INTERNATIONAL STANDARD

ISO 20022-4

First edition
2013-05-01

Financial services — Universal financial industry message scheme —

Part 4: XML Schema generation

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

Partie 4: Génération de schéma XML

Reference number
ISO 20022-4:2013(E)

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 20022-4 was prepared by Technical Committee ISO/TC 68, *Financial services*.

This first edition cancels and replaces ISO/TS 20022-4:2004.

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.

— This part of ISO 20022 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.

— ISO 20022-8 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 4:
XML Schema generation

1 Scope

This part of ISO 20022 was prepared to complement the ISO 20022 Metamodel, as specified in ISO 20022-1, with the XML syntax transformation rules to be applied by the ISO 20022 Registration Authority in order to translate an ISO 20022 compliant MessageDefinition into an XML Schema for the description and validation of XML Messages.

It specifies the transformation rules from level 3 to level 4. It is a deterministic transformation, meaning that the resulting XML Schema 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.

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

RFC 5141, Available at http://www.rfc-archive.org/getrfc.php?rfc=5141

3 Terms and definitions

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

4 Background

XML is a standard defined by W3C (the World Wide Web Consortium) that is used for the representation (i.e. the syntax) of standardized ISO 20022 MessageDefinitions. XML leaves a lot of freedom for the exact way it is used in a particular application. Therefore, merely stating that XML is used is not sufficient to guarantee predictability; it is also necessary to explain how it will be used.

This part of ISO 20022 contains a set of XML design rules. These design rules define how a MessageDefinition is transformed into an ISO 20022 XML Schema.

A valid XML document (referred to hereafter as ‘XML Instance’ or ‘Instance’) is any XML document that has an associated description and that complies with the constraints expressed in that description. The associated description in this case is derived from the MessageDefinition.

This part of ISO 20022 also describes how a MessageSet can be converted into XML Schemas by specifying how a MessageDefinition is transformed into an XML Schema. This XML Schema will then make it possible to use a validating XML Schema parser to automatically verify that a given XML Instance complies with the constraints (or a subset of constraints) described in the MessageDefinition.

This part of ISO 20022 is limited to explaining how a given MessageDefinition will be mapped into XML; 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 Preconditions

The MessageSet used as input for the transformation is a valid instance of the MessageSet meta-class.

5.2 Transformation constraints

An ISO 20022 XML Schema contains an XML comment at the top, which contains the following metadata:

— the ISO 20022 RA issued release number;

EXAMPLE R6.1.0.2

— the generation date;

— documentation text (optional).

Apart from the RA administered XML comment line, an ISO 20022 XML Schema contains only information relevant for the validation of XML Instances by an XML Schema parser. For example, it does not contain documentation (definitions, etc.) or implementation information (relationships between similar Components, etc).

An ISO 20022 XML Schema is an implementation of the MessageDefinition. The MessageDefinition is always the definitive source.

XML Schema Definition (XSD) Elements are local within their complexType, except for rootElement ComplexTypes, which are global. This approach is commonly called the ‘Venetian blind’ approach.

All aspects of dictionary management, e.g. reuse and pointers, are managed at the level of the MessageDefinition. These aspects, among others, are covered in ISO 20022-3.

All complexType and simpleType elements in the XML Schema appear after the root element, in alphabetical order, using the type "name" attribute as sort key.

5.3 Namespaces

There are several namespace declarations used in the XML Schema.

1) The targetNamespace, which is the namespace to which all XSD Elements and Types belong. The URN, which shall be in accordance with RFC 5141, consists of:

   — a fixed part consisting of the URN of the namespace for ISO documents, urn:iso:std:iso:20022:tech;

   — a part unique to each schema, xsd: *MessageDefinitionIdentifier*; the structure and meaning of *MessageDefinitionIdentifier* are explained in ISO 20022-3.

2) The XML Schema namespace.

3) The default namespace, which is the same as the TargetNamespace.

EXAMPLE Namespace declarations in an ISO 20022 XML Schema:
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="urn:iso:std:iso:20022:tech:xsd:camt.007.002.01" elementFormDefault="qualified"
targetNamespace="urn:iso:std:iso:20022:tech:xsd:camt.007.002.01">

EXAMPLE Namespace declarations in an ISO 20022 XML Instance:
<Doc:Document xmlns:Doc="urn:iso:std:iso:20022:tech:xsd:camt.007.002.01">
or, in case the Sender decides to use the default namespace,
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:camt.007.002.01">

5.4 Granularity of schemas

There is one well-formed and valid XML Schema per MessageDefinition.

5.5 XML MessageInstances

5.5.1 Encoding

The encoding shall be UTF-8, and shall be identified as UTF-8 in the XML prolog of the document.

EXAMPLE <?xml version="1.0" encoding="UTF-8"?>

5.5.2 Default namespace

The document may declare any namespace as default.

5.5.3 DOCTYPE

The document shall not include a DOCTYPE declaration.

5.5.4 SchemaLocation

The document may declare any schemaLocation. However, the schema shall be resolved from the namespace alone and not from the schemaLocation.

5.5.5 XML name abbreviation algorithm

Names of XML Elements and XML Attributes appearing in XML Instances are derived from their MessageDefinition names according to the algorithm posted on the ISO 20022 website, whereby the algorithm is part of this International Standard.

5.6 Completeness

The list of transformation rules described in this subclause is complete. Therefore, no other transformation rules are applicable and no other information may be added to the XML Schema outside of what is allowed by the transformation rules given below.

The XML Schema is a representation of the MessageDefinition.

5.7 Method

5.7.1 General

A MessageDefinition is composed of a limited number of distinct modelling patterns. For a depiction of the Message Metamodel, see ISO 20022-1, Figure 7.

By defining the transformation rules from those patterns to ISO 20022 XML Schema, any MessageDefinition can be transformed into its corresponding ISO 20022 XML Schema.

5.7.2 Relationship between metamodel concepts and XML Schema artefacts

5.7.2.1 MessageSet

Each MessageSet is transformed into an artefact of MIME type application/zip, containing all MessageDefinition XML Schemas belonging to that MessageSet. It may also contain images of the sequence diagrams relevant to this MessageSet.

5.7.2.2 MessageDefinition

A MessageDefinition is transformed into an XML Schema Document using the MessageDefinitionIdentifier as the filename suffixed with ".xsd".

EXAMPLE camt.001.001.01.xsd

The XML Schema Document consists of the following elements.

a) The XML prolog containing XML attribute "version" with value "1.0" and XML attribute "encoding" with value "UTF-8" (see also 5.5.1).

b) The RA administered XML comment line defined in 5.2.

c) The XML Element "xs:schema" containing

   — XML Attributes declaring the namespaces defined in 5.3,

   — an XML Attribute with name "elementFormDefault" and value "qualified".

d) An XML Element "xs:element" containing

   — XML attribute with name "name" and value the MessageDefinition rootElement's Name,

   — XML attribute with name "type" and value the MessageDefinition rootElement's Name.

e) An XML Element "xs:complexType" whereby the MessageDefinition rootElement's Name is the value of the XML Attribute "name". The XML Element "xs:complexType" contains

   — an XML Element "xs:sequence". Within that XML Element "xs:sequence" the value of MessageDefinition is used to create (following the abbreviation algorithm in 5.5.5) the value of the XML Attribute "name" of the XML Element "xs:element" and is also copied into the XML Attribute "type".

NOTE The version number (indicated as "Vxx" where "xx" is numerical) is not part of that name. This means that in the mapping table used by the transformation algorithm, all versions (shown as "Vxx") are mapped to an empty value.

f) An XML Element "xs:complexType" whereby the MessageDefinition's value is the value of the XML Attribute "name". The XML Element "xs:complexType" contains

— an XML Element "xs:sequence". Within that XML Element "xs:sequence", each of the MessageBuildingBlocks is transformed into an XML Element "xs:element", preserving the order of the MessageBuildingBlocks. The transformation rules for MessageBuildingBlocks are given in 5.7.2.6.

EXAMPLE


```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:iso:std:iso:20022:tech:xsd:camt.040.001.02"
        targetNamespace="urn:iso:std:iso:20022:tech:xsd:camt.040.001.02"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">
       <xs:element name="Document" type="Document"/>
       <xs:complexType name="Document">
         <xs:sequence>
          <xs:element name=" NtfctnOfIntrst"
"type="NotificationOfInterestV02"/>
         </xs:sequence>
       </xs:complexType>
       <xs:complexType name="NotificationOfInterestV02"/>
</xs:schema>
```

5.7.2.3 MessageComponent

A MessageComponent is transformed into an XML Element "xs:complexType" whereby the MessageComponent's Name is the value of the XML Attribute "name". The XML Element "xs:complexType" contains

— an XML Element "xs:sequence". Within that XML Element "xs:sequence", each of the MessageElements is transformed into an XML Element "xs:element", preserving the order of the MessageElements. The transformation rules for MessageElements are given in 5.7.2.6, 5.7.2.8 and 5.7.2.9.

EXAMPLE

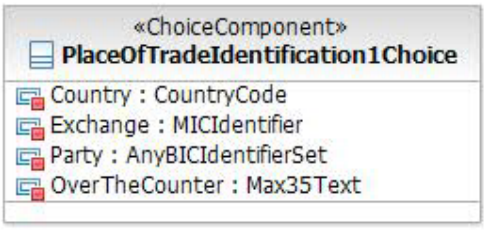

```
<xs:complexType name="PartyIdentification8">
  <xs:sequence>
    <xs:element name="Nm" type="Max70text" minOccurs="0" maxOccurs="1"/>
    <xs:element name="CtryOfRes" type="CountryCode" minOccurs="0"
maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

5.7.2.4 ChoiceComponent

A ChoiceComponent is transformed into an XML Element "xs:complexType" whereby the ChoiceComponent's Name is the value of the XML Attribute "name". The XML Element "xs:complexType" contains

— an XML Element "xs:choice". Within that XML element "xs:choice", each of the MessageElements is transformed into an XML Element "xs:element", preserving the order of the ChoiceComponent's MessageElements. The transformation rules for Message Elements are given in 5.7.2.6, 5.7.2.8 and 5.7.2.9.

EXAMPLE




```
<xs:complexType name="PlaceOfTradeIdentification1Choice">
   <xs:choice>
    <xs:element name="Ctry" type="CountryCode"></xs:element>
    <xs:element name="Xchg" type="MICIdentifier"></xs:element>
    <xs:element name="Pty" type="AnyBICIdentifier"></xs:element>
    <xs:element name="OverTheCntr" type="Max35Text"></xs:element>
   </xs:choice>
</xs:complexType>
```

5.7.2.5 ExternalSchema

An ExternalSchema is transformed into an XML Element "xs:complexType" with XML Attribute "name".

The ExternalSchema's Name is the value of the XML Attribute "name".

The XML Element "xs:complexType" contains the XML Element "xs:sequence" that contains the XML Element "xs:any" which has the following XML Attributes:

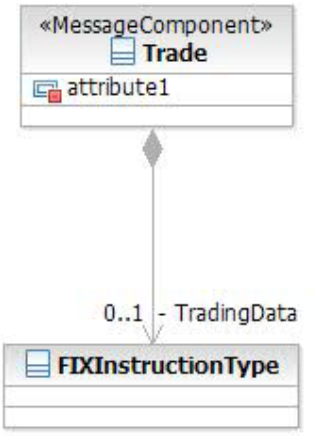
— the Property ProcessContents is copied into the XML Attribute "processContents";

— the Property Namespace is copied into the XML Attribute "namespace".

NOTE 1 The possible values for the XML Attribute "namespace" are ((##any | ##other) | List of (anyURI). In case of List of (anyURI), all values are white space separated.

NOTE 2 XML Attributes "xs:minOccurs" and "xs:maxOccurs" are not present in the XML Schema declaration. This means the complexType only contains the xs:any and this xs:any has a cardinality of exactly one.

EXAMPLE



XML SCHEMA snippet

```
<xs:complexType name="Trade">
   <xs:sequence>
        <xs:element name="TrdngDta" type="FIXInstructionType" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="FIXInstructionType">
   <xs:sequence>
        <xs:any namespace="www.fixprotocol.org" processContents="strict"/>
   </xs:sequence>
</xs:complexType>
```

5.7.2.6 MessageElement

MessageElement is transformed into XML Element "xs:element".

Order of the MessageElements as defined in the MessageComponentTypes shall be preserved in the corresponding XML Schema.

MessageElement Name is used to create (following the abbreviation algorithm in 5.5.5) the value of the XML Attribute "name" of the XML Element "xs:element".

MessageElement's cardinality:

— MinimumOccurrence is copied into the XML Attribute "minOccurs" of the XML Element "xs:element";

— MaximumOccurrence is copied into the XML Attribute "maxOccurs" of the XML Element "xs:element".

The name of the MessageElement Type is copied into the XML Attribute "type" of the XML Element "xs:element". If the MesssageElement Type is an XSD Datatype (see 5.7.3.2) then that name is preceded by "xs:".

5.7.2.7 MessageBuildingBlock

See transformation rules for MessageElement.

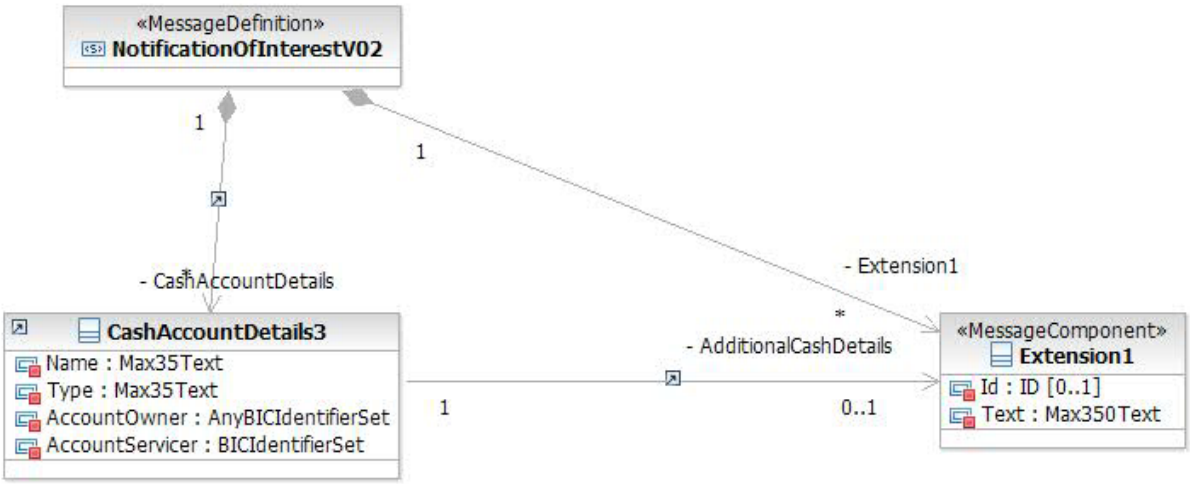
5.7.2.8 MessageAssociationEnd where isComposite is true

See transformation rules for MessageElement.

5.7.2.9 MessageAssociationEnd where isComposite is false

See transformation rules for MessageElement, except that the XML Element "xs:element" has as type "xs:IDREF" instead of the name of the MessageElement Type.

EXAMPLE



XML SCHEMA snippet

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:iso:std:iso:20022:tech:xsd:camt.040.001.02"
targetNamespace="urn:iso:std:iso:20022:tech:xsd:camt.040.001.02"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="Document" type="Document"/>
 <xs:complexType name="Document">
    <xs:sequence>
      <xs:element name="NtfctOfIntrst" type="NotificationOfInterestV02"/>
    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="NotificationOfInterestV02">
    <xs:sequence>
           <xs:element name="CshAcctDtls" type="CashAccountDetails3"/>
           <xs:element name="Xtnsn1" type="Extension1"/>
    </xs:sequence>
 </xs:complexType>
 <xs:complexType name="Extension1">
    <xs:sequence>
           <xs:element name="Id" type="xs:ID" minOccurs="0" maxOccurs="1"/>
           <xs:element name="Txt" type="Max350Text"/>
    </xs:sequence>
           </xs:complexType>
 <xs:complexType name="CashAccountDetails3">
    <xs:sequence>
           <xs:element name="Nm" type="Max35Text"/>
           <xs:element name="Tp" type="Max35Text"/>
           <xs:element name="AcctOwnr" type="AnyBICIdentifierSet"/>
           <xs:element name="AccntSrvcr" type="AnyBICIdentifierSet"/>
           <xs:element name="AddtnlCshDtls" type="xs:IDREF"/>
    </xs:sequence>
 </xs:complexType>
</xs:schema>
```

XML INSTANCE

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:camt.040.001.02">
 <NtfctOfIntrst>
```

```
    <CshAcctDtls>
        <Nm>A</Nm>
        <Tp>b</Tp>
        <AccntOwnr>ABNANL2A</AccntOwnr>
        <AccntSrvcr>CHASUS33</AccntSrvcr>
        <AddtnlCshDtls>ref1</AddtnlCshDtls>
    </CshAcctDtls>
    <Xtnsn1>
        <Id>ref1</Id>
        <Txt>additionalInfo</Txt>
    </Xtnsn1>
  </NtfctOfIntrst>
</Document>
```

5.7.3 ISO 20022 DataType transformation to XSD Schema

5.7.3.1 General

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

NOTE See Figure 15 and Figure 7 in ISO 20022-1.

5.7.3.2 XSD Datatypes

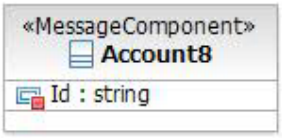
An XSD Datatype is found in the metamodel in the package TypeLibrary, subpackage XML_Schema.

Each XSD Datatype Name is transformed to the value of the XML Attribute "type" as shown in Table 1.

Table 1 — XSD Datatype transformation

XSD Datatype Name	XML Schema
string	xs:string
boolean	xs:boolean
decimal	xs:decimal
integer	xs:integer
dateTime	xs:dateTime
base64Binary	xs:base64Binary
duration	xs:duration
gDay	xs:gDay
gMonthDay	xs:gMonthDay
gMonth	xs:gMonth
gYear	xs:gYear
gYearMonth	xs:gYearMonth
date	xs:date
time	xs:time
ID	xs:ID

EXAMPLE




```
<xs:complexType name="Account8">
  <xs:sequence>
    <xs:element name="Id" type="xs:string"></xs:element>
  </xs:sequence>
</xs:complexType>
```

5.7.3.3 User-defined DataTypes

5.7.3.3.1 General

A derived DataType is an instance of a DataType metaclass, as specified in ISO20022-1, that is not one of the XSD built-in datatypes.

A derived DataType is based on an XSD DataType. The XSD facets for each derived DataType are expressed as Meta-properties, and the actual values for these facets are provided as properties of the derived DataTypes.

Table 2 — DataType Properties transformed to XSD facets

	pattern	length	min. length	max. length	min. inclusive	max. inclusive	min. exclusive	max. exclusive	total Digits	fraction Digits
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						

5.7.3.3.2 DataType Amount

5.7.3.3.2.1 CurrencyIdentifierSet is not empty

XML Element "xs:simpleType" is created with XML Attribute "name" with value DataType Amount Name suffixed by "_SimpleType". It contains XML Element "xs:restriction", which contains:

- XML Attribute "base" that has value "xs:decimal";
- the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet. Properties are only transformed if their value is not empty.

DataType Amount is transformed into XML Element "xs:complexType". It contains:

- XML Attribute "name" with value DataType Amount Name;
- XML Element "xs:simpleContent", which contains the XML Element "xs:extension", which contains
  - XML Attribute "base" with value DataType Amount's Name suffixed by "_SimpleType",
  - XML Element "xs:attribute", which has
    - XML Attribute "name", which has as value the Name of Property CurrencyIdentifierSet on which the abbreviation algorithm in 5.5.5 has been applied,
    - XML Attribute "type" which has as value the Name of Property CurrencyIdentifierSet's Type,
    - XML Attribute "use" with value "required".

The Amount's Property Type is transformed as per the rules defined for that DataType.

EXAMPLE


```
<xs:simpleType name="ActiveCurrencyAndAmount_SimpleType">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0"/>
        <xs:totalDigits value="18"/>
        <xs:fractionDigits value="5"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="CurrencyCode">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-Z]{3,3}"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="ActiveCurrencyAndAmount">
    <xs:simpleContent>
        <xs:extension base="ActiveCurrencyAndAmount_SimpleType">
            <xs:attribute name="Ccy" use="required" type="CurrencyCode"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

5.7.3.3.2.2 CurrencyIdentifierSet is empty

DataType Amount is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Amount Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base" with the value "xs:decimal",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE

```
<xs:simpleType name="AmountWithNoCurrency">
  <xs:restriction base='xs:decimal'>
        <xs:minInclusive value="0"/>
        <xs:totalDigits value="18"/>
        <xs:fractionDigits value="5"/>
 </xs:restriction>
 </xs:simpleType>
```

5.7.3.3.3 DataType Quantity

DataType Quantity is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Quantity Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base" which has value "xs:decimal",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="Number">
  <xs:restriction base='xs:decimal'>
    <xs:minInclusive value='1'/>
    <xs:maxInclusive value='100000'/>
 </xs:restriction>
 </xs:simpleType>
```

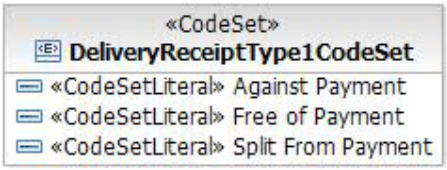
5.7.3.3.4 DataType CodeSet

DataType CodeSet is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType CodeSet Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base", which has value "xs:string",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty),
  - no transformation for Property IdentificationScheme,
  - for each CodeSetLiteral, an XSD Enumeration using namespace prefix "xs:".

The body of the EnumerationLiteral 'specification' Property is copied into the value of the XSD Enumeration.

EXAMPLE



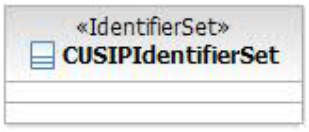

```
<xs:simpleType name="DeliveryReceiptType1CodeSet">
    <xs:restriction base="xs:string">
        <xs:enumeration value="APMT"/>
        <xs:enumeration value="FREE"/>
        <xs:enumeration value="DSPA"/>
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.5 DataType IdentifierSet

DataType IdentifierSet is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value IdentifierSet Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base", which has value "xs:string",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty),
  - no transformation for Property IdentificationScheme.

EXAMPLE




```
<xs:simpleType name="CUSIPIdentifierSet">
    <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z]{2,2}[0-9]{2,2}[a-zA-Z0-9]{1,30}"/>
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.6 DataType Rate

DataType Rate is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Rate Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base", which has value "xs:decimal",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty),
  - no transformation for Properties baseValue and baseUnitCode.

EXAMPLE

```
<xs:simpleType name="PercentageRate">
    <xs:restriction base="xs:decimal">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.7 DataType Indicator

DataType Indicator is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Indicator Name;
- XML Element "xs:restriction", which contains
  - XML Attribute "base", which has value "xs:boolean",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty),
  - no transformation for Properties meaningWhenFalse and meaningWhenTrue.

EXAMPLE

```
<xs:simpleType name="YesNoIndicator">
    <xs:restriction base="xs:boolean">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.8 DataType Text

DataType Text is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Text Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:string",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="Max35Text">
    <xs:restriction base="xs:string">
        <xs:maxLength value="35" />
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.9 DataType DateTime

DataType DateTime is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType DateTime Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:dateTime",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISODateTime">
    <xs:restriction base="xs:dateTime">
    </xs:restriction>
</xs:simpleType>
```

Examples of valid values:
2000-12-20T20:00:00.000Z
2000-12-20T20:00:00Z

5.7.3.3.10 DataType Time

DataType Time is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Time Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:time",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISOTime">
    <xs:restriction base="xs:time">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.11 DataType Date

DataType Date is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Date Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:date",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute

"value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISODate">
    <xs:restriction base="xs:date">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.12 DataType Duration

DataType Duration is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Duration Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:duration",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISODuration">
    <xs:restriction base="xs:duration">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.13 DataType Year

DataType Year is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Year Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:gYear",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISOYear">
    <xs:restriction base="xs:gYear">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.14 DataType Month

DataType Month is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Month Name;
- XML Element "xs:restriction" which contains
  - with XML Attribute "base", which has value "xs:gMonth",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISOMonth">
    <xs:restriction base="xs:gMonth">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.15 DataType Day

DataType Day is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Day Name;
- XML Element "xs:restriction" which contains
  - with XML Attribute "base", which has value "xs:gDay",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISODay">
    <xs:restriction base="xs:gDay">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.16 DataType MonthDay

DataType MonthDay is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType MonthDay Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:gMonthDay",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISOMonthDay">
    <xs:restriction base="xs:gMonthDay">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.17 DataType YearMonth

DataType YearMonth is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType YearMonth Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:gYearMonth",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="ISOYearMonth">
    <xs:restriction base="xs:gYearMonth">
    </xs:restriction>
</xs:simpleType>
```

5.7.3.3.18 DataType Binary

DataType Binary is transformed into XML Element "xs:simpleType". It contains:

- XML Attribute "name" with value DataType Binary Name;
- XML Element "xs:restriction" which contains
  - XML Attribute "base", which has value "xs:base64Binary",
  - the Properties as defined in Table 2, which are transformed into XSD Constraining Facets with the same name and preceded by "xs:", and the value of the Property is copied into the attribute "value" of the XSD Constraining Facet (properties are only transformed if their value is not empty).

EXAMPLE


```
<xs:simpleType name="BinaryContent">
    <xs:restriction base="xs:base64Binary">
    </xs:restriction>
</xs:simpleType>
```

Bibliography

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

[2] UML2 (Unified Modeling Language) is available on the Object Management Group Website

[3] URN namespace for ISO documents IETF RFC 5141

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

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

[6] W3C Recommendation: XML Schema Part 2: Datatypes Second Edition (28 October 2004)

ICS 03.060

Price based on 22 pages