INTERNATIONAL STANDARD

ISO 20022-8

First edition
2013-05-01

Financial services — Universal financial industry message scheme —

Part 8: ASN.1 generation

Services financiers — Schéma universel de messages pour l'industrie financière —

Partie 8: Génération ASN.1

Reference number
ISO 20022-8:2013(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 20022-8 was prepared by Technical Committee ISO/TC 68, *Financial services*.

ISO 20022 consists of the following parts, under the general title *Financial services — Universal financial industry message scheme*:

— *Part 1: Metamodel*

— *Part 2: UML profile*

— *Part 3: Modelling*

— *Part 4: XML Schema generation*

— *Part 5: Reverse engineering*

— *Part 6: Message transport characteristics*

— *Part 7: Registration*

— *Part 8: ASN.1 generation*

ISO 20022-1:2013, ISO 20022-2:2013, ISO 20022-3:2013, ISO 20022-4:2013, ISO 20022-5:2013, ISO 20022-6:2013, ISO 20022-7:2013 and ISO 20022-8:2013 will be implemented by the Registration Authority by no later than the end of May 2013, at which time support for the concepts set out within them will be effective. Users and potential users of the ISO 20022 series are encouraged to familiarize themselves with the 2013 editions as soon as possible, in order to understand their impact and take advantage of their content as soon as they are implemented by the Registration Authority. For further guidance, please contact the Registration Authority.

For the purposes of research on financial industry message standards, users are encouraged to share their views on ISO 20022:2013 and their priorities for changes to future editions of the document. Click on the link below to take part in the online survey:

http://www.surveymonkey.com/s/20022_2013

Introduction

This International Standard defines a scalable, methodical process to ensure consistent descriptions of messages throughout the financial services industry.

The purpose of this International Standard is to describe precisely and completely the externally observable aspects of financial services messaging in a way that can be verified independently against operational messaging.

The trigger for the creation of this International Standard was the rapid growth in the scale and sophistication of messaging within financial services during the 1990s using ISO 15022. The financial services industry (from here on referred to as "the industry") created the first version of this International Standard as the successor to ISO 15022 in response to that trigger. Since ISO 15022, the industry has broadened the scope from securities to the entire industry for this International Standard.

This International Standard is based on open technology standards, which historically have evolved more rapidly than the industry itself. Consequently, this International Standard adopted a model-driven approach where the model of the industry's messaging can evolve separately from the evolution of the messaging technology standards. The period during which this International Standard has emerged followed the widespread adoption of the World Wide Web (the Web) for business. XML (eXtensible Mark-up Language) emerged as the *de facto* standard for document representation on the Web and it became the first syntax for ISO 20022.

The modelling process is further refined into three levels which, in addition to the messaging technology standard, is why this International Standard is based on four levels: the Scope level, the Conceptual level, the Logical level and the Physical level.

This four-level approach is based on the first four levels of the Zachman Framework. The remaining two levels of the Zachman Framework are equivalent to the implementations and the operational levels, respectively.

In ISO 20022-1, the first, second and third levels are described in UML (Unified Modelling Language) because it is widely supported and supports multiple levels of abstraction. The models created in accordance with this International Standard are technology independent in that they do not require any particular physical expression or implementation. Such models aim to describe all parts of the message exchange. The models form the definition of the protocol between participants exchanging messages. This International Standard defines a method that describes a process by which these models can be created and maintained by the modellers.

The models and the Physical level artefacts are stored in a central repository, serviced by a Registration Authority. This International Standard's repository is available on the World Wide Web and offers public access for browsing.

The Repository is organized into two areas:

— A DataDictionary containing the industry model elements likely to have further or repeated use.

— A BusinessProcessCatalogue that contains models describing specific message definitions and business processes, and physical syntax implementations.

This International Standard is organized into the following parts.

— ISO 20022-1 describes in MOF (Meta-Object Facility) the metamodel of all the models and the Repository.

— ISO 20022-2 covers the UML profile, a grounding of general UML into a specific subset defined for this International Standard (to be used when UML is selected to define the models).

— ISO 20022-3 describes a modelling method to produce models for this International Standard.

— ISO 20022-4 covers XML schema generation rules to transform a Logical level model into a Physical level description in the syntaxes.

— ISO 20022-5 covers logical model alignment and reverse engineering of existing message syntaxes.

— ISO 20022-6 covers message transport characteristics that define the quality of service required by the business process definitions so that they can operate successfully.

— ISO 20022-7 describes the process of managing the registration of models and physical syntax implementations.

— This part of ISO 20022 gives ASN.1 syntax generation rules to transform a Logical level model into a Physical level description in ASN.1.

Financial services — Universal financial industry message scheme —

Part 8:
ASN.1 generation

1 Scope

This part of ISO 20022 describes the transformation rules to generate ASN.1 abstract syntax from an ISO 20022 compliant MessageDefinition. The generated abstract syntax is for the description and validation of Messages.

The transformation rules are a transformation from Level 3 to Level 4. It is a deterministic transformation, meaning that the resulting ASN.1 is completely predictable for a given MessageDefinition. There is neither manual input to the transformation itself nor manual adjustment to the result of the transformation.

This part of ISO 20022 is the ASN.1 equivalent of ISO 20022-4. In ISO 20022-4 the abstract syntax generated is XML Schema; in this part of ISO 20022 it is ASN.1. In ISO 20022-4 the only encoding supported is UTF-8 XML; in this part there are multiple encodings supported for ASN.1. These include all the standard encodings, but in addition the ability to register custom encodings in ECN.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8825-5:2008, *Information technology — ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1 — Part 5*

ISO 20022-1, *Financial services — Universal financial industry message scheme — Part 1: Metamodel*

W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, (28 October 2004)

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 20022-1 apply.

NOTE Within ASN.1 notation, XSD refers to the XSD Module defined by ISO/IEC 8825-5.

4 Background

Abstract Syntax Notation One (ASN.1) is an International Standard and flexible notation that describes data structures for representing, encoding, transmitting and decoding data. It provides a set of rules for describing

the structure of objects that are independent of machine-specific encoding techniques and it is a precise, formal notation that removes ambiguities.

ASN.1 is a joint ISO/IEC and ITU-T standard, originally defined in 1984 as part of CCITT X.409:1984. ASN.1 constituted its own standard, X.208, in 1988 due to wide applicability. The substantially revised 1995 version is covered by the X.680 series. The current version is backward compatible with the 1995 version.

ISO/IEC 8824 covers the ASN.1 abstract syntax; ISO/IEC 8825 covers encodings for the ASN.1 abstract syntax. The abstract syntax is roughly analogous to XML Schema, and the encodings roughly analogous to XML. The XML Schema Definition Language (XSD) Recommendation is written in terms of the XML Recommendation, which makes substituting XML for a different encoding impractical. XSD effectively limits the encoding to XML.

Parts 1 to 5 of ISO 20022-5 published in 2004 use only one syntax, XML Schema, and only one encoding, UTF-8 encoded XML. Support for ASN.1 has been added to ISO 20022 in order to support scenarios in which its syntax and encodings are more suitable. It is up to users of this International Standard to decide for which scenarios ASN.1 is suitable.

The consensus that led to the inclusion of ASN.1 was a history of its usage in messaging, good support, and a reputation for small and fast to parse messages. Encoding Control Notation had been identified as a way to support existing encodings within 20022 without compromising the abstract syntax.

The addition of multiple encodings to ISO 20022 gives up wire level compatibility in return for broadening the domain in which ISO 20022 can standardize the Business Process and Message Definitions. The Messages remain syntactically compatible, which is less than wire-level compatible and more than semantically compatible. In practice, we expect an ISO 20022 Messaging Library to be identical for encodings based on ASN.1 and XSD.

ASN.1 may be used as a schema language for XML. The generated ASN.1 Abstract Syntax will contain XER tags to guide the encoding as XML. An XML Message will be equivalent, whether encoded from the ISO 20022 XSD or the ISO 20022 ASN.1 Abstract Syntax. For any Message expressed in ASN.1, when encoded to XML using XER, will produce an XML Message that is valid against the XSD for the same Message Definition. It should not matter whether the XML is produced using the XSD or ASN.1 as the ASN.1 Abstract Syntax and XSD are equivalent through their XML being compatible; the design of the ASN.1 is limited to features and design idioms with a direct equivalent in XSD. The scopes of an ASN.1 Module and an XSD Schema namespace are equivalent.

The ASN.1 generation takes a single parameter as input, which is a MessageSet. The output of the generation is ASN.1 Abstract Syntax.

Each Message Definition is mapped to a single Module containing a single PDU. Each Module is stored in a separate file with a file extension of ".asn".

Each MessageSet is mapped to a collection of PDUs. Each MessageSet is stored as a Zip file containing all the ".asn" files for the MessageDefinitions.

The default encodings are Aligned PER and XML using XER. The use of every other encoding shall be Registered. Any encoding that is not an International Standard shall also be described using ECN and that description shall be Registered.

NOTE This part of ISO 20022 explains how a given MessageDefinition will be mapped into ASN.1; it does not explain how to create a MessageDefinition. This information can be found in ISO 20022-3.

5 ISO 20022 transformation rules for MessageSet

5.1 Registration and Repository

ASN.1 is present in the Repository as a SyntaxScheme. Every ISO/IEC 8825 Encoding shall be added to the Repository as an EncodingScheme for the ASN.1 Syntax Scheme. The Registration Authority may register additional EncodingSchemes in the Repository for ASN.1 if they can be completely and precisely described in ECN. The definition in ECN shall be included in the Repository.

5.2 Preconditions

The input parameter to the generation is a single Message Set.

The MessageSet used as input for the transformation is a valid instance of the MessageSet metaclass.

5.3 Transformation constraints

The generated Abstract Syntax contains a Comment at the start of the document containing metadata. This information comes from the generator, not the MessageSet.

— The ISO 20022 RA issued release number.

EXAMPLE R6.1.0.2.

— The generation date in ISO 8601 format.

EXAMPLE 2009-06-30Z.

— Documentation text (optional). The content of this field is undefined in this International Standard.

EXAMPLE R6.1.0.2 2009-06-30Z.

The Abstract Syntax is a syntactical representation of the MessageDefinition. The Abstract Syntax does not contain the full semantics of a Message. The MessageDefinition is always the definitive description of the semantics of a Message.

The Abstract Syntax is at Level 4 of the ISO 20022 levels. Information about types such as Message Components and Message Elements that are common across Message Definitions is lost at Level 4. For example, the commonality of a single Message Element in two different Message Definitions will be lost in the Abstract Syntax because they will generate to separate Modules for each Message Definition. The semantic relationship between PDUs can only be understood at Level 3 and Level 2.

All Types appear in the ASN.1 in alphabetical order, using the ASN.1 Type Name as the sort key.

5.4 Module Header

5.4.1 General

Each item in the Module Header appears on a separate line.

5.4.2 Module Name

The Module Name is the concatenation of "ISO20022-" and the MessageDefinitionIdentifier property.

The resulting Module Name might require alterations to be legal ASN.1:

a) if the first character of the Module Name is not an an alphabetic character, then add a prefix of "U";

b) if the first character is a lower case alphabetic character, then convert it to its equivalent upper case alphabetic character;

c) any other invalid characters in the name are substituted with hyphens.

NOTE 1 This step is necessary as ASN.1 has tighter requirements for a Module Name than ISO 20022 has for a Message Definition Identifier.

NOTE 2 If the Module Name ever changes, a) and b) become effective. Until then, they bring no value.

EXAMPLE ISO20022-AccountDetailsConfirmationV02.

5.4.3 Module identification

Each Module is identified by an Object Identifier. The Object Identifier comes from the Message Definition's OID property.

EXAMPLE AccountDetailsConfirmationV02 {iso(1) standard(0) unifi(20022) accountDetailsConfirmationV02(12345) }.

5.4.4 Definition of the tagging environment

The tagging environment is defined as "XER INSTRUCTIONS" and "AUTOMATIC TAGS".

5.4.5 Definition of the extensibility environment

All XSD datatypes used in the MessageSet are Imported from OID {joint-iso-itu-t(2) asn1(1) specification(0) modules(0) xsd-module(2)}.

The Module for XSD is always generated; the name of the document is always XSDv2.asn.

NOTE The name of the document is the filename for a file system. The Imports for the generated Module include every type from the XSD Module used.

The Imports include any type XSD type used.

5.5 Granularity of Modules

There is one well-formed and valid ASN.1 Module per MessageDefinition.

NOTE 1 The declaration of which types are PDUs is not declared in an ASN.1 Module. The generated Module will not indicate which types are PDUs.

NOTE 2 For ISO 20022 any generated ASN.1 Type not referenced by another Type can be considered to be a PDU. This holds true because this International Standard only generates types referred to directly or indirectly from the Message Definition.

NOTE 3 The PDU is analogous to the XML Schema generation's global root element, derived from MessageDefinition.rootElement.

5.6 Encoding Messages

5.6.1 Encoding

ASN.1 is to be added to the Repository as a SyntaxScheme.

Every ISO/IEC 8825 Encoding is to be added to the Repository as an EncodingScheme for the ASN.1 Syntax Scheme.

The Registration Authority may register additional EncodingSchemes in the Repository for ASN.1 if they can be completely and precisely described in ECN. The definition in ECN shall be included in the Repository.

An ISO 20022 Valid Message shall comply with a SyntaxScheme and EncodingScheme in the Repository. No other SyntaxScheme or EncodingSchemes are valid for ISO 20022.

5.7 Completeness

The list of transformation rules described in this clause is complete, which means that no other transformation rules are applicable. Therefore, no other information may be added to the ASN.1 outside of what is allowed by the transformation rules given below.

The Module is a representation of the MessageDefinition.

5.8 Method

5.8.1 General

A MessageSet is composed of a limited number of distinct modeling patterns (see ISO 20022-1, Figure 7 for a depiction of the Message Metamodel).

By defining the transformation rules from those patterns to ASN.1, any MessageSet and its MessageDefinitions can be transformed into its corresponding ASN.1 Modules.

5.8.2 Relationship between metamodel concepts and ASN.1 artefacts

5.8.2.1 MessageSet

Each MessageSet is transformed into an artefact of MIME type application/zip containing a file for the MessageDefinition Module and a single file for the XSD Library Module.

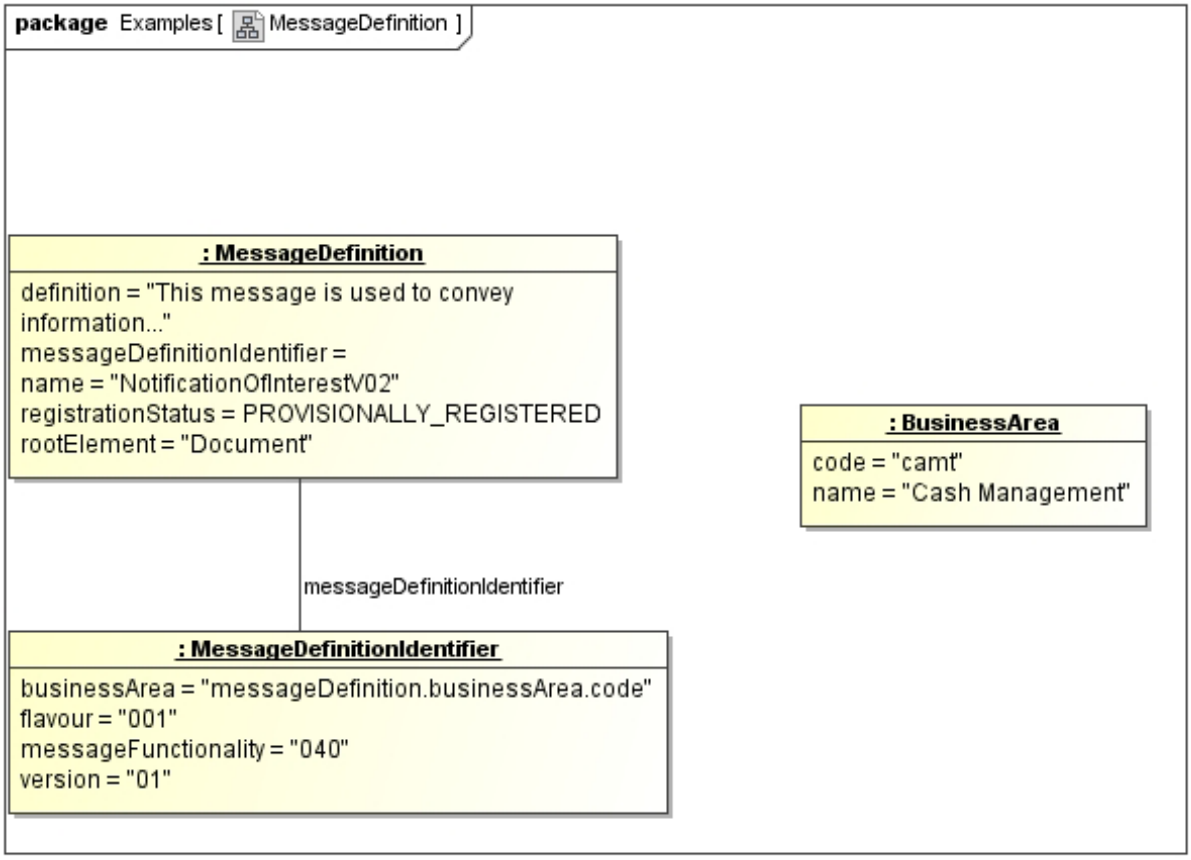
5.8.2.2 MessageDefinition

The file name for the MessageDefinition Module is the MessageDefinitionIdentifier with an extension of ".asn".

EXAMPLE camt.001.040.01.asn.

Each MessageDefinition Module contains a root type with the value of Property rootElement.

EXAMPLE




```
/* Generated by SWIFTStandards Workstation (build:R5.1.0.4) on 2006 Sep
08 11:58:39 */
Camt-001-040-01

DEFINITIONS XER INSTRUCTIONS AUTOMATIC TAGS ::= BEGIN IMPORTS
    String, Decimal, Date, DateTime
    FROM XSD;

Camt-001-040-01-OID OBJECT IDENTIFIER ::= {iso(1) registration-authority(1) unifi(20022)
cash-management(0)}

Document ::= [ELEMENT] Document-1 IDENTIFIED BY
{camt-001-040-01-OID abstract-syntax{1})}

Document-1       ::= [NAME AS "Document"] SEQUENCE {
notificationOfInterestV02 [NAME AS " NotificationOfInterestV02"] Camt-001-040-01 }
```

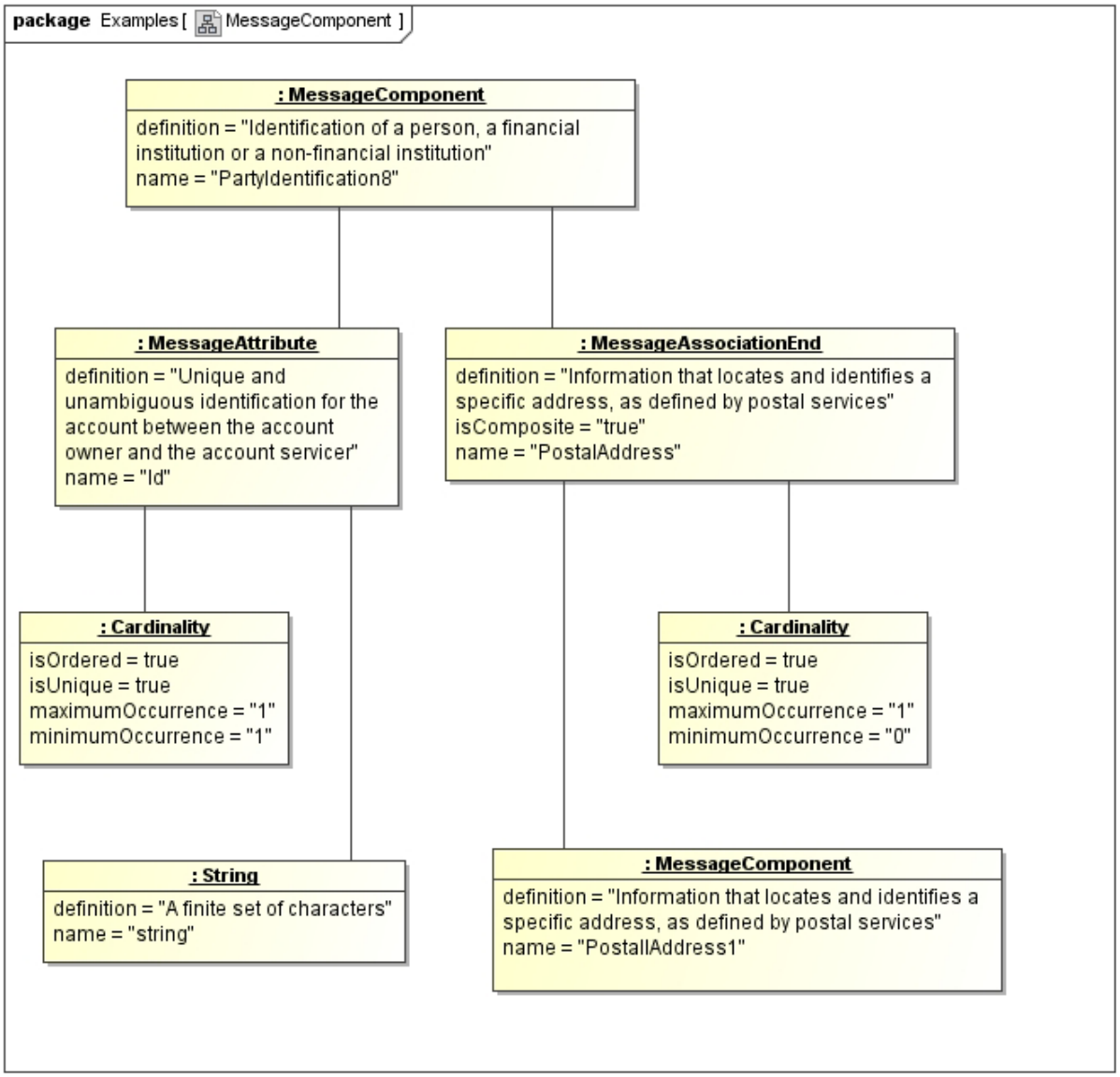
5.8.2.3 MessageComponent

A MessageComponent is transformed into an ASN.1 SEQUENCE whereby the MessageComponent's Name is the value of the SEQUENCE name.

— The SEQUENCE preserves the order of the MessageElements. The transformation rules for MessageElements are given in 5.8.2.6, 5.8.2.8 and 5.8.2.9.

— Each Component within the SEQUENCE is followed by a comma, except for the last one.

EXAMPLE



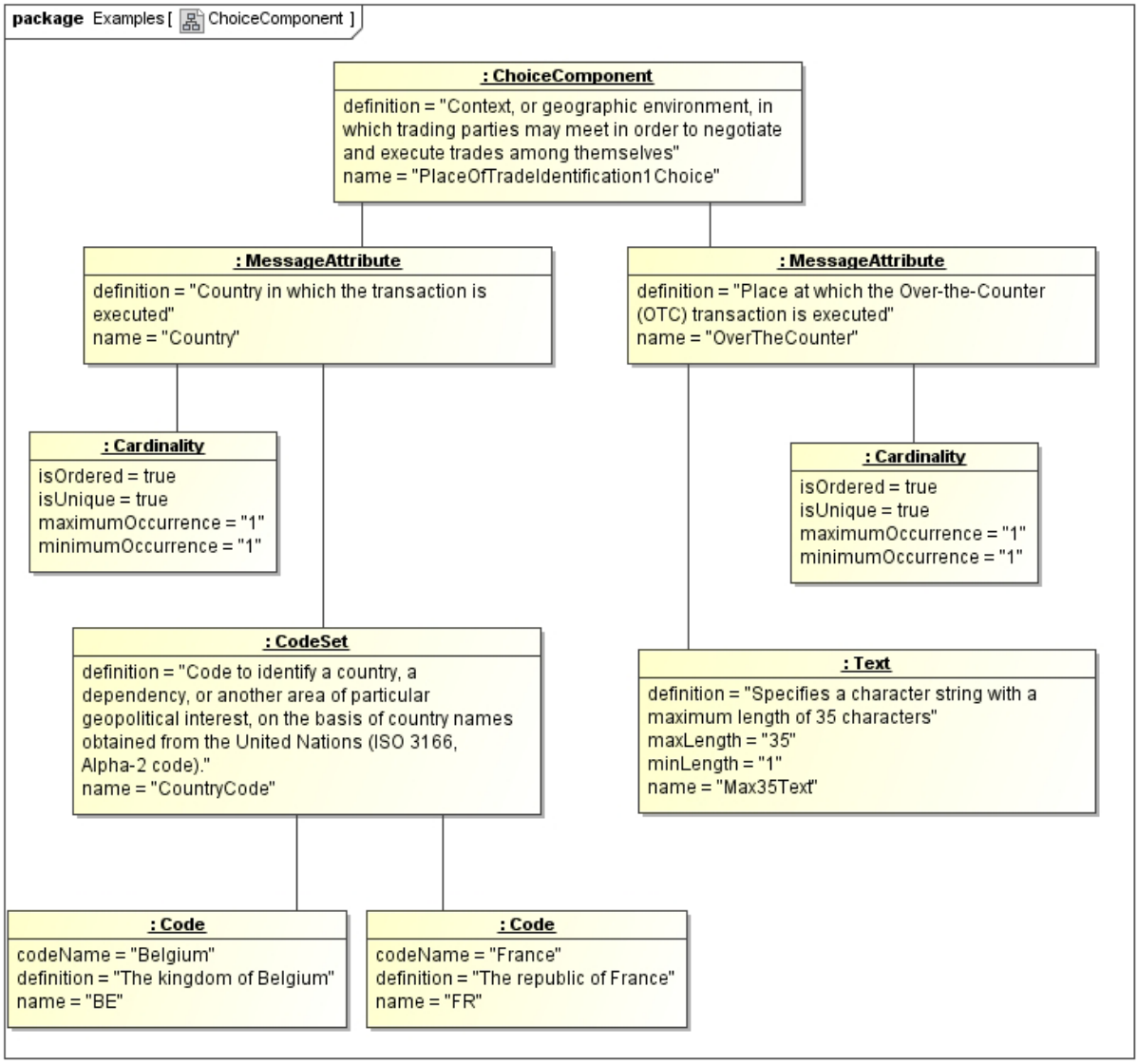

```
AccountIdentification3Choice ::= SEQUENCE {
        Id                  [NAME AS CAPITALIZED] String,
        postalAddress       [NAME AS CAPITALIZED] PostallAddress1 OPTIONAL
    }
}
```

5.8.2.4 ChoiceComponent

A ChoiceComponent is transformed into a SEQUENCE of CHOICE.

Each Alternative within the CHOICE is followed by a comma, except for the last one.

EXAMPLE



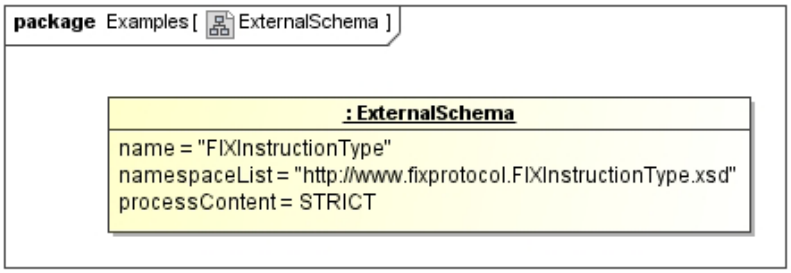

```
PlaceOfTradeIdentification1Choice ::= SEQUENCE {
    choice [UNTAGGED] CHOICE {
        country [NAME AS CAPITALIZED] CountryCode,
        overTheCounter [NAME AS CAPITALIZED] Max35Text
    }
}
```

5.8.2.5 ExternalSchema

ExternalSchema is transformed into a TypeAssignment as follows.

— ExternalSchema Name is transformed into its Typereference.

— Type is transformed into "XSD.AnyType".



EXAMPLE

FIXInstructionType ::= XSD.AnyType.

5.8.2.6 MessageElement

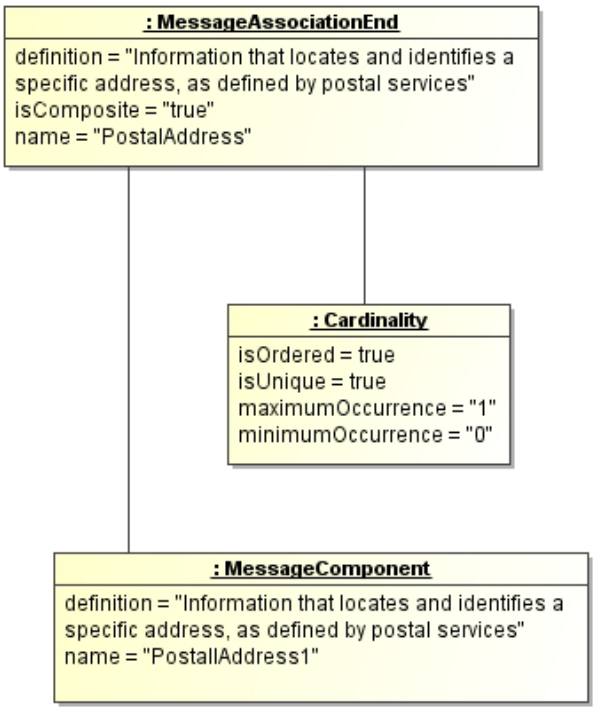
MessageElement in a MessageComponent is transformed into a Component.

MessageElement in a ChoiceComponent is transformed into an Alternative.

MessageElement Name is transformed into the Identifier of the Component or Alternative, whereby the first character is put in lowercase.

- If MaximumOccurrence is greater than "1" then the Component or Alternative Identifier is concatenated with "-list",
  - followed by "[UNTAGGED] SEQUENCE (SIZE (",
  - followed by the value of MinimumOccurrence,
  - followed by ".."
- If MaximumOccurrence contains a number then followed by the value of MaximumOccurrence
- If MaximumOccurrence contains "UNBOUNDED" then followed by "MAX",
  - followed by ")) OF",
  - followed by the Identifier of the Component or Alternative,
  - followed by the XER encoding instruction "[NAME AS ",
  - followed by the result of the abbreviation algorithm for that MessageElement Name, in quotes,
  - followed by "]"
  - followed by the MessageElement Type Name
- If MinimumOccurrence is "0" and MaximumOccurrence is "1" then followed by " OPTIONAL"
  - followed by "(CONSTRAINED BY {/*", followed by "NameAndDefinition ", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10), followed by "*/})"

EXAMPLE




```
postalAddress  [NAME ASpstlAddrss] PostallAddress1 OPTIONAL,
```
(CONSTRAINED BY {/*"NameAndDefinition PostalAdress.
information that locates and identifies a specific address, as defined by postal services"*/})

5.8.2.7 MessageBuildingBlock

See transformation rules for MessageElement.

5.8.2.8 MessageAssociationEnd where isComposite is true

See transformation rules for MessageElement.

5.8.2.9 MessageAssociationEnd where isComposite is false

The transformation rules for MessageElement apply, except that the MessageElement Type Name is replaced by XSD.IDREF.

5.8.2.10 Constraint

A Constraint is translated into an ASN.1 user defined constraint.

Concatenate "Constraint ", Constraint Name and the Constraint Expression.

EXAMPLE Constraint ValidIBAN. The account shall contain a valid IBAN number.

5.8.3 ISO 20022 DataType transformation to ASN.1

5.8.3.1 General

There are two kinds of DataTypes: XSD DataTypes and user-defined DataTypes, each with their own set of transformation rules.

5.8.3.2 XSD DataTypes

An XSD DataType is found in the metamodel in the package TypeLibrary, subpackage XML_Schema.

Each reference to an XSD DataType is transformed into a reference to its equivalent ASN.1 Type according to ISO/IEC 8825-5.

EXAMPLE Metamodel::DataTypes::MonthDay is transformed to XSD.GMonthDay.

5.8.3.3 user-defined DataTypes

5.8.3.3.1 General

A user-defined DataType is an instance of a DataType metaclass (as specified in ISO 20022-1) that is not one of the XSD built-in DataTypes. It is based on an XSD DataType.

5.8.3.3.2 Facets

The XSD facets for each user-defined DataType are expressed as Meta-properties, and the actual values for these facets are provided as properties of the derived DataTypes.

a) Each facet is transformed by the rules defined in ISO/IEC 8825-5:2008, Clause 12.

b) Each Type that has an ASN.1 User-defined Constraint shall be structured as follows:

- "(CONSTRAINED BY {/* "
- for the facets that are mapped in ISO/IEC 8825-5:2008, Clause 12 into ASN.1 User-defined Constraints, the following rule applies:
  - append the DataType Property Name, followed by ": " followed by the DataType Property Value that is prefixed and suffixed by a double quotation mark (this rule is applied repeatedly for each facet, with the results separated by ", "),
  - append Newline;
- append "NameAndDefinition", followed by the concatenation of DataType, CodeSet, IdentifierSet Name or the concept on which the facets are defined, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10) ensuring numbers are preceded and followed by ";
- append " */})"

EXAMPLE (CONSTRAINED BY {/* fractionDigits: "5", totalDigits: "18" */}).

Table 1 — Permissible DataType facets

	pattern	length	minLength	maxLength	minInclusive	maxInclusive	minExclusive	maxEclusive	totalDigits	fractionDigits
Text	x	x	x	x						
Indicator	x									
Amount	x				x	x	x	x	x	x
Quantity	x				x	x	x	x	x	x
Decimal	x				x	x	x	x	x	x
CodeSet	x	x	x	x						
IdentifierSet	x	x	x	x						
Rate	x				x	x	x	x	x	x
Integer	x				x	x	x	x	x	
DateTime	x				x	x	x	x		
Date	x				x	x	x	x		
Time	x				x	x	x	x		
Duration	x				x	x	x	x		
Day	x				x	x	x	x		
Month	x				x	x	x	x		
Year	x				x	x	x	x		
MonthDay	x				x	x	x	x		
YearMonth	x				x	x	x	x		
Binary	x	x	x	x						

5.8.3.3.3 Amount

Amount Name is transformed into the Typereference Name.

Amount is transformed into a new Type that contains an ASN.1 SEQUENCE whereby the Amount's Name is the value of the SEQUENCE name. Amount's Name is the name of the Identifier. The SEQUENCE contains the following Components.

— If not empty, Property CurrencyIdentification is the first Component. It has the CurrencyIdentifierSet as the Identifier and the XER instructions "[NAME AS "Ccy"]" and "[ATTRIBUTE]".

— The second Component's Name is "base" and has XSD.DECIMAL as the Identifier and the XER instruction " [UNTAGGED]".

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

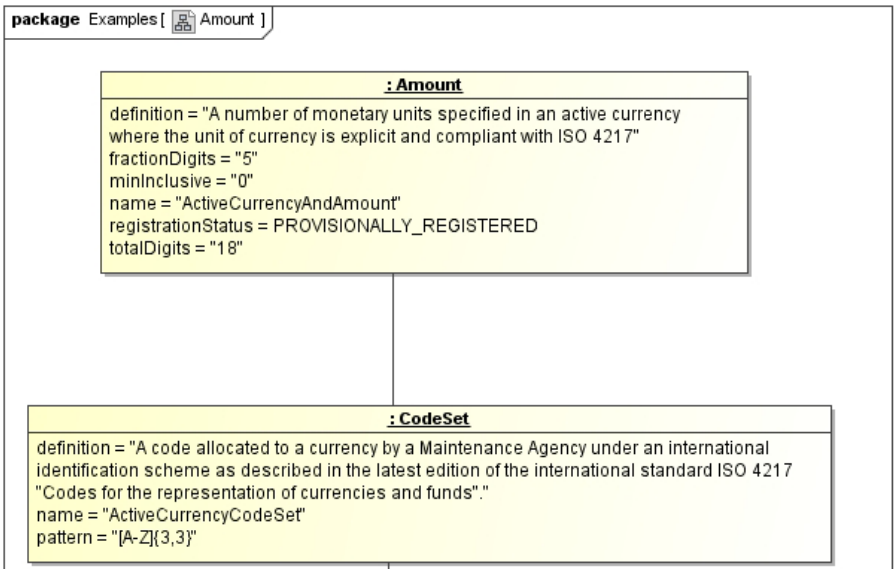
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ActiveCurrencyAndAmount ::= SEQUENCE {

CurrencyIdentification [NAME AS "Ccy"] [ATTRIBUTE] ActiveCurrencyCodeSet

base [UNTAGGED] XSD.DECIMAL

(CONSTRAINED BY {/* fractionDigits="5" totalDigits="18" minInclusive ="0" " NameAndDefinition ActiveCurrencyAndAmount.

A number of monetary units specified in an active currency where the unit of currency is explicit and compliant with ISO 4217" */})

}

5.8.3.3.4 Quantity

Quantity is transformed into a new Type.

Quantity::name is used as the Typereference Name.

The Type is a subtype of XSD.Decimal.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

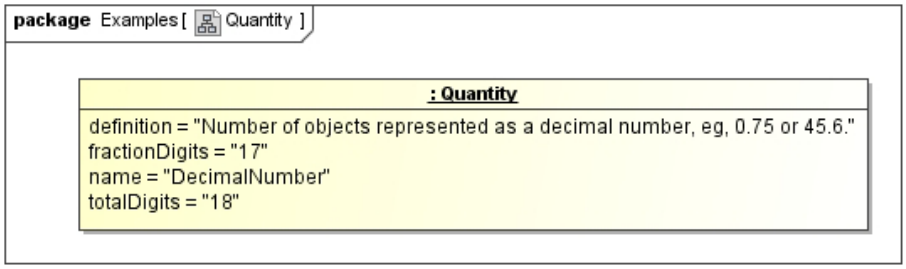
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



DecimalNumber::= XSD.Decimal (CONSTRAINED BY {/* fractionDigits="17" totalDigits="17" "
NameAndDefinition DecimalNumber.

Number of objects represented as decimal number, e.g. 0.75 or 45.6"*/})

5.8.3.3.5 DataType CodeSet

CodeSet is transformed into an ENUMERATED Type.

CodeSet::name is used as the Typereference Name.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

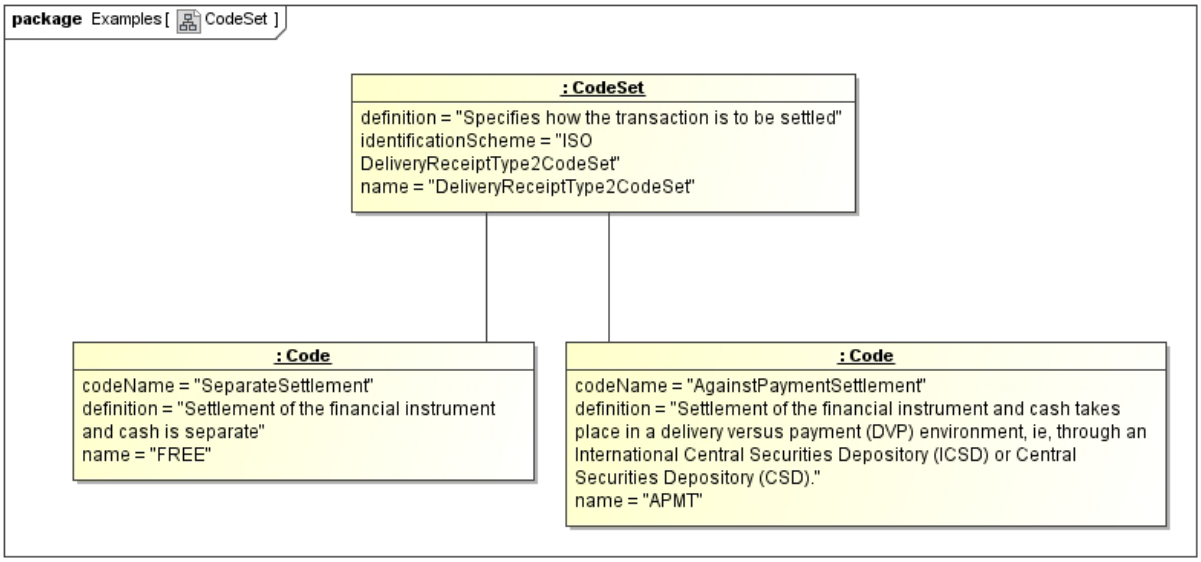
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



DeliveryReceiptType2CodeSet ::= ENUMERATED {FREE, APMT} (CONSTRAINED BY {/*" NameAndDefinition DeliveryReceiptType2CodeSet.

Specifies how the transaction is to be settled"*/})

5.8.3.3.6 DataType IdentifierSet

IdentifierSet is transformed into a new Type.

IdentifierSet::name is used as the Typereference Name.

The Type is a subtype of XSD.String.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

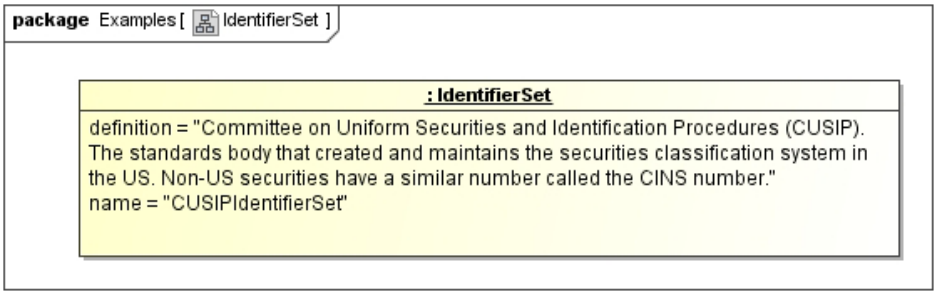
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



CUSIPIdentifierSet::= XSD.String (CONSTRAINED BY {/*" NameAndDefinition CUSIPIdentifierSet.

Committee on Uniform Securities and Identification Procedures (CUSIP). The standards body that created and maintains the securities classification system in the US. Non-US securities have a similar number called the CINS number"*/})

5.8.3.3.7 DataType Rate

Rate is transformed into a new Type.

Rate::name is used as the Typereference Name.

The Type is a subtype of XSD.Decimal.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

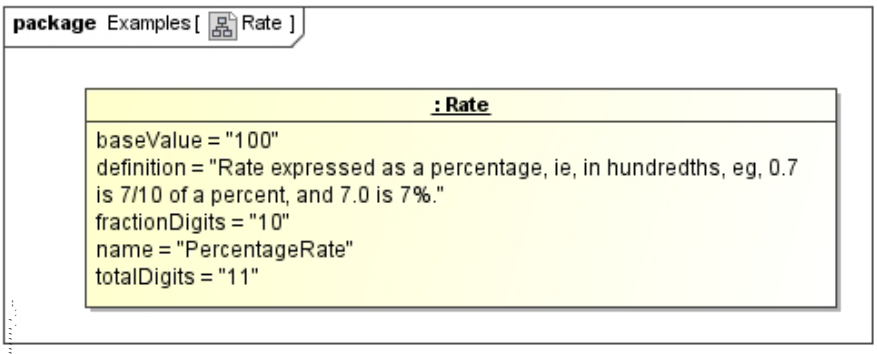
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



PercentageRate::= XSD.Decimal (CONSTRAINED BY {/* fractionDigits="10" totalDigits="11" " NameAndDefinition PercentageRate.

Rate expressed as a percentage, i.e. in hundredths, e.g. 0.7 is 7/10 of a percent, and 7.0 is 7%"*/})

5.8.3.3.8 DataType Indicator

Indicator is transformed into a new Type.

Indicator::name is used as the Typereference Name.

The Type is a subtype of XSD.Boolean.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

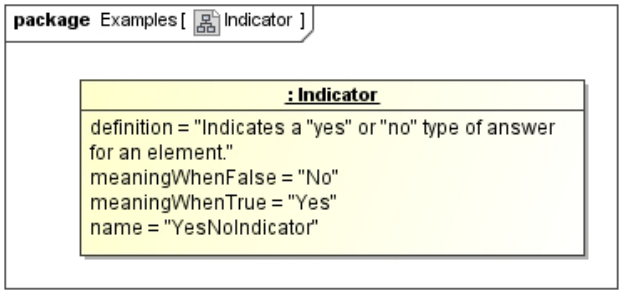
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



YesNoIndicator::= XSD.Boolean (CONSTRAINED BY {/*" NameAndDefinition YesNoIndicator.

Indicates a "yes" or "no" type of answer for an element"*/})

5.8.3.3.9 DataType Text

Text is transformed into a new Type.

Text::name is used as the Typereference Name.

The Type is a subtype of XSD.String.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

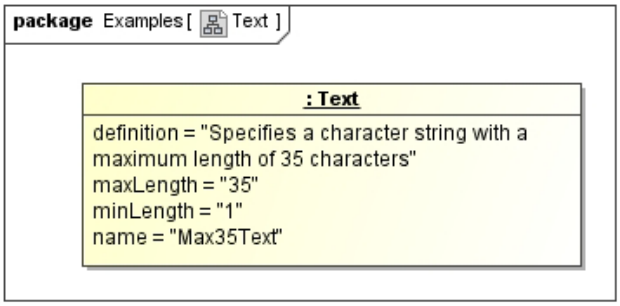
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation ofMessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



Max35Text::= XSD.String (CONSTRAINED BY {/* maxLength="35" minLength="1" " NameAndDefinition Max35Text.

Specifies a character string with a maximum length of 35 characters"*/})

5.8.3.3.10 DataType DateTime

DateTime is transformed into a new Type.

DateTime::name is used as the Typereference Name.

The Type is a subtype of XSD.DateTime.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

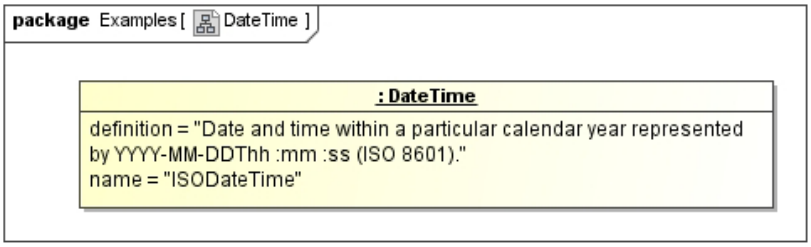
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISODateTime::= XSD.DateTime (CONSTRAINED BY {/*" NameAndDefinition ISODateTime.

Date and time within a particular calendar year represented by YYYY-MM-DDThh:mm:ss (ISO 8601)"*/})

5.8.3.3.11 DataType Time

Time is transformed into a new Type.

Time::name is used as the Typereference Name.

The Type is a subtype of XSD.Time.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

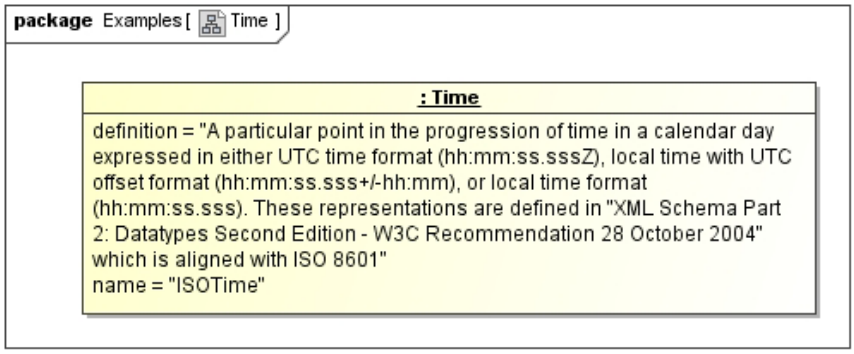
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISOTime::= XSD.Time (CONSTRAINED BY {/*" NameAndDefinition ISOTime.

A particular point in the progression of time in a calendar day expressed in either UTC time format (hh:mm:ss.sssZ), local time with UTC offset format (hh:mm:ss.sss+/-hh:mm), or local time format (hh:mm:ss.sss). These representations are defined in "XML Schema Part 2: Datatypes Second Edition - W3C Recommendation 28 October 2004" which is aligned with ISO 8601"*/})

5.8.3.3.12 DataType Date

Date is transformed into a new Type.

Date::name is used as the Typereference Name.

The Type is a subtype of XSD.Date.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

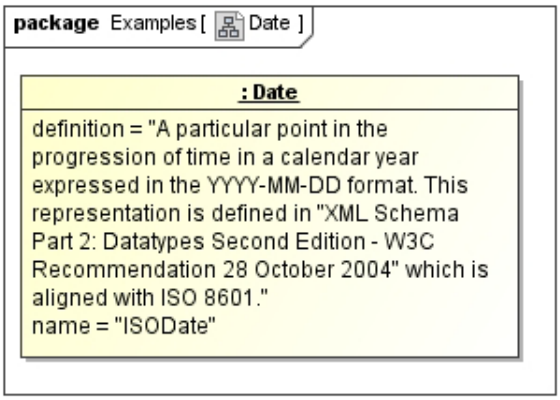
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISODate::= XSD.Date (CONSTRAINED BY {/*" NameAndDefinition ISODate.

A particular point in the progression of time in a calendar year expressed in the YYYY-MM-DD format. This representation is defined in "XML Schema Part 2: Datatypes Second Edition - W3C Recommendation 28 October 2004" which is aligned with ISO 8601."*/})

5.8.3.3.13 DataType Duration

Duration is transformed into a new Type.

Duration::name is used as the Typereference Name.

The Type is a subtype of XSD.Duration.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

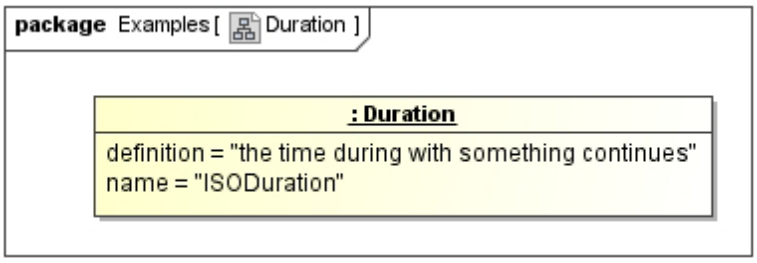
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISODuration::= XSD.Duration (CONSTRAINED BY {/*" NameAndDefinition ISODuration.

the time during which something continues "*/})

5.8.3.3.14 DataType Year

Year is transformed into a new Type.

Year::name is used as the Typereference Name.

The Type is a subtype of XSD.Year.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

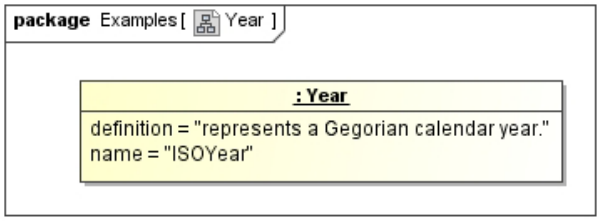
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISOYear::= XSD.Year (CONSTRAINED BY {/*" NameAndDefinition ISOYear.

represents a Gregorian calendar year."*/})

5.8.3.3.15 DataType Month

Month is transformed into a new Type.

Month::name is used as the Typereference Name.

The Type is a subtype of XSD.Month.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

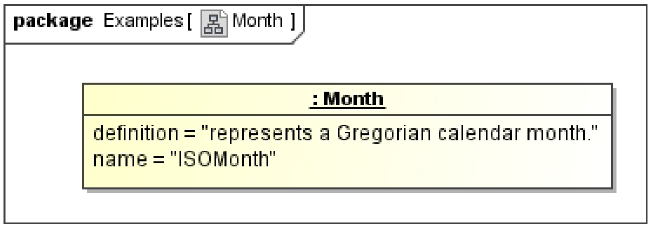
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISOMonth::= XSD.Month (CONSTRAINED BY {/*" NameAndDefinition ISOMonth.

represents a Gregorian calendar month."*/})

5.8.3.3.16 DataType Day

Day is transformed into a new Type.

Day::name is used as the Typereference Name.

The Type is a subtype of XSD.Day.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

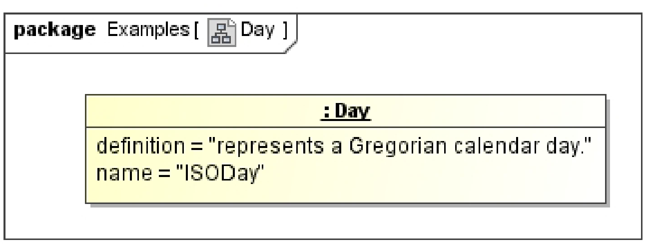
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISODay::= XSD.Day (CONSTRAINED BY {/*" NameAndDefinition ISODay.

represents a Gregorian calendar day."*/})

5.8.3.3.17 DataType MonthDay

MonthDay is transformed into a new Type.

MonthDay::name is used as the Typereference Name.

The Type is a subtype of XSD.MonthDay.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

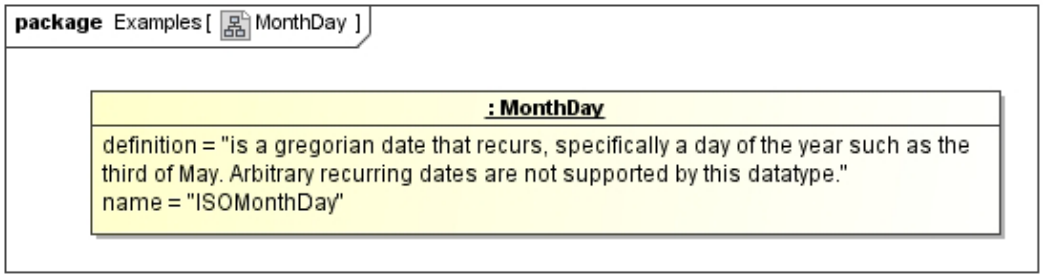
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISOMonthDay::= XSD.MonthDay (CONSTRAINED BY {/*" NameAndDefinition ISOMonthDay.

is a gregorian date that recurs, specifically a day of the year such as the third of May. Arbitrary recurring dates are not supported by this datatype."*/})

5.8.3.3.18 DataType YearMonth

YearMonth is transformed into a new Type.

YearMonth::name is used as the Typereference Name.

The Type is a subtype of XSD.YearMonth.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

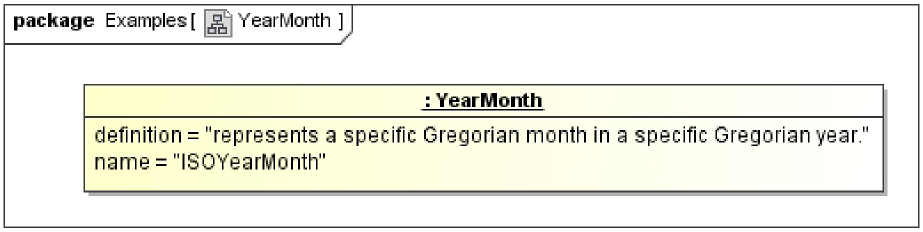
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



ISOYearMonth::= XSD.YearMonth (CONSTRAINED BY {/*" NameAndDefinition ISOYearMonth.

represents a specific Gregorian month in a specific Gregorian year."*/})

5.8.3.3.19 DataType Binary

Binary is transformed into a new Type.

Binary::name is used as the Typereference Name.

The Type is a subtype of XSD.Base64Binary.

The Properties as defined in Table 1 are transformed according to the facet transformations specified in 5.8.3.3.2.

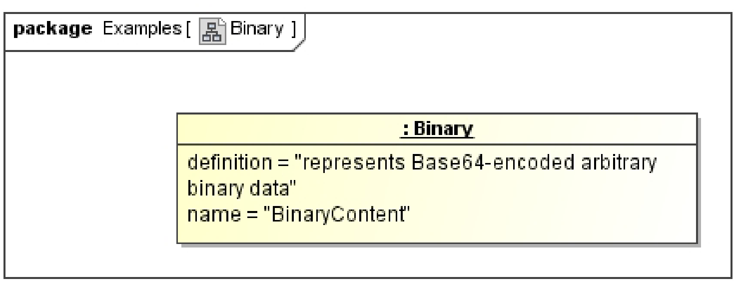
Properties are only transformed if their value is not empty.

If no Properties are transformed, then append "(CONSTRAINED BY {/*".

Append "NameAndDefinition", followed by the concatenation of MessageElement Name, ".", Newline and MessageElement Definition, followed by the transformation for each Constraint (see 5.8.2.10).

If no Properties are transformed, then append "*/})".

EXAMPLE



BinaryContent::= XSD.Base64Binary (CONSTRAINED BY {/*" NameAndDefinition BinaryContent.

represents Base64-encoded arbitrary binary data."*/})

Bibliography

[1] ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

[2] ISO 15022 (all parts), *Securities — Scheme for messages (Data Field Dictionary)*

[3] ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

[4] ISO/IEC 8824-2, *Information technology — Abstract Syntax Notation One (ASN.1): Information object specification*

[5] ISO/IEC 8824-3, *Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification*

[6] ISO/IEC 8824-4, *Information technology — Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

[7] ISO/IEC 8825-1, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

[8] ISO/IEC 8825-2, *Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*

[9] ISO/IEC 8825-3, *Information technology —- ASN.1 encoding rules: Specification of Encoding Control Notation (ECN)*

[10] ISO/IEC 8825-4, *Information technology — ASN.1 encoding rules: XML Encoding Rules (XER)*

[11] ISO/IEC 8825-6, *Information technology —- ASN.1 encoding rules: Registration and application of PER encoding instructions*

[12] W3C Recommendation: XML 1.0 specification (4th edition, 16 August 2006)

[13] W3C Recommendation: XML Schema Part 1: Structures Second Edition (28 October 2004)

ICS 03.060

Price based on 25 pages