GS1 Technical XML User Guide to Release 2.0

version 2.0
May 2005

Document Summary

Document Item	Current Value
Document Title	GS1 Technical XML User Guide to Release 2.0
Date Last Modified	2005-05-10
Current Document Version	2.0
Status	For Publishing
Document Description (one sentence summary)	This document provides users with the technical guidelines to the structure and design of the EAN.UCC XML standard release 2.0

Acknowledgements

Name	Organization
Dipan Anarkat	GS1 HO
Marilyn Dodd	3M Company
John Duker	Procter & Gamble Company
Anders Grangard	GS1 France
Anthony Hoang	WWRE
Ewa Iwicka	GS1 HO
Christian Przybilla	GS1 Germany
Sylvia Webb	GEFEG US

Log of Changes in version 2.0

Section #	Summary of Change
1	Added FAQ 12 referring to the use of time-related data types
3	Updated website addresses hosting EAN.UCC XML standards
8.1	Added section explaining the way of handling date and time in EAN.UCC XML standards
8.1.1	Added section explaining the use of milliseconds
8.1.2	Added section explaining the use of time zones
10	Added description and usage of 'message' component introduced in the version 2.0.2 of the EAN.UCC XML standards
10.1.2	Extended explanation of the Transaction layer functionality, modified examples (the misleading Order example has been replaced by Trade Item)
10.2.3.3	Added section explaining the Link Command functionality

Disclaimer

Table of Contents

0 SCOPE OF THE DOCUMENT

This "GS1 Technical XML User Guide to Release 2.0" covers only technical aspects of the EAN.UCC Business Message Standard Release 2.0. It does not explain the technical details of any earlier releases. The major differences between the current one (2.0) and the previous release (1.3.1) are listed in an Appendix.

Should you need further information about the release 1.3.1, please refer to the document "How to use EAN.UCC XML Standards version 1.3.1"

"GS1 Technical XML User Guide to Release 2.0" does not contain any guidelines for use of the particular business messages in specific business processes. This aspect is covered in the Business Message Standards, separately for each business message.

1 FREQUENTLY ASKED QUESTIONS

1. What are the prerequisites for using XML?
   Answer: In its simplest configuration, you need a software tool to edit and validate the XML documents. The software tools can be either embedded into the application used to view the XML document or standalone ones. Although XML documents can be edited with any word processor, using a dedicated tool significantly simplifies it and reduces the number of errors. The validating tool verifies whether the XML document – the actual business message – conforms to the XML schema – the standardised structure and content definition of the message. For more details see: Prerequisites for using XML.

2. If I want to use the EAN.UCC XML standards, what documents should I download and how are they related?
   Answer: The description of the message and its full UML model are presented in the Business Message Standard (BMS) document. The actual XML schemas and the example instance files and their HTML representation are contained in the Implementer's Packet. Both the BMS and the content of the Implementer's Packet have to be downloaded and analysed prior to any implementation attempt. More details concerning the set of standard documents can be found in: GS1 set of XML standards.

3. What questions should be asked to software providers when purchasing a product to support EAN.UCC XML messages?
   Answer: All companies that are interested in implementing the EAN.UCC XML standards and are planning to buy software tools to support it, should make sure that the product they have chosen is suitable for such implementation. The Checklist for Solution Providers can facilitate this verification.

4. What is the relationship between the schema and the BMS? Why are the restrictions on data in BMS different from those in the schema?
   Answer: The Business Message Standard document is based on the actual user requirements. It is a base for the XML standard development.

However, while building XML schemas, the developers have to conform to the data types built in XML specifications and included in every XML-aware application. In addition, if possible, they try to enable reusability of definitions, in order to facilitate processing and lower the costs of application development. Therefore, schemas tend to be less restrictive than the BMS. In case of any discrepancies, i.e. whenever the schema definition is more permissive than the BMS, the latter one should be used as a base for populating the business messages with business data. For more detailed information see: GS1 set of XML standards.

5. What is the GS1 strategy for publishing the XML standards? When do the new versions become official and when can they be implemented?
   Answer: The EAN.UCC XML standards are first published as a draft version, after their technical approval. At this stage the standards can be used as a base for testing and piloting. The messages become the final standard after they are ratified by the GS1 Management Board. After this ratification they can be used for the final implementation. For more details on the approval and publication process, refer to: GS1 XML Publication Strategy.

6. What is the GS1 strategy towards the XML standards versioning? Are the new versions compatible with the old ones?
   Answer: Beginning from the Release 2.0, all changes in the EAN.UCC XML standards are reflected either as major or minor versions. All minor versions are backward compatible within the same major version. Thus, they can reflect only such changes that allow the message recipient to successfully validate a business document based on an old schema against a new schema. Major versions are applied if this compatibility is broken. For further explanations concerning backward compatibility and reflecting versions in schemas and instance documents, see: GS1 XML versioning strategy.

7. What is context and how is it supported in the EAN.UCC XML standards?
   Answer: Context reflects the circumstances in which the business information exchange occurs and on which the information needs depend. Those circumstances are classified in context categories. In every information component there are certain attributes valid in any situation and other dependent on business process, industry sector or geographical location. Those common attributes are context-independent and the different ones are context-specific.
   The context categories in EAN.UCC XML standards are represented in namespaces. For more details please refer to Context.

8. How are namespaces used in EAN.UCC XML standards?
   Answer: In the EAN.UCC XML standards, namespaces are used for reflecting context of the message and the major version number. More information about the namespace structure, its use and standard prefixes can be found in: GS1 Namespaces.

9. Are the namespace prefixes provided by GS1 mandatory, or can the user provide their own prefixes?

Answer: In the EAN.UCC System it is mandatory to use the standard namespace prefixes, in order to support the global interoperability of the standards. For more details see: GS1 namespace prefix.

10. How are the code lists represented in EAN.UCC standards? How can new codes be added?
    Answer: The code lists that are developed and maintained outside of the EAN.UCC System – referred to as the external ones, are only referenced in the XML standards. Their content is not copied to the EAN.UCC schemas and the users have to load them to their applications at their own discretion.
    The internal code lists – developed and maintained by GS1 – are represented in the separate XML schemas. Each code list can be versioned independently from the other schemas, so adding new codes is possible just by changing the minor version of the schema. The list containing new codes is still backward compatible with earlier schemas within the same major version. Detailed explanation can be found in: Code lists.

11. How are the XML built-in types used in EAN.UCC Schemas? Why are they sometimes different from data types represented in the BMS?
    Answer: EAN.UCC standards use the W3C defined data types wherever possible. Currently 10 built-in types are used. In Business Message Standards, which contain the UML representation of the business document, some of the data types may be presented in a different format than the one used in the schema. This is due to the fact that the UML notation is independent from the syntax to which it is finally mapped. For more information about the W3C defined data types used in the EAN.UCC XML standards, please refer to XML Built-in Types.

12. How to handle the date and time related data in case of different time zones? What is the maximum precision of time related data types in EAN.UCC XML Standards?
    Answer: The time zone information for non-UTC times within EAN.UCC system must be expressed by specifying the difference between the local time and UTC. Besides, ‘time’ and ‘dateTime’ data types allow use of additional digits increasing the precision of fractional seconds. In EAN.UCC standards 3 precision digits are used to denote the miliseconds. For more details on using the time related data types please refer to Handling Date and Time.

13. What is the Standard Business Document Header and how is it used in the EAN.UCC schemas? How does it relate to the AS2 Envelope used in previous releases?
    Answer: The Standard Business Document Header (SBDH) is the UN/CEFACT standard, containing information about the routing and processing of the business document. In the EAN.UCC XML 2.0 architecture, the SBDH replaces the Envelope layer used in previous releases of XML standards. For more information, please refer to Standard Business Document Header.

14. What is the functionality of the Transaction element in the EAN.UCC standards?
    Answer: Transaction element enables sending multiple documents as one interchange. The transaction element offers functionality of processing the group of documents together. If one of them fails, all of them will be discarded. More detailed information about the transaction layer are provided in Transaction.

15. How can I express the relationship between two objects that need to be linked, e.g. SSCC of the pallet and cases placed on that pallet, or between the stores and the distribution centres?
    Answer: Two objects can be logically linked together by the use of the Link Command. It provides the possibility of establishing parent-child or peer relationships between several documents. For more information about this and the other commands used in the EAN.UCC standards, please refer to Command.

16. What is the functionality of the proxy schemas? Are they normative?
    Answer: EAN.UCC standard business documents are defined using multiple XML schemas. Because certain parsers do not support multiple schema locations in the instance file, the proxy schema including and importing all the required schemas is provided for every business document. The proxy schema is not normative, it is developed to assist some users. If a parser utilized by the user company supports multiple schema locations, it does not have to be implemented. For more information see: Proxy files.

17. How do I use the extension element in EAN.UCC XML schemas release 2.0? What can be placed there and what are the limitations and implications for validation?
    Answer: In the release 2.0 a new extension component has been introduced. It is a placeholder in the form of the XML '##any' element. Its validation is optional and if no schema definition for the extension components is found, the validator will not report an error.
    From the technical point of view, any content can be placed in the extension element, but in order to maintain global interoperability, users are advised to use only the information components approved within GSMP. More details about the extension element and its validation can be found in the Extension Mechanism.

18. What software tools can I use to process (edit, validate, transform) the XML messages? Are there any free or low cost tools available for companies beginning to work with XML?
    Answer: There are a lot of different types of tools for processing XML, available from many providers. Since many new ones are developed quite frequently and the existing ones are constantly updated, it is better to either make a search on the Internet to find the latest ones or consult the specific provider's website for the most recent updates. The XML Tools section provides some advice on the keywords which may facilitate such search and lists the tools that are used for testing the correctness of the EAN.UCC schemas.

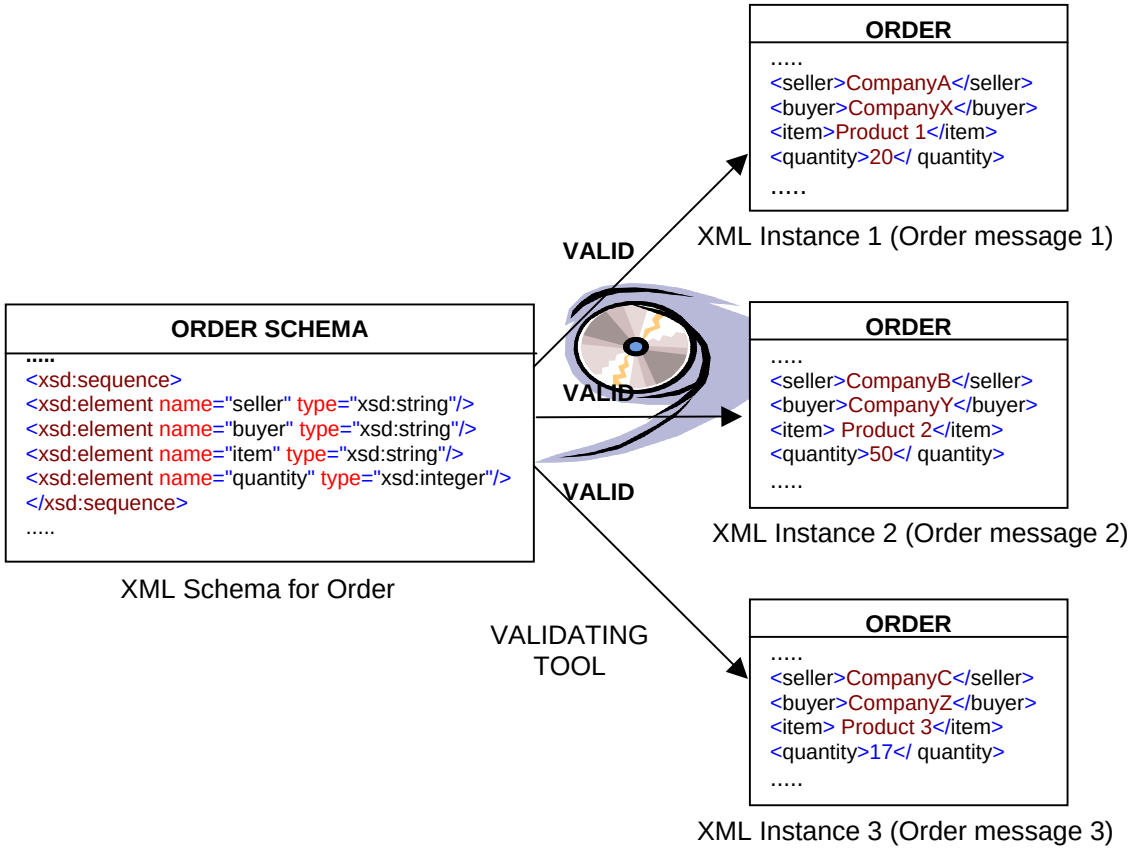
2 GENERAL PICTURE HOW TO USE THE XML STANDARD

2.1 PREREQUISITES FOR USING XML

The use of the XML standards requires certain software tools. In the simplest case, these are an XML processor and validator.

The XML processor is usually included in the application used to view the XML document. It verifies whether the XML file is well formed, which means that it follows all the rules defined in the W3C recommendation for XML syntax.

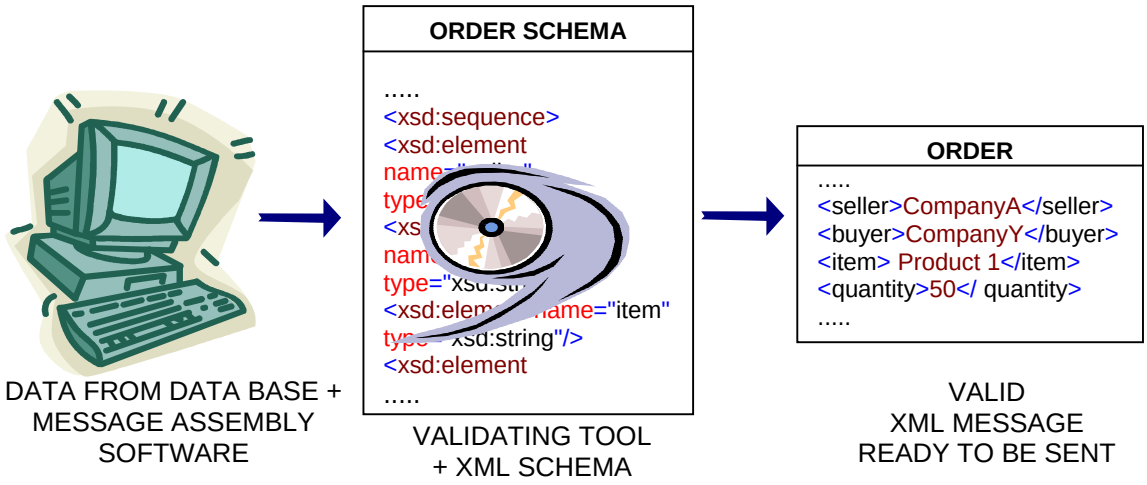
The validator checks whether the XML instance document conforms to the XML schema. XML schemas contain definitions and structures which can be used in XML instance documents. The instance documents contain the actual data with their respective tags in the form of elements and their attributes. Validating software compares the instance document to its schema. This includes checking that the document contains only legal tags, if the data conforms to the format specified in the schema, whether the structure of its content is correct, etc. One schema can define multiple data representations, contained in different XML instance documents. However, the content of all those documents must remain within the limits and restrictions specified in the schema.



Validating different instances of XML business documents with one schema

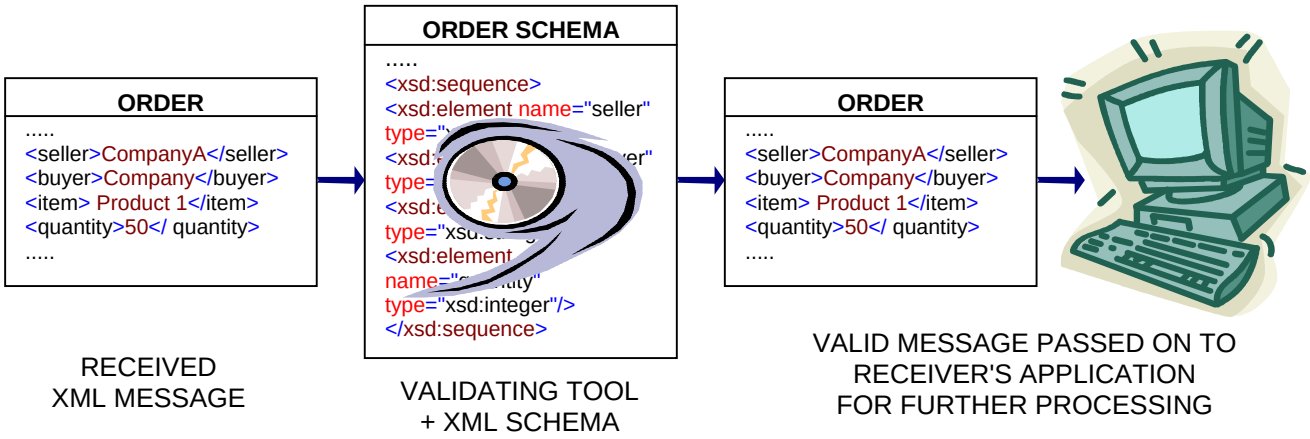
Validation can be used in business scenarios, where each of the trading partners involved in the data exchange holds a copy of a standard schema and validates each instance document sent or received. Of course, there have

to be separate schemas for each type of business document, as they all have different content and structure, e.g. Order, Despatch Advice, Invoice, etc. Documents that are not valid (do not conform to the respective schema) are rejected. At the sender's side, the validating software should be installed at the document generation point. Thus, each business message is validated and the possible errors can be corrected before sending.



Validating XML documents at the sender's side

At the receiver's side the validation takes place in the receiving point of the exchange software, before any data is transmitted to the users' business application.



Validating XML documents at the receiver's side

Usually, the XML tools combine the functionalities of parsing (extracting data and tags from the native XML document), editing, checking well-formednes and validating in one software unit, but there are also a number of self contained validators and editors. For more information refer to the XML Tools.

2.2 GS1 SET OF XML STANDARDS

GS1 produces several work products as part of it's standards for implementation. Two of those work products are schemas and Business Message Standards or BMS.

Message standards allow users to convert business documents into a format that can be electronically exchanged. XML business documents are referred to as “messages”, or “documents”, and their format is defined in the EAN.UCC XML data format message standards. The exchange of these business documents is a component of overall e-Commerce.

EAN.UCC Schema describes the structure of an XML document. The purpose of an XML Schema is to define the legal building blocks of an XML document, EAN.UCC schema design allows developers to supply information embedded within XML documentation.

EAN.UCC XML Standards are an organized suite of XML Schema Modules. The current release is an architecture upgrade from the previous version. The changes have been incorporated to take full advantage of all the features of XML Schema specifications.

Business Message Standards are the artefact of the GSMP that documents the formally approved standards for a business message. Each Business Message Standard brings together the appropriate classes, attributes, and values needed to fulfil the message objective. Specific definitions are provided to ensure clarity around class, attributes, and values. Syntax constraints are identified. The standard also includes the high level and detail level class diagrams depicting the scope of the message, and the relationship of its elements to each other. These diagrams allow parties to see data relationships and to determine where and how to interface extensions to fulfil a business function. Each standard contains a series of extracts from the Global Data Dictionary. Relevant attribute items within a specific class name are presented, identified by type and use.

In the BMS, the data formats restrictions can be different from those in the schemas. This is due to the fact that in the schemas the reusable formats of data types are used, to allow the application designers to reuse the length of data fields. On the other hand, the BMS is based on the particular business requirements, so it can specify the format of each data field, taking into account the semantic meaning of data. Thus, when populating the business message with business data, the BMS format should be used, while the design of application data base should be based on the restrictions specified in the schema. These differences appear mostly in data types defined using the string restriction. For example in the Trade Item message:

BMS class and required number of characters	**Schema type required number of characters**
DescriptionShort: 30	DescriptionType: 1..70
FunctionalName: ..35	DescriptionType: 1..70
TradeItemDescription: ..143	LongDescriptionType: 1..1000
AdditionalTradeItemDescrition ..350	LongDescriptionType: 1..1000

The EAN.UCC XML standards are published as a set of documents per business message. Users need to download all of those documents to implement a given message. Those documents include:

2.2.1 IMPLEMENTERS PACKET

The implementer's packet is a ZIP file, comprised of all the XML files necessary for validating a given XML message. It consists of:

- TableofContents.txt – a text file listing all the files included in the given packet
- Instance File folder, containing one (out of many possible) sample XML file for the message
- HTML Sample folder, containing the HTML representation of the sample XML file from the Instance File folder
- Schemas folder, with the following content
  - EAN.UCC folder – contains two subfolders:
    - Common – includes schemas from the common library, with the target namespace: xmlns:eanucc="urn:ean.ucc:2". These are files that can be reused in many business documents, in any context
    - [Business Process Area name] – includes schemas from the particular business area that are necessary to validate the given business message. These schemas have the target namespace: xmlns:[context-specific prefix]="urn:ean.ucc:[context value(s)]:2" For more details see Context and Versioning
  - SBDH folder – contains the Standard Business Document Header schemas
  - Proxy schema – for the particular business message

2.2.2 BUSINESS MESSAGE STANDARD (BMS)

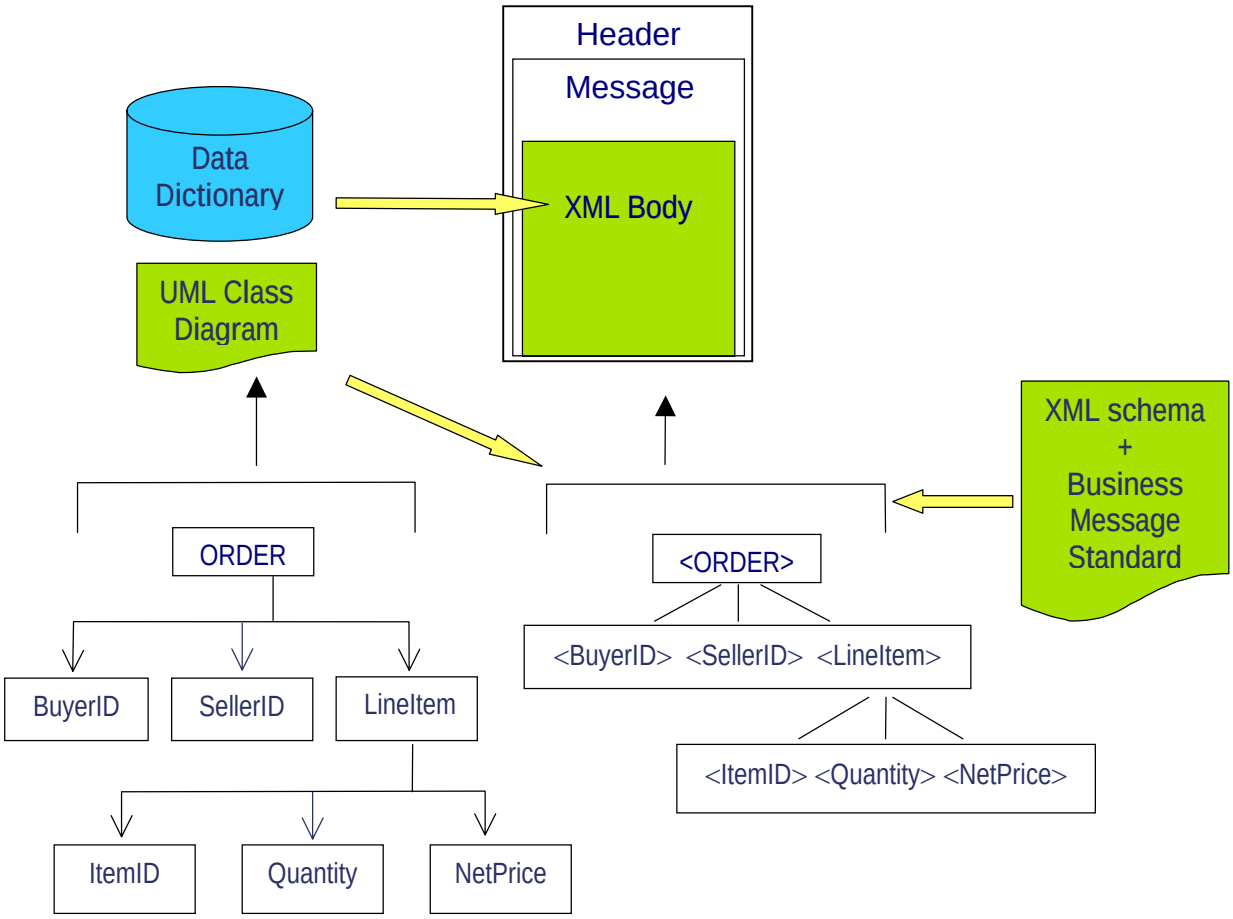
This is the document containing Business Solution Document (BSD) for the given message and the full UML model of the message.

The Global Data Dictionary report lists all the message model components (classes, role names, enumerated values and attributes), their definitions, cardinality, data field length and the title of the XML schema, where those components are defined.

The purpose of the Business Message Standards is to provide the necessary information to implement a particular message as a part of the EAN.UCC System.

2.3 MAPPING FROM UML TO XML

The EAN.UCC XML standards are developed based on UML Class Diagrams. A Class Diagram is mapped into an XML Schema or Schemas. Afterwards, a sample XML file is developed, complying to data structures defined in the schema and data definitions from the Global Data Dictionary. The Class Diagrams with the model description and the sample XML file are then used to create the Business Message Standard, published together with the set of schemas.



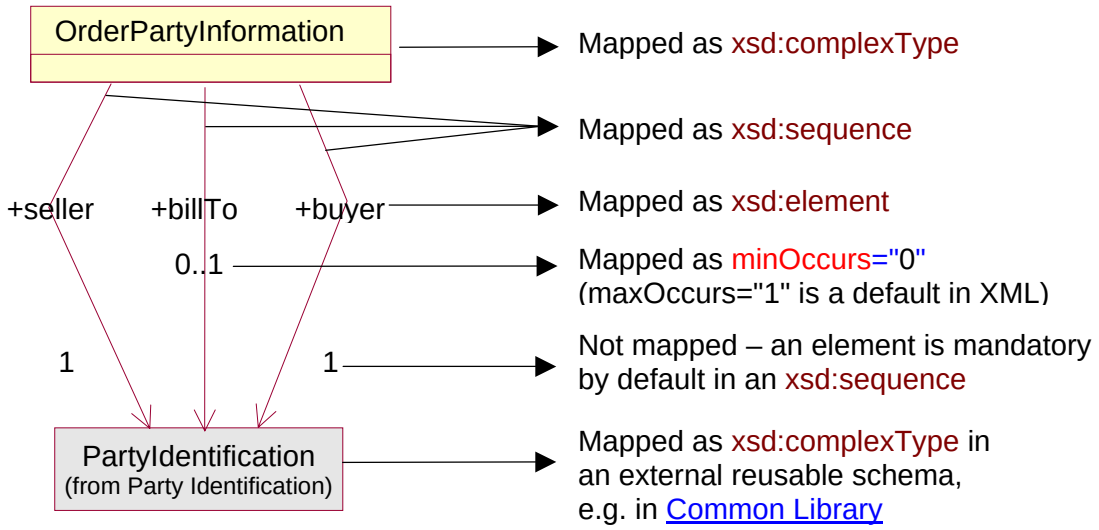
The general picture of EAN.UCC XML standards development

The actual mapping of the specific components of UML to XML is driven by the XML syntax constraints and the chosen schema design model. However, some general rules are applied whenever it is possible (exceptions from those rules are rare and caused by the syntax constraints). The user does not have to understand all the details and complexities of UML to XML mapping, but the basic familiarity of key principles can be helpful in reading the EAN.UCC standards.

The basic rules for mapping of UML components to XML can be illustrated by the three examples below:

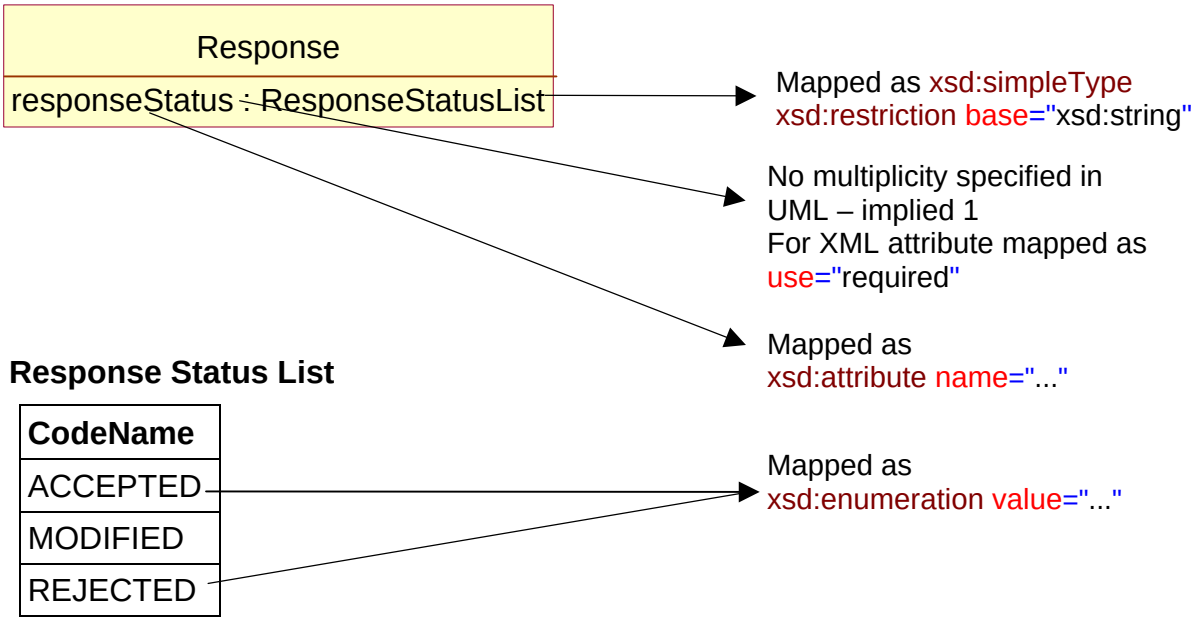
Example 1

Mapping of UML components as XML Complex Types and Elements



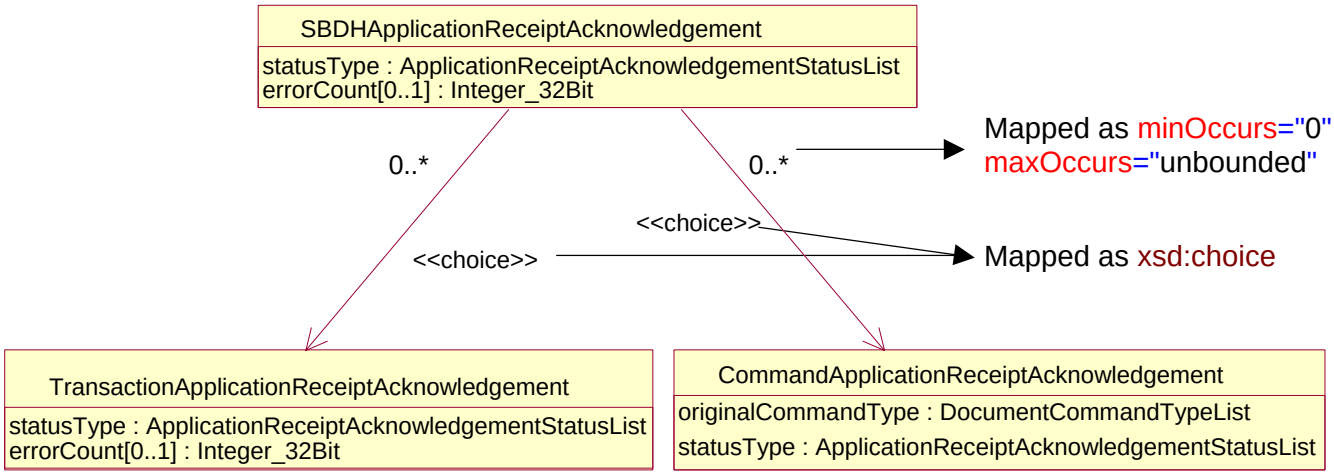
Example 2

Mapping of UML components as XML Simple Types and Attributes



Example 3

Mapping of UML components as XML Choice



3 GS1 XML PUBLICATION STRATEGY

The EAN.UCC XML standards have to undergo a strict formal approval process in order to ensure their accuracy and integrity within the XML messages themselves, as well as with the other EAN.UCC standards.

Once the XML messages are developed as a part of the Global Standard Management Process (GSMP), they are reviewed by the Information Technical Requirements Group (ITRG). After verifying that they meet business requirements and are technically correct, the ITRG formally approves them in a voting process. At that time, the standards receive a DRAFT status and are published on the GSMP website. The draft messages are published in order to allow users to analyse their content, test and plan pilot implementations. The overall content of those messages is not likely to be changed, but there may be the need to amend them before their final release, so they should not be considered as being a base for real implementation.

The standards become final when they are ratified by the GS1 Management Board. This typically happens once or twice a year, depending on the number of messages being developed and any major architectural changes requiring new release of all the XML messages. Upon this ratification, the messages are published as a final standard on the GSMP website. The ratified messages can be used for the final implementation.

The standards from all the previous releases (from 1.0 to 1.3.1) can be downloaded from the following link: http://www.ean-int.org/xml and the new ones (2.0, 2.0.2 and 2.0.2) form: http://www.ean-ucc.org/global_smp/ean.ucc_standards.html

NOTE: Beginning from the Release 2.0, all the minor versions of XML schemas will be published as a part of a given major version of the standards. The next major release will always contain only the latest minor versions schemas of the previous release.

4 GS1 NAMESPACES

4.1 GS1 NAMESPACE STRUCTURE

The namespaces in GS1 reflect the context and the major version of the schema components. The GS1 namespaces have the format of the Uniform Resource Names (URN). All URNs have the following structure:

<URN> ::= "urn:" <NID> ":" <NSS>

where :
<NID> is the Namespace Identifier
<NSS> is the Namespace Specific String.

In all the GS1 namespaces, “ean.ucc” is used for the NID. The NSS of all URNs assigned by GS1 have the following hierarchical structure, reflecting the basic context categories and major version number:

urn:ean.ucc:_____:_____:_____:_____
BP IC GP Major Version

where:
BP – Business Process
IC – Industry Classification
GP – Geopolitical Context

Note: NSS in GS1 namespaces is case-sensitive, even though in the RFC 2141 is specified as not sensitive.

Every component has all the context categories defined – either specific, global or all. The namespace does not specify the context category if it is global. For example, a component that is common across all geographical regions and industry classifications, within the Global Data Synchronization Network (GDSN) Business Process has the following namespace:
gdsn="urn:ean.ucc:gdsn:2"

4.2 GS1 NAMESPACE PREFIX

EAN.UCC XML schemas do not use any default namespace. The GS1 information components must be assigned to a namespace that reflects the context it was defined in. This namespace (context) should be explicitly specified for each component.

The GS1 standard specifies the standard namespace prefixes. Usually, they are created from the name of the business process:

- "pay" for PAY process, e.g.: **pay**="urn:ean.ucc:pay:2"
- "align" for ALIGN process, e.g.: **align**="urn:ean.ucc:align:2"
- "plan" for PLAN process, e.g.: **plan**="urn:ean.ucc:plan:2"
- "deliver" for DELIVER process, e.g.: **deliver**="urn:ean.ucc:deliver:2"
- "order" for ORDER process, e.g.: **order**="urn:ean.ucc:order:2"
- "gdsn" for GLOBAL DATA SYNCHRONISATION process, e.g.: **gdsn**="urn:ean.ucc:gdsn:2"
- "eanucc" for all schemas COMMON to all the contexts: **eanucc**="urn:ean.ucc:2"

The local extensions also have their own namespace and standard prefixes, e.g. TradeItem components specific for Sweden use a separate namespace and the "sw" prefix: **sw**="urn:ean.ucc:align:sweden:2"

Use of these standard namespace prefixes is mandatory in the EAN.UCC System. Although, from the XML syntax point of view, any prefix can be used as long as it points to the correct namespace, assigning non-standard prefixes would compromise the global interoperability of standards.

In addition, certain mapping and processing XML tools are prefix sensitive and use of non-matching prefixes by the business partners causes serious validation and processing problems.

4.3 XML SCHEMA NAMESPACE AND PREFIX

For the XML schema components, the World Wide Web Consortium (W3C) URL http://www.w3.org/2001/XMLSchema is used as the namespace. Again, the prefix for that namespace has been standardised in the EAN.UCC System and all the existing schema modules use xsd: for the XML Schema namespace.

5 GS1 XML VERSIONING STRATEGY

Beginning from release 2.0, versioning of EAN.UCC XML schemas is no longer coupled with the publication of EAN.UCC Business Message Standards (BMS), as they were in previous releases (1.1, 1.3, 1.3.1 and 1.3.2) This means that all schemas, including the ones for business documents, and common library schemas do not necessarily hold the same version number.

Thanks to this policy, a change in the common library does not necessitate the re-publishing of all schemas as it was the case in previous releases. It also ensures the backward-compatibility between two successive releases of EAN.UCC schemas.

Backward compatibility means that a message sender can create a business document based on an old schema, despatch it to a recipient, who can successfully validate it against a new schema.

5.1 MINOR VERSIONS

Minor versions are interim releases of an XML schema that contain only the changes that are backwardly compatible with the existing version of this schema.

Changes that can be incorporated in a Minor Version:

- Adding new optional elements or optional attributes
- Changing attributes cardinality from mandatory to optional
- Changing element multiplicity from old schema [0..1] to new schema [0..*]
- Changing element multiplicity from old schema [1..1] to new schema [1..*]

- Adding a term to an enumerated list

5.2 MAJOR VERSIONS

Major versions are the releases of a suite of XML schemas that contain changes not backwardly compatible with the existing suite of the same schemas.

All the schemas within a given Business Process have the same major version, to ease the implementation and ongoing maintenance of EAN.UCC XML schema versioning. It means that if one schema in a suite of XML schemas within a particular business process needs to be upgraded to the next major version, all the schemas within that business process are upgraded to the next major version. The business processes are defined by business users during the business requirements gathering.

Changes that require incrementing the major version:
- Changing an attribute cardinality from optional to mandatory
- Adding a mandatory element
- Changing an attribute or element tag name
- Changing element multiplicity from old schema [0..*] to new schema [0..1]
- Changing element multiplicity from old schema [1..*] to new schema [1]
- Changing the sequence of elements in a <xsd:sequence> tag

5.3 CONTEXT CATEGORIES SPECIFYING VERSION NUMBERS

5.3.1 VERSION NUMBERS IN XML SCHEMAS

The major version number of the schema is specified in the schema namespace.

Example
urn:ean.ucc:gdsn:fmcg:1

The minor versions are reflected in the version attribute of the xsd:schema element. For the ease of implementation, the internal schema version attribute reflects the entire version (both major & minor) of the schema.

Example
```
<xsd:schema targetNamespace="urn:ean.ucc:gdsn:fmcg:2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:ean.ucc:gdsn:fmcg:2" elementFormDefault="unqualified"
attributeFormDefault="unqualified" version="2.3">
```

This approach ensures backward compatibility. Due to the fact that the minor versions are not reflected in the namespace of a schema, instance documents can be validated against the older minor schema. In addition, any schemas that ‘include’ this schema do not need to change because the target namespace of the included components is the same as the target namespace of the including schema.

5.3.2 VERSION NUMBERS IN THE BUSINESS DOCUMENTS – XML INSTANCE

Since the major version is specified in the namespace, the business document (xml instance document) contains the major version by default. However, the complete version also needs to be specified in the instance document, as the application systems that receive the business documents need to know what version the trading partners are using.

EAN.UCC business documents use the “documentStructureVersion” attribute defined in the DocumentType complex type to reflect the complete version of the business document schema, e.g. for the Order.xml message, the “documentStructureVersion” attribute will reflect the version number of the Order.xsd schema. The versions of the Common Library schemas are not reflected in the instance document, as they do not affect the validation process, due to their backward compatibility.
This type is defined as:

```
<xsd:complexType name="DocumentType" abstract="true">
<xsd:attribute name="contentVersion" type="eanucc:VersionType" default="2.0"/>
<xsd:attribute name="documentStructureVersion" type="eanucc:VersionType"
default="2.0"/>
        <xsd:attribute name="lastUpdateDate" type="xsd:date"/>
        <xsd:attribute name="creationDate" type="xsd:dateTime" use="required"/>
        <xsd:attribute name="documentStatus" type="eanucc:DocumentStatusListType"
use="required"/>
</xsd:complexType>
```

Please note, that starting from the release 2.0, the attribute “documentStructureVersion” no longer has a fixed value, as it did in previous releases. Instead, trading partners will specify the value of this attribute to reflect the major and minor version they are using.

5.3.3 VERSION NUMBERS IN THE BUSINESS DOCUMENTS AND IN THE STANDARD BUSINESS DOCUMENT HEADER

The Standard Business Document Header schema contains the 'TypeVersion' element, which is the place holder for the version of the business document or business documents sent with the one header. The SBDH specification requires that all the documents sent with one header have the same version number. To comply with this requirement, GS1 recommends that only the major version number (the one specified in the namespace) of the business documents is indicated in the 'TypeVersion' element of the SBDH. Thus, it is possible to send in one message only documents of the same type (only invoices, only orders, etc) of the same major version. The minor versions of those documents can be different, as they are compatible within the same major version.

6 CONTEXT

Business messages are always used in business domains and processes under certain real-world circumstances. There are many similarities and also many differences between them. EAN.UCC schemas are based on vocabulary of components that represent those common and different circumstances. To manage this complexity, a concept of business context has been introduced. It reflects the circumstances in which the business information exchange occurs.

An example can be a trade item description. There are certain attributes valid for every type of product, e.g. brand name, identification number, etc. However, the pharmaceutical industry needs certain specific attributes, different from the automotive and textile sectors. Thus, those different attributes are context-specific, while the context in this case is the industry sector, determining certain information needs. The common attributes are context-independent.

6.1 CONTEXT CATEGORIES

The context in which the business collaboration takes place can be specified by a set of categories and their associated values. There are 8 categories in total, but in EAN.UCC XML messages currently only three of them are used:

- Business Process, in which collaboration takes place, e.g. ordering, delivery, etc.
- Industry Sector, in which the business partners are involved, e.g. FMCG, Hardlines, etc.
- Geopolitical, reflecting the geographical factors that influence the business semantics, it can be either country-specific, e.g. only for France or Sweden, limited to certain economic regions, e.g. NAFTA, European Union, and finally, it can be applicable everywhere in the world – in this case the context is defined as Global

Other categories can be introduced according to needs, in further releases of the EAN.UCC XML standards.

Components that can be used in the entire context category, across all its values, has an implied value 'In All Contexts'. In this case, the value is omitted, for clarity.

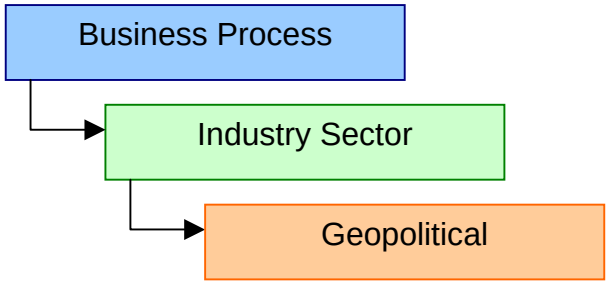
Some components can be associated with more than one context category value, e.g. 'Pharmaceutical and Metallurgical'.

6.2 REPRESENTING CONTEXT IN EAN.UCC XML STANDARDS

In general context is a non-hierarchical concept since all categories are at the same level of importance. However, in the XML schemas hierarchy has been introduced in order to facilitate context management.

The hierarchy order starts with the Business Process Category, followed by

the Industry Sector and then the Geopolitical Category.



The hierarchy of context categories used in EAN.UCC XML schemas

In the schemas, context is represented in namespaces.

7 CODE LISTS

There are two types of code lists in EAN.UCC XML messages: external and internal ones.

7.1 EXTERNAL CODE LISTS

External code lists are defined and maintained outside the GS1 community, usually by other standard bodies, e.g. ISO. Their examples are:

- Country Codes - ISO 3166-1:1997, defined in Country.xsd schema, as the ISO3166_1CodeType
- Country Subdivision Codes - ISO 3166-2:1998, defined in Country.xsd schema, as the ISO3166_2CodeType
- Currency Codes - ISO 4217:2001, defined in MonetaryAmount.xsd schema, as the CurrencyISOCodeType

All of them are defined as a part of the common library. They are defined as strings restricted to an appropriate number of characters, but do not contain the code list values. The reason behind this decision is that enumerating the lists content would cause problems with their maintenance, which is done outside of the EAN.UCC System and would breach the copyright ownership.

The users are advised to consult the website of the International Standard Organisation www.iso.org to check how the lists can be acquired and used.

7.2 INTERNAL CODE LISTS

Internal code lists are those developed and maintained within the EAN.UCC System. Starting from release 2.0, those code lists are defined in separate schemas and are no longer embedded in the common library schemas. This practice requires giving up some advantages, mainly the ability for parsers to enforce strict validation. In other words, the code list embedded within the schema cannot be changed, without affecting the validation process. On the contrary, if a code list is only referenced in the given schema and defined

externally, various versions of that code list can be used as a base for validation.

On the other hand, this ensures that any change to a code list schema does not trigger a new release of the entire set of EAN.UCC schemas, which would require additional upgrades for the users. Defining code lists outside of the main schema document allows upgrading lists as soon as business demand arises and facilitates version management.

Code lists are defined as xsd:enumeration. The lists are included or imported into the schema document, which uses it.

Advantages for implementers of this approach include reduced maintenance costs, faster implementation of list changes, reduced versions of schemas and simpler publications. In addition, the handling of code lists is consistent with the new, similar handling of components.

7.2.1 INTERNAL CODE LISTS IN CONTEXT

Code list schemas follow the same principles as all the other EAN.UCC schemas, with respect to namespaces. If a given code list is used across different contexts, it is defined within the common library namespace, e.g. TimePeriodList.xsd or AllowanceChargeList.xsd.

Code lists used just in one context are defined in the namespace of the schema document that uses them. For example, ReplenishmentRequestStatusList.xsd and CollaborationPriorityCodelist.xsd are defined in the Plan namespace (xmlns:plan="urn:ean.ucc:plan:2").

This approach ensures backward compatibility. Due to the fact that the minor versions are not reflected in the namespace of a schema, instance documents can be validated against the older minor schema. In addition, any schemas that ‘include’ this schema do not need to change because the target namespace of the included components is the same as the target namespace of the including schema.

7.2.2 INTERNAL CODE LIST VERSIONING

Each code list is defined in its own schema file, which is versioned separately. When a change is made to a code list schema, it is versioned in the same way as any other component. When new code definitions are approved by the respective Business Requirements Group (BRG), the code list schema is updated and the minor version number within the code list schema is incremented. This allows users to have access to the codes in the most efficient manner.

Addition of code values constitutes a minor version change. If a code deletion or modification of definition is necessary, it will only be implemented under a major version change.

8 XML BUILT-IN TYPES

The XML standard developed by the World Wide Web Consortium (W3C) lists 44 data types that are embedded in the specification. This means that these data types can be used in XML schemas with no need to define them. These data types should be implicitly understood by all the XML-aware software tools. The built-in data types have certain standard facets to represent them in the schema or to restrict their range.

In EAN.UCC standards, only a subset of the built-in types is used. Some of the W3C data types have different representation in the Business Message Standards than the standard one used in the schema. This difference is due to the fact that the BMS contains a UML representation of the business document, which is meant to be independent from the syntax to which it will finally be mapped.

The list of W3C data types used in EAN.UCC standards, together with their representation in the schema and the BMS is presented below:

W3C Data Type	W3C XSD	EAN.UCC BMS Data Type
string		AN String
boolean	true/false 1/0	Y/N
decimal	totalDigits=5 fractionDigits=3	Numeric 5,3
integer	totalDigits=5	Number 5
nonNegativeInteger	totalDigits=5	Number 5
float		
dateTime	ISO 8601 Year: YYYY (e.g. 1997) Year and month: YYYY-MM (e.g. 1997-07) Complete date: YYYY-MM-DD (e.g. 1997-07-16) Complete date plus hours and minutes: YYYY-MM-DDThh:mmTZD (e.g. 1997-07-16T19:20+01:00) Complete date plus hours, minutes and seconds: YYYY-MM-DDThh:mm:ssTZD (e.g. 1997-07-16T19:20:30+01:00) Complete date plus hours, minutes, seconds and a decimal fraction of a second YYYY-MM-DDThh:mm:ss.sTZD (e.g. 1997-07-16T19:20:30.45+01:00) where: YYYY = four-digit year MM = two-digit month (01=January,	CCYYMMDDThh:mm:ss

	etc.) DD = two-digit day of month (01 through 31) hh = two digits of hour (00 through 23) (am/pm NOT allowed) mm = two digits of minute (00 through 59) ss = two digits of second (00 through 59) s = one or more digits representing a decimal fraction of a second TZD = time zone designator (Z or +hh:mm or -hh:mm)	
time	ISO 8601 – structures related to time: hh:mm:ss.s – explanations and examples – see dateTime	hh:mm:ss
date	ISO 8601 ISO 8601 – structures related to date: YYYY-MM-DD – explanations and examples – see dateTime	CCYYMMDD

W3C data types currently not used by EAN.UCC:

- normalizedString
- token
- Name
- NCName
- ID
- IDREF
- IDREFS
- ENTITY
- ENTITIES
- NMTOKEN
- NMTOKENS
- nonPositiveInteger
- negativeInteger
- long
- language
- int
- short
- byte
- unsignedLong
- unsignedInt
- unsignedShort
- unsignedByte
- positiveInteger
- double
- duration
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- hexBinary
- Base64Binary
- anyURI
- QName
- NOTATION

8.1 HANDLING DATE AND TIME

For data elements where time is required, two built-in data-types are used within EAN.UCC XML standards: 'dateTime' and 'time'. The value space of ‘time’ and ‘dateTime’ XSD data types is defined in article 5.3 of ISO 8601, the details can be found at the following links:

http://www.w3.org/TR/xmlschema-2/#time
http://www.w3.org/TR/xmlschema-2/#dateTime

As per the value space of ‘time’ and ‘dateTime’, additional fractional seconds (milliseconds) and Time zone information can specified.

8.1.1 USE OF MILLISECONDS

Both ‘time’ and ‘dateTime’ data types allow use of additional digits increasing the precision of fractional seconds if desired, in the format ss.s.

‘ss.s’ denotes two digits of second (00 through 59) followed by one or more digits representing a decimal fraction of a second (milliseconds). The fractional seconds part is separated from the two digits of second by the use of a ‘dot’ as a separator.

Though any number of digits for the fractional seconds is supported, only 3 precision digits should be used within EAN.UCC XML messages to denote the milliseconds.

Example 1

The following values are true for the attribute 'creationDate' which is of type XSD 'dateTime' in all versions of EAN.UCC XML Standards.

- creationDate="2003-03-22T09:30:47"
  The example above indicates 47 seconds and 0 milliseconds.
- creationDate="2003-03-22T09:30:47.0"
  The example above indicates 47 seconds and 0 milliseconds and is equivalent to example 1 from above.
- creationDate="2003-03-22T09:30:47.233"
  The example above indicates 47 seconds and 233 milliseconds.

8.1.2 USE OF TIME ZONES

Both ‘time’ and ‘dateTime’ data types allow specifying the time zone, following the time information. The time zone information for non-UTC times within EAN.UCC System must be expressed by specifying the difference between the local time and UTC. This is indicated by immediately following the time representation by a sign, + or -, followed by the difference from UTC represented as hh:mm (note: the minutes part is required).

Example 2

- creationDate="2003-03-22T09:30:47.233-05:00"
  The example above indicates Eastern Time (ET), which is 5 hours behind UTC.
- creationDate="2004-11-06T12:43:17.000+09:00"
  The example above indicates Tokyo Time, which is 9 hours ahead of UTC.

Example 3

A composite example of the XSD data type ‘dateTime’ following the above mentioned guidelines:

- creationDate="2003-03-22T09:30:47.233-05:00"

where:

- '2003' denotes the year
- '03' denotes the month
- '22' denotes the day
- 'T' denotes the time separator
- '09' denotes the hours
- '30' denotes the minutes
- '47' denotes the seconds
- '.' denotes the fractional seconds separator
- '233' denotes the fractional seconds (milliseconds)
- '-' denotes the time zone offset indicator indicating 'behind UTC'
- '05' denotes the hours
- ':' denotes the minutes separator and '00' is the minutes

9 STANDARD BUSINESS DOCUMENT HEADER

The Standard Business Document Header (SBDH) replaces the Envelope layer including the AS2 Message Header, used in previous EAN.UCC XML standards releases. The SBDH provides information about the routing and processing of the XML instance document. The SBDH is designed to be independent of the specific transport protocol used. The information contained in the SBDH can be used by communication applications to determine routing whether the transport protocol used is ebMS, AS2, or any other protocol.

The SBDH can also optionally provide business scope and business service information. In EAN.UCC schemas, the SBDH is designed to be an integral part of the XML instance document (in other standards, the SBDH may be an object associated with the XML instance document).

The SBDH schema contains an element specifying the version number of the document or documents contained. For more details on how to use this data element, please refer to Version numbers in the business documents and the Standard Business Document Header.

The detailed UN/CEFACT specification for the SBDH and the guidelines on how to use it can be downloaded from the GS1 website.

10 MESSAGE LAYER

The message layer defines actions that should be performed on the specific document or documents by the receiving application. Those tasks are defined as commands.

Use of this layer allows a reduction in the number of documents needed to perform an efficient exchange of business information. The same document can be used with different commands. Hence, no separate documents like 'Add Order, 'Change Order' or 'Delete Order' are needed. The same 'Order' document can be sent with a relevant command, e.g. 'add', 'change' or 'delete'.

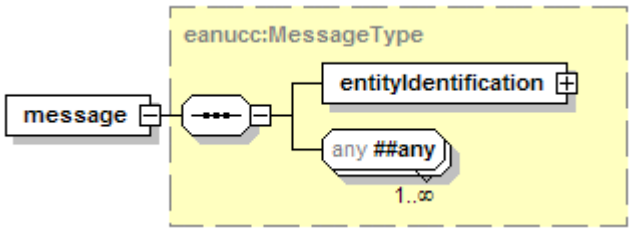
In a similar way, several documents can reuse the same commands, so adding the new commands when such a need is identified, is quick and easy, with no need to make any change to the other document schemas. The component that needs to be added then is a new item in the 'documentCommandListType', defining the type of action that should be performed on the corresponding documents by the receiving end.

The Message layer contains three main components:

- transaction
- command
- interface to the Document Layer

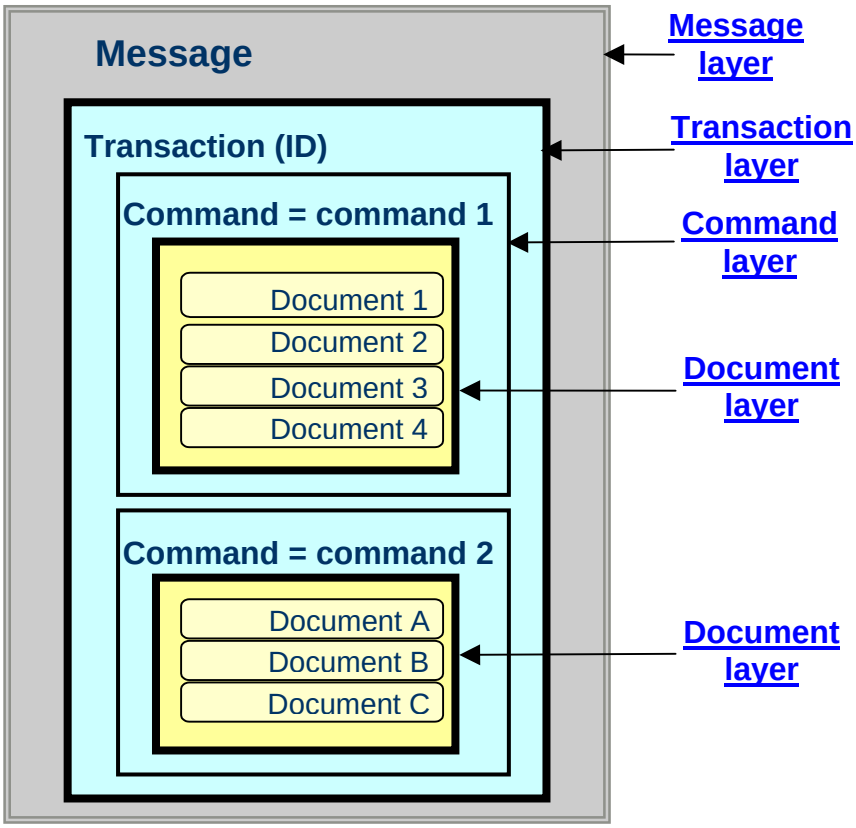
The Message layer is linked to the Header layer by the 'message' component, introduced in version 2.0.2 of the EAN.UCC XML Standards. It acts simply as a placeholder allowing for appending multiple transactions in one header.

The 'message' component is identified by the entity identification, and contains a placeholder where the transaction component can be inserted. This placeholder has a form of the XML schema '##any' element. It can have multiple occurrence, so that many transactions can be appended.



The 'message component structure

The XML 'any' element has a built-in attribute: processContents indicating how an XML processor should handle the validation of XML documents against the elements specified by the 'any' element. In the message component, the value of this attribute is set to 'strict', it means that the XML processor must obtain the schema for the required namespaces and validate any element from those namespaces.



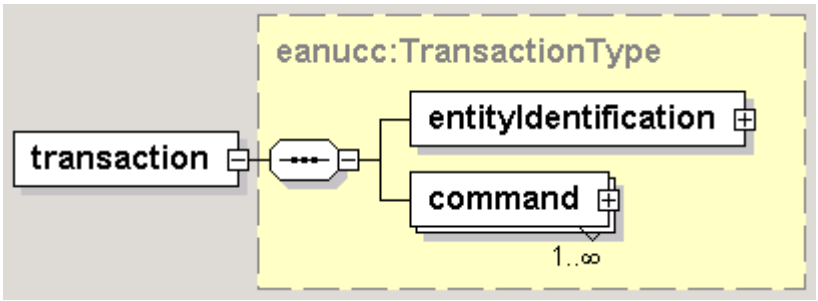
Message layer overview

Each message sent can contain several transactions and every transaction can hold multiple commands concerning one or more documents. This architecture allows great flexibility of business information exchange.

10.1 TRANSACTION

10.1.1 TRANSACTION STRUCTURE

A transaction provides functionality of submitting multiple commands simultaneously. It allows the users to process the group of messages together. If one of them fails, all of them may then be discarded.

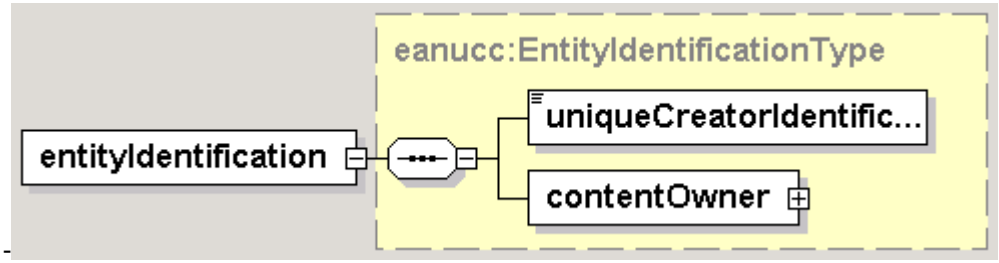


Transaction elements

The Transaction schema contains two main components:

1. Identification of the transaction - a component called entity identification, consisting of:

a. the party that created the set of commands
b. a unique identifier assigned by that party



The entity identification structure

2. The commands themselves. The ‘command’ element is a container, where one or multiple commands that form one transaction can be inserted

The number of commands in one transaction is unlimited from the XML syntax point of view, however, it is limited by the bandwidth and application processing capabilities. All messages sent in one transaction should be of the same type, e.g. only Orders or Invoices.

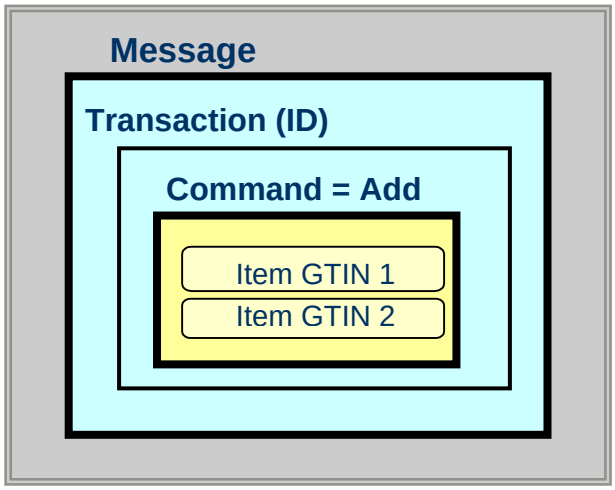
10.1.2 TRANSACTION FUNCTIONALITY

Transaction applies the principle of 'all or nothing'. If one of the documents in a transaction is not valid, then all of them are rejected.

The transaction element can be considered as a flag indicating that if some of the documents nested within it fail the validation, the processing of all the remaining ones should stop. Obviously, this functionality has to be pre-programmed into the processing application. It is important to note that this relates only to technical level validation, it does not cover the business level validation, since it is not possible to verify the conformance to the business rules at the moment of 'unwrapping' the transaction layer.

The technical validation checks if the tag names, order and data field format are inconsistent with the schema. For example, the GTIN is defined in the schema as strictly 14 digits long, therefore, if in the instance file it has 13 or 15 digits, the document fails the technical validation. On the contrary, if the check digit in the GTIN is not correct, the file will still be considered as valid, as this information can only be verified at the business application level.
In the same way the cardinality of components is checked – if a mandatory component is missing, the file will not be valid, but if the given component had been defined as optional, the file will be valid, even if from business point of view the information should be present, the file will still be technically valid and the full transaction content will be accepted.

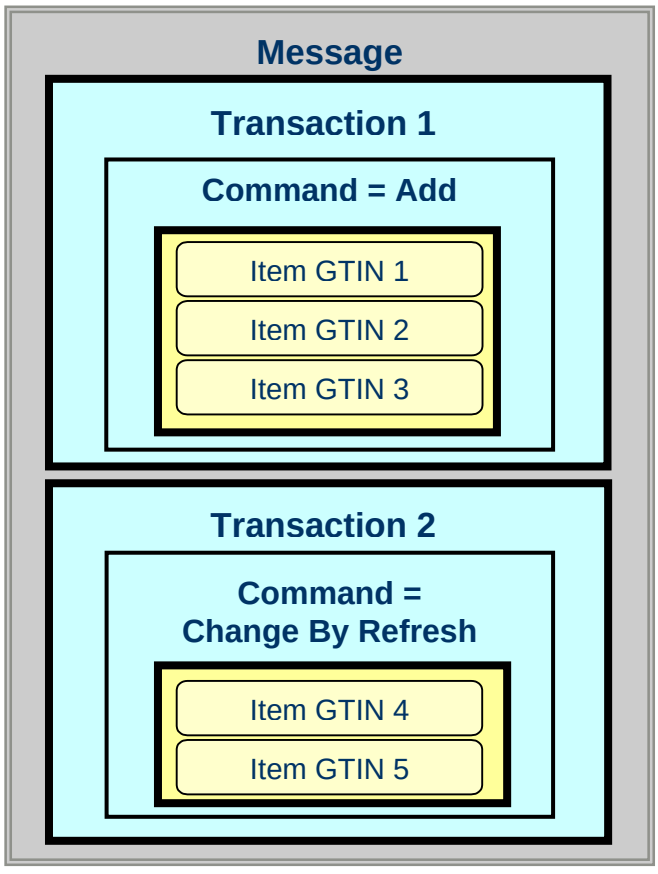
Example 1
A sender needs to send two TradeItem messages, one for GTIN 1, the second for GTIN 2. The product 1 is related to product 2. Instead of sending them separately, in two distinct transmissions, he can transmit them together in one transaction:



Both documents have to be correct (valid), otherwise both will be rejected.

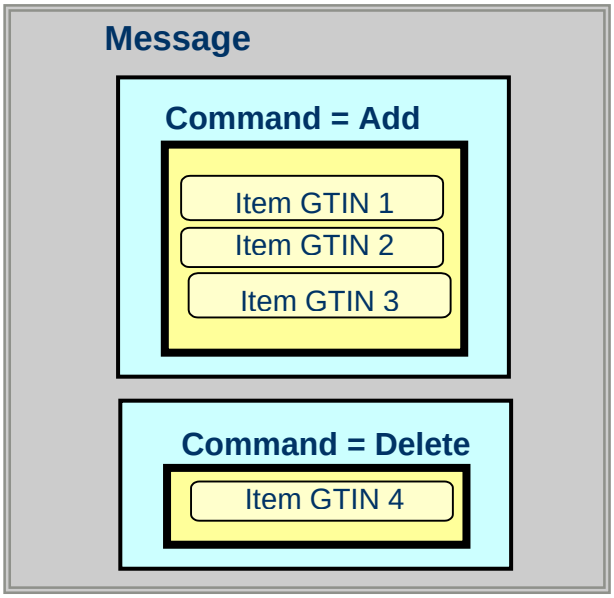
Example 2

A sender needs to send three new TradeItem messages to a customer and amend two other TradeItem descriptions, which had been sent earlier. All five TradeItem documents can be sent together in one message – three with a command ADD and two with a command CHANGE BY REFRESH. Both commands could be inserted in one transaction, but since the new TradeItem messages are not related to the amended ones, they can be placed in two separate transactions, sent in one message:



If any of the first three messages (TradeItem for GTIN 1, 2 or 3) is not valid, all three will be rejected but the messages from the second transaction (TradeItem for GTIN 4 and 5) will still be processed.

Example 3
A sender needs to send three new TradeItem messages to a customer and cancel one sent previously. Those messages are not interrelated and even if one of them fails, the sender wants to maintain the other as valid ones. All four messages are sent in one transmission, three with a command ADD and one with a command DELETE, but without the transaction layer:



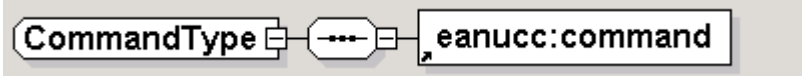
If any of those messages (TradeItem for GTIN 1, 2, 3 or cancellation of TradeItem for GTIN 4) are not valid, the other ones will still be processed.

10.2 COMMAND

Commands are used by a trading partner to instruct the receiving application about an action that should be performed on a given document. All commands are transitive and are discarded after executing the task.

Some of the actions to be performed are defined by the command itself, while other ones by an attribute of the command.

The 'command' is an abstract element, extended by the following elements:
'documentCommand'
'documentIdentificationCommand'
'linkCommand`



The Command Type structure

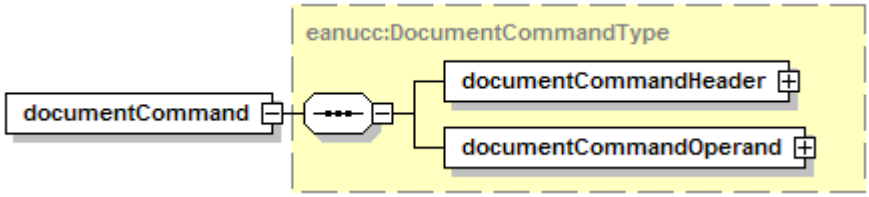
The substituting elements are defined in DocumentCommand.xsd, DocumentIdentificationCommand.xsd and LinkCommand.xsd.

10.2.1 DOCUMENT COMMAND

The Document Command is used to instruct the recipient of that command to perform a particular action related to the documents within the command. The document in question has to be present, i.e. it must be sent together with the command. The actions, which can be performed on it include:

- 'Add'
- 'Refresh'
- 'Correct'
- 'Delete'

The Document Command is an extension of the abstract Command. It contains two main components: the 'documentCommandHeader' and 'documentCommandOperand':



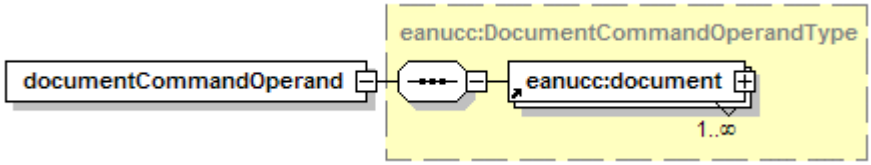
The Document Command structure

The 'documentCommandHeader' specifies the action that should be performed on the given document. Those tasks, defined as the required attribute of the header element, include:

ADD	– the receiving application is instructed to store the document or documents
CHANGE_BY_REFRESH	– the receiving application is instructed to update the existing document or documents, by total replacement
CORRECT	– the receiving application is instructed to update the existing document or documents, by total replacement, skipping certain business specific validation rules. The syntactical and content validation rules still apply. This command is used in cases where the specific validation rules would otherwise prevent the application from changing data. It can be used only if the correction does not impact the integrity of the corrected data. Otherwise, correction should be performed by sending two commands: DELETE (with the old document) and ADD (with the new document) in one transaction.
DELETE	– the receiving application is instructed to delete the document or documents

The 'documentCommandOperand' contains an abstract 'document' element that can be extended by the actual business document elements, defined in

the business message schemas. Those are the documents upon which one of the actions defined in the header element should be performed.



The Document Command Operand structure

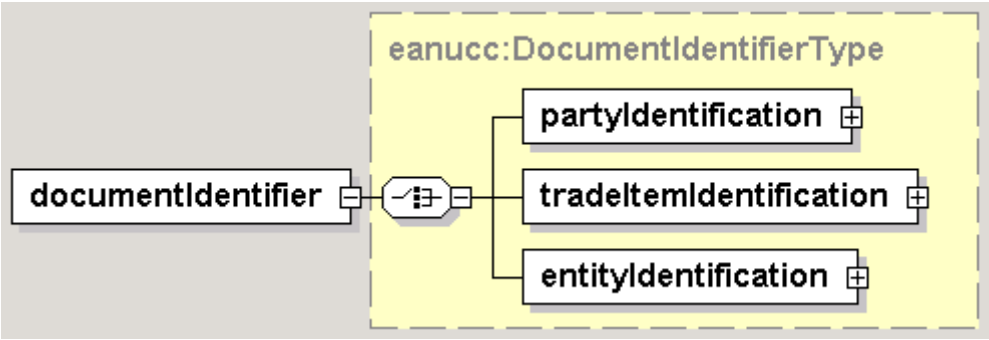
The list of the possible documents contains all the business messages defined within the given major version.

10.2.2 DOCUMENT IDENTIFICATION COMMAND

This is an extension of the abstract Command, defining operations that require just an identification of the relevant documents. The only operation currently supported by this method is DELETE, defined as a value of an attribute of the 'documentIdentificationCommand' element.

The Document Identification Command structure

The document on which the action should be performed is identified in the 'documentIdentifierList' element.



The Document Identifier structure

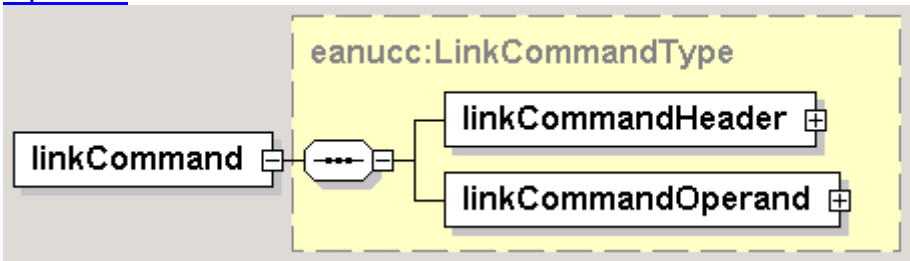
It can be specified in three ways:

1. By 'partyIdentification' – using GLN and optionally, an additional identifier of the concerned party.
2. By 'tradeItemIdentification' – using GTIN and optionally, an additional identifier of the trade item concerned by the document in question.
3. By 'entityIdentification' - a combination of the unique document identifier assigned by its creator and the unique identification of that document creator (GLN and optionally, an additional identifier).

10.2.3 LINK COMMAND

- This type of command allows to establish parent-child or peer relationships between several entities.

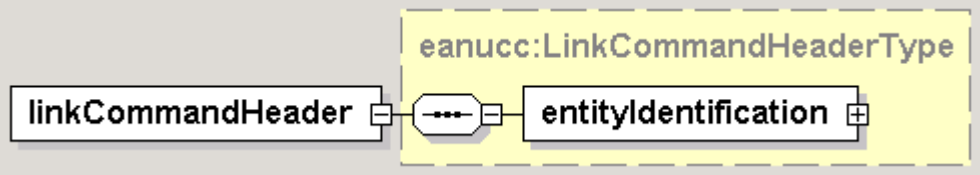
The Link Command consists of two major components: Link Header and Link Operand.



The Link Command structure

10.2.3.1 Link Header

The 'linkCommandHeader' specifies whether the function of the command is to link or unlink the documents concerned. They are defined as the enumeration values: LINK and UNLINK of the element's attribute 'LinkCommandListType'.

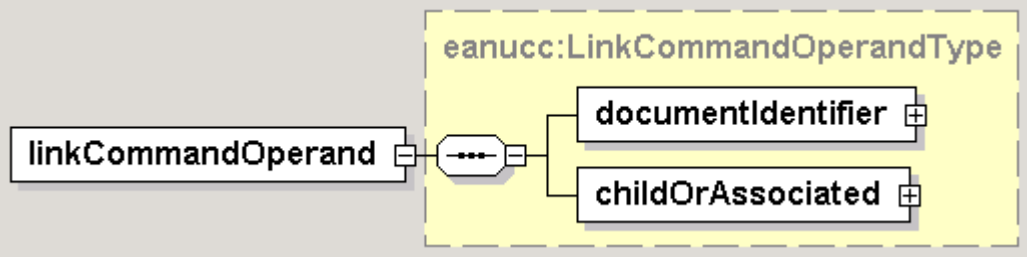


The Link Command Header structure

The header contains also the unique command identifier assigned by its creator, combined with the creator's identification (GLN and optionally, an additional one).

10.2.3.2 Link Operand

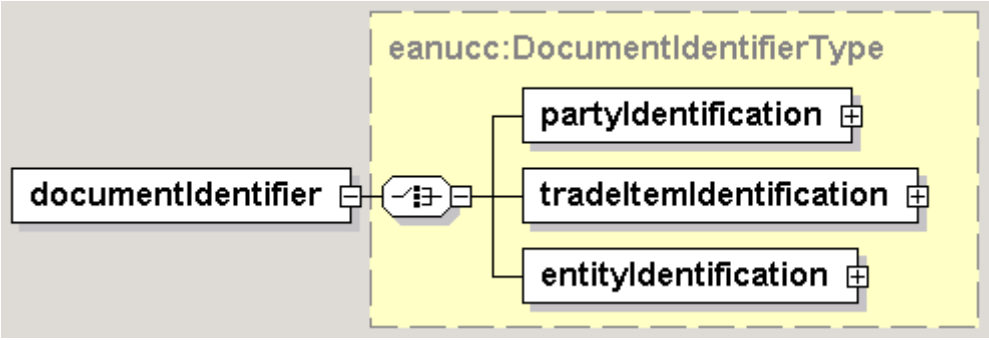
The 'linkCommandOperand' specifies the parent and children, identified by a document identifier.



The Link Command Operand structure

The term document here can be identified in three ways:

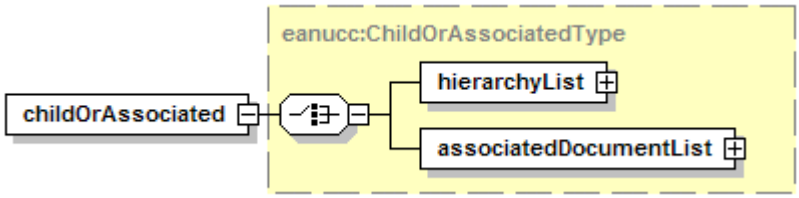
- by Entity Identification
- by Party Identification
- by Trade Item Identification



The Document Identifier structure

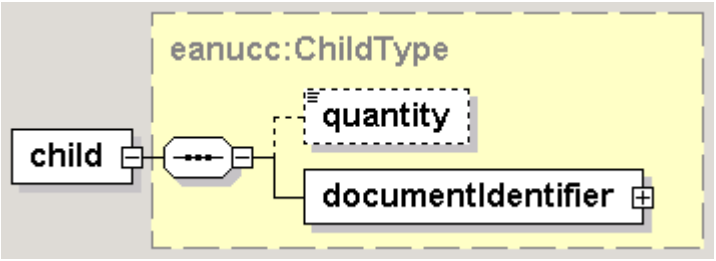
The first document identifier occurring in the sequence of Link Command Operand sub elements, defines the parent object.

The children objects can be identified using either a Hierarchy List or Associated Document List method.



The child or associated element structure

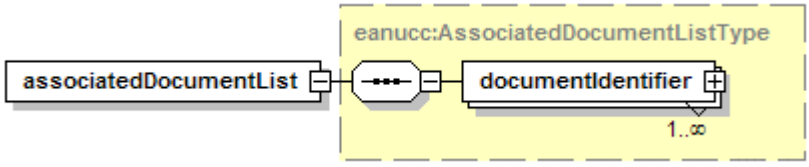
The 'hierarchyList' element allows to make a list of child documents, with just one document per child. By using this method it is also possible to specify the quantity of each child.



The child element structure

Therefore, the Hierarchy List can be used to specify the number of lower items in a packaging unit and the content of mixed containers.

The 'associatedDocumentList' allows to link some documents to a given parent.



The associated document list element structure

Within both parent and child elements, there is a choice of the components

that are meant to be linked or unlinked. It can be ‘partyIdentification’, defining the trading partner that is to be linked to given information. The second option is the ‘tradeItemIdentification’, defining the product to be linked. The last possibility is to specify the document, using the ‘entityIdentification’.

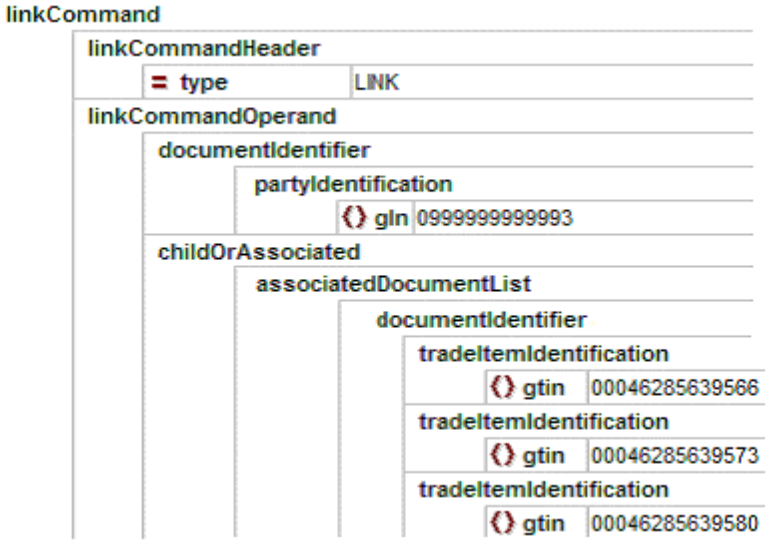
10.2.3.3 Link Command functionality

As mentioned before, the ‘LinkCommand’ purpose is to create a parent – child relationship between parties, trade items and other entities, e.g. documents. The parent and children are identified by the ‘partyIdentification’, ‘tradeItemIdentification’, or ‘entityIdentification’, depending on the nature of the linked entities. Although they are all referred to as the ‘documentIdentifier’, their use is not restricted to documents.

Example 1

A store may receive various items via different distribution centres. In order to specify where the goods are being delivered, the supplier can use the LinkCommand to link products identified by ‘tradeItemIdentification’ to a specific location identified by ‘partyIdentification’.

The purpose of this command is to make a link between components, therefore the value of the attribute of the ‘linkCommandHeader’ element is LINK. The first occurrence of the ‘documentIdentifier’ element within the ‘linkCommandOperand’ identifies the parent of the linked components, so the ‘partyIdentification’ element has as a value the GLN of the given distribution centre. The child components are the GTINs of the given items. Since the quantity of the items is not important in this case, the items are specified in the ‘associatedDocumentList’.



The instance of the Link Command establishing a relationship between a location and trade items

Note that the actual business message, e.g. Despatch Advice, is not attached to this command. The link created is independent of any changes of business

documents. When the relationship is no longer valid, the same command with the attribute UNLINK can be sent to announce this fact.

11 DOCUMENT LAYER

Document layer contains the actual business documents. This layer is fully documented in EAN.UCC Business Message Standards.

11.1 PROXY FILES

Functionally, a business document is a part of the messaging architecture, which requires the presence of other layers – the Standard Business Document Header, transaction and command.

Some parsers (e.g. Xerces versions earlier than 2.0) do not support multiple schema locations in the instance file. In order to facilitate the process of creating valid XML messages, integrated into the whole architecture, the proxy schemas have been created for each business document that include and import all the required files. The proxy schemas are non-normative; they are created only to support implementation of the EAN.UCC XML standards. An example of a proxy schema file for the Request For Payment message follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:ean.ucc:2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:import namespace="urn:ean.ucc:pay:2"
schemaLocation="RequestForPayment.xsd"/>
<xsd:include schemaLocation="DocumentCommand.xsd"/>
<xsd:include schemaLocation="AllowanceCharge.xsd"/>
<xsd:include schemaLocation="PaymentTerms.xsd"/>
<xsd:include schemaLocation="TradeItemIdentification.xsd"/>
<xsd:include schemaLocation="Transaction.xsd"/>
</xsd:schema>
```

In the instance document these multiple file names must be replaced with the single proxy file name. If this file is specified using the 'xsi:schemaLocation' attribute, the parser is able to validate the XML instance document. An excerpt of a sample xml instance file referring to a proxy schema follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a sample file-->
<sh:StandardBusinessDocument
xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader
"xmlns:eanucc="urn:ean.ucc:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.unece.org/cefact/namespaces/
StandardBusinessDocumentHeader StandardBusinessDocumentHeader.xsd
urn:ean.ucc:2 RequestForPaymentProxy.xsd" xmlns:pay="urn:ean.ucc:pay:2">
```

11.2 BUSINESS DOCUMENTS

The business documents are organised by their respective business processes. Each business document requires all the included and imported files to be present for the successful validation. Each Implementers Packet for every message contains a text file called Table of Content, listing all the files

that have been used to create a given business message schema and are necessary for the business document validation.

11.3 COMMON LIBRARY

In design of each business document, some components defined in the common library are used. The common library contains all the files that may be used in more than one business document and more than one context.

The components that are used only within one context, are defined within the namespace of that particular context. Finally, components that are not used in other messages are defined locally within the business message schema.

This approach allows reusing the same information constructs in all business messages, increases interoperability and simplifies schema maintenance.

In previous EAN.UCC XML standards releases (1.1, 1.3 and 1.3.1), the common library components had no separate namespace. They took on the namespace of the schema documents in which they were used. This means that a change to the common library affected not just the rest of the common library, but also the business documents.

Common library schema documents follow the same versioning strategy as all other schema documents. Starting from version 2.0, the common library files have their own target namespace: xmlns:eanucc="urn:ean.ucc:2". This namespace includes only the major version but not the minor one, and only backward compatible changes are made in minor versions. Like other schema documents, they use the version attribute on the xsd:schema element to indicate the minor version number.

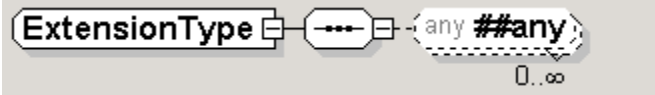
Common library schema documents have namespaces separate from the business documents that use them, so that they can be versioned separately. Therefore, a change in any of those components is reflected only in the version of those documents where this component is used, not in the whole release.

However, when the common library changes to a new major version, all business documents that use the library change to the new version as well.

12 EXTENSION MECHANISM

12.1 GENERAL EXTENSION COMPONENT

Beginning from release 2.0, every business document schema contains a placeholder where some additional, context-specific components can be inserted. This placeholder has a form of the XML schema '##any' element.



The structure of the Extension component

The XML 'any' element has a built-in attribute: processContents. It acts as an indicator of how an application or XML processor should handle the validation of XML documents against the elements specified by the 'any' element. In the EAN.UCC extension component, the value of this attribute is set to 'lax', it means that the XML processor attempts to obtain the schema for the required namespaces and validate any element from those namespaces, but if the schema cannot be obtained, no errors will occur.

12.1.1 VALIDATION OF THE EXTENSION COMPONENTS

Although the XML syntax rules for the 'any' element allow any well-formed XML to be placed within the instance file, the users must remember that their business partners will not be able to validate the non-standard information. The message recipient must be provided with the relevant schema validating the extended part of the message. It is recommended to use only the extension attributes approved within the GSMP process. Otherwise, the messages will not be standard and globally interoperable.

However, because the processContents attribute is set to 'lax', the validation of the extension element is optional: an XML schema parser will validate elements for which it can find declarations (extension schemas) and raise errors if they are invalid. If it does not find context specific schemas for a certain set of context specific elements, the parser will not report errors for those elements. The placeholder is an optional element of the root element.

12.1.2 QUALIFICATION OF THE EXTENSION COMPONENTS

All the components placed within the extension element are specific to certain context, thus they belong to the context-specific namespace, which has its own namespace prefix assigned. Those context-specific extension schema modules should have the 'elementFormDefault' and 'attributeFormDefault' attributes set to qualified. This way they will be prefixed and easily recognisable in the instance document.

13 XML TOOLS

There is a large choice of tools available on the market: from the simple, web based and free validators, through Integrated Development Environments, combining in one package the validator, parser, documentation generator, repository interface, scripting, transformation of XML documents to other formats and many other features.

Obviously, the more sophisticated products tend to be more expensive and the beginning user may either not need all the above mentioned features or does not know from the start which functionalities could be useful and worth paying for. There is a wide range of free of charge software tools for XML processing, which may be helpful in building the understanding of XML standards and beginning their use. A quite comprehensive list can be found at

the following links: http://www.garshol.priv.no/download/xmltools/ or the list maintained by W3C: http://www.w3.org/XML/Schema#Tools.

For searching the internet for the available XML software, the following key words may be helpful:

- 'XML tools'
- 'XML parser'
- 'XML validator'
- 'XML integrated development environment'
- 'XSLT editor'
- 'XSLT generator'
- 'XSLT engine'

There is also a special tool available, helping the business experts and analysts, developers and implementers to understand, visualise and analyse the structure of the EAN.UCC Data Models and XML Artefacts: EAN.UCC Schemas Reader - specialized standards and data models reader. Link: http://www.gefeg.com/en/edifix/reader-eanucc.html

For companies that need to decide which tools they should choose to be able to implement the EAN.UCC standards, the Checklist for Solution Providers should be helpful.

The EAN.UCC schemas and sample files are tested using the latest versions of the following validators:

- XML Schema Validator (XSV)
- XMLSpy (version 4.4 or higher) from Altova
- Xerces (version 2.6.2) from Apache

14 CHECKLIST FOR SOLUTION PROVIDERS

Before purchasing a product to support EAN.UCC messages, users should make sure that the standard is sufficiently supported. The questions presented below should help in this evaluation.

1. Does your product support EAN.UCC XML schemas?

2. What version or versions of EAN.UCC XML schemas does your product support?

3. Once EAN.UCC releases a new version of the standard, how quickly does your product support this new version?

4. How does your product handle multiple versions of EAN.UCC standards?

5. Does your company participate in the EAN.UCC Global Standards Management Process (GSMP)? If so, in what capacity?

6. Explain how your product imports EAN.UCC schema.

7. Can your product generate XML instance documents based on the imported schema?

8. What different ways does your product have to display EAN.UCC schema and XML?

9. Does your product support editing EAN.UCC XML messages, validation of messages, or both editing and validation?

10. How does your editing/validation tool work with EAN.UCC XML messages to ensure data conforms to the standard?

11. When XML is being validated against EAN.UCC schema, how efficient is this process?

12. Can validation be enabled/disabled by trading partner and/or message? How is this done?

13. How easily does your product map from EAN.UCC messages to other formats (e.g., user-defined record layouts, other XML formats)? Please explain how your mapping tool works.

14. Do you have clients who use your product to exchange and translate EAN.UCC messages? How many and with which messages?

15. If you have clients using your product with EAN.UCC messages, please provide a client list including contact references.

16. Please explain your product support structure. Is there support staff knowledgeable in EAN.UCC messaging?

17. Does your product allow for dynamically validating XML payload using EAN.UCC schemas residing outside the firewall?

18. Does your product support the W3C Recommendations for XML Schemas?

19. Does your product support <<include>> of schemas from multiple locations and <<import>> from multiple namespaces?

APPENDIX: MAJOR CHANGES OF DESIGN BETWEEN RELEASE 1.3.1 AND 2.0

1. Use of namespaces:
    a. Separate namespaces for different context
    b. Major versions reflected in the namespace

2. New versioning strategy
    a. Backward compatible changes reflected in minor versions
    b. Not compatible changes reflected in major versions

3. Envelope layer replaced by the Standard Business Document Header

4. New code lists management
    a. Each code list defined in the separate schema
    b. Each code list versioned separately (life cycle separation)
    c. Adding new codes does not require upgrade of the whole message suite

5. New extension mechanism
    a. New extension component introduced in all business documents schemas
    b. xsi:type no longer used

6. New way of managing the Common Library – each common component schema is independent of changes in the rest of the release

7. GTIN and GLN are used as mandatory primary identifiers for Trade Item and Party identification respectively.