GS1 XML Release 1.31

Technical User Guide

Issue 1, Nov-2003

Document Summary

Document Item	Current Value
Document Title	GS1 XML Release 1.31 Technical User Guide
Date Last Modified	Nov-2003
Current Document Issue	Issue 1
Status	Approved
Document Description (one sentence summary)	This document provides users with the technical guidelines to the structure and design of the GS1 XML standard release 1.31

Disclaimer

Table of Contents

1. What's In This Release?

GS1 XML Standards are an organized suite of XML Schema Modules. The schemas are organized into three distinct layers, which perform specific functions. These layers are in turn supported by the reusable components defined in the common library.

- **The AS2 message transport and routing protocol**: This layer provides the message header, which contains the information for routing and transporting only.
- **The GS1 Message Architecture**: The message architecture layer combines the business documents in one message transaction, allowing the sender to package one or more documents in a transmission. This reduces the transmission and handshaking costs incurred if one were to send a document at a time. It also supports the increased flexibility of sending one or many documents in one transmission, allowing the sender to mix and match the set of documents sent according to the individual transmission needs.
- **The Business Documents**
- **A Common Component Library**: The common components are of 2 types:
  - The basic building blocks that provide the reusable common components (i.e. date, Quantity, MeasurementType).
  - The reusable components built on top of the basic building blocks that define the lowest common denominator that is required in a core business document

2. Implementation Details

The World Wide Web Consortium (W3C) released XML Schema as a W3C Recommendation in three parts on May 2, 2001:

- Part 0 Primer (http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/)
- Part 1: Structures (http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/)
- Part 2: Datatypes (http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/)

The schemas conform to and are implemented using this Recommendation. They have been implemented and tested using the following software:

- XML Schema Validator (XSV) version: XSV 1.203.2.47.2.4.2.8/1.106.2.25.2.2 of 2002/05/25
- XML Schema Validator from University of Edinburgh/W3C. www.ltg.ed.ac.xsv-status.html
- XMLSpy version 4.4 from Altova. www.xmlspy.com
- Xerces (exposes Java API), version 1.4.4, from Apache. xml.apache.org

3. Namespaces

All files are in the **eanucc** namespace except for extensions, which are in specific extension namespaces.

Namespaces allow extensions to occur upon data structures without naming collisions. Collisions would otherwise occur when one business document had to include multiple extensions (i.e. fmcg extensions may be incorporated in one business document).

4. The Schema Organization

The overall architecture of the GS1 XML schemas is divided into three layers, performing different functions. The layers are: Envelope, Message and Document. Every layer is wrapped inside the other, all of them combined together form an XML message.

Each layer can be created separately from the two others, if needed even by various parties. For example the Envelope layer can be created by the communication service provider, the Message layer by the functionality service provider (e.g. an exchange), and the Document layer is created by the business partner.

The layers can also be replaced with other messaging systems, without affecting the lower layers.

The structure and functions of each layer are described below.

4.1. Envelope layer

The Envelope layer provides essential transport information, assuring that a message will be delivered from a sender to receiver. Besides, it identifies the message set that is sent together in one envelope as well as the version used to create the content.

GS1 does not specify any standard for transport of XML messages. Instead, it allows for using other standards defined by other organisations, e.g. W3C or ebXML. As soon as other bodies develop their standards, they will be supported by GS1, upon their evaluation.

The currently recommended and supported standard is AS2 Envelope, based on IETF AS2 Specification of the Internet Engineering Task Force. It is an extended MIME structure, allowing for a Secure MIME. The S/MIME allows encoding and decoding messages for security purposes.

The transport protocol used by AS2 standard is HyperText Transfer Protocol.

Table 4-1 Envelope Elements

Elements & subelements	Attributes	Values
1 envelope	communicationVersion	Default value: 1.3.1
1.1 messageHeader	creationDate	
1.1.1 userId		
1.1.2 password		
1.1.3 messageIdentifier		
1.1.4 to		
1.1.5 from		
1.1.6 representingParty		
1.2 body		
1.2.1 ##any		

The **envelope** element contains an attribute specifying the communication version. It describes the schema version used to create the XML content.

Whenever the GS1 schemas are updated to the later version, this number is updated accordingly.

The **Envelope.xsd** schema contains message header and body. The header defines all the message address information and body constitutes a container where the actual business document can be attached.

4.1.1. Envelope Header

The header contains elements and attributes needed to support the AS2 protocol, i.e. how, from where and to where the message is routed. The information contained in the message header is mapped and transferred to the respective HTTP fields. It allows a transportation service to identify where and how to send a particular message.

4.1.1.1. Address information

The address information include tags ‘to’ and ‘from’, where users can decide whether to use a Global Location Number or other, ‘alternate’ party identifier. It is also possible to use both, thus conforming to specific requirements of certain industries.

Another address component is a ‘representingParty’ element. It allows to identify the actual trading partner, on behalf of which the message is being sent. Its value can be different than the one in the ‘from’ element. In a B2B environment, the business messages can be sent by service providers such as exchanges or agents. In that case, the ‘from’ element identifies the actual message sender – an intermediary party, whereas the ‘representingParty’ identifies the real trading partner. It can be also used if a ‘parent’ organisation sends a message on behalf of one of its subsidiaries.

If the sending party, however, acts only on its own behalf, then the values in ‘from’ and ‘representingParty’ elements will be the same. As the element ‘from’ is an optional one, it can be skipped then, in order to avoid redundancy, and the sender’s information will be contained in the ‘representingParty’ element.

4.1.1.2. Sender’s authentication

The sender of the message can be authenticated by the ‘userId’, indicating the individual sender or the machine from which the message was sent. The ID should comply with the ‘login’ ID assigned for that user in the communication application.

The additional security measure is an optional ‘password’. The password is meant to authenticate the User ID. If the EDIINT transport is used, the digital signature provides the necessary authentication and the ‘password’ element can be skipped.

4.1.1.3. Message identification

Each message sent is assigned an identifier which should be unique for a given sender. The messages with duplicated identifiers are not valid.

An additional information about the message is the ‘creationDate’, being an attribute of the ‘messageHeader’ element. It is a mandatory information, indicating the date and time when the message was created.

4.1.2. Envelope Body

The body of the envelope is a tag allowing to insert a message payload, consisting of any number of any kind of GS1 messages or transactions.

The body connects information contained in the envelope header with business information. It provides a way to separate the payload from the transport and routing information contained in the envelope header.

4.2. 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 to reduce the number of messages needed to perform an efficient exchange of business messages. The same message can be used with different commands. Hence, no separate messages like 'Add Item', 'Change Item' or 'Delete Item' are needed. The same 'Item' message can be send with the 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, will be 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 and an interface to the Document Layer.

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

4.2.1. Transaction

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.

Table 4-2 Transaction Elements

Elements & Subelements	Attributes	Values
1 entityIdentification		
2 command		

The Transaction schema contains two main components: identification of the parties that created the set of commands and the one that is the owner of the message content. The second component are the commands themselves.

4.2.1.1. Identification of the Transaction

The Transaction is identified by the combination of two identifier elements. The first one is the **uniqueCreatorIdentification**, it is an identifier assigned by the party creating the transaction. The second identifier, the **contentOwner**, is the one specifying the assigning party itself.

Both those identifiers are mandatory, to make the Transaction identification really unique.

The Content Owner can be identified either by a GLN or alternate identifier that value can be specified as one of the **AlternatePartyIdentificationListType**.

4.2.1.2. Command Element

The **command** element is a container, where one or multiple commands that form one transaction can be inserted.

4.2.2. Commands

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** and **linkCommand**, defined in DocumentCommand.xsd, **DocumentIdentificationCommand.xsd** and **LinkCommand.xsd**, respectively.

4.2.2.1. 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. The actions, which can be performed on it include **Add**, **Refresh** and **Delete**.

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

Table 4-3 DocumentCommand Elements

Elements & Subelements	Attributes	Values
1 documentCommandHeader	Type	Enumeration Values: ▪ ADD ▪ CHANGE_BY_REFRESH ▪ DELETE
1.1 entityIdentification		
2 documentCommandOperand		

The **documentCommandHeader** specifies the action that should be performed on the given document. Those tasks, defined as the required attribute of the header element, are listed in Table 4-4:

Table 4-4 Tasks

Task	Definition
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
DELETE	The receiving application is instructed to delete the document or documents

The **documentCommandOperand** contains an abstract element that can be extended by the actual document elements, defined in the actual business message schemas. Those are the documents against which one of the actions defined in the header element should be performed. The list of the possible documents includes:

- Align messages
  - TradeItem
  - Price
  - PriceBracket
  - Party
- Catalogue Item Synchronisation messages
  - Catalogue Item Confirmation
  - Catalogue Item Link
  - Catalogue Item Notification

- Catalogue Item Publication
- Catalogue Item Registration Response
- Catalogue Item Subscription
- Data Synchronisation Data Pool Profile
- Registry Catalogue Item
- Request For Catalogue Item Notification

- Party Synchronisation messages
  - PartyConfirmation
  - PartyNotification
  - PartyPublication
  - PartyRegistration
  - PartySubscription
  - RequestForPartyNotification
- Order message
  - Order
- Deliver messages
  - BookingRequest
  - BookingRequestAcknowledgement
  - DespatchAdvice
  - FullTruckloadLoadTender
  - InventoryStatusAdvice
  - PickupManifest
  - PickupNotificationLTL
  - ReceivingAdvice
  - WarehouseShippingOrder
- Pay messages
  - ControlTotal
  - DebitCreditAdvice
  - RequestForPayment
  - Settlement
  - SimpleInvoice
- Plan messages
  - Event
  - ExceptionCriteria
  - ExceptionNotification
  - Forecast
  - ForecastBulkData

- ForecastRevision
- PerformanceHistory
- ProductActivity
- ProductActivityBulkData
- RetailEvent
- TradeItemInformationRequest
- TradeItemLocationProfile

Those documents extend the abstract 'document' element from the DocumentCommand.xsd

4.2.2.2. 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.

Table 4-5 DocumentIdentificationCommand Elements

Elements & subelements	Attributes	Values
1 documentIdentificationCommand	type	Enumeration value: DELETE
1.1 documentIdentifierList		
1.1.1 documentIdentifier	contentVersion	
1.1.1.1 partyIdentification		
1.1.1.2 tradeItemIdentification		
1.1.1.3 typedEntityIdentification		

The document on which the action should be performed is identified in the **documentIdentifierList** element. It can be specified in three ways:

1. **partyIdentification**: Using GLN or an alternate identifier of the concerned party.
2. **tradeItemIdentification**: Using GTIN or an alternate identifier of the trade item concerned by the document in question.
3. **entityIdentification**: Being the combination of the unique document identifier allocated by its creator and the unique identification of that document creator (GLN or the alternate one).

The documents that can be used with this command are specified as attribute values of the **typedEntityIdentification** element (see Section 4.4.1, Common Identifiers).

4.2.2.3. Link Command

This type of command allows the establishment of parent-child or peer relationships between several documents.

Table 4-6 LinkCommand Elements

Elements & Subelements	Attributes	Values
1 linkCommandHeader	type	Enumeration values: ▪ LINK ▪ UNLINK
1.1 entityIdentification		
2 linkCommandOperand		
2.1 documentIdentifier		
2.1.1 partyIdentification		
2.1.2 tradeItemIdentification		
2.1.3 typedEntityIdentification		
2.2 childOrAssociated		
2.2.1 hierarchyList		
2.2.1.1 child		
2.2.1.1.1 quantity		Default = 1
2.2.1.1.2 documentIdentifier	contentVersion	
2.2.1.1.2.1 partyIdentification		
2.2.1.1.2.2 tradeItemIdentification		
2.2.1.1.2.3 typedEntityIdentification		
2.2.2 associatedDocumentList		
2.2.2.1 documentIdentifier		
2.2.2.1.1 partyIdentification		
2.2.2.1.2 tradeItemIdentification		
2.2.2.1.3 typedEntityIdentification		

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

4.2.2.4. 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 of the element's attribute: **LinkCommandListType**.

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

4.2.2.5. Link Operand

The **linkCommandOperand** specifies the parent and children, identified by an entity identifier. The entity can be either a document - identified by a **typedEntityIdentification**, a party - identified by GLN or one of the alternate identifiers, or trade item - identified by GTIN or any alternate identifier.

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

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

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

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 a given information. The second option is the **tradeItemIdentification**, defining the product to be linked. The last possibility is to specify the document, using the **typedEntityIdentification**. It defines the type of the document, by the **entityTypeList** attribute, containing the enumeration list of three possible document types (see Section 4.4.1, Common Identifiers)

This choice provides also the unique identifier of the given document, by the combination of a unique identifier assigned by the creator of the command and identifier of that party itself.

4.3. Document Layer

Document layer contains the actual business documents. This layer is documented in GS1 Business Message Standards.

4.3.1. Proxy Files

Functionally a business document is a part of the messaging architecture, which requires the presence of other layers – an envelope, transaction and command.

Some parsers (e.g. Xerces) do not support multiple schema locations in the instance file in the current version. 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 includes and imports all the required files. Below is a sample proxy schema file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ean-ucc.org/schemas/1.3.1/eanucc"
xmlns="http://www.ean-ucc.org/schemas/1.3.1/eanucc" xmlns:eanucc="http://www.ean-
ucc.org/schemas/1.3.1/eanucc" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:include schemaLocation="As2Envelope.xsd"/>
    <xsd:include schemaLocation="DocumentCommand.xsd"/>
    <xsd:include schemaLocation="Transaction.xsd"/>
    <xsd:include schemaLocation="Event.xsd"/>
    <xsd:import namespace="http://www.ean-ucc.org/schemas/1.3.1/fmcg"
schemaLocation="FmcgIdentification.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 and even track errors in schema. An excerpt of a sample xml instance file referring to the above proxy schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a sample file-->
<eanucc:envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:eanucc="http://www.ean-ucc.org/schemas/1.3.1/eanucc" xmlns:fmcg="http://www.ean-
ucc.org/schemas/1.3.1/fmcg" xsi:schemaLocation="http://www.ean-
ucc.org/schemas/1.3.1/eanucc EventProxy.xsd" communicationVersion="1.3.1">
```

4.3.2. Extensions

The GS1 XML standards contain two extensions for FMCG trade item. One of them provides additional identification – **FmcgIdentification.xsd**, the second one contains additional information about trade item that are specific for FMCG industry - **FmcgItem.xsd**. The final user can decide whether to use the 'general' trade item information or the FMCG one. This can be achieved using the so called late binding mechanism in the XML file. The original **AlternateTradeItemIdentificationType**, **TradeItemMarkingType** or **TradeItemMeasurementsType** can be extended by the respective FMCG type, using the **xsi:type** attribute of the relevant element in the XML instance file – i.e. in the actual business message.

4.3.3. Business Documents

The business documents are organised by their respective business processes. Each business document requires all the included files to be present for the successful validation.

4.3.3.1. ALIGN Business Process:

TradeItem.xsd

Includes: TradeItemComponents.xsd

Price.xsd

Includes: AlignComponentLibrary.xsd

PriceBracket.xsd

Includes: AlignComponentLibrary.xsd

Party.xsd

Includes: AlignComponentLibrary.xsd

4.3.3.1.1. Common Files for ALIGN:

AlignComponentLibrary.xsd

Includes: PriceComponents.xsd
TradeItemComponents.xsd
PartyComponents.xsd

PriceComponents.xsd

Includes: Terms.xsd

TradeItemComponents.xsd

Includes: Terms.xsd

PartyComponents.xsd

Includes: Terms.xsd
PayComponentLibrary.xsd

4.3.3.2. Catalogue ITEM Synchronisation

CatalogueItemConfirmation.xsd

Includes: CatalogueItemComponents.xsd

CatalogueItemLink.xsd

Includes: CatalogueItemComponents.xsd

CatalogueItemNotification.xsd

Includes: CatalogueItemComponents.xsd
Terms.xsd

CatalogueItemPublication.xsd

Includes: CatalogueItemComponents.xsd

CatalogueItemRegistrationResponse.xsd

Includes: CatalogueItemComponents.xsd

CatalogueItemSubscription.xsd

Includes: CatalogueItemComponents.xsd

DataSynchronisationDataPoolProfile.xsd

Includes: CatalogueItemComponents.xsd

DataSynchronisationError.xsd

Includes: CatalogueItemComponents.xsd

RegistryCatalogueItem.xsd

Includes: CatalogueItemComponents.xsd

RequestForCatalogueItemNotification.xsd

Includes: CatalogueItemSubscription.xsd

4.3.3.3. Common Files for Catalogue ITEM Synchronisation:

CatalogueItemComponents.xsd

Includes: Terms.xsd

4.3.3.4. PARTY Synchronisation

PartyConfirmation.xsd

Includes: PartySynchronisationComponents.xsd

PartyNotification.xsd

Includes: PartySynchronisationComponents.xsd
Party.xsd

PartyPublication.xsd

Includes: Terms.xsd

PartyRegistration.xsd

Includes: PartySynchronisationComponents.xsd
Party.xsd

PartyRegistrationResponse.xsd

Includes: Terms.xsd

PartySubscription.xsd

Includes: PartySynchronisationComponents.xsd

RequestForPartyNotification.xsd

Includes: PartySubscription.xsd

4.3.3.4.1. Common Files for PARTY Synchronisation:

PartySynchronisationComponents.xsd

Includes: PartyComponents.xsd
CatalogueItemComponents.xsd

4.3.3.5. ORDER Business Process:

Order.xsd

Includes: OrderComponentLibrary.xsd

4.3.3.5.1. Common Files for ORDER:

OrderComponentLibrary.xsd

Includes: Terms.xsd

4.3.3.6. DELIVER business process:

BookingRequest.xsd

Includes: DeliverComponentLibrary.xsd

BookingRequestAcknowledgement.xsd

Includes: DeliverComponentLibrary.xsd

DespatchAdvice.xsd

Includes: DeliverComponentLibrary.xsd

FullTruckloadLoadTender.xsd

Includes: DeliverComponentLibrary.xsd

InventoryStatusAdvice.xsd

Includes: DeliverComponentLibrary.xsd

PickupManifest.xsd

Includes: DeliverComponentLibrary.xsd

PickupNotificationLTL.xsd

Includes: DeliverComponentLibrary.xsd
TradeItemComponents.xsd

ReceivingAdvice.xsd

Includes: DeliverComponentLibrary.xsd

WarehouseShippingOrder.xsd

Includes: DeliverComponentLibrary.xsd

4.3.3.6.1. Common Files for DELIVER:

OrderComponentLibrary.xsd

Includes: Terms.xsd

4.3.3.7. PAY business process:

ControlTotal.xsd

Includes: PayComponentLibrary.xsd

DebitCreditAdvice.xsd

Includes: PayComponentLibrary.xsd

RequestForPayment.xsd

Includes: PayComponentLibrary.xsd

Settlement.xsd

Includes: PayComponentLibrary.xsd

SimpleInvoice.xsd

Includes: PayComponentLibrary.xsd
RequestForPayment.xsd

4.3.3.7.1. Common Files for PAY:

PayComponentLibrary.xsd

Includes: Terms.xsd

4.3.3.8. PLAN business process:

Event.xsd

Includes: PlanComponentLibrary.xsd

ExcieptionCriteria.xsd

Includes: PlanComponentLibrary.xsd

Forecast.xsd

Includes: PlanComponentLibrary.xsd

ForecastBulkData.xsd

Includes: PlanComponentLibrary.xsd

ForecastRevision.xsd

Includes: PlanComponentLibrary.xsd

PerformanceHistory.xsd

Includes: PlanComponentLibrary.xsd

ProductActivity.xsd

Includes: PlanComponentLibrary.xsd

ProductActivityBulkData.xsd

Includes: PlanComponentLibrary.xsd

RetailEvent.xsd

Includes: PlanComponentLibrary.xsd

TradeItemInformationRequest.xsd

Includes: PlanComponentLibrary.xsd

TradeItemInformationLocationProfile.xsd

Includes: PlanComponentLibrary.xsd

4.3.3.8.1. Common Files for Plan:

PlanComponentLibrary.xsd

Includes: Terms.xsd

4.4. Common Library

Each document is constructed using components defined in the Common Library. Locally defined are only components not used in other messages. This approach allows to reuse the same information constructs in all business messages.

The use of the most common identification components is explained below. The meaning and usage other Common Library files is described in details in the respective Business Message Standards

documents. They are not covered in this document, since they do not provide any additional technical functionalities, but have strictly business function.

4.4.1. Common Identifiers

The identifiers used throughout all the schemas are defined in **Identification.xsd**. They are:

- Party Identification
- Trade Item Identification
- Entity Identification
- Typed Entity Identification
- Document Identification

4.4.1.1. Party Identification

A party can be identified either just by a primary identifier – GLN (the recommended option), an alternate identifier, or by a combination of primary and additional identifiers. All identifiers must represent the same party. This must not be used to represent the link between different parties, for that purpose the **PartyContainment** mechanism has been developed, defined in **PartyComponents.xsd**.

Both alternate and additional identifiers must comply to one of the identifier types listed in the **AlternatePartyIdentificationListType**, which is a mandatory attribute of each of those identifier elements.

The **AlternatePartyIdentificationListType** contains the following values:

- **BUYER_ASSIGNED_IDENTIFIER_FOR_A_PARTY**: Party identifier assigned by the buyer
- **DEA_DRUG_ENFORCEMENT_AGENCY**
- **DUNS**: 9-digit number assigned by DUN & Bradstreet
- **DUNS_PLUS_FOUR**: 9-digit number consisting of 9-digit DUNS and 4-digit number assigned by the user
- **HIN_CANADIAN_HEALTHCARE_IDENTIFICATION_NUMBER**
- **SCAC (US Standard Carrier Alpha Code):** The organisation maintaining the SCAC lists and transportation operating in North America is the National Motor Freight Transportation Association (NMFTA).
- **SELLER_ASSIGNED_IDENTIFIER_FOR_A_PARTY**: Party identifier assigned by the seller
- **TD_LINK_TRADE_DIMENSIONS**
- **UCC_COMMUNICATION_IDENTIFICATION**
- **UN_LOCATION_CODE**: United Nations Location Code (UN/ECE Recommendation 20)

For each party only one primary identifier is allowed (the recommended choice is GLN), but it can be used with several additional alternate identifiers. If it is used as a primary party identification, it can be combined with DUNS used as an additional party identification. Otherwise, any alternate identifier can be selected as the primary party identification, with reference to additional party identification alternates. If DUNS number is used as a primary party identification, it can be cross-referenced to another alternate number such as a buyer assigned customer number. However, GLN may never be used as the additional alternate.

4.4.1.2. Trade ITEM Identification

A trade item can be identified either just by a primary identifier – GTIN (the recommended one) or an alternate one, or by a combination of a primary and additional identifiers.

The alternate code can be used to cross-reference the vendors internal trade item number to the GTIN in a one to one relationship. It should not be used to group trade item or define trade item variants.

The **AlternateTradeItemIdentificationListType** contains the following values:

- BUYER_ASSIGNED
- SUPPLIER_ASSIGNED
- INDUSTRY_ASSIGNED

Those values are used to specify the type of the alternate identifier.

4.4.1.3. Entity Identification

Entities can be documents, commands or transaction. The entity identifier allows uniquely identify each instance of an XML document, command or transaction. It consists of two parts:

- **uniqueCreatorIdentification** – an identifier assigned by the party creating the command or document; this identifier must be unique for that given party, if they duplicate at any point, the message will be discarded
- **contentOwner** – an identifier of the party creating the entity. It is of the Party Identification type, so it can be a GLN or an alternate ID (see 4.4.1.1 Party Identification above)

The combination of those two components assures a unique identification of each single entity.

4.4.1.4. Typed Entity Identification

In certain situations, only a reference, instead of the whole document is included in the message, for instance in other business document or in the command layer – in Document Identification Command (see 4.2.2 Command Layer). The referenced document in this case is identified by the Typed Entity Identification. In addition to the Entity Identification structure, it contains the **typedEntityIdentification** element, with an attribute **EntityTypeListType**. That attribute lists all the document types that can be referenced by this identifier:

- CATALOGUE_ITEM_CONFIRMATION
- CATALOGUE_ITEM_LINK
- CATALOGUE_ITEM_NOTIFICATION
- CATALOGUE_ITEM_PUBLICATION
- CATALOGUE_ITEM_REGISTRATION_ACKNOWLEDGEMENT
- CATALOGUE_ITEM_SUBSCRIPTION
- CONTROL_TOTAL
- DATA_SYNCHRONISATION_DATA_POOL_PROFILE
- DEBIT_CREDIT_ADVICE
- DESPATCH_ADVICE
- FULL_TRUCKLOAD_LOAD_TENDER
- INVENTORY_STATUS_ADVICE
- ORDER

- PARTY_DOCUMENT
- PICKUP_MANIFEST
- PICKUP_NOTIFICATION_LTL
- PRICE_BRACKET_DOCUMENT
- PRICE_DOCUMENT
- RECEIVING_ADVICE
- REGISTRY_CATALOGUE_ITEM
- REQUEST_FOR_CATALOGUE_ITEM_NOTIFICATION
- REQUEST_FOR_PAYMENT
- RESPONSE
- SETTLEMENT
- TRADE_ITEM_DOCUMENT
- WAREHOUSE_SHIPPING_ORDER

4.4.1.5. Document Identification

Document identification allows references to other documents to be made in three ways: by an identification of the document itself – as Typed Entity Identification, by Party Identification or Trade Item Identification.

In a business document exchange, there is sometimes a need to refer to some other document that had been previously sent. For example Despatch Advice referring to Order. Such functionality is provided by the Typed Entity Identification.

If a reference is necessary to be made to the Party or Item information sent in another document, it is possible via Party or Trade Item Identification respectively.

Note: For the Party structure, see Section 4.4.1.1, Party Identification

Note: For the Trade Item Identification structure see Section 4.4.1.2, Trade ITEM Identification.