361f495647dd4876ec)Description

A Realization signifying a NIEM subsetting relationship between a client derived (subset) element and a supplier base (reference) element. The <<Subsets>> Realization must be between the same meta-types: either Properties, Classifiers, or <<InformationModel>> packages. The <<Subsets>> Realization must be between elements owned by different <<InformationModel>> packages. The targetNamespace of the distinct <<InformationModel>> packages must be identical. The defaultPurpose of client and supplier may be one of the following combinations: client is subset, supplier is reference; client is reference, supplier is reference; client is extension, supplier is extension; client is constraint, supplier is exchange, subset, extension, or reference

##### Constraints

###### SubsetNamesMustMatch

**context** Subsets **inv:**

// pseudo code for specifygin constraint in terms of tag values of foreign stereotype instances  
self.base\_Realization->forAll(r|  
 (r.client->forAll(c|  
 r.supplier->forAll(s|  
 ( s.name=c.name)  
 and (s.oclIsKindOf(Package) implies (c.oclIsKindOf(Package) and (s.getTargetNamespace()=c.getTargetNamespace())))  
 and (s.oclIsKindOf(Classifier) implies (c.oclIsKindOf(Classifier) and (s.getNearestPackage().getTargetNamespace()=c.getNearestPackage().getTargetNamespace())))  
 and (s.oclIsKindOf(Property) implies \*c.oclIsKindOf(Property) and (s.getNearestPackage().getTargetNamespace()=c.getNearestPackage().getTargetNamespace())))  
   
 )  
 )  
 )  
)

### <Enumeration> [DefaultPurposeCode](#_4b2f5b68c9069dda1904776486664fb6)

The possible purposes for an information model. This enumeration provides the allowed values for the defaultPurpose attribute of the InformationModel stereotype. The values correspond to the schema purpose codes for an MPD artifact.

The NDR and MPD define kinds of schema documents.  The NDR defines schema document kinds in terms of conformance targets, defined at the schema level, with each document kind adhering to a set of conformance rules.  The MPD further elaborates these document kinds into artifact types and defines a catalog which associates the specific purpose of an artifact with the location of that artifact.

The NDR defines conformance targets using the following definitions:

CTAS defines several terms used normatively within this specification.

[Definition: *conformance target*]

The term conformance target is as defined by CTAS, which states:

A *conformance target* is a class of artifact, such as an interface, protocol, document, platform, process or service, that is the subject of conformance clauses and normative statements. There may be several conformance targets defined within a specification, and these targets may be diverse so as to reflect different aspects of a specification. For example, a protocol message and a protocol engine may be different conformance targets.

[Definition: *conformance target identifier*]

The term conformance target identifier is as defined by CTAS, which states:

A *conformance target identifier* is an internationalized resource identifier that uniquely identifies a conformance target.

[Definition: *effective conformance target identifier*]

The term effective conformance target identifier is as defined by CTAS Section 4, [*Semantics and Use*](http://reference.niem.gov/niem/specification/conformance-targets-attribute/3.0/NIEM-CTAS-3.0-2014-07-31.html#definition_effective_conformance_target_identifier), which states:

An *effective conformance target identifier* of a conformant document is an internationalized resource identifier reference that occurs in the document’s effective conformance targets attribute.

##### Literals

###### subset

A NIEM *schema document subset* is a set of XML schema documents that constitutes a reduced set of components derived from a NIEM reference schema document or document set associated with a given numbered release or domain update.

[Definition: *schema document subset*]

An XML schema document set based on a reference schema document set intended to ensure that any instance XML document valid to the schema document subset is also valid to the reference schema document set.

The primary purpose for a schema document subset is to reduce and constrain the scope and size of a full NIEM reference schema document set for use within an IEPD. A schema document subset is derived from a reference schema document set (such as a NIEM release) by applying subset operations. Also, note that employing a subset of a reference schema document set within an IEPD is optional; it is completely valid to reuse NIEM reference schema documents as-is within IEPDs.

The fundamental rule for a valid NIEM schema document subset is formally stated follows:

Rule 4-1. Fundamental NIEM Subset Rule

[Rule 4-1] (Schema-subset) (Constraint)

A schema document subset ($SUBSET) for a given reference schema document set ($REFERENCE) MUST be defined such that for all instance XML documents ($XML), where $XML is valid to $SUBSET, $XML is valid to $REFERENCE.

A schema document subset is composed of XML schema documents. A schema document subset can essentially be a reference schema document set (i.e., a NIEM release) that has been modified by applying subset operations to support business requirements represented in an IEPD. A subset derived from a reference schema document set may differ from that reference such that its content has been reduced and/or constrained.

[Definition: *subset schema document*]

An XML schema document that meets all of the following criteria:

* It is built from a reference schema document set where one or more reference schema documents have been substituted by corresponding subset schema documents.
* It is built from a reference schema document by applying subset operations to the XML schema statements in a reference schema document.
* It is explicitly designated as a subset schema document. This is accomplished by declaration in the relevant MPD catalog or by a tool-specific mechanism outside the subset schema document.
* It has a target namespace previously defined by a reference schema document. That is, it does not provide original definitions and declarations for schema components, but instead provides an alternate schema representation of components that are defined by a reference schema document.
* It does not alter the business semantics of components in its namespace. The reference schema document defines these business semantics.
* It is intended to express the limited vocabulary necessary for an IEPD and to support XML Schema validation for an IEPD.

 NIEM subset operations are essentially reduction operations that remove or constrain portions of a reference schema document set, thereby building a profile of the set. They do not expand the scope (i.e., relax constraints) or change the semantics of reference schema document set content.

Because NIEM adopts an optional and over-inclusive data representation strategy, most elements in a NIEM reference schema have zero to unbounded cardinality. So, elements with cardinality minOccurs="0" are optional and may be omitted from a subset schema document if not needed for business reasons. It is also valid to constrain element cardinality within a subset schema document, as long as doing so does not break the subset relationship with the reference schema document set. For example, a reference schema document element with cardinality (minOccurs="0", maxOccurs="unbounded") may be constrained to (0,1) or (1,1) in a subset schema document. However, if a reference schema document element’s cardinality is (1,unbounded), it may not be constrained to (0,1) since this breaks the subset relationship. The interval (0,1) is not contained within, and instead, overlaps the interval (1,unbounded).

The following list describes valid subset operations that are considered non-normative and informative only. In most cases, they can be applied to a schema document set and result in a corresponding schema document subset. However, it is possible to apply them in combinations that will break the subset relationship, or even result in invalid schemas. Apply these operations carefully and thoughtfully!

* 1. Remove an XML comment.
  2. Remove an xs:annotation and its children xs:documentation and xs:appinfo.
  3. Increase the value of an xs:element/@minOccurs as long as it remains less than or equal to its corresponding @maxOccurs value).
  4. Decrease the value of an xs:element/@maxOccurs as long as it remains greater than or equal to its corresponding @minOccurs value.
  5. Remove an xs:element if its @minOccurs="0".
  6. Remove an xs:complexType or xs:simpleType if not supporting an xs:element or xs:attribute declaration, or another xs:complexType or xs:simpleType definition.
  7. Remove an xs:attribute with @use="optional" from an xs:complexType.
  8. Change an xs:attribute/@use="optional" to @use="prohibited".
  9. Change an xs:attribute/@use="optional" to @use="required".
  10. Remove an xs:element declaration if it is not supporting an element use.
  11. Remove an xs:enumeration from an xs:simpleType as long as it is not the only remaining xs:enumeration.
  12. Remove an element with representation term AugmentationPoint if it is not being used for element substitution.
  13. Add or apply a constraining facet to an xs:simpleType.
  14. Remove an xs:import and its associated schema document if the schema document is not used within the document set.
  15. Change a concrete xs:element declaration to @abstract="true".
  16. Change an xs:element/@nillable="true" to @nillable="false".
  17. Substitute an xs:element/@substitutionGroup member for its associated substitution group head.
  18. Substitute a composition of xs:element/@substitutionGroup members for their associated substitution head (subject to cardinality and unique particle attribution (UPA) constraints[http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ - cos-nonambig](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#cos-nonambig)). The composition is an ordered sequence of the @substitutionGroup member elements. Each substitute element may bound its cardinality such that the total cardinality sum is within the bounds of the @substitutionGroup head cardinality. Order and cardinality of the replacement sequence must conform to XML Schema UPA constraints.
  19. Replace a wildcard (subject to cardinality, UPA, and namespace constraints) with a composition, i.e., an ordered sequence of elements. Each element may further bound cardinality within the bounds of the wildcard. Order and cardinality of replacement sequence must conform to XML Schema UPA constraints. The namespace of each element must conform with namespace constraints specified by the wildcard (if any).

###### constraint

[Definition: *constraint schema document set*]

A set of related constraint schema documents that work together, such as a constraint schema document set built by adding constraints to a schema document subset.

A constraint schema document set is an XML schema document set that is used to express business rules for a class of instance XML document, and is not assumed to be a definition for the semantics of the components it contains and describes. Instead, a constraint schema document set uses the XML Schema Definition Language to add constraints to components defined or declared by other schema documents, usually a schema document subset; but they can be applied to extension schema documents as well.

A constraint schema document set validates additional constraints imposed on an instance XML document only after it is known to be NIEM-conformant (i.e., has been validated with a reference schema document set, or schema document subset, and applicable extension schema documents).

To use a constraint schema document set to tighten constraints on an IEP, a two-pass validation technique is employed. In the first pass, an IEP is validated against the schema document subset and extension schema documents. This pass ensures that IEP semantics and structure conform to the NIEM model and NDR. In the second pass, an IEP is checked against a constraint schema document set, which may contain constrained versions of the subset schema documents and extension schema documents. This pass ensures that the IEP also satisfies the additional constraints (i.e., business rules that the first pass was unable to validate). A constraint schema document set need not validate constraints that are applied by other schema documents.

Constraint schema document sets are generally useful when it is necessary to impose restrictions that are more complex than cardinality. If only cardinality restrictions are needed, then it is easier and more efficient to set these directly in the subset schema documents and avoid the use of a constraint schema document set. Otherwise, a constraint schema document set may be necessary.

Use of a constraint schema document set is one option for tightening constraints on NIEM IEPs beyond what NIEM itself provides. This particular technique uses the XML Schema Definition Language. NIEM also allows other methods that do not use XML Schema. For example, the use of ISO Schematron is the preferred method for applying business rules. However, other constraint or business rule methods are also acceptable. That said, at this time there are no normative rules for how these business rule techniques should be employed in NIEM IEPDs. Therefore, if other techniques are used, it is a developer responsibility to incorporate appropriate artifacts and clear documentation.

Note that one disadvantage to use of constraint schema document sets is that they do not provide clear visibility or explanation of the constraints they enforce; nor do they provide clear validation failure messages. On the other hand, a standard business rule language such as ISO Schematron provides facilities for better understanding of the business rules, their intent, and error handling of failures.

A common practice for creating an IEPD constraint schema document set is to start with a valid NIEM schema document subset and modify it to further restrict the class of instance XML documents (IEPs) that will validate with this constraint schema set. However, an extension schema document can also be used to derive a constraint schema document. Using this technique, the namespace of that schema document would reuse the target namespace of the schema document from which it is derived.

There is no restriction on the number of constraint schema document sets (and passes) that an IEPD can employ. As in other advanced situations, developers must clearly document their intentions for and use of multiple constraint schema document sets.

In general, constraint schema documents in a constraint schema document set have far fewer requirements than other classes of NIEM schema documents. Since they work in tandem with NIEM normative schema documents, these schema documents are allowed to use the XML Schema Definition language in any way necessary to express business rules. This means that to constrain instance XML documents, these schema document can employ XML Schema constructs that are not allowed in NIEM conformant schema documents.

###### extension

[Definition: *extension schema document*]

An **extension schema document** is a schema docuemt that is intended to provide definitions of schema components that are intended for reuse within a more narrow scope than those defined by a reference schema document. It is a conformance target of this specification. An extension schema document MUST conform to all rules of this specification that apply to this conformance target. An XML document with a conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument MUST be an extension schema document.

Characteristics of an extension schema document include:

* It is explicitly designated as an extension schema document via the conformance targets attribute.
* It provides the broadest, most fundamental definitions of components in its namespace.
* It provides the authoritative definition of business semantics for components in its namespace.
* It contains components that, when appropriate, use or are derived from the components in reference schema documents.
* It is intended to express the additional vocabulary required for an information exchange, above and beyond the vocabulary available from reference schemas, and to also support additional XML Schema validation requirements for an exchange.
* It satisfies all rules specified in this document for extension schema documents.

An extension schema in an information exchange specification serves several functions. First, it defines new content within a new namespace, which may be an exchange-specific namespace or a namespace shared by several exchanges. This content is NIEM-conformant but has fewer restrictions on it than do reference schema documents. Second, the extension schema document bases its content on content from reference schema documents, where appropriate. Methods of deriving content include using (by reference) existing schema components, as well as creating extensions and restrictions of existing components.

For example, an information exchange specification may define a type for an exchange-specific phone number and base that type on a type defined by the NIEM Core reference schema document. This exchange-specific phone number type may restrict the NIEM Core type to limit those possibilities that are permitted of the base type. Exchange extensions and restrictions must include annotations and documentation to be conformant, but they are allowed to use restriction, choice, and some other constructs that are not allowed in reference schema documents.

Note that exchange specifications may define schemas that meet the criteria of reference schemas for those components that its developers wish to nominate for later inclusion in NIEM Core or in domains.

From the MPD:

In general, an extension schema document contains components that use or are derived from the components in reference schema documents. It is intended to express additional vocabulary above and beyond the vocabulary available from reference schema documents.

A developer who determines that NIEM is missing elements required for a given information exchange has several options to account for such requirement shortfalls. Using rules and techniques defined in the NDR:

* + Extend an existing NIEM data component (if possible).
  + Augment an existing NIEM data type (through NIEM Type Augmentation).
  + Build a new NIEM-conformant data component.
  + Employ NIEM adapter types for components from an external standard that does not conform to NIEM.

A NIEM extension schema document may contain data components built from any of the options above. Employment of extension schema documents in an IEPD is entirely optional.

Multiple extension schema documents are allowed in a single IEPD. Developers will likely want to reuse many of their extension schema documents in other IEPDs. Therefore, the best practice for extension is to group all data components designed to be reused into one extension schema document or document set, and group IEPD-specific data components into another. Then the reusable extension components can be more easily redeployed in other IEPDs as needed.

Extension schema documents generally contain new data component declarations that may (though not necessarily) be derived from or reference existing NIEM data component. This being the case, reference schema documents do not exist for new data components found within extension schema documents. Therefore, extension schema documents must satisfy the more rigorous documentation requirements of reference schema documents. Per the NDR, the definition or declaration of each new data component in an extension schema document must include an xs:annotation element that provides its semantics and NIEM-specific relationships.

###### incremental

###### reference

[Definition: *reference schema document*]

A **reference schema document** is a schema document that is intended to provide the authoritative definitions of broadly reusable schema components. It is a conformance target of this specification. A reference schema document MUST conform to all rules of this specification that apply to this conformance target. An XML document with a conformance target identifier of http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument MUST be a conformant reference schema document.

A reference schema document is a schema document that is intended to be the authoritative definition schema for a namespace. Examples include NIEM Core and NIEM domains.

Some characteristics of a reference schema document:

* It is explicitly designated as a reference schema via the conformance targets attribute, per Rule 4-5, [*Schema claims reference schema conformance target*](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_4-5) *(REF).*
* It provides the broadest, most fundamental definitions of components in its namespace.
* It provides the authoritative definition of business semantics for components in its namespace.
* It is intended to serve as the basis for components in information exchanges and extension schema documents.
* It satisfies all rules specified in the Naming and Design Rules for reference schemas.

Any schema that defines components that are intended to be incorporated into NIEM Core or a NIEM domain will be defined as a reference schema.

The rules for reference schema documents are more stringent than are the rules for other classes of NIEM-conformant schemas. Reference schema documents are intended to support the broadest reuse. They are very uniform in their structure. As they are the primary definitions for schema components, they do not need to restrict other data definitions, and they are not allowed to use XML Schema’s restriction mechanism (e.g., Rule 9-29, [*Complex content uses extension*](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_9-29) *(REF)*). Reference schema documents are intended to be as regular and simple as possible.

From the MPD:

A reference schema document generally applies to NIEM releases, core updates, and domain updates. Though not common, it is also valid to use a reference schema document or document set within an IEPD.

A NIEM reference schema document is intended to be the authoritative definition schema document for a NIEM target namespace, therefore, all NIEM releases, core updates, and domain updates are composed of a reference schema document set and associated namespaces. As a standalone artifact set, a reference schema document set is always harmonized such that all types and properties are semantically unique (i.e., multiple versions of semantically identical types or properties do not exist within the set).

As authoritative definitions, NIEM reference schema document sets satisfy more rigorous documentation requirements.  Typically reference schema documents contain data components with the most relaxed cardinality (zero to unbounded). However, this is not an absolute requirement. If necessary, cardinality in reference schema documents may be constrained to model reality. For example, in NIEM 3.0 a nc:Location2DGeospatialCoordinateType contains both a nc:GeographicCoordinateLatitude element and a nc:GeographicCoordinateLongitude element. Each of these elements has cardinality minOccurs="1" and maxOccurs="1". Any other cardinality for these elements has no meaning. On the other hand, one might claim that NIEM should constrain nc:PersonType to a single occurrence of the element nc:PersonBirthDate. Every person has one and only one birth date. Unfortunately, also in reality, criminal persons often present multiple identities with multiple birth dates; and so the capability to represent such is an important data requirement for NIEM.

###### replacement

###### external

[Definition: *external schema document*]

Any XML schema document that is not one of:

* a reference schema document,
* an extension schema document, or
* an XML schema document that has the structures namespace as its target namespace.

All MPD classes may contain *external schema documents* that do not conform to NIEM. Data components declared and defined in external schema documents require NIEM *external adapter types* to identify the fact they do not conform to NIEM.

[Definition: *external adapter type*]

A NIEM-conformant type that adapts external components for use within NIEM. An external adapter type creates a new class of object that embodies a single concept composed of external components. A NIEM-conformant schema defines an external adapter type.

## Profile : NIEM\_PSM\_Profile

### Overview

The NIEM PSM Profile comprises stereotypes that are used in NIEM PSMs. These  
 stereotypes need not be used with a NIEM PIM, but they may be in order to provide  
 additional platform-specific markup.Further, the NIEM PIM Profile imports the NIEM  
 Common Profile and, therefore, includes all the stereotypes and metaclasses covered by  
 that profile.

### <Stereotype> [XSDAnyProperty](#_592b854febaf5d00d830ef91173335e3)

##### Extends

##### PropertyDescription

XSDAnyProperty stereotype represents a property that is unrestricted with respect  
 to the properties type, which is implemented in XML Schema as the xsd:any  
 particle.

##### Properties

###### processContents

Determines how or if the value of a NIEM property should be processed; values  
 are: "lax", "skip", and "strict".

|  |  |
| --- | --- |
| **Type** | XSDProcessContentsCode |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### valueNamespace

The namespace in which values of this property must be defined. Implemented in  
 XML Schema as the value of the namespace attribute on the xsd:any element.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### XSDAnyPropertyType

An XSDAnyProperty must have an empty type and must not be a derived union or subset any other property.

**context** XSDAnyProperty **inv:**

self.base\_Property.type.oclIsUndefined() and  
 not(self.base\_Property.isDerivedUnion) and  
 self.base\_Property.subsettedProperty->isEmpty()

### <Stereotype> [XSDDeclaration](#_fe37b7788f1a1dcf7c9a73e489c79423)

##### Generalization

##### [References](#_c7b8a68ef50d3d361f495647dd4876ec)Description

The XSDDeclaration stereotype is a specialization of the common References  
 stereotype. However, it is constrained such that its client must be an XSDProperty  
 Property and its supplier must be an XSDProperty Property or a Namepsace Package. By  
 default, the namespace of the global XSD�property declaration referenced by  
 XSDProperty is the namespace of its class. The XSDDeclaration stereotype allows the  
 modeler to specify the namespace a XSDProperty will reference based on the namespace  
 of another XSDProperty or the target namespace of a Namespace Package. Specifically,  
 the client of the XSDDeclaration Realization shall reference the namespace indicated  
 by the supplier of the XSDDeclaration Realization, the client of the maps to one of  
 the following: an attribute use schema component or a particle component whose term  
 property is an element declaration schema component. In the first case, the supplier  
 maps to the attribute declaration schema component for the attribute use component. In  
 the second case, the supplier maps to the element declaration schema component for the  
 particle schema component.

### <Stereotype> [XSDProperty](#_ef2da866bc4fca08abc4202c19cac9e0)

##### Extends

##### PropertyDescription

An XSDProperty Property represents a NIEM property, which is implemented in XML  
 Schema as either an attribute declaration and use or an element declaration and  
 particle. If an XSDProperty Property is the client of a References Realization, then  
 the supplier of the Realization defines the declaration of the NIEM property.  
 Otherwise, the declaration of the NIEM property is defined implicitly to be the  
 top-level attribute or element definition of the same name within the target namespace  
 of the Namespace Package that contains the XSDProperty Property. All NIEM properties  
 represented by XSDProperty Properties with the same name within the same package that  
 are not clients of References Realizations share the same implicit attribute or  
 element declaration.

##### Properties

###### fixed

If present, implemented as the value of the fixed attribute of the xsd:attribute  
 or xsd:element.

|  |  |
| --- | --- |
| **Type** | String |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

###### kind

Indicates whether the NIEM property is implemented in XML Schema as an attribute  
 declaration and attribute use or element declaration and element particle: if  
 "attribute", the NIEM property is implemented in XML Schema as an attribute  
 declaration and attribute use; if "element", the NIEM property is implemented as an  
 element declaration and element particle.

|  |  |
| --- | --- |
| **Type** | XSDPropertyKindCode |
| **Multiplicity** | 1 |
| **Ordering** |  |
| **Composition** | none |

###### nillable

Implemented in XML Schema as the value of the nillable attribute on the  
 xsd:element element. Note that an XSDProperty that represents an XML attribute may  
 not have a nillable value.

|  |  |
| --- | --- |
| **Type** | Boolean |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### NDR3 [Rule 10-10] (REF,EXT). Element use from external adapter type defined by external schema documents

[[NDR] Rule 10-10](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-10), Element use from external adapter type defined by external schema documents (REF, EXT): Section 10.2.3.2, External adapter types

**context** XSDProperty **inv:**

(  
 self.namespace.stereotypedBy('AdapterType')  
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
)  
   
implies  
self.clientDependency->select(r|r.stereotypedBy('References')).supplier  
->forAll(s|s.namespace.namespace.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel).defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::external)

###### NDR3 [Rule 10-13] (REF). External attribute use only in external adapter type

[[NDR] Rule 10-13](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-13), External attribute use only in external adapter type (REF): Section 10.2.3.3, External attribute use

**context** XSDProperty **inv:**

(  
 (self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
 and  
 self.base\_Property.namespace.namespace.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(im|(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset)or(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference))  
   
 and  
 not(self.base\_Property.namespace.stereotypedBy('AdapterType'))   
)   
implies  
self.base\_Property.clientDependency->select(d|d.stereotypedBy('References')).supplier.namespace.namespace.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(typeInformationModel|typeInformationModel.defaultPurpose<>NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::external)

###### NDR3 [Rule 10-14] (REF,EXT). External attribute use has data definition

[[NDR] Rule 10-14](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-14), External attribute use has data definition (REF, EXT): Section 10.2.3.3, External attribute use

**context** XSDProperty **inv:**

(  
 (self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)  
 and  
 self.base\_Property.clientDependency->select(d|d.stereotypedBy('References')).supplier.namespace.namespace  
 ->forAll(typeInformationModel|typeInformationModel.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel).defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::external)  
)   
implies  
self.base\_Property.ownedComment.\_'body'->exists(b|not(b.oclIsUndefined()) and b<>'')

###### NDR3 [Rule 10-15] (SET). External attribute use not an ID

[[NDR] Rule 10-15](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-15), External attribute use not an ID (SET): Section 10.2.3.3, External attribute use

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)  
implies  
self.base\_Property.type->forAll(t|(t.name<>'ID')and(t.namespace.oclAsType(NamedElement).name<>'XMLPrimitiveTypes'))

###### NDR3 [Rule 10-16] (REF,EXT). External element use has data definition

[[NDR] Rule 10-16](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-16), External element use has data definition (REF, EXT): Section 10.2.3.4, External element use

**context** XSDProperty **inv:**

(  
   
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and   
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 self.clientDependency->select(d|d.stereotypedBy('References'))->exists(d|d.supplier.namespace.namespace  
 ->exists(typeInformationModel|typeInformationModel.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel).defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::external)  
 )   
)   
implies  
 self.ownedComment.\_'body'->exists(b|not(b.oclIsUndefined()) and b<>'')

###### NDR3 [Rule 10-20] (REF,EXT). Association element type is an association type

[[NDR] Rule 10-20](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-20), Association element type is an association type (REF, EXT): Section 10.3.2, Association element declarations

**context** XSDProperty **inv:**

(   
 (  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 not(self.name.oclIsUndefined())  
 and   
 self.name.endsWith('Association')  
 and  
 not(self.type.oclIsUndefined())  
 )   
 implies  
 self.type->forAll(t|t.stereotypedBy('AssociationType')or t.oclIsKindOf(AssociationClass))  
)  
and  
(  
 (  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 not(self.type.oclIsUndefined())  
 and  
 self.type->forAll(t|t.stereotypedBy('AssociationType')or t.oclIsKindOf(AssociationClass))  
 and  
 not(self.name.oclIsUndefined())  
 )   
 implies  
 self.name.endsWith('Association')  
)

###### NDR3 [Rule 10-34] (REF,EXT). Augmentation element type is an augmentation type

[[NDR] Rule 10-34](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-34), Augmentation element type is an augmentation type (REF, EXT): Section 10.4.5, Augmentation element declarations

**context** XSDProperty **inv:**

(  
 (  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 not(self.name.oclIsUndefined())  
 and  
 self.name.endsWith('Augmentation')  
 and  
 not(self.type.oclIsUndefined())  
 )   
 implies  
 self.type.stereotypedBy('AugmentationType')  
)  
and  
(  
 (  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 not(self.type.oclIsUndefined())  
 and  
 self.type.stereotypedBy('AugmentationType')  
 )   
 implies  
 self.name.endsWith('Augmentation')  
)

###### NDR3 [Rule 10-35] (REF,SET). Augmentation elements are not used directly

[[NDR] Rule 10-35](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-35), Augmentation elements are not used directly (REF, SET): Section 10.4.5, Augmentation element declarations

**context** XSDProperty **inv:**

(  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 self.name.endsWith('Augmentation')  
 and   
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
 implies  
   
 (  
 self.namespace->forAll(t|t.stereotypedBy('PropertyHolder'))  
 )

###### NDR3 [Rule 10-39] (REF,EXT). Metadata element declaration type is a metadata type

[[NDR] Rule 10-39](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-39), Metadata element declaration type is a metadata type (REF, EXT): Section 10.5.2, Metadata element declarations

**context** XSDProperty **inv:**

(  
 (  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
   
 self.name.endsWith('Metadata')  
 and   
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
 implies  
 self.type->forAll(t|t.stereotypedBy('MetadataType'))  
)  
and  
(  
 (  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 self.type->forAll(t|t.stereotypedBy('MetadataType'))  
 and   
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
 implies  
 self.name.endsWith('Metadata')  
)

###### NDR3 [Rule 10-40] (REF,EXT,SET). Metadata element has applicable elements

[[NDR] Rule 10-40](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-40), Metadata element has applicable elements (REF, EXT, SET): Section 10.5.2, Metadata element declarations

**context** XSDProperty **inv:**

The rule is definitional.

###### NDR3 [Rule 10-47] . Attribute name begins with lower case letter

[[NDR] Rule 10-47](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-47), Attribute name begins with lower case letter (REF, EXT): Section 10.8.1, Character case

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute) implies  
 self.base\_Property.name.firstToUpper()<>self.base\_Property.name

###### NDR3 [Rule 10-62] (REF,EXT). Element with simple content has representation term

[[NDR] Rule 10-62](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-62), Element with simple content has representation term (REF, EXT): Section 10.8.7, Representation terms

**context** XSDProperty **inv:**

if(  
self.namespace.namespace.stereotypedBy('InformationModel')  
 and   
not( self.type.oclIsUndefined())   
and   
self.type.oclIsKindOf(DataType)   
and  
 (   
 not(self.stereotypedBy('XSDProperty'))  
 or  
 (  
 self.stereotypedBy('XSDProperty')  
 and  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 )  
 )   
 )   
then (  
 self.name.match('.\*Name.\*') or  
 self.name.match('.\*Text.\*') or  
 self.name.match('.\*List.\*') or  
 self.name.match('.\*Quantity.\*') or  
 self.name.match('.\*Percent.\*') or  
 self.name.match('.\*Rate.\*') or  
 self.name.match('.\*Value.\*') or  
 self.name.match('.\*Numeric.\*') or  
 self.name.match('.\*Measure.\*') or  
 self.name.match('.\*Indicator.\*') or  
 self.name.match('.\*URI.\*')   
 or self.name.match('.\*ID.\*')  
 or self.name.match('.\*Time.\*') or  
 self.name.match('.\*Date.\*') or  
 self.name.match('.\*Duration.\*') or  
 self.name.match('.\*DateTime.\*') or  
 self.name.match('.\*Code.\*') or  
 self.name.match('.\*Video.\*') or  
 self.name.match('.\*Sound.\*') or  
 self.name.match('.\*Picture.\*') or  
 self.name.match('.\*Graphic.\*') or  
 self.name.match('.\*BinaryObject.\*') or  
 self.name.match('.\*Amount.\*')   
 )   
 else(true)endif

###### NDR3 [Rule 10-63] (REF,EXT). Name has representation term when appropriate

[[NDR] Rule 10-63](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-63), Name has representation term when appropriate (REF, EXT): Section 10.8.7, Representation terms

**context** XSDProperty **inv:**

Constraint is non-computable.

###### NDR3 [Rule 10-64] (REF,EXT). Name has representation term only when appropriate

[[NDR] Rule 10-64](http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/NIEM-NDR-3.0-2014-07-31.html#rule_10-64), Name has representation term only when appropriate (REF, EXT): Section 10.8.7, Representation terms

**context** XSDProperty **inv:**

Constraint is non-computable.

###### NDR3 [Rule 11-12] (REF,EXT). Element name is upper camel case

Rule 11-12. Element name is upper camel case

[Rule 11-12] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@name)]">

<sch:assert test="matches(string(@name), '^([A-Z][A-Za-z0-9\-]\*)+$')"

>The name of an element declaration MUST be upper camel case.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (  
 not(self.stereotypedBy('XSDProperty'))  
 or  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 )   
 )   
implies  
self.name.match('^([A-Z][A-Za-z0-9\\-]\*)+$')

###### NDR3 [Rule 11-13] (REF,EXT). Element type does not have a simple type name

Rule 11-13. Element type does not have a simple type name

[Rule 11-13] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)]">

<sch:assert test="not(ends-with(@type, 'SimpleType'))"

>The {type definition} of an element declaration MUST NOT have a {name} that ends in 'SimpleType'.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies  
not(self.type.name.endsWith('SimpleType'))

###### NDR3 [Rule 11-14] (REF,EXT). Element type is from conformant namespace

Rule 11-14. Element type is from conformant namespace

[Rule 11-14] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)]">

<sch:assert test="for $type-qname in resolve-QName(@type, .),

$type-namespace in namespace-uri-from-QName($type-qname) return

$type-namespace = nf:get-target-namespace(.)

or exists(nf:get-document-element(.)/xs:import[

xs:anyURI(@namespace) = $type-namespace

and empty(@appinfo:externalImportIndicator)])"

>The {type definition} of an element declaration MUST have a {target namespace} that is the target namespace, or one that is imported as conformant.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and  
 (  
 not(self.stereotypedBy('XSDProperty'))  
 or  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 )   
 )   
implies   
self.type.namespace.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant

###### NDR3 [Rule 11-15] (REF,EXT). Name of element that ends in "Abstract" is abstract

Rule 11-15. Name of element that ends in Abstract is abstract

[Rule 11-15] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@name[ends-with(., 'Abstract')]]">

<sch:assert test="exists(@abstract) and xs:boolean(@abstract) = true()"

>An element declaration with a name that ends in 'Abstract' MUST have the {abstract} property with a value of "true".</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')   
 and  
 self.name.endsWith('Abstract')  
 )   
implies   
self.isDerivedUnion

###### NDR3 [Rule 11-16] (REF,EXT). Name of element declaration with simple content has representation term

Rule 11-16. Name of element declaration with simple content has representation term

[Rule 11-16] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@name and @type

and (some $type-qname in resolve-QName(@type, .) satisfies (

nf:get-target-namespace(.) = namespace-uri-from-QName($type-qname)

and nf:resolve-type(., $type-qname)/xs:simpleContent))]">

<sch:assert test="some $representation-term in ('Amount', 'BinaryObject', 'Graphic', 'Picture', 'Sound', 'Video', 'Code', 'DateTime', 'Date', 'Time', 'Duration', 'ID', 'URI', 'Indicator', 'Measure', 'Numeric', 'Value', 'Rate', 'Percent', 'Quantity', 'Text', 'Name', 'List') satisfies

ends-with(@name, $representation-term)"

>The name of an element declaration that is of simple content MUST use a representation term.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and   
 self.type.oclIsKindOf(DataType)   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies (  
 self.name.match('.\*Name.\*') or  
 self.name.match('.\*Text.\*') or  
 self.name.match('.\*List.\*') or  
 self.name.match('.\*Quantity.\*') or  
 self.name.match('.\*Percent.\*') or  
 self.name.match('.\*Rate.\*') or  
 self.name.match('.\*Value.\*') or  
 self.name.match('.\*Numeric.\*') or  
 self.name.match('.\*Measure.\*') or  
 self.name.match('.\*Indicator.\*') or  
 self.name.match('.\*URI.\*')   
 or self.name.match('.\*ID.\*')  
 or self.name.match('.\*Time.\*') or  
 self.name.match('.\*Date.\*') or  
 self.name.match('.\*Duration.\*') or  
 self.name.match('.\*DateTime.\*') or  
 self.name.match('.\*Code.\*') or  
 self.name.match('.\*Video.\*') or  
 self.name.match('.\*Sound.\*') or  
 self.name.match('.\*Picture.\*') or  
 self.name.match('.\*Graphic.\*') or  
 self.name.match('.\*BinaryObject.\*') or  
 self.name.match('.\*Amount.\*')   
 )

###### NDR3 [Rule 11-17] (SET). Name of element declaration with simple content has representation term

Rule 11-17. Name of element declaration with simple content has representation term

This rule only checks the cases not testable in the (REF, EXT) version.

[Rule 11-17] (SET) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[@name and @type

and (nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ReferenceSchemaDocument'))

or nf:has-effective-conformance-target-identifier(., xs:anyURI('http://reference.niem.gov/niem/specification/naming-and-design-rules/3.0/#ExtensionSchemaDocument')))

and (some $type-qname in resolve-QName(@type, .) satisfies (

nf:get-target-namespace(.) != namespace-uri-from-QName($type-qname)

and nf:resolve-type(., $type-qname)/xs:simpleContent))]">

<sch:assert test="some $representation-term in ('Amount', 'BinaryObject', 'Graphic', 'Picture', 'Sound', 'Video', 'Code', 'DateTime', 'Date', 'Time', 'Duration', 'ID', 'URI', 'Indicator', 'Measure', 'Numeric', 'Value', 'Rate', 'Percent', 'Quantity', 'Text', 'Name', 'List') satisfies

ends-with(@name, $representation-term)"

>the name of an element declaration that is of simple content MUST use a representation term.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and   
 self.type.oclIsKindOf(DataType)   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies (  
 self.name.match('.\*Name.\*') or  
 self.name.match('.\*Text.\*') or  
 self.name.match('.\*List.\*') or  
 self.name.match('.\*Quantity.\*') or  
 self.name.match('.\*Percent.\*') or  
 self.name.match('.\*Rate.\*') or  
 self.name.match('.\*Value.\*') or  
 self.name.match('.\*Numeric.\*') or  
 self.name.match('.\*Measure.\*') or  
 self.name.match('.\*Indicator.\*') or  
 self.name.match('.\*URI.\*')   
 or self.name.match('.\*ID.\*')  
 or self.name.match('.\*Time.\*') or  
 self.name.match('.\*Date.\*') or  
 self.name.match('.\*Duration.\*') or  
 self.name.match('.\*DateTime.\*') or  
 self.name.match('.\*Code.\*') or  
 self.name.match('.\*Video.\*') or  
 self.name.match('.\*Sound.\*') or  
 self.name.match('.\*Picture.\*') or  
 self.name.match('.\*Graphic.\*') or  
 self.name.match('.\*BinaryObject.\*') or  
 self.name.match('.\*Amount.\*')   
 )

###### NDR3 [Rule 11-18] (REF,EXT). Element substitution group defined by conformant schema

Rule 11-18. Element substitution group defined by conformant schema

[Rule 11-18] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@substitutionGroup)]">

<sch:let name="namespace" value="namespace-uri-from-QName(resolve-QName(@substitutionGroup, .))"/>

<sch:assert test="$namespace = nf:get-target-namespace(.)

or exists(ancestor::xs:schema[1]/xs:import[exists(@namespace)

and $namespace = xs:anyURI(@namespace)

and empty(@appinfo:externalImportIndicator)])"

>An element substitution group MUST have either the target namespace or a namespace that is imported as conformant.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies   
self.subsettedProperty.namespace.namespace->forAll(m|m.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant)

###### NDR3 [Rule 11-19](REF,EXT). Attribute type defined by conformant schema

Rule 11-19. Attribute type defined by conformant schema

[Rule 11-19] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:let name="namespace" value="namespace-uri-from-QName(resolve-QName(@type, .))"/>

<sch:assert test="$namespace = (nf:get-target-namespace(.), xs:anyURI('http://www.w3.org/2001/XMLSchema'))

or exists(ancestor::xs:schema[1]/xs:import[exists(@namespace)

and $namespace = xs:anyURI(@namespace)

and empty(@appinfo:externalImportIndicator)])"

>The type of an attribute declaration MUST have the target namespace or the XML Schema namespace or a namespace that is imported as conformant.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)  
implies   
self.base\_Property.type.namespace.namespace->forAll(m|m.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant)

###### NDR3 [Rule 11-20] (REF,EXT). Attribute name uses representation term

Rule 11-20. Attribute name uses representation term

[Rule 11-20] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@name)]">

<sch:assert test="some $representation-term in ('Amount', 'BinaryObject', 'Graphic', 'Picture', 'Sound', 'Video', 'Code', 'DateTime', 'Date', 'Time', 'Duration', 'ID', 'URI', 'Indicator', 'Measure', 'Numeric', 'Value', 'Rate', 'Percent', 'Quantity', 'Text', 'Name', 'List') satisfies

ends-with(@name, $representation-term)"

>An attribute name MUST end with a representation term.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies(  
 self.base\_Property.name.endsWith('List') or  
 self.base\_Property.name.endsWith('Name') or  
 self.base\_Property.name.endsWith('Text') or  
 self.base\_Property.name.endsWith('Quantity') or  
 self.base\_Property.name.endsWith('Percent') or  
 self.base\_Property.name.endsWith('Rate') or  
 self.base\_Property.name.endsWith('Value') or  
 self.base\_Property.name.endsWith('Numeric') or  
 self.base\_Property.name.endsWith('Measure') or  
 self.base\_Property.name.endsWith('Indicator') or  
 self.base\_Property.name.endsWith('URI') or   
 self.base\_Property.name.endsWith('ID')  
 or self.base\_Property.name.endsWith('Time') or  
 self.base\_Property.name.endsWith('Duration') or  
 self.base\_Property.name.endsWith('Date') or  
 self.base\_Property.name.endsWith('DateTime') or  
 self.base\_Property.name.endsWith('Code') or  
 self.base\_Property.name.endsWith('Video') or  
 self.base\_Property.name.endsWith('Sound') or  
 self.base\_Property.name.endsWith('Picture') or  
 self.base\_Property.name.endsWith('Graphic') or  
 self.base\_Property.name.endsWith('BinaryObject') or  
 self.base\_Property.name.endsWith('Amount')   
)

###### NDR3 [Rule 11-21] (REF,EXT). Element or attribute declaration introduced only once into a type

Rule 11-21. Element or attribute declaration introduced only once into a type

[Rule 11-21] (REF, EXT) (Constraint)

Within the schema, an element declaration or attribute decaration MUST NOT be introduced more than once into a type definition. This applies to content acquired by a type by any means, including from a base type definition, via element substitution groups, or through the use of attribute groups.

This rule ensures that a type definition does not incorporate a component multiple times. As information exchange specifications often contain multiple versions of schemas, including reference schemas as well as subset and constraint schemas, it may be easy to omit an element or attribute in one version of the schema, only to reincorporate it via an extension. This can cause difficulties in integrating such schemas, as it may be impossible to use a reference schema if an attribute is added twice, in both a base type and an extension type, since that would make it an invalid schema.

Incorporating a component multiple times can also make it difficult to avoid violating XML Schema’s unique particle attribution constraint, which is described by XML Schema Structures Section 3.8.6[, *Constraints on Model Group Schema Components*](http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/#cos-nonambig). This can create difficulty if an element is added both directly, and via a substitution group head. In such a case, a parser may not be able to determine which element use is responsible for an element in an instance, which is a violation of the UPA constraint.

This rule is also intended to prevent developers from creating complicated sequences of recurring elements. Such definitions are difficult for developers to satisfy in code, and can cause havoc with XML Schema language binding tools. If an element is needed more than once, or if a particular sequence of elements is needed, a developer should consider the use of flexible content models (via substitution groups) along with additional rules.

**context** XSDProperty **inv:**

(  
 not(self.name.oclIsUndefined())  
 and  
 (self.name<>'')  
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies   
self.namespace->select(c|c.oclIsKindOf(Classifier)).oclAsType(Classifier).allAttributes()  
->select(a|(a.name=self.name)and(a<>self))->isEmpty()

###### NDR3 [Rule 11-22] (REF,EXT). Element reference defined by conformant schema

Rule 11-22. Element reference defined by conformant schema

[Rule 11-22] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(ancestor::xs:complexType[empty(@appinfo:externalAdapterTypeIndicator)]) and @ref]">

<sch:let name="namespace" value="namespace-uri-from-QName(resolve-QName(@ref, .))"/>

<sch:assert test="$namespace = nf:get-target-namespace(.)

or exists(ancestor::xs:schema[1]/xs:import[exists(@namespace)

and $namespace = xs:anyURI(@namespace)

and empty(@appinfo:externalImportIndicator)])"

>An element reference MUST be to a component that has a namespace that is either the target namespace of the schema document in which it appears, or which is imported as conformant by that schema document.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 )   
implies   
self.clientDependency->select(d|d.stereotypedBy('References')).supplier->select(s|s.oclIsKindOf(Property)).namespace.namespace  
->forAll(m|m.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant)

###### NDR3 [Rule 11-23] (REF,EXT). Referenced attribute defined by conformant schemas

Rule 11-23. Referenced attribute defined by conformant schemas

[Rule 11-23] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[@ref]">

<sch:let name="namespace" value="namespace-uri-from-QName(resolve-QName(@ref, .))"/>

<sch:assert test="some $namespace in namespace-uri-from-QName(resolve-QName(@ref, .)) satisfies (

$namespace = nf:get-target-namespace(.)

or ancestor::xs:schema[1]/xs:import[

@namespace

and $namespace = xs:anyURI(@namespace)

and empty(@appinfo:externalImportIndicator)])"

>An attribute {}ref MUST have the target namespace or a namespace that is imported as conformant.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)  
implies   
self.base\_Property.clientDependency->select(d|d.stereotypedBy('References')).supplier->select(s|s.oclIsKindOf(Property)).namespace.namespace  
->forAll(m|m.appliedStereotype('Namespace').oclAsType(NIEM\_UML\_Profile::NIEM\_Common\_Profile::Namespace).isConformant)

###### NDR3 [Rule 11-31] (REF,EXT). Standard opening phrase for element

Rule 11-31. Standard opening phrase for element

[Rule 11-31] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[ends-with(@name, 'AugmentationPoint')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(starts-with(lower-case(normalize-space(.)), 'an augmentation point '))"

>The data definition for an augmentation point element SHOULD begin with standard opening phrase "an augmentation point...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Augmentation')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(some $phrase in ('supplements ', 'additional information about ')

satisfies starts-with(lower-case(normalize-space(.)), $phrase))"

>The data definition for an augmentation element SHOULD begin with the standard opening phrase "supplements..." or "additional information about...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Metadata')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '(metadata about|information that further qualifies)'))"

>The data definition for a metadata element SHOULD begin with the standard opening phrase "metadata about..." or "information that further qualifies...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Association') and empty(@abstract)

]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)),

'^an?( .\*)? (relationship|association)'))"

>The data defintion for an association element that is not abstract SHOULD begin with the standard opening phrase "an (optional adjectives) (relationship|association)...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[xs:boolean(@abstract) = true()

]/xs:annotation/xs:documentation[1]">

<sch:report test="not(starts-with(lower-case(normalize-space(.)), 'a data concept'))"

>The data defintion for an abstract element SHOULD begin with the standard opening phrase "a data concept...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Date')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^an?( .\*)? (date|month|year)'))"

>The data defintion for an element with a date representation term SHOULD begin with the standard opening phrase "a(n?) (optional adjectives) (date|month|year)...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Quantity')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^an?( .\*)? (count|number)'))"

>The data defintion for an element with a quantity representation term SHOULD begin with the standard opening phrase "an (optional adjectives) (count|number)...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Picture')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^an?( .\*)? (image|picture|photograph)'))"

>The data defintion for an element with a picture representation term SHOULD begin with the standard opening phrase "an (optional adjectives) (image|picture|photograph)".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Indicator')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^true if .\*; false (otherwise|if)'))"

>The data defintion for an element with an indicator representation term SHOULD begin with the standard opening phrase "true if ...; false (otherwise|if)...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Identification')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^an?( .\*)? identification'))"

>The data defintion for an element with an identification representation term SHOULD begin with the standard opening phrase "(a|an) (optional adjectives) identification...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[ends-with(@name, 'Name')]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^(a|an)( .\*)? name'))"

>The data defintion for an element with a name representation term SHOULD begin with the standard opening phrase "(a|an) (optional adjectives) name...".</sch:report>

</sch:rule>

<sch:rule context="xs:element[@name]/xs:annotation/xs:documentation[1]">

<sch:report test="not(matches(lower-case(normalize-space(.)), '^an? '))"

>The data defintion for an element declaration with a name SHOULD begin with the standard opening phrase "(a|an)".</sch:report>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (  
 (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element)  
 or  
 not(self.stereotypedBy('XSDProperty'))  
 )   
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')   
 and  
 not(self.name.oclIsUndefined())  
 and  
 (self.name<>'')  
 )   
implies   
if (self.name.endsWith('AugmentationPoint')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.startsWith('an augmentation point '))   
 else if (self.name.endsWith('Augmentation')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.startsWith('supplements ')or b.startsWith('additional information about '))   
 else if (self.name.endsWith('Metadata')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('(metadata about|information that further qualifies)'))   
 else if (self.name.endsWith('Association')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^an?( .\*)? (relationship|association)'))   
 else if (self.type.oclIsUndefined()) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('a data concept'))   
 else if (self.name.endsWith('Date')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^an?( .\*)? (date|month|year)'))   
 else if (self.name.endsWith('Quantity')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^an?( .\*)? (count|number)'))   
 else if (self.name.endsWith('Picture')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^an?( .\*)? (image|picture|photograph)'))   
 else if (self.name.endsWith('Indicator')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^true if .\*; false (otherwise|if)'))   
 else if (self.name.endsWith('Identification')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^an?( .\*)? identification'))   
 else if (self.name.endsWith('Name')) then self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('^(a|an)( .\*)? name'))   
 else (self.ownedComment.\_'body'.toLower().normalizeSpace()->exists(b|b.match('(a|an)')))   
 endif  
 endif   
 endif  
 endif  
 endif  
 endif   
 endif   
 endif   
 endif   
 endif   
endif

###### NDR3 [Rule 9-37] (REF,EXT). Untyped element is abstract

Rule 9-37. Untyped element is abstract

[Rule 9-37] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:schema/xs:element[empty(@type)]">

<sch:assert test="exists(@abstract)

and xs:boolean(@abstract) = true()"

>A top-level element declaration that does not set the {type definition} property via the attribute "type" MUST have the {abstract} property with a value of "true".</sch:assert>

</sch:rule>

</sch:pattern>

Untyped element declarations act as wildcards that may carry arbitrary data. By declaring such types abstract, NIEM allows the creation of type independent semantics without allowing arbitrary content to appear in XML instances.

**context** XSDProperty **inv:**

(  
 self.namespace.namespace.stereotypedBy('InformationModel')   
 and  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 )   
 implies  
(self.type.oclIsUndefined() implies self.isDerivedUnion)

###### NDR3 [Rule 9-38] (REF,EXT). Element of type xs:anySimpleType is abstract

The type xs:anySimpleType does not have any concrete semantics; The use of xs:anySimpleType is limited to the case where an abstract element is of type xs:anySimpleType, to act as a base for concrete implementations of the element.

[Rule 9-38] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:element[exists(@type)

and resolve-QName(@type, .) = xs:QName('xs:anySimpleType')]">

<sch:assert test="exists(@abstract)

and xs:boolean(@abstract) = true()"

>An element declaration that has a type xs:anySimpleType MUST have the {abstract} property with a value of "true".</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(  
 (not(self.stereotypedBy('XSDProperty')) or (self.appliedStereotype('XSDProperty').oclAsType(NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDProperty).kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::element))  
 and  
 self.namespace.namespace.stereotypedBy('InformationModel')  
 and   
 (self.type.name='anySimpleType')  
 and   
 (self.type.\_'package'.name='XMLPrimitiveTypes')  
)   
implies   
self.isDerivedUnion

###### NDR3 [Rule 9-43] (REF). No element disallowed derivation

Rule 9-43. No element disallowed derivation

[Rule 9-43] (REF) (Constraint)

<sch:pattern>

<sch:rule context="xs:element">

<sch:assert test="empty(@final)"

>An element xs:element MUST NOT have an attribute {}final.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

self.namespace.namespace.stereotypedBy('InformationModel')  
implies  
(  
 self.namespace.namespace.appliedStereotype('InformationModel').oclAsType(NIEM\_UML\_Profile::NIEM\_PIM\_Profile::InformationModel)  
 ->forAll(im|  
 (im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::reference)  
 or(im.defaultPurpose=NIEM\_UML\_Profile::NIEM\_PIM\_Profile::DefaultPurposeCode::subset)  
 )  
 implies  
 not(self.isLeaf)  
)

###### NDR3 [Rule 9-50] (REF,EXT). No attribute type of xs:ID

Rule 9-50. No attribute type of xs:ID

[Rule 9-50] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:ID')"

>A schema component MUST NOT have an attribute {}type with a value of xs:ID.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='ID')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-51] (REF,EXT). No attribute type of xs:IDREF

Rule 9-51. No attribute type of xs:IDREF

[Rule 9-51] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:IDREF')"

>A schema component MUST NOT have an attribute {}type with a value of xs:IDREF.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='IDREF')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-52] (REF,EXT). No attribute type of xs:IDREFS

Rule 9-52. No attribute type of xs:IDREFS

[Rule 9-52] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:IDREFS')"

>A schema component MUST NOT have an attribute {}type with a value of xs:IDREFS.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='IDREFS')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-53] (REF,EXT). No attribute type of xs:ENTITY

Rule 9-53. No attribute type of xs:ENTITY

[Rule 9-53] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:ENTITY')"

>A schema component MUST NOT have an attribute {}type with a value of xs:ENTITY.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='ENTITY')and(t.\_'package'.name='XMLPrimitiveTypes')) )

###### NDR3 [Rule 9-54] (REF,EXT). No attribute type of xs:ENTITIES

Rule 9-54. No attribute type of xs:ENTITIES

[Rule 9-54] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:ENTITIES')"

>A schema component MUST NOT have an attribute {}type with a value of xs:ENTITIES.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='ENTITIES')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-55] (REF,EXT). No attribute type of xs:anySimpleType

Rule 9-55. No attribute type of xs:anySimpleType

[Rule 9-55] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute[exists(@type)]">

<sch:assert test="resolve-QName(@type, .) != xs:QName('xs:anySimpleType')"

>A schema component MUST NOT have an attribute {}type with a value of xs:anySimpleType.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies   
self.base\_Property.type->forAll(t|not((t.name='anySimpleType')and(t.\_'package'.name='XMLPrimitiveTypes')))

###### NDR3 [Rule 9-56] (REF,EXT). No attribute default values

Rule 9-56. No attribute default values

[Rule 9-56] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute">

<sch:assert test="empty(@default)"

>An element xs:attribute MUST NOT have an attribute {}default.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

This constraint enforced by provisioning, there are no @default attributes generated for an xs:attribute within a target InformationModel schema.

###### NDR3 [Rule 9-57] (REF,EXT). No attribute fixed values

Rule 9-57. No attribute fixed values

[Rule 9-57] (REF, EXT) (Constraint)

<sch:pattern>

<sch:rule context="xs:attribute">

<sch:assert test="empty(@fixed)"

>An element xs:attribute MUST NOT have an attribute {}fixed.</sch:assert>

</sch:rule>

</sch:pattern>

**context** XSDProperty **inv:**

This constraint enforced by provisioning, there are no @fixed attributes generated for an xs:attribute within a target InformationModel schema.

###### XSDProperty AttributeKind

If kind=attribute, then an XSDProperty must have multiplicity 1..1, must not be a derived union and must not subset any other property. If the type is not empty, it must be a DataType.

**context** XSDProperty **inv:**

(self.kind=NIEM\_UML\_Profile::NIEM\_PSM\_Profile::XSDPropertyKindCode::attribute)   
implies(  
 (self.base\_Property.upper=1)  
 and (self.base\_Property.lower=1)   
 and not (self.base\_Property.isDerivedUnion)  
 and self.base\_Property.subsettedProperty->isEmpty()  
 and( not(self.base\_Property.type.oclIsUndefined()) implies self.base\_Property.type.oclIsKindOf(DataType) )  
 )

###### XSDProperty Owner

An XSDProperty must be the ownedAttribute of a DataType or a Class stereotyped as a NIEMType.

**context** XSDProperty **inv:**

self.base\_Property.namespace.namespace.stereotypedBy('InformationModel')

### <Stereotype> [XSDRepresentationRestriction](#_c8fef7fb3c39b93b2966597e780b26b2)

##### Extends

##### DataTypeDescription

XSDRepresentationRestriction specifies a restriction on the representation in an  
 XML schema of the values of a base DataType.

##### Properties

###### whiteSpace

whiteSpace is a restriction on the value space of the DataType. It is  
 implemented in XML Schema as the value of the value attribute on the xsd:whiteSpace  
 element, the child of the xsd:restriction element which is the immediate child of  
 the xsd:simpleType element.

|  |  |
| --- | --- |
| **Type** | XSDWhiteSpaceCode |
| **Multiplicity** | 0..1 |
| **Ordering** |  |
| **Composition** | none |

##### Constraints

###### must have one generalization

A DataType with an XSDRepresentationRestriction must have exactly one

generalization.

**context** XSDRepresentationRestriction **inv:**

self.base\_DataType.generalization->size()=1

### <Stereotype> [XSDSimpleContent](#_2382969db62499e782e09fe85837f61b)

##### Extends

##### RealizationDescription

XSDSimpleContent represents a relationship between two type definitions: the first  
 is a complex type definition with simple content whose content type is the second.  
 This relationship is implemented in XML Schema through base attribute on the  
 xsd:extension or xsd:restriction element of the first type definition, the actual  
 value of which resolves to the second type definition. Section 3.4 of XML Schema  
 Structures addresses simple content types in XML Schema; Sections 6.5.1, 6.5.2, and  
 7.4 of NIEM NDR v1.3 address simple content types in NIEM-conformant XML  
 Schema.

##### Constraints

###### Client must be a <<NIEMType>>

The client of an XSDSimpleContent Realization must be a Class stereotyped as a NIEMType.

**context** XSDSimpleContent **inv:**

self.base\_Realization.client->forAll(client|client.oclIsKindOf(Classifier) and client.namespace.stereotypedBy('InformationModel'))

###### supplier must be a <DataType>

The suppler of an XSDSimpleContent Realization must be a DataType.

**context** XSDSimpleContent **inv:**

self.base\_Realization.supplier->forAll(s|s.oclIsKindOf(DataType))

### <Enumeration> [XSDProcessContentsCode](#_6057dd3a5a78e3c24dd275e3132ced75)

XSDProcessContentsCode supports the processContents attribute of the  
 XSDAnyProperty stereotype.

##### Literals

###### strict

###### lax

###### skip

### <Enumeration> [XSDPropertyKindCode](#_c99713ce9776ea74e37055d9ba6c7754)

XSDPropertyKindCode supports the kind attribute of XSDProperty by providing values  
 to specify if an XSD property is represented as an xsd:element or  
 xsd:attribute.

##### Literals

###### element

###### attribute

### <Enumeration> [XSDWhiteSpaceCode](#_7b3de59b6bd2d58d0e6f7cf6d38f5a89)

Enumeration XSDWhiteSpaceCode supports the whiteSpace attribute of the  
 XSDWhiteSpaceCode attribute as per the XSD definitions.

##### Literals

###### replace

###### collapse

###### preserve

## Profile : NIEM\_UML\_Profile

### Overview

The NIEM UML Profile imports the NIEM PIM Profile, the NIEM PSM Profile and the  
 Model Package Description Profile, so all three of these profiles can effectively be  
 imported just by importing the single NIEM UML Profile.

# NIEM-UML Transformation Reference