GDSN 1.2

Operations Manual

Issue 2, Apr-2005

Document Summary

Document Item	Current Value
Document Title	GDSN 1.2 Operations Manual
Date Last Modified	Apr-2005
Current Document Issue	Issue 2
Status	Approved
Document Description (one sentence summary)	This document is the user operations manual for GDSN.

Log of Changes

Date of Change	Version	Reason for Change	Summary of Change	CCR #
January 27, 2005	1.1	Review Comment	Added section 2.2.3.1 with an example.	Does Not Apply
January 27, 2005	1.1	Review Comment	Rewrote Sections 2.2.2 and 2.2.5 to consistently show that the Delete Command acts on a pointer and not a document.	Does Not Apply
January 27, 2005	1.1	Review Comment	Deleted from Section 4.2.3 wording that indicated that the RFCIN generates a subscription in the GS1 GR.	Does Not Apply
January 27, 2005	1.1	Review Comment	Corrected Section 4.2.4 to state that updates happen only if there is a publication that matches the subscription.	Does Not Apply
April 27, 2005	1.2	To Incorporate Approved Standards Enhancements	Updated SBDH Chapter with Task Group approved content; added MessageID and Content Owner Clarifications	Does Not Apply

Disclaimer

Table of Contents

1. Introduction

The GDSN (Global Data Synchronization Network) Task Group Operations Manual is a guide for implementers of the Network. It is a ‘living document’ that will be periodically updated on an as needed basis. This implementer’s guide should be referenced along with the Business Requirements Analysis Documents (BRAD) and the Business Message Standards (BMS) defined by the GS1 Global Standards Management Process (GSMP). The BRADs and BMSs are the normative documents – they specify the standards; how to implement the standards is described in this operations manual. The authors and editors of the operations manual are responsible for ensuring that there are no discrepancies between the normative documents and this manual. Because the standards for GDSN are still being defined, standards gaps will arise between the documented standards and the learnings forged by implementation. These implementation learnings will be captured in the operations manual, being the reference point collecting all the decisions reached by the GDSN group before these make their way as standards in the BRAD or BMS.

When this occurs, and standards gaps arise between the BRAD and BMS and the Operations Manual, the GSMP Change Request System will be used to note those gaps. A Change Request will be submitted through the GSMP Change Request process, generating a tracking number which will be referenced in the manual and eventually in the BRAD and / or BMS.

The Operations Manual also describes how to use the current version of the GS1 XML. Because the Operations Manual is a living document, the portion of the manual relating to XML will be upgraded each time the XML version changes. The older versions will be dropped from the manual when they are no longer being used. To make these changes readily available, the manual will not be issued in printed form, but will be available for download from the GDS web site at GS1.org.

1.1. Purpose of This Document

The purpose of this document is to explain how to use the GS1 Standards in the Global Data Synchronisation Network (GDSN). It answers the questions – what do I need to do to build a STANDARDS COMPLIANT APPLICATION. Other GDSN documents provide an overview of GDSN or the business requirements behind GDSN and this document serves as an operations manual and an aide to the GDSN user and implementer community. It is supplemented by the GDSN business requirements documents and provides greater detail on how to implement GDSN. This document is intended to be a living document. It’s content will be expanded and will change as the needs of the users change. This document and updated versions of this document will be located and can be downloaded from the following site:

http://www.gs1.org/productssolutions/gdsn/implementation/index.html

1.2. Intended Audience

This document should be of use to implementers of the GS1 Global Registry, GS1 Member Organizations, GDSN Data Pools, and GDSN Trading Partners organizations. This document is intended for businesspersons and developers of GDSN who will implement GDSN from the operations guide. In most cases they will not have attended GDSN meetings. The information in this document is organized in such a way as to be meaningful to them. In addition, the document is intended for the participants in the standards development process for GDSN.

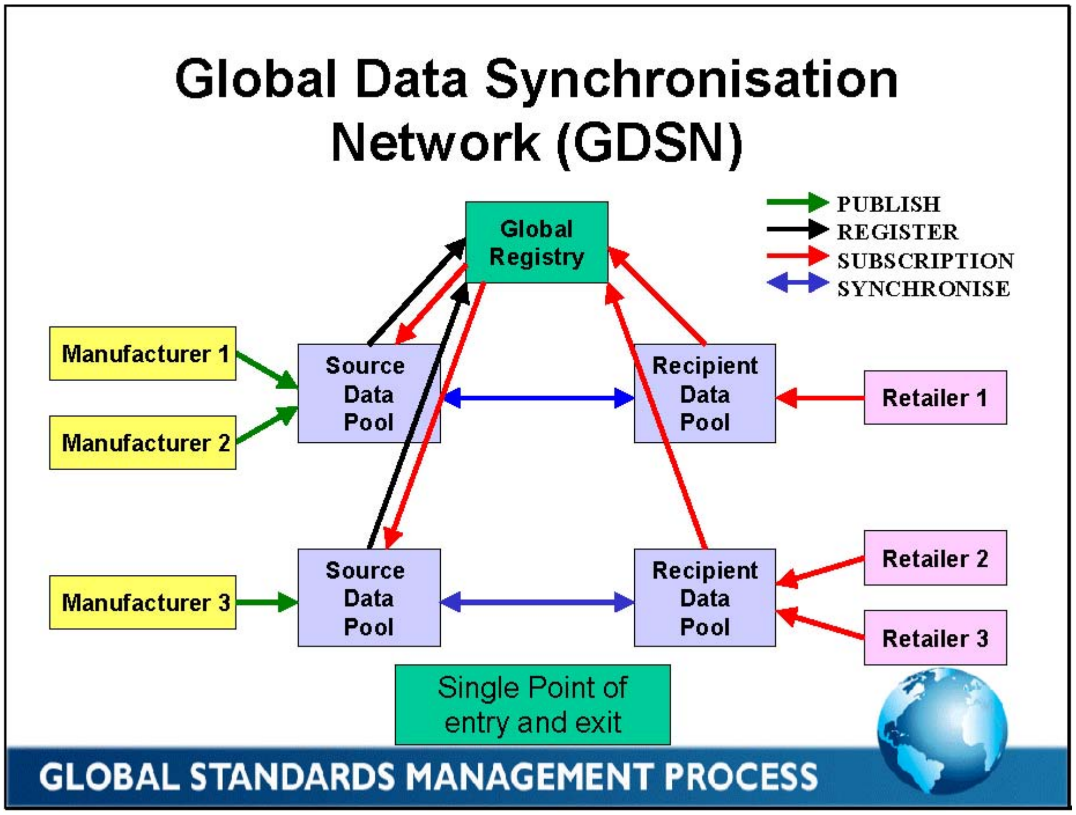
1.3. General Overview of GDSN

The GDSN is a concept illustrated in Figure 1, developed by GS1 and various industry groups, including the Global Commerce Initiative (GCI), to help industry streamline supply chain transactions and reduce supply chain costs. The GDSN is an Internet-based, interconnected network of interoperable data pools and a global registry, the GS1 Global Registry, that enables companies around the world to exchange standardised and synchronised supply chain data with their trading partners. The GDSN assures that data exchanged between trading partners is accurate and compliant with universally supported GS1 XML Standards. The GDSN consists of trading partners (supplier and retailers), data pools (services that hold and process trading partner data), and the GS1 Global Registry (a worldwide directory to help the GDSN community locate data sources and manage ongoing synchronisation relationships between trading partners).

Some key aspects of the GDSN are the following:

- The model supports a rich network of interoperable data pools
- Suppliers and Retailers have a single point of entry to the GDSN
- Suppliers and Retailers do not have direct access to the GS1 Global Registry, unless they act as their own data pools
- Catalogue Items are uniquely identified in the GDSN using the unique combination of Global Trade Item Number (GTIN), Global Location Number (GLN) of the data source, and Target Market (TM).
- Parties are identified in the GDSN using GLN.

Figure 1-1 Source GS1 Global Standard Management Process (GSMP)

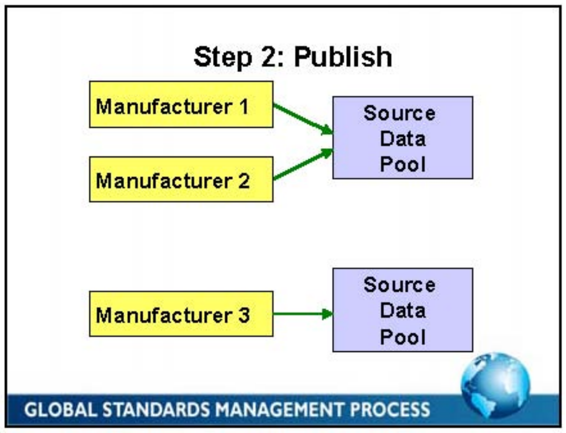


1.3.1. What is GDSN and How Does it Work?

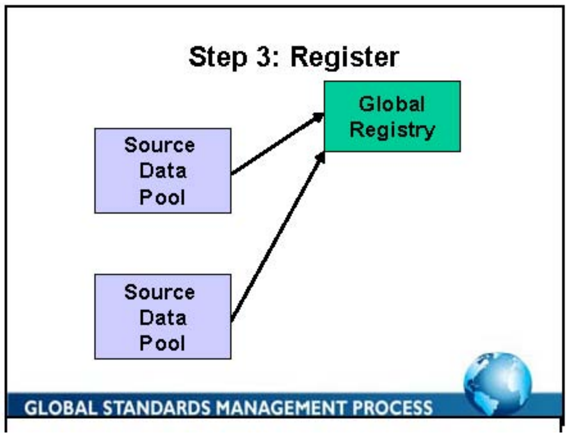
One of the key advantages of the GDSN is that trading partners have a single point of entry to the GDSN through the certified data pool of their choice therefore avoiding having to pay subscriptions to multiple data pools either within the same geographic location or across multiple geographic locations. Therefore, trading partner's access to the GS1 Global Registry is only available through a certified data pool. It has to be noted that trading partners can act as their own certified data pools.

Within the GDSN, Catalogue Items are identified using the GTINs and GLNs of the data source (supplier), and target market while Parties are identified using a GLN. Suppliers and retailers willing to synchronise item, location (party) and price data with each other should perform the 5 following basic steps detailed below:

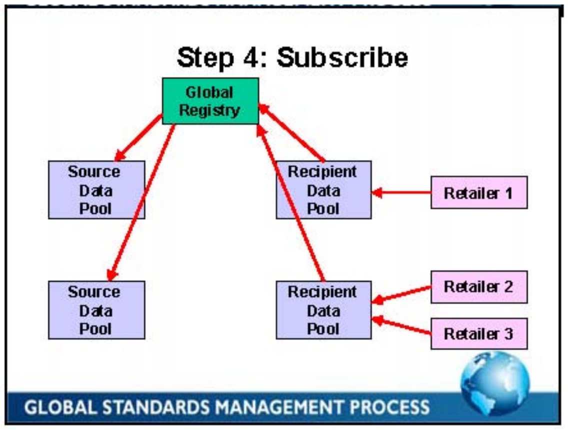
- **Step 1:** Suppliers prepare internal data and systems to match GS1 standards (GTIN, GLN, Global Product Classification - GPC, etc.)
- **Step 2:** Suppliers publish their accurate and standardised data to an GS1 certified source data pool of their choice (in-house or third party).



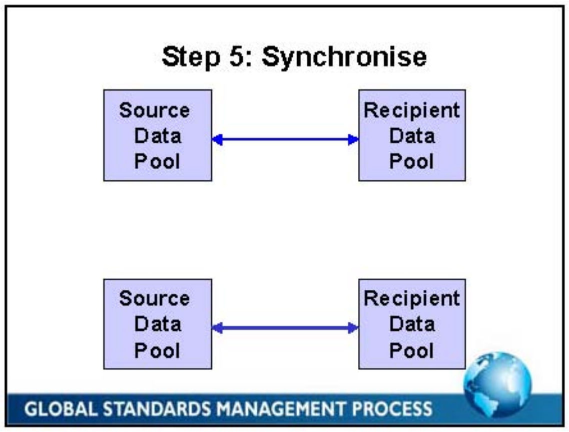
- **Step 3:** The data pool registers a small subset of the supplier's information about each item (or party) to the GS1 Global Registry. The GS1 Global Registry holds this information and the location of each item's (or party's) data pool



- **Step 4:** Retailers search the GS1 Global Registry via their chosen GS1 certified data pool for the item(s) or party for which they want to subscribe. Subscription can be done by any combination of GTIN, GLN, GPC and/or Target Market at the exception of GTIN and GPC that are mutually exclusive. The GS1Global Registry identifies the source data pool(s) of the requested item(s) or party(s).



- **Step 5:** Data pools process the exchange of information between the trading partners via their respective data pools. As product attributes change, suppliers immediately resend their updated information to their chosen data pool, which ensures that all retailers within the GDSN community who are subscribing to this information are notified of the updated information via their recipient data pools.



2. XML Schema Organization and Architectural Principles

2.1. What are GS1 Standards and How Do They Work?

GS1 XML Standards are standards for collaborative commerce defined by a global body of users: the GS1 member companies under the guidance of GS1 and its member organizations. The key features of the standards are that they are customer driven, based on global consensus, and designed jointly by GS1 and GS1 US member companies in an open process that encourages global participation.

The standards provide a flexible and extensible approach for transacting business-to-business electronic commerce. They have multi-sector and global applicability. GS1 XML Standards build on top of the World Wide Web Consortium (W3C) XML specifications. The W3C is a standards organization for developing interoperable technologies (specifications, guidelines, software, and tools) for Internet-based commerce and communication. Key features of the GS1 architecture are a reduced standards development cycle, consistent standards development, reusability, and message interoperability. The GS1 XML organization is modular. GS1 XML architecture uses this feature to organize schemas into logical layers where each layer performs a specific function. There are 3 layers in the Architecture:

- **Document Layer** contains individual business components (Trade Item etc.). The instance documents made from these schemas represent one or more business documents.
- **Message Layer** interfaces with the Document Layer and contains commands and transactions. Each XML instance contains one or more transactions; each transaction contains one or more commands to be carried out on a business document and each command contains one or more business documents.
- **Envelope** or the **Standard Business Document Header (SBDH) Layer** (depending on the version of the standard) contains the message header which provides transport & routing information.

In addition, the **Common Library** consists of common and reusable components that are used in all the other layers. Unique global GS1 identifiers (**GTIN and GLN**) establish the foundational data details about a party or trade item.

2.2. Architectural Principles of GS1 XML Standards

GS1 XML architecture guidelines impact the construction of the EAN UCC schemas. Users should understand these guidelines. This will ensure that GS1 XML conventions are followed consistently throughout the GDSN. Consistency facilitates interoperability.

There are four main concepts behind the architecture of the GS1 XML Standards:

- Message layering and separation
- Unique global identifiers
- Reusable documents and
- Context sensitive message definitions.

The purpose of layering is to organize the schemas into layers of functions where each layer performs a specialized function. The reason for this separation is so that if any of the layers need to change, the underlying layers will not be impacted. Such is the case as the version 1.3.2 AS2 and ebMS headers are replaced with the SBDH of v2. The support of the SBDH has had no impact on the Message or Document Layers or on their function.

In a single message, the layers can be either wrapped inside one another, as is the case with an AS2 Envelope 1.3.2 message or they can stand alone or wrap, as in the example of the SBDH v2 message. Using the SBDH, the message and document layer can be transported in the same or in a separate MIME part. (See UN/CEFACT SBDH Specification for details at:

http://www.disa.org/cefact-groups/atg/downloads/index.cfm

2.2.1. Benefits of Layering

Layering reduces the overall message set needed to support the GDSN – there is no need to create new document types because it decouples the document from the action taken on the document. This makes for quicker and more efficient enabling of messages. Commands are reusable across documents therefore they fit well with the principle of component reusability. A layer can be replaced without impacting other layers, as in the example of the SBDH replacing the Envelope Layer.

2.2.2. Global Entity Identification

The GLN and GTIN are used to globally identify parties and trade Items in GDSN. A similar concept is used to identify the commands, transactions and documents in the GS1 message. This is called the EntityIdentification, which is composed of the owner's GLN and the owner's unique identifier. The owner must ensure that the generated identifier is unique among entities within the owner's domain. The combination of GLN and the unique identifier makes the EntityIdentification globally unique. In version 1.3.1 and 1.3.2 this included the document type as well, which has been removed from version 2 because it breaks backward compatibility. The EntityIdentification is used to identify transactions, commands and documents. An example of the **EntityIdentification** is:

GLN: 1234567890123

Identifier: B89890

Another benefit of the EntityIdentification is that it is used as a "pointer" to the document on a Delete Command. Using the pointer eliminates the need to actually send the document to be deleted, as the reference to that document –its EntityIdentification – is sent instead. In fact, Delete documents are not used in GS1 XML at all. Only the pointer to them is used. The identification of functionality and business processes are dependent upon the use of Entity Identification.

2.2.3. Component Reuse

The concept of component reuse is practiced throughout GS1 XML:

- Component reuse is practiced when the document is reused by various commands.
- Document reuse is practiced across business process choreographies where the same document and command can be used across different business processes. Reusing components provides for consistency across the business processes.
- Document components are reused across documents. For example, the allowanceOrCharge and paymentTerms components which are optionally used in the TradeItem in GDSN may also be used in the non-GDSN Order and Invoice.
- Flexibility is maintained because XML has the ability to change a component by restriction or extension as needed.

The benefits of component reuse are that the component only needs to be defined once and may be reused across all the business processes as defined. The definition of that component can be found in the Global Data Dictionary (GDD). Reused components are defined in the GDD and once they are created as data objects in the application they maintain the same spelling, capitalization, naming and children.

2.2.3.1. An Example of Component Reuse

XML Schemas allow developers to precisely develop the structure of XML based formats in the instance documents. All XML instance documents are developed according to schemas and validated by parsing tools. So, the schemas form the rules that make up the instance documents that carry the data. An XML parser is a software tool that is used to extract data from an instance document. There are 2 steps involved:

- First the parser ensures the data is valid and built according to the rules written in the schema
- Next, the parser extracts the data from the instance document.

```
<xsd:complexType name="PartyIdentificationType">
  <xsd:sequence>
    <xsd:element name="gln" type="eanucc:GlobalLocationNumberType"/>
    <xsd:element name="additionalPartyIdentification"
    type="eanucc:AdditionalPartyIdentificationType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:simpleType name="GlobalLocationNumberType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{13}"/>
    </xsd:restriction>
 </xsd:simpleType>
```

The schema provides the formatting rules, for instance in what sequence will the data elements appear and what will be the cardinality or repeatability of the data. The resulting format can be used and reused to carry different sets of data. The example shows the formatting rules for the GS1 XML representation of the Global Location Number (GLN). Because the rules are reused, the format of that data does not change, only the data values change. The example of 2 instance documents carrying GLN data values is shown here, based upon the schema snippet above:

First example, reusing the GLN rules in an exception or error message:

```
<gdsn:gDSNException xsi:schemaLocation="urn:ean.ucc:2 ../Schemas/GDSNExceptionProxy.xsd">
 <originatingMessageIdentifier>
  <uniqueCreatorIdentification>WG-000001</uniqueCreatorIdentification>
   <contentOwner>
    <gln>0999999999991</gln> …
```

and second example, reusing the GLN in a Catalogue Item Confirmation:

```
<gdsn:catalogueItemConfirmation creationDateTime="2004-01-10T12:00:01"
documentStatus="ORIGINAL" lastUpdateDate="2004-02-26" xsi:schemaLocation="urn:ean.ucc:2
../Schemas/CatalogueItemConfirmationProxy.xsd">
   <contentVersion>
    <versionIdentification>2.0</versionIdentification>
   </contentVersion>
   <documentStructureVersion>
    <versionIdentification>2.0</versionIdentification>
   </documentStructureVersion>
   <catalogueItemConfirmationIdentification>
    <uniqueCreatorIdentification>WG-000001</uniqueCreatorIdentification>
    <contentOwner>
     <gln>0012345000359</gln> …
```

In both of these messages the concept of reusing rather than recoding is applied. Component reuse in GS1 is important because it provides consistency in the GS1 XML representation of the standard. It allows the component in the schema which defines the rules for a GTIN component or a GLN component or any other component to be represented identically in the instance document, regardless of whether the GTIN is used in a Price document or in an Item document. This increases consistency and eliminates ambiguous and multiple formats for the representation of a single concept.

2.2.4. The Envelope Layer of Version 1.3.2

In version 1.3.2, the Envelope Layer contains the tags that allow the GS1 XML message to be routed from a sender to a receiver by supplying information such as sender, recipient, guaranteed message delivery information and message identification to the communications software. The tags for mapping to the recommended protocol, AS2, were supplied by those protocol specifications defined by the IETF standards body. In version 1.3.2, the envelope components are stored in the following set of schemas:

- AS2Envelope.xsd.

This is a sample of the 1.3.2 Envelope Layer AS2 header:

```
<?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/1.3.2/eanucc" xsi:schemaLocation="http://www.ean-
ucc.org/schemas/1.3.2/eanucc CatalogueItemNotificationProxy.xsd" communicationVersion="1.3.2">
 <messageHeader creationDate="2001-08-02T12:00:00">
  <userId>0012345000065</userId>
  <messageIdentifier>10844772600209399</messageIdentifier>
  <to>
   <gln>0012345000065</gln>
  </to>
  <from>
   <gln>0614141000012</gln>
  </from>
  <representingParty>
   <gln>0012345000065</gln>
  </representingParty>
 </messageHeader>
 <body>
```

In version 2 and beyond, the Envelope Layer has been eliminated and replaced by the SBDH. The impact on GDSN is that instead of passing the AS2 Envelope layer tags; the SBDH is passed. The SBDH is a UN/CEFACT standard, containing information about the routing and processing of the business document. The SBDH components are stored in the following set of schemas:

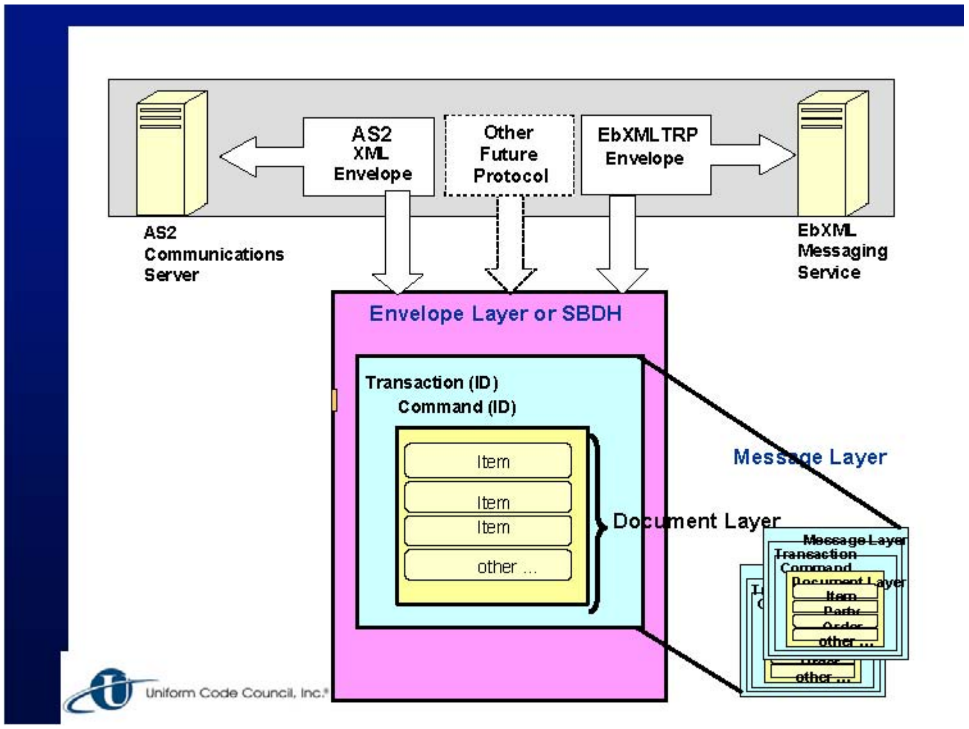
- BasicTypes.xsd, BusinessScope.xsd, DocumentIdentification.xsd, Manifest.xsd, Partner.xsd and StandardBusinessDocumentHeader.xsd.

Details about the SBDH can be found in the document available for download from the following link:

http://www.disa.org/cefact-groups/atg/downloads/index.cfm

The document is the "UN/CEFACT Standard Business Document Header Technical Specification Version 1.3". (More information on using the SBDH in GS1 schemas can be found in the Section 3, Communication Protocol)

Figure 2-1 The GS1 Architectural Layers and their Relation to Over-the-Wire Protocols



2.2.5. The Message Layer of Versions 1.3.2 and 2

The Message Layer defines what action is to take place on the transactions. Examples of commands or actions are: Add, Delete, Correct, Link and Unlink. The possible values that a command can act on are defined within the Message Layer. In version 1.3.2 and version 2, these command components are stored in the following set of schemas (only those relevant to GDSN will be listed here):

- Command.xsd, DocumentCommand.xsd, DocumentIdentificationCommand.xsd and LinkCommand.xsd.

There are technical capabilities that are intrinsic to this EAN·UCC architecture. However, the GDSN processes constrain these very flexible technical capabilities in order to easily facilitate adoption and implementation. 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.

Transactions encapsulate the document or set of documents that will be acted upon by the command. Because transactions enable sending multiple documents as one interchange, the use of the transaction allows for a reduction in the number of messages needed to transmit a group of documents for a more efficient exchange of business information. The transaction offers functionality of processing the group of documents together. If one of them fails, all of them will be discarded. The Transaction is made up of:

- An identification of the transaction, a component called entity identification. Entity identification consists of:
    - the party that created the set of commands
    - a unique identifier assigned by that party
- The commands themselves. The ‘command’ element is a container, where one or more commands that form the transaction can be inserted.

This is a sample of transactions and commands within an GS1 XML instance document.

```
<!-- this is the start of the first transaction -->
<eanucc:transaction>
 <entityIdentification>
  <uniqueCreatorIdentification>OJGROWER-TRANS-12345 </uniqueCreatorIdentification>
  <contentOwner>
   <gln>0614141000012</gln>
  </contentOwner>
 </entityIdentification>
 <!-- this is the start of the first command within the first transaction -->
 <command>
  <eanucc:documentCommand>
   <documentCommandHeader type="ADD">
    <entityIdentification>
     <uniqueCreatorIdentification>OJGROWER-ITEM-12345 </uniqueCreatorIdentification>
     <contentOwner>
      <gln>0614141000012</gln>
     </contentOwner>
    </entityIdentification>
   </documentCommandHeader>
   <documentCommandOperand>
```

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. In GDSN, the convention is that all documents sent in one message transmission should be of the same type, e.g. only Trade Items or Party documents. (See Section 4, GDSN Functional Description for more detail on how to use transactions.)

Example 1

A sender sends two Item documents, one for GTIN 1, the second for GTIN 2. The demand for product 1 is dependent on the availability of product 2. Instead of sending them separately, in two distinct transmissions, they can be transmitted together in one transaction. Both documents have to be valid, otherwise both will be rejected.

Example 2

A sender sends three new item documents to a customer and amends two other items, which had been sent earlier. All five item documents can be sent in one message – three with a command of ADD and two with a command of CHANGE BY REFRESH. The commands are placed in two separate transactions and sent in one message. If one of the first three messages (Item for GTIN 1, 2 or 3) is not valid, all three will be rejected but the messages from the second transaction (Item for GTIN 4 and 5) will be processed.

Example 3

A sender sends three new item documents to the customer and cancels one previously sent item. Those items are not interrelated and even if one of them fails, the sender wants the others processed. All four items are sent in one transmission, three with a command ADD and one with a command DELETE, in 4 separate transactions. If one of those transactions (including Item for GTIN 1, 2, 3 or cancellation of Item for GTIN 4) are not valid it will fail. The others will still be processed.

A single document type can be acted on by different commands. Hence, no separate documents for ‘Add Party, ‘Change Party or ‘Delete Party exist. For example, there is one ‘Party document sent with the relevant command, e.g. ‘add’, ‘change’ or ‘correct. This practice is efficient, as only one vote is needed to reach consensus on the business document as opposed to 3 votes for the Add Party, Delete Party or Modify Party for example.

Commands are used by a trading partner to instruct the receiving application about an action that should be performed on a given document.

The ‘command’ is an element, extended by the following elements:

- ‘documentCommand’
- ‘documentIdentificationCommand’
- ‘linkCommand’

The substituting elements are defined in DocumentCommand.xsd, DocumentIdentificationCommand.xsd and LinkCommand.xsd. The document is sent together with the command for all the actions except Delete.

2.2.5.1. The Document Command

DocumentCommand contains two main components: the ‘documentCommandHeader’ and ‘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, include:

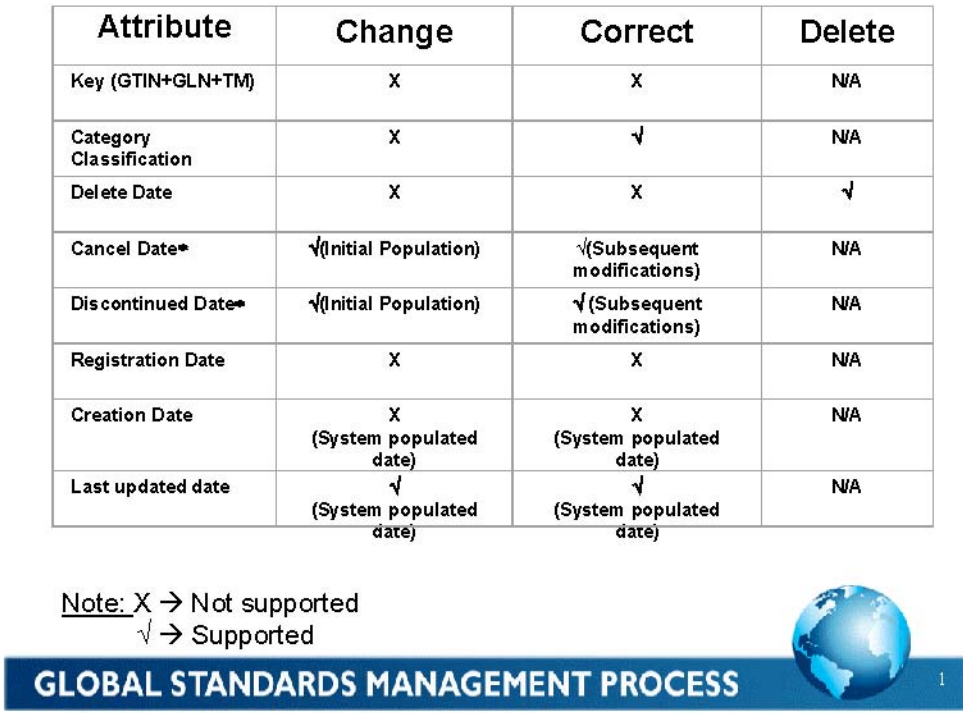
- **ADD** – the receiving application is instructed to store the document or documents
- **CHANGE_BY_REFRESH** – the receiving application is instructed to update the existing document or documents, by total replacement
- **CORRECT** – the receiving application is instructed to update the existing document 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. Those are the documents upon which one of the actions defined in the header element should be performed. The list of the possible documents contains all the business messages defined within the given major version.

2.2.5.1.1. When to Use the Change, Correct and Delete Commands

Table 2-1 describes when the Change, Correct and Delete Commands should be used.

Table 2-1 Change versus Correct versus Delete Commands



Attribute	Change	Correct	Delete
Key (GTIN+GLN+TM)	X	X	N/A
Category Classification	X	√	N/A
Delete Date	X	X	√
Cancel Date•	√(Initial Population)	√(Subsequent modifications)	N/A
Discontinued Date•	√(Initial Population)	√(Subsequent modifications)	N/A
Registration Date	X	X	N/A
Creation Date	X (System populated date)	X (System populated date)	N/A
Last updated date	√ (System populated date)	√ (System populated date)	N/A

Note: X → Not supported
√ → Supported

GLOBAL STANDARDS MANAGEMENT PROCESS

1

2.2.5.2. The Document Identification Command

The documentIdentificationCommand defines 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 on which the action should be performed is identified in the 'documentIdentifierList' element. The Document Identifier structure is specified in three ways:

1. By 'partyIdentification' – using GLN of the concerned party.
2. By 'tradeItemIdentification' – using GTIN 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 or the alternate one).

2.2.5.3. The Link Command

The real world items are represented in GS1 XML as documents. Whereas in the real, physical world there would be a relationship between the parents and children of items, in GS1 XML that relationship is established between Item documents or Party documents.

The Link Command establishes a parent-child or peer relationship between several documents. The Link Command consists of two major components: Link Header and Link Operand. The 'linkCommandHeader' specifies whether the function of a 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 header also contains the unique command identifier assigned by its creator, combined with the creator's identification (GLN). The 'linkCommandOperand' specifies the parent and children, identified by a document identifier. The Link Command Operand identifies the document with an Entity Identification of the document.

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 lists the 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' links documents to a given parent. Within both parent and child elements, there is a choice of components that are meant to be linked or unlinked. The first option is 'partyIdentification', defines the trading partner that is to be linked to given information. The second option is 'tradeItemIdentification', defining the product to be linked. The last option is to specify the document, using the 'entityIdentification'. These options give the user the ability to link and unlink any 2 GLNs (using partyIdentification), any 2 GTINs (using tradeItemIdentification) or any 2 documents (using entityIdentification). This is true for both the parent document and the child document. Both support all 3 options.

The link command can be used to express the relationship between two objects that need to be linked, e.g. GTINs of a pallet and cases placed on that pallet, or GLNs between the stores and the distribution centres. These 2 objects can be logically linked together by the use of the Link Command which provides the possibility of establishing parent-child or peer relationships between several documents. The list of the possible documents to use with the link command include Trade Item and Party documents.

2.2.6. Document Layer of Versions 1.3.2 and 2

The Document Layer contains the data, information or business documents which contain business data that the command operates upon. Examples of contents in the Document Layer are: Trade Item, Party and Price. Documents contain business level information and can have more than one instance and version. The document is transported by itself in the message or if used with the delete command, a reference to the document is sent instead of the actual document.

The business data is exchanged in support of business processes. The document layer provides the complete definition of a particular business process. The business document is a collection of extensions and a core part. Together this constitutes the business data that is exchanged between trading partners. The core part defines the industry and geography neutral business data. In the core part the components are not context-dependent. The extension part defines the industry and geography specific business data. The extension part is the part of the business document that is specific to one or more contexts. It is created using common components as a basis which are then extended to meet the needs of the business context. Likewise restricting common components could create an extension. The result is a customized business process for a specific industry or location. Multiple extensions may be added to a core business process to add functionality. (Please see Section 5, Use of Extensions for further information on contexts and extensions.)

2.2.7. Common Library

The reusable elements and attributes that are common across all industries and business processes, for example, the representation of the GLN, Text Description or Measurement, is coded once and reused throughout the suite of standards.

In version 1.3.2, these common components are stored in a set of schemas known as the common library (only those relevant to GDSN will be listed here):

- AlignComponentLibrary.xsd, CatalogueItemComponents.xsd, Components.xsd, ExtendedTypes.xsd, PartyComponents.xsd, PriceComponents.xsd, PartySynchronisationComponents.xsd, and TradeItemComponents.xsd.

In v2, these common components have been relocated into their own individual physical files. This coupled with the support for minor releases, impacts the GDSN community:

- Individual Business Process messages are independent of the entire release.
- They can be published as draft.
- They can be published as soon as the message is completed without waiting for the entire suite of messages to be released.

2.3. Validation of GS1 XML

The GS1 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
- MSXML parser (version 4.0 or higher) from Microsoft and

Xerces (version 2.6.2) from Apache.

2.4. GS1 XML Schema and GDSN

GDSN schemas are based on the conventions of the GS1 XML Standard. These standards are released on a version basis. GDSN is currently supported by version 1.3.2 of GS1 XML Standards. The next version published is version 2 (v2). The table below describes the differences between these 2 versions.

Table 2-2 Overview of the Difference between versions 1.3.2 and 2

	Version 1.3.2	Version 2.0	Comments
	Business Terms		
1	Business terms and definitions come from BMS 1.3.1	Business terms and definitions come from BMS 1.3.1	No change
	Transport Function Header		
2	Supported by the Envelope Layer AS2 header tags	Supported by the UN/CEFACT Standard Business Document Header. Envelope Layer AS2 Header is removed.	Major change
	Namespaces		
3	Concept of one version, one namespace for all schemas <xsd:schema targetNamespace=http://www.ean-ucc.org/scheams/1.3.2/eanucc xmlns=http://www.ean-ucc.org/schemas/1.3.2/eanucc …	Differing namespaces supported for schema modules: *Higher level XSDs:* <xsd:schema targetNamespace="urn:ean.ucc:align:2" xmlns:align="urn:ean.ucc:align:2" xmlns:eanucc="urn:ean.ucc:2" … *version="2.0">* *Codelists and Component XSDs:* <xsd:schema version="2.0" targetNamespace="urn:ean.ucc:2" xmlns:eanucc="urn:ean.ucc:2" …	Major change

	Version 1.3.2	Version 2.0	Comments
4	Namespace format is url	Namespace format is urn targetNamespace=”urn:ean.ucc:plan:2”	Minor change
5	Major version only represented in the namespace	Major and Minor version represented in: Major version in namespace Minor version in internal schema version attribute XML instance in the documentStructureVersion and DocumentIdentification TypeVersionExample: targetNamespace=”urn:ean.ucc:plan:2” … xmlns:eanucc:”urn:ean.ucc:plan:2” … version=”2.3”	Major change – allows for minor changes to messages
Backward and Forward Compatibility			
6	Backward Compatibility of XML instance is not supported	Backward Compatibility of XML instance is supported within one major release	Major Change
7	Forward Compatibility of XML instance is not supported	Forward Compatibility of XML instance is not supported	No Change
Context			
8	A message’s context is not defined by the sender on-demand. In version 1.3.1 the context was predefined by the BRG.	A message’s context is defined by the sender on-demand.	Major Change
9	Single or multiple context co-located in one XML instance as decided by BRG.	Multiple contexts can be co-located in one XML instance as decided by sender	Major Change
10	Extensions in XML Instance use xsi:type and in the schema have optional extension tag implemented with any element & skip validation rules	Extensions supported by optional extension tag in the schema implemented with the any element supporting lax validation rules. No xsi:type support.	Xsi:type eliminated Skip validation of extensions changed to lax
11	Incorporate Candidate Attributes with their own namespace different from release namespace: <xsd:schema targetNamespace="http://www.ean-ucc.org/eanucc/extensions/itemidentification/1" … "xmlns="http://www.ean-ucc.org/eanucc/extensions/itemidentification/1"	Candidate Attributes more consistent with release: <xsd:schema targetNamespace="urn:ean.ucc:sweden:2" … xmlns:sw="urn:ean.ucc:sweden:2" … version="2.0"> where the 2 after sweden indicates the major release of the candidate attribute	Minor change
Codelists and Components			
12	Codelists are co-located in main schema modules	Codelists are located in separate schema modules	Major change
13	Components are located in a centralized set of files (components.xsd etc.) causing circular includes, multiple includes, no separate lifecycle, memory stack overflow.	Components are located in separate schema modules and a separate lifecycle is supported	Major change
14	Components did not have a separate namespace (were chameleon)	All components have a namespace indicating the major version number and a schema version attribute showing the minor version	Major change

	Version 1.3.2	Version 2.0	Comments
		Publication	
15	Messages are interdependent –they are released in one zip file at one publication date	Messages are stand-alone – they are released in separate Implementer's Packets per message as soon as complete. Only modules that are necessary for instance and schema validation are included. Sample contents of Party Implementer's Packet: *Instance File* PartySample.xml, v2.0 *Schemas* FunctionalModeCodeList.xsd; Party.xsd; PartyDocument.xsd; AccountNumberTypeList.xsd; AllowanceCharge.xsd; AllowanceChargeList.xsd; AllowanceOrChargeList.xsd; AlternatePartyIdentificationList.xsd; Command.xsd; CommunicationChannelCodeList.xsd;Contact.xsd; DaysOfTheWeekList.xsd; Description.xsd; Document.xsd; DocumentCommand.xsd; DocumentCommandList.xsd; DocumentStatusList.xsd; EffectiveDateList.xsd; EntityIdentification.xsd; Extension.xsd; FinancialInstitutionInformation.xsd; Measurement.xsd; MonetaryAmount.xsd; NameAndAddress.xsd; PartyIdentification.xsd; PartyRoleList.xsd; PaymentFormatList.xsd; PaymentMethod.xsd; PaymentMethodList.xsd; PaymentTerms.xsd; PaymentTermsEventList.xsd; PaymentTermsTypeList.xsd; Percentage.xsd; ResponseStatusList.xsd; RoutingNumberTypeList.xsd; SettlementList.xsd; TimePeriodList.xsd; Transaction.xsd; *SBDH* BasicTypes.xsd; BusinessScope.xsd; DocumentIdentification.xsd; Manifest.xsd; Partner.xsd; StandardBusinessDocumentHeader.xsd *Proxies* PartyProxy.xsd, v2.0	Major change
	No Configuration management in Implementer's Packet	Implementer's Packet published with Configuration management for ease of storing and validating – schemas unzip into the correct directory	Major change

3. Communication Protocol

In version 1.3.2 and v2 of the schemas, the AS2 protocol is the sole approved communications protocol for GDSN. (The reader is referred to the following document for information on the use of this protocol: “AS2 Transport Communications Guide for the GS1 GDSN Community” (version 1.1 refers to v1.3.2 schemas and version 2.0 refers to the version 2.0 schemas).) This document is accessed from the following link:

http://www.gs1.org/productssolutions/gdsn/

In addition to the GS1 standards on use of AS2 in GS1, the GDSN Architecture Group, a sub-team of the GDSN Task Group was chartered with the definition of technical specifications for GDSN. Their recommendations on AS2 use are as follows:

- **AS2 Operations**

  AS2 will operate asynchronously for all collaborations: request – response, with a synchronous MDN on each individual message transfer.

- **Secure Channel**

  AS2 will use channel encryption (SSL) to obviate message encryption.

3.1. Standard Business Document Header (SBDH) of Version 2

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 GS1 schemas v2 and beyond, 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).

(Detailed UN/CEFACT specification for the SBDH and the guidelines on how to use it can be downloaded from the UN/CEFACT website:

http://www.disa.org/cefact-groups/atg/index.cfm)

3.1.1. How to Use SBDH Tags in GDSN

This document describes how to use and populate the SBDH tags for use in GDSN July 2005 certification / production. No optional tags will be used for the July certification of the October 29, 2005 Production deployment.

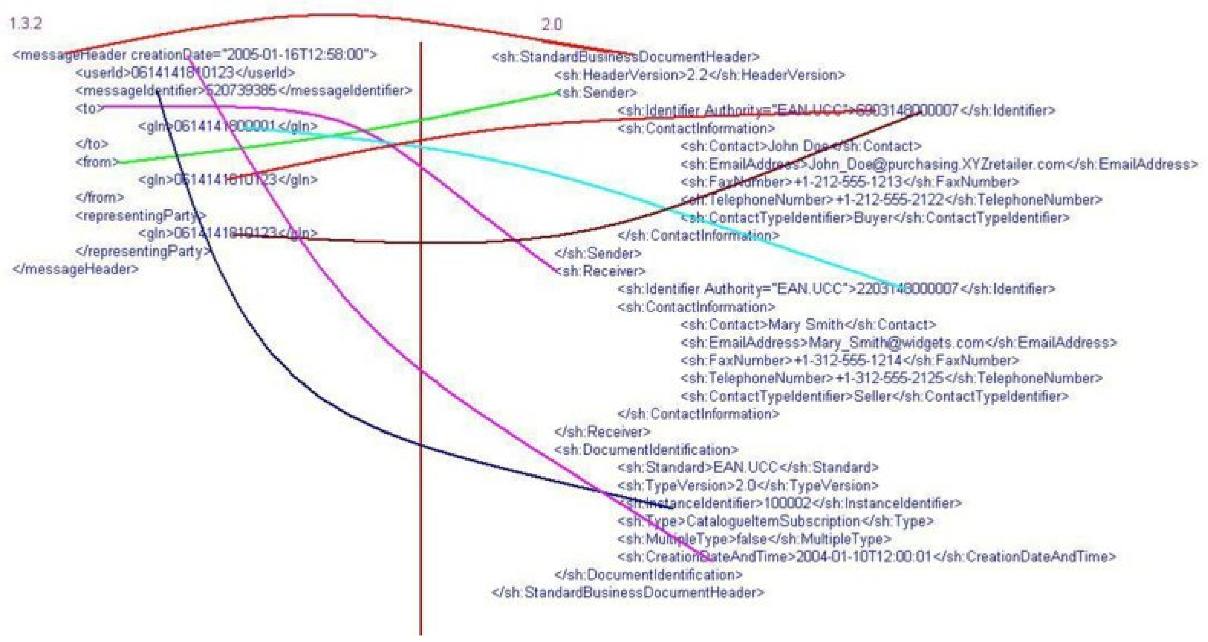
However in the GS1 Response messages the business scope tag which is optional and recursive, will be used but is restricted to only one occurrence. Additionally, the correlation ID in the business scope tag will match the message ID in the corresponding message.

3.1.1.1. AS2 Envelope Layer is Retired

AS2 tags are no longer available. Therefore, in GDSN GS1 version 2.0 XML these tags MUST NOT be used.

Figure 3-1 displays is the mapping diagram from 1.3.2 to 2.0.

Figure 3-1 Mapping Diagram from 1.3.2 to 2.0



```
<messageHeader creationDate="2005-01-16T12:58:00">
        <userId>0614141810123</userId>
        <messageIdentifier>520739385</messageIdentifier>
        <to>
                <gln>0614141800001</gln>
        </to>
        <from>
                <gln>0614141810123</gln>
        </from>
        <representingParty>
                <gln>0614141810123</gln>
        </representingParty>
</messageHeader>
```

```
<sh:StandardBusinessDocumentHeader>
        <sh:HeaderVersion>2.2</sh:HeaderVersion>
        <sh:Sender>
                <sh:Identifier Authority="EAN.UCC">6903148000007</sh:Identifier>
                <sh:ContactInformation>
                        <sh:Contact>John Doe</sh:Contact>
                        <sh:EmailAddress>John_Doe@purchasing.XYZretailer.com</sh:EmailAddress>
                        <sh:FaxNumber>+1-212-555-1213</sh:FaxNumber>
                        <sh:TelephoneNumber>+1-212-555-2122</sh:TelephoneNumber>
                        <sh:ContactTypeIdentifier>Buyer</sh:ContactTypeIdentifier>
                </sh:ContactInformation>
        </sh:Sender>
        <sh:Receiver>
                <sh:Identifier Authority="EAN.UCC">2203148000007</sh:Identifier>
                <sh:ContactInformation>
                        <sh:Contact>Mary Smith</sh:Contact>
                        <sh:EmailAddress>Mary_Smith@widgets.com</sh:EmailAddress>
                        <sh:FaxNumber>+1-312-555-1214</sh:FaxNumber>
                        <sh:TelephoneNumber>+1-312-555-2125</sh:TelephoneNumber>
                        <sh:ContactTypeIdentifier>Seller</sh:ContactTypeIdentifier>
                </sh:ContactInformation>
        </sh:Receiver>
        <sh:DocumentIdentification>
                <sh:Standard>EAN.UCC</sh:Standard>
                <sh:TypeVersion>2.0</sh:TypeVersion>
                <sh:InstanceIdentifier>100002</sh:InstanceIdentifier>
                <sh:Type>CatalogueItemSubscription</sh:Type>
                <sh:MultipleType>false</sh:MultipleType>
                <sh:CreationDateAndTime>2004-01-10T12:00:01</sh:CreationDateAndTime>
        </sh:DocumentIdentification>
</sh:StandardBusinessDocumentHeader>
```

3.1.1.2. The SBDH Tags

1. Standard Business Document Tag

This is a root tag and as such is always mandatory within the SBDH object. You can note that it is **OPTIONAL** under the UN/CEFACT SBDH standard however, as published for use in GDSN it is mandatory as noted prior.

The GS1 part of the message is wrapped by the header (and NOT sent in a separate MIME part), and the tag **MUST BE USED** for GDSN GS1 XML July 2005 certification.

Request SBDH Sample:

```
<sh:StandardBusinessDocumentHeader>
  <sh:HeaderVersion>1.0</sh:HeaderVersion>
  <sh:Sender>
   <sh:Identifier Authority="EAN.UCC">2116990776066</sh:Identifier>
  </sh:Sender>
  <sh:Receiver>
   <sh:Identifier Authority="EAN.UCC">0614141810017</sh:Identifier>
  </sh:Receiver>
  <sh:DocumentIdentification>
   <sh:Standard>EAN.UCC</sh:Standard>
   <sh:TypeVersion>2.0.2</sh:TypeVersion>
   <sh:InstanceIdentifier>11128210773015440</sh:InstanceIdentifier>
   <sh:Type>CatalogueItemSubscription</sh:Type>
   <sh:CreationDateAndTime>2005-04-06T16:57:45</sh:CreationDateAndTime>
  </sh:DocumentIdentification>
</sh:StandardBusinessDocumentHeader>
```

- Header version MUST be 1.0 for this certification.
- Sender and receiver tags will be used ONLY once even though it is a recursive tag.
- GS1 will be used as the 'Authority' value.

2. Document Identification Tag

DocumentIdentification is **MANDATORY** under the UN/CEFACT SBDH standard. In GS1 XML it **MUST** be used.

- The Standard tag will always be populated as: **GS1** as in the example below.
- The Type version is always the current schema version. As of the date of this document, April 8, 2005, the current schema version is **2.0.2** and should be populated as such.
- The InstanceIdentifier **MUST** always be populated with the identification of the current message instance. This applies to both the originating message as well as the responding message. There are options for populating the InstanceIdentifier:
  - Populate with the same data as the messageIdentifier in version 1.3.2.
  - Please notice this in the sample request and sample response messages included
- The Type indicates the type of document being sent. [in the sample above, this tag states 'catalogueItemSubscription']. This tag is **MANDATORY** under the UN/CEFACT SBDH standard. In GS1 XML it **MUST** be used. According to Task Group agreement (see operations manual) the GDSN standard is to limit to 1 document type within 1 message. This same information may be extracted from the child of the payload tag "documentCommandOperand".
- CreationDateAndTime is **MANDATORY** under the UN/CEFACT SBDH standard. In GS1 XML it **MUST** be used to set the date and time when the 'document originating application' or the parser created the document. This role will typically be acted by the trading partner and will typically differ from the time stamping of the message by the AS2 software.

```
<sh:DocumentIdentification>
   <sh:Standard>EAN.UCC</sh:Standard>
   <sh:TypeVersion>2.0</sh:TypeVersion>
   <sh:InstanceIdentifier>100002</sh:InstanceIdentifier>
   <sh:Type>CatalogueItemNotification</sh:Type>
   <sh:CreationDateAndTime>2004-01-10T12:00:01</sh:CreationDateAndTime>
</sh:DocumentIdentification>3. Business Scope Tag
```

The Business Scope and Scope tags are **OPTIONAL** under the UN/CEFACT SBDH standard but as stated above, this optional class will be used in GDSN for **responding messages ONLY**

The sender 's gln is now the receiver's gln and the originating instance id is now in the RequestingDocumentInstanceIdentifier.

Response SBDH Sample with Business Scope included:

```
 <sh:StandardBusinessDocumentHeader>
  <sh:HeaderVersion>1.0</sh:HeaderVersion>
  <sh:Sender>
   <sh:Identifier Authority="EAN.UCC">0614141810017</sh:Identifier>
  </sh:Sender>
  <sh:Receiver>
   <sh:Identifier Authority="EAN.UCC">2116990776066</sh:Identifier>
  </sh:Receiver>
  <sh:DocumentIdentification>
   <sh:Standard>EAN.UCC</sh:Standard>
   <sh:TypeVersion>2.0.2</sh:TypeVersion>
   <sh:InstanceIdentifier>11128210773015440</sh:InstanceIdentifier>
   <sh:Type>GDSNResponse</sh:Type>
   <sh:CreationDateAndTime>2005-04-06T04:57:57</sh:CreationDateAndTime>
  </sh:DocumentIdentification>
  <sh:BusinessScope>
   <sh:Scope>
    <sh:Type>GDSN</sh:Type>
    <sh:CorrelationInformation>
<sh:RequestingDocumentInstanceIdentifier>11128210773015440</sh:RequestingDocumentInstanceI
dentifier>
    </sh:CorrelationInformation>
   </sh:Scope>
  </sh:BusinessScope>
 </sh:StandardBusinessDocumentHeader>
```

3.1.1.3. The SBDH and the GS1 Payload

GS1 has created a new structure to accommodate the multiple transactions within one message requirement when using the SBDH. This message construct is shown below:

```
  <xsd:complexType name="MessageType">
  <xsd:sequence>
   <xsd:element name="entityIdentification" type="eanucc:EntityIdentificationType"/>
   <xsd:any namespace="##any" processContents="strict" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
```

The start of the instance document which follows the UN/CEFACT SBDH begins at the Message Layer Tag: <eanucc:message>

3.1.1.3.1. SBDH Example

The following is a snippet from a sample XML document that shows how this construct can be used in conjunction with the SBDH

```
<sh:StandardBusinessDocument>    /*MANDATORY*/
    <sh:StandardBusinessDocumentHeader>    /*MANDATORY*/
        <sh:HeaderVersion>1.0</sh:HeaderVersion>  /*MANDATORY*/
        <sh:Sender>              /*PRIMARY MANDATORY*
                    MUST NOT USE CONTACT
                                                   INFORMATION /
            <sh:Identifier Authority="EAN.UCC">6903148000007</sh:Identifier>
        </sh:Sender>
        <sh:Sender>      /*SECONDARY OPTIONAL*/
            <sh:Identifier Authority="EAN.UCC">6903148000008</sh:Identifier>
        </sh:Sender>
        <sh:Receiver>            /*PRIMARY MANDATORY
                    MUST NOT USE CONTACT
                                                   INFORMATION */
            <sh:Identifier Authority="EAN.UCC">2203148000007</sh:Identifier>
        </sh:Receiver>

        <sh:Receiver>            /*SECONDARY OPTIONAL */
            <sh:Identifier Authority="EAN.UCC">2203148000007</sh:Identifier>
        </sh:Receiver>
        <sh:DocumentIdentification> /* MUST NOT USE  MULTIPLE TYPE - */
            <sh:Standard>EAN.UCC</sh:Standard>
            <sh:TypeVersion>2.0.2</sh:TypeVersion>
            <sh:InstanceIdentifier>100002</sh:InstanceIdentifier>
            <sh:Type> CatalogueItemNotification</sh:Type>
            <sh:CreationDateAndTime>2004-01-10T12:00:01</sh:CreationDateAndTime>
         </sh:DocumentIdentification>
                                                  /*MUST NOT USE MANIFEST */
```

The tags below must be used for responding messages and only responding messages:

```
        <sh:BusinessScope>
          <sh:Scope>
            <sh:Type>GDSN</sh:Type>
            <sh:CorrelationInformation>
            <sh:RequestingDocumentInstanceIdentifier>100002
          </sh:RequestingDocumentInstanceIdentifier>
            </sh:CorrelationInformation>
          </sh:Scope>
         </sh:BusinessScope>
    </sh:StandardBusinessDocumentHeader>
        <eanucc:message> /* START OF THE GS1 PAYLOAD*/
          …
           <eanucc:transaction>
          …
            <command>
             <eanucc:documentCommand>
              <documentCommandHeader type="ADD">
        …
        </eanucc:message>
  </sh:StandardBusinessDocument>
```

4. GDSN Functional Description

This section describes the entities and the functions that make up GDSN and provides more detail on the GDSN network and the standards supporting the network.

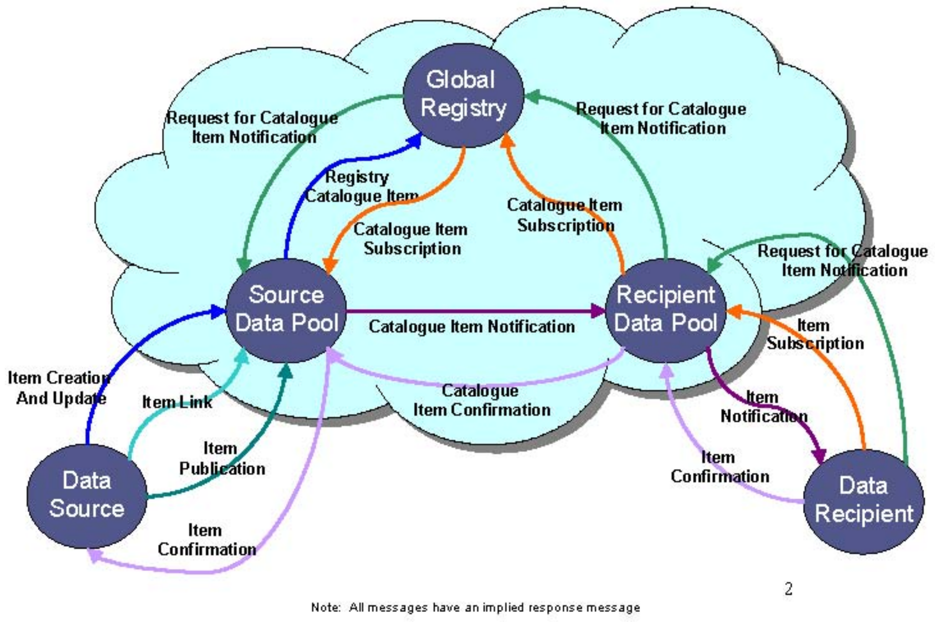
4.1. The GDSN Entities

The key entities taking part in GDSN are:

- The Trading Partners (Data Source and Data Recipient)
- The Data pools (Source Data Pool and Recipient Data Pool)
- And the GS1 Global Registry.

Figure 4-1 depicts the flow of information among the entities.

Figure 4-1 GDSN Choreography


Global Registry
Source Data Pool
Recipient Data Pool
Data Source
Data Recipient
Request for Catalogue Item Notification
Request for Catalogue Item Notification
Request for Catalogue Item Notification
Registry Catalogue Item
Catalogue Item Subscription
Catalogue Item Subscription
Catalogue Item Notification
Catalogue Item Confirmation
Item Creation And Update
Item Link
Item Publication
Item Confirmation
Item Subscription
Item Notification
Item Confirmation
2
Note: All messages have an implied response message

4.1.1. The GS1 Global Registry

The GS1 Global Registry is an GS1 registry for Party, Item and Subscription Data. The GS1 Global Registry stores Base Item data, Party information and facilitates the routing of subscription information to Data Pools. The Data Pool can act as Source Data Pool, Recipient Data Pool or both. Source Data Pools act on behalf of the Trading partner which is the source of item information. They interact with the data source (trading partner), the GS1 Global Registry and the Recipient Data Pool, providing the item information to the GS1 Global Registry. The Source Data Pool in turn provides this information to the Recipient Data Pool. The Recipient Data Pool provides the information to their member trading partners who have set up subscriptions to receive this item data. Recipient Data Pools act on behalf of the Trading Partner acting as the receiver of item information. They interact with the data recipient Trading Partner, the GS1 Global Registry and the Source Data Pool. They receive subscription information from their trading partners and submit it to the GS1 Global Registry which delivers it to the Source Data Pool. As a result of this chain of events the recipient will receive item information published by the source trading partner.

4.1.1.1. Joining the GDSN

Trading partners wishing to participate in the GDSN should contact the GS1 Member Organisation (MO) in the country in which their company is headquartered. Data Pools must be certified to be a part of the GDSN. All interested parties should visit the GS1 site for instructions on how to become members:

http://www.gs1.org/productssolutions/gdsn/

4.1.2. The GDSN Data Pools

Data Pools interoperate within the GDSN facilitating the synchronization of data for their Trading Partner, the retail and manufacturer companies. In order to be a part of the GDSN, the Data Pool and the Trading Partner must become registered with the GS1 Global Registry.

4.2. Functionality

GDSN trading partners, data pools and the GS1 Global Registry interact with each other through standard interface messages. The defined standard interface messages, their UML class models and use cases are found in the Business Message Standard (BMS) documents. Class diagrams are transformed into W3C XML Schema Definition Language schemas. The business intent is preserved from BMS to schema realization, although demands of XML syntax, cross-business process consistency and component reuse may warrant minor deviations between the BMS document and schema. Such discrepancies may be found in string lengths for example.

A BMS is first published as a draft, having gone through business and technical user approval. Later documents and interfaces become a final standard once they are ratified by the GS1 and GS1 US Management Boards. Ratification typically occurs once or twice a year, depending on the number of messages being developed and whether there are major architectural changes requiring a new version. Upon ratification, the messages are published as a final standard on the GS1 website (www.GS1.org). The ratified messages can be used for final implementation.

A Trade or Catalogue Item, is any product or service upon which there is a need to retrieve pre-defined information and that may be priced, ordered or invoiced at any point in any supply chain. The term “trade item” can represent any level of product containment. A Catalogue Item is uniquely defined within the GS1 Global Registry by the combination of the item’s Global Trade Identification Number (GTIN), Global Location Number (GLN), and Target Market. There is a difference between a Trade Item and a Catalogue Item. The Trade Item embodies the representation of the definition of a trade item and may or may not include hierarchy structure. The Trade Item key is a GTIN. The Trade Item is registered by the GTIN Owner and allows for the GTIN Owner to define specific non-changeable attributes. The Catalogue Item embodies the representation of the usage of an item, or in other words the content of an item. The key of the Catalogue Item is the GTIN + GLN + TM (Target Market). It is registered by the Information Provider who may or may not be equivalent to the GTIN Owner and allows for the flexibility for Information Providers to allow for varying data across GLN’s and/or Target Markets. The Target Market (TM) is a 3-digit numerical code that indicates the country level or higher geographical definition in which the information provider will make the item available to buyers. This indicator does not govern where a buyer may re-sell the item to consumers. The Target Market Country Code is taken from the ISO 3166-1 list. Each Trade Item is treated as a separate and independent item within the GS1 Global Registry. (For more details on the functionality please see the BMS.)

4.2.1. Registry Catalogue Item Add

The ADD Registry Catalogue Item (ADD RCI) function is used by the Source Data Pool whenever one of its member Parties wishes to add a new item to the GS1 Global Registry. ADD RCI functionality provides the Source Data Pool with the ability to register base item information in the GS1 Global Registry on behalf of its member source trading partners. The process for a Source Data Pool to prepare and submit an ADD RCI message to the GS1 Global Registry on behalf of one of its member parties in order to register a new item in the GS1 Global Registry is:

- A Data Source Trading Partner has a new item and wishes to register information about the item in the GS1 Global Registry. The Data Source Trading Partner communicates to the source Data Pool its wish to register a new item in the GS1 Global Registry in a manner agreed upon by the two parties. The Data Source Trading Partner communicates the item attribute information to its Source Data Pool.
- The Source Data Pool prepares and submits a valid, well-formed ADD Registry Catalogue Item (ADD RCI) message to the GS1Global Registry. (ADD is the document command type).
- The information in the message is validated through the Data Validation Rules (DO WE HAVE A VALIDATION ENGINE?) to ensure that all item compliance rules have been successfully met. The item is registered and stored in the GS1 Global Registry.

- The GS1 Global Registry sends back a response indicating whether the Item was successfully registered. If the message has not passed all validation checks, the response message will include an error message with specific information on why the item was not registered successfully. If no errors are produced, the Item is successfully registered and its information is now stored in the GS1 Global Registry.

4.2.2. Party Registration

The **Party Registration** is used by a Data Pool to register one of its member Parties in the GS1 Global Registry in order to begin performing actions on behalf of its member. In the GS1 Global Registry, each Party is treated as a separate and independent GLN and is registered separately by a Data Pool. Trading Partners involved with GDSN require Party data information to achieve data synchronization. A Party must be registered in the GS1 Global Registry in order to register or subscribe to item data. An item cannot be registered if an associated Party does not exist in the GS1 Global Registry. Each Party within the GS1 Global Registry must belong to a Data Pool. The Data Pool will be responsible for communicating all Party, Item, and Subscription information to the GS1 Global Registry on behalf of its member Trading Partners. The process for a Source or Recipient Data Pool to register Party information for one of its member Trading Partners is:

- The GS1 Global Registry Administrator has successfully registered the Data Pool in the GS1 Global Registry.
- The Data Pool’s member trading partner communicates its Party information to the Data Pool in a process agreed upon by the two parties.
- The Data Pool prepares and submits a valid, well-formed ADD Party Registration message (Party.xml) containing all of the necessary Party information regarding its member Trading Partner’s location to the GS1 Global Registry. (ADD is the document command.)
- The GS1 Global Registry sends back a response indicating whether the Party was successfully registered. If the Party was not successfully registered, the response message will include an error message containing specific information on why the message could not be sent successfully. If no errors are produced, the Party is successfully added to the GS1 Global Registry, and a successful response message is returned to the Data Pool GLN. When the Party record is successfully registered, the GS1 Global Registry will store the Party’s Data Pool information for future distribution. The Party’s Data Pool information identifies the GDSN certified Data Pool that the Party selected as their Data Pool.

The DELETE Party function will be used when a Data Pool member wishes to remove one or more of its organizational units from the GS1 Global Registry. The CHANGE Party function will be used when a Data Pool member’s Party information has changed, and the Data Pool needs to communicate these changes to the GS1 Global Registry on behalf of its member. The Party Change functionality is necessary in order to ensure that a Party’s information is kept up to date in the GS1 Global Registry. (DELETE and CHANGE_BY_REFRESH are document commands.)

4.2.3. Request for Catalogue Item Notification

The Request for Catalogue Item Notification (RFCIN) function is used for an item that already has a subscription in place whenever a Recipient Data Pool member desires to obtain Catalogue Item information from Data Source Parties. The RFCIN is a one-time request to item information that has been registered in the GS1 Global Registry. This functionality is used by Recipient Data Pools to request publication of a Catalogue Item from the Source Data Pool member (data source) to be sent to the data recipient. This one-time publication request does not establish a permanent subscription to the item information. An RFCIN can reset the synch list and reactivate an existing subscription. The CIN must have been previously published to the data recipient for this to occur. The process for a Recipient Data Pool to prepare and submit an RFCIN message to the GS1 Global Registry on behalf of one of its member Parties is:

- A Data Recipient Trading Partner wishes to examine the attributes of an item (or items) registered in the GS1 Global Registry, but does not yet wish to permanently subscribe to the item information. The Data Recipient communicates its request to its Recipient Data Pool to receive a one-time publication of information for a particular item or items. Item data can be requested using the following filter criteria:
  - GTIN (request for specific item information)
  - Category (request for information on all items within a particular GPC Item Classification)
  - Target Market (TM) (all items within a particular Target Market)
  - GLN (all items from a particular Data Source GLN).

or any combination of these 4 with the exception of GTIN and GPC which are redundant.

The Data Recipient will indicate the type of Notification (New Item or Initial Item Load). To restart synchronisation from scratch, for items to be resent whether they were previously rejected or not, set the:**RFCIN isReload = falseActions:**1. Sync List is reset 2. CIN response will have the following values:documentStatus = **Original**IsReload = **False**Command = **Add**The documentStatus of the RFCIN command is ignored for the purposes of determining it's impact on the synchlist and the status of the CIN that is generated.

resend synchronised items, only the items not previously rejected will be resent. Set the:**RFCIN isReload = trueActions:**1. Sync List is not reset2. CIN response will have the following values:documentStatus = **Copy**IsReload = **True**Command = **Add**

The documentStatus of the RFCIN command is ignored for the purposes of determining it's impact on the synchlist and the status of the CIN that is generated. his will set the "is Reload" indicator in the RFCIN accordingly. The CIC States will be set to the following:

- Notifications Continue
  - No CIC sent
  - ACCEPTED
    - Data has been received by the Recipient, but no business decision has been made on the data
- REVIEW
  - Request to the data source to "review" their data because the data recipient has received discrepant data which they cannot synchronise
- SYNCHRONISED
  - Data is integrated and in synch
- Discontinue Notifications
  - REJECTED
    - Recipient requests that no further updates are desired
- The Source Data Pool receives the RFCIN information and notifies its applicable member Parties.

The item GTIN referenced in the RFCIN.xml message must be registered in the GS1 Global Registry; the Category referenced in the RFCIN.xml message must be a valid GPC brick code classification and the Target Market referenced in the RFCIN.xml must be a valid 3-digit Country Code from the ISO-3166-1 list; and the party GLN referenced must be registered in the GS1 Global Registry. If the Data Recipient wishes to continue to receive information about this item, the Recipient Data Pool will need to submit a Catalogue Item Subscription (CIS) on behalf of the Data Recipient Party to set up a permanent subscription.

4.2.4. Catalogue Item Subscription

The Catalogue Item Subscription (CIS) is a message used by Recipient Data Pools on behalf of their member Data Recipient Trading Partners to request item information within the Global Data Synchronization Network (GDSN).Once a subscription is active and there is a publication that matches the subscription, the Recipient Data Pool receives updates of the matching data, and passes this data along to its member Parties. The process for a Recipient Data Pool to prepare and submit a CIS message to the GS1 Global Registry on behalf of one of its member Parties is:

- A Data Recipient sets up a subscription to item information registered in the GS1 Global Registry.
- The Recipient Data Pool prepares and submits a valid CIS message to the GS1 Global Registry. (The document command type is ADD.) The Recipient Data Pool includes the filter criteria in the subscription message.
- The GS1 Global Registry resolves the request by validating through the Data Validation Rules and sends the information to the appropriate Source Data Pool(s), who then communicate this information to their appropriate member Parties. If the information in the message passes all validation checks, a successful response message will be generated and returned to the Recipient Data Pool. If the document does not pass all validation checks, a response will be generated and returned to the Recipient Data Pool with a specific error message. (Please refer to the GS1 Global Registry Data Validation Rules document for more information regarding specific error messages.) The GS1 Global Registry determines which Source Data Pool(s) to route the subscription information to, based on the filter criteria included in the CIS message. The GS1 Global Registry delivers the message to the appropriate Source Data Pool.
- The Source Data Pool receives the CIS information and notifies its applicable member trading partner of the Data Recipient's subscription information.

The DELETE Catalogue Item Subscription function is used whenever a Recipient Data Pool Trading Partner decides to cancel a subscription to item information that has been previously submitted to the GS1 Global Registry. (The document command type is DELETE.) There is no way to change (CHANGE_BY_REFRESH) a CIS. The original subscription must be deleted and the new subscription added.

4.3. Technical Specifications for GDSN

The GDSN Architecture Group, a sub-team of the GDSN Task Group was chartered with the definition of technical specifications for GDSN. Their recommendations are as follows:

- **AS2 Operations**

  AS2 will operate asynchronously for all collaborations: request – response, with a synchronous MDN on each individual message transfer.
- **Secure Channel**

  AS2 will use Secure Socket Layer (SSL) channel encryption to obviate message encryption.
- **Retry Interval**

  The Retry Interval will be 60 minutes.
- **Protocol Failure and Message Attempts (Retries)**

  Protocol failure occurs when a single message transfer (transmit message and receive MDN) does not complete properly. Possible failure modes are:

  - Failure on transmission,
  - Failure to receive MDN.

- **Three Total Message Attempts**

  There will be three attempts to send and complete a message transfer: one initial attempt and two retries. A third failure will result in no further attempts and constitute a transaction failure. Based on a 60 minute Retry Interval and three attempts, the total minimum elapsed time for consecutive Protocol Failures would be just over 120 minutes.

- **Response Message Timeout and Retries**

  A Response Message Timeout occurs when a request message has been successfully sent (transmit message and synchronous MDN) and there is no Response Message.

- **Three Total Message Attempts**

  There will be three attempts to initiate the transaction with the request message: one initial attempt and two retries. The request message will contain the same identifier for each attempt. Based on a 60 minute Retry Interval and three attempts, the total minimum elapsed time for consecutive Response Message Timeouts would be just over 180 minutes.

- **Retry Counter**

  There will be only one retry counter. A Response Message Timeout or a Protocol Failure retry will increment the retry counter. If a Protocol Failure occurs during a Response Message Timeout, a retry will be attempted after the Retry Interval has expired providing the retry count is not exceeded.

- **Transaction Timeout**

  A Transaction Timeout occurs when the request message – response message scenario does not complete before the total elapsed retry intervals expires (approximately 180 minutes maximum).

- **Failed Application Acceptance after MDN response**

  If an MDN has been sent by the message subsystem indicating successful receipt (non-repudiation) of an incoming message, and then this same message fails application acceptance – cant' be processed by the application system - then human intervention is necessary to deal with ambiguity of the transaction failure on the receiving end while indicating apparent success to the sender. Human intervention is recommended, because this instance will be (hopefully) rare, and there is a subsequent reduction in overall message traffic versus sending acceptance messages every time.

- **Service Levels and Availability**

  8 Second Nominal System Response

  (MDN or message response? ED. Need clarification from Architecture Group) for all participants.

- **180 Minute Worst Case System Response**

  Window waiting for response to three Response Message Timeouts for all participants.

- **45 Hours Cumulative Down Time per Year**

  Service windows will be performed within the 180 minute worst case system response.

The following collaboration sequence depicts the technical specifications for GDSN as defined by the GDSN Architecture Group. The expected Response Application System Actions are:

- On Positive Acceptance: Begin Transaction
- Failed some evaluation: Signal Error
- Passed some evaluation: Respond to the Request
- On Positive Receipt and no expected Acceptance Signal: Transaction success, otherwise fail

Figure 4-2 On Positive Acceptance: Transaction Success, Otherwise Fail

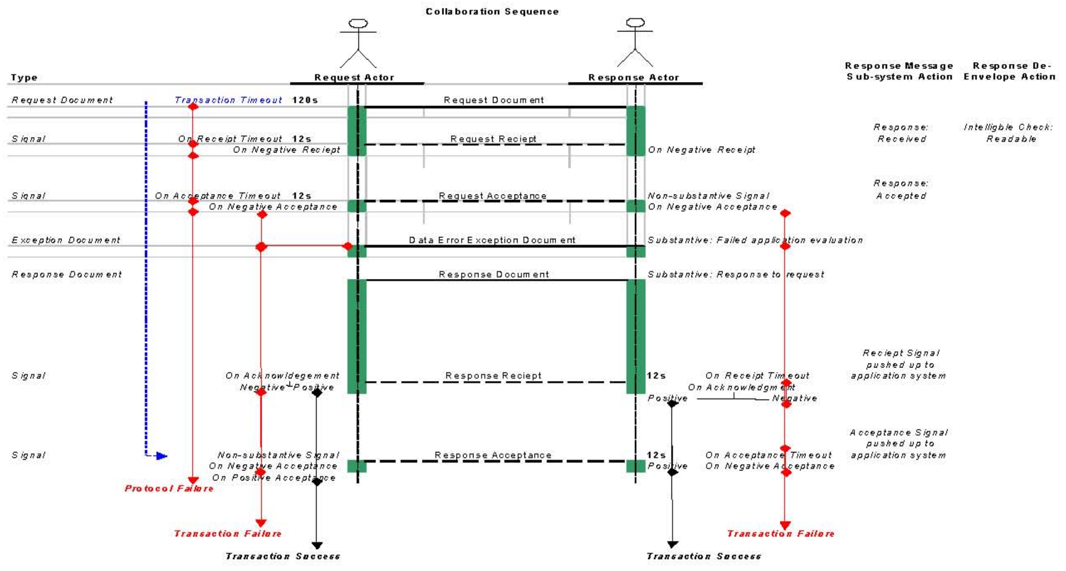


Figure 5. The GDSN Architecture Group GDSN Collaboration Sequence

4.4. Where Are the Standards Located?

Both the Draft and Final Business Message Standards (BMS) from previous, current and new releases can be downloaded from the following link:

http://www.gs1.org/services/gsmp/technical/xml/

GS1 XML schemas and the example instance files for GDSN are placed in Implementer's Packets and made available for download.

4.4.1. Where are the v1.3.2 Schemas Hosted and Located?

In GDSN, the version 1.3.2 schemas are hosted by the GS1 Global Registry in 2 environments, test and production. The URLs of these directories is:

http://www.gs1globalregistry.net/eanucc/

The sample XML instance file below points to the test or production sites, having set the schemaLocation attribute appropriately to locate the schema on the GS1 Global Registry hosting site. The first code snippet shows that the schema location of the Proxy schema can be found under the 1.3.2/ directory. The second snippet shows the location of the schema is not within the 1.3.2 directory but within the extensions part of the directory. This is because an extension, hardlines, is referenced. Because the hardlines major release is v0, the namespace and location reference ean.ucc:hardlines:0. (For more information on extensions and candidate attributes please refer to Section 5, Use of Extensions.)

```
<eanucc:envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:eanucc="http://www.ean-ucc.org/schemas/1.3.2/eanucc" xsi:schemaLocation="http://www.ean-
ucc.org/schemas/1.3.2/eanucc 1.3.2/CatalogueItemNotificationProxy.xsd"
communicationVersion="1.3.2">


<hl:hardlinesTradeItemExtension xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:hl="http://www.ean-ucc.org/eanucc/extensions/hardlines/0"
xsi:schemaLocation="http://www.ean-ucc.org/eanucc/extensions/hardlines/0
extensions/hardlines/0/hardlines.xsd" version="0">
```

4.4.2. Where Are the v2 Schemas Hosted and Located?

V2 schemas are not yet ready to be hosted on the GS1 Global Registry but when they become available the link will be published in an update to this document. Once they are hosted there will be a test and production site as well. The first code snippet shows that the schema location of the Proxy schema can be found under the ean.ucc/align/2.0/ directory. The second snippet shows the location of the schema is not within the 1.3.2 directory but within the extensions part of the directory. This is because an extension, hardlines, is referenced. Because the hardlines major release is v1, the namespace and location reference ean.ucc:hardlines:1.

```
<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
../../../../1_XSD/sbdh/StandardBusinessDocumentHeader.xsd urn:ean.ucc:align:2
../../../../2_XSD_PROXY/ean.ucc/align/2.0/TradeItemProxy.xsd" xmlns:align="urn:ean.ucc:align:2">


<hl:hardlinesTradeItemExtension xmlns:eanucc="urn:ean.ucc:2" xmlns:hl="urn:ean.ucc:hardlines:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.ean-
ucc.org/eanucc/extensions/hardlines/1 ../../../../1_XSD/ean.ucc/hardlines/1/Hardlines.xsd">
```

4.5. What Implementers Should Download

Implementers will need to download the:

- BMS and
- Implementer’s Packet.

The purpose of the BMS is to provide the necessary information to implement a particular message as a part of the GS1 System. The user requirements for the business message document contain the full UML model of the message, the XML instance sample document with its HTML representation and the Global Data Dictionary (GDD) report. The GDD report lists the message model components (classes, role names, enumerated values and attributes), their definitions, cardinality, data field length and title of the XML schema where those components are defined.

The Implementer’s Packet is a ZIP file, comprising all the XML files necessary for validating a given XML message. In v1.3.2 all the schema modules for a release are located in one ZIP file. In v2, the Implementer’s Packets contain the modules required for validating one BMS Message only.

4.6. How the v2 Schemas are Delivered to Users

In version 1.0, all GS1 schemas evolved together, changing from version to version as a group. This meant that all the schemas could be kept in the same directory, organized by version number. For each version number, there was a complete set of schemas. In version 2.0, schema components evolve separately from each other. Schema components with different version numbers may be used together. Different business processes, even business documents within a business process, may use different minor versions of the same schema components.

As an example, assume that:

- Version 2.2 of the Catalogue Item Confirmation (CIC) business document uses version 2.1 of the Party Identification schema (part of the common library).
- Version 2.2 of Catalogue Item Notification (CIN) also uses version 2.1 of Party Identification.
- A new requirement on Party Identification is identified and a new version 2.2 of the Party Identification schema is created with a new value added to one of its code list types.
- CIN must use this new Party identification and therefore CIN is upgraded from version 2.2 to 2.3.
- CIC does not upgrade to version 2.2 of Party Identification; therefore CIC continues to use version 2.1 of Party Identification.

4.6.1. Storing and Maintaining the v2 Schemas

In addition to delivery of the schemas an approach for storing and centrally managing the schemas and related files is provided that is easy to handle. This involves keeping track of major and minor versions, and the relationships between them.

The tracking of minor version numbers and schema file versions that a particular business document uses are accomplished by unzipping the Implementer's Packet. For example, we need to keep track of the fact that version 2.2 of CatalogueItemConfirmation.xsd uses version 2.1 of PartyIdentification.xsd, while version 2.3 of CatalogueItemNotification.xsd uses version 2.2 of PartyIdentification.xsd. To do this, schemas need to be organized into "packets", one per business document, which are intended to be used together to validate that business document.

4.7. The Implementer's Packet

To track the v2 schemas, the Implementer's Packet will be made available to users of the GS1 XML Standard. The Implementer's Packet contains:

- The TableofContents.txt– a text file listing all the files included in the packet. All schemas referenced in the instance or proxy of the instance and all schemas included or imported (directly or indirectly) by those schemas are listed in the Table of Contents. The name of the file will normally take on the name of the first sample instance, unless overridden. For example, CatalogueItemNotificationSample.xml, will generate the manifest with the name CatalogueItemNotification.spp.
- The Instance File folder, containing 1 or more sample XML files for the BMS message
- A Schemas folder containing:
  - GS1 folder with 2 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] – including schemas from the particular business area that are necessary to validate the given business message.
  - SBDH folder – contains the SBDH schemas

- Proxy Schemas –non-normative schemas which facilitate the use of the multiple GS1 XML schemas with certain parsers which do not support multiple schema locations in an instance file. The proxy schema includes and imports the required schema modules for a BMS message and is provided to assist users. If the parser of choice supports multiple schema locations in the instance file, use of the proxy becomes optional. Multiple schemas are required to create an instance document because functionally a business document is impacted by the architecture requiring the presence of other layers – the Standard Business Document Header, transaction and command. Reusability concerns are another reason for multiple schema modules.

Within the packets, version number directories are not used in the directory structure but instead relative paths in the schema locations, in both the instance(s) and the schemas are used. This is due to backward compatibility.

4.7.1. Why are Implementers Packets of v2 Useful?

Three types of information must be tracked in the schema maintenance system:

1. Multiple versions of a schema module. For example, the ability to track both version 2.1 and 2.2 of Party Identification. These two minor versions may be both in use in the same release, by different business documents.
2. Relationships between schema files. For example, the ability to track that CIC is using version 2.1 of Party Identification, while CIN is using version 2.2 of Party Identification. It is not indicated in the business document schemas which minor version of components is used; this is intentional, to allow backward compatibility. Therefore, there needs to be an external mechanism that keeps track of this relationship.
3. Multiple minor versions of sample instances and proxy schemas. These are designed so that, information does not appear in the HTML unless it is present in the instance.

4.7.2. A Sample of An Implementer’s Packet

The name of the packet is: Party_Implementer’s_Packet.zip. From the name, the functional aspect of the schemas can be derived (see section below on File Naming Conventions). The contents of this file are listed in the Table of Contents and are shown here. In addition, the relative path name as displayed in the zip file is added here for further clarification:

Party Implementer's Table of Contents:

	[Relative Path]
Instance File	[Instance File\]
PartySample.xml, version 2.0	
Schemas	
Align	[Schemas\ean.ucc\align\]
FunctionalModeCodeList.xsd, version 2.0	
Party.xsd, version 2.0	
PartyDocument.xsd, version 2.0	
common	[Schemas\ean.ucc\common\]
AccountNumberTypeList.xsd, version 2.0	
AllowanceCharge.xsd, version 2.0	
AllowanceChargeList.xsd, version 2.0	
AllowanceOrChargeList.xsd, version 2.0	
AlternatePartyIdentificationList.xsd, version 2.0	
Command.xsd, version 2.0	
CommunicationChannelCodeList.xsd, version 2.0	
Contact.xsd, version 2.0	
DaysOfTheWeekList.xsd, version 2.1	
Description.xsd, version 2.0	
Document.xsd, version 2.0	
DocumentCommand.xsd, version 2.1	
DocumentCommandList.xsd, version 2.0	
DocumentStatusList.xsd, version 2.0	
EffectiveDateList.xsd, version 2.0	
EntityIdentification.xsd, version 2.0	
Extension.xsd, version 2.0	
FinancialInstitutionInformation.xsd, version 2.0	
Measurement.xsd, version 2.0	
MonetaryAmount.xsd, version 2.0	
NameAndAddress.xsd, version 2.0	
PartyIdentification.xsd, version 2.0	
PartyRoleList.xsd, version 2.0	
PaymentFormatList.xsd, version 2.0	
PaymentMethod.xsd, version 2.0	
PaymentMethodList.xsd, version 2.0	
PaymentTerms.xsd, version 2.0	
PaymentTermsEventList.xsd, version 2.0	
PaymentTermsTypeList.xsd, version 2.0	
Percentage.xsd, version 2.0	
ResponseStatusList.xsd, version 2.0	
RoutingNumberTypeList.xsd, version 2.0	
SettlementList.xsd, version 2.0	
TimePeriodList.xsd, version 2.0	
Transaction.xsd, version 2.0	
sbdh	[Schemas\sbdh]
BasicTypes.xsd	
BusinessScope.xsd	
DocumentIdentification.xsd	
Manifest.xsd	
Partner.xsd	
StandardBusinessDocumentHeader.xsd	
PartyProxy.xsd, version 2.0	[Schemas\]

When the implementer unzips the packet, the files will be located in their respective directories. Each time the Party_Implementer's_Packet.zip is upgraded, the implementer will unzip the new packet in another part of the directory, thus avoiding collisions between files of different versions.

All schemas referenced in the Party packet and included in the Party packet are required for successful validation of the Party sample. The name of the file takes on the name of the PartySample.xml instance. In the sample above, most of the schema modules happen to be in version 2, however, DaysOfTheWeekList.xsd and DocumentCommand.xsd are of version 2.1.

Because this is the first instance of Party.xsd, it is version 2.0. Party.xsd is the main schema module. The version number of the main functional schema module is inputted by the application into the documentStructureVersion and the SBDH Document Identification TypeVersion. In that way the instance document will correctly map to the appropriate Implementer's Packet of schemas. If another one of the referenced modules included or imported into Party.xsd was upgraded, let us say AllowanceOrCharge.xsd was upgraded to version2.4, then Party.xsd will be upgraded to v2.1 and a new Implementer's Packet will be created. The application creating the new modified Party.xml by using the new Implementer's Packet will insert a value of 2.1 into the documentStructureVersion and the SBDH DocumentIdentification TypeVersion.

4.8. File Naming Conventions

GS1 XML schema files shall be named according to the functional aspect of the schema itself and GS1 XML instance document files shall be named according to the functional aspect of the schema followed by the term "Sample".

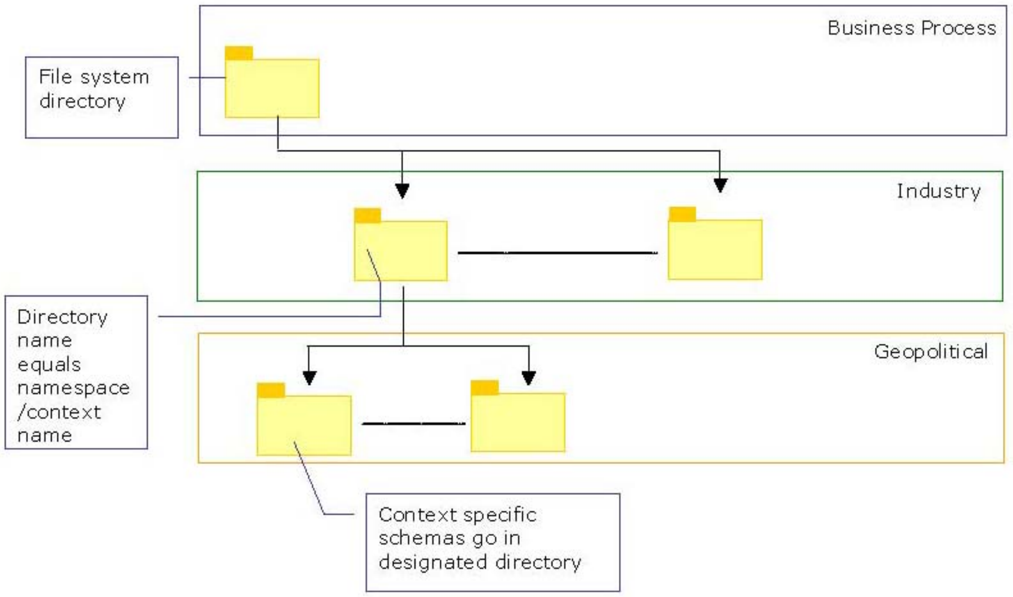
For example:

- A Catalogue Item Notification schema document will be named CatalogueItemNotification.xsd.
- Schemas serving as a library of information components will be given a name that reflects the functional aspect of the library.
- Codelist definition schemas will be given the exact name of the codelist defined within the schema itself (i.e. a country codelist schema will be name CountryCodeList.xsd).

To make GS1 schema documents easier to find and to use, to avoid naming conflicts between contexts, and to control access, they shall be bundled into logical groups of related schemas that share the same context and thus the same namespace. These logical groups of schemas will be identified with their corresponding context identifier/namespace and shall follow the rules of the context category hierarchy. This convention will have a direct impact on how all GS1 source files are distributed and represented on a file system. The path qualifier for the source files (schemas) in a file system will be directly correlated to the context category hierarchy.

GS1 source files will be distributed and represented as a file system logically grouping schemas with their corresponding context identifier namespace. The following diagram illustrates this correlation:

Figure 4-3 File Naming Conventions and File Directory


Business Process
File system directory
Industry
Directory name equals namespace /context name
Geopolitical
Context specific schemas go in designated directory

4.9. Why are Proxy Schemas Useful?

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 GS1 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="../../../../1_XSD/ean.ucc/pay/2.0/RequestForPayment.xsd"/>

<xsd:include schemaLocation="../../../../1_XSD/ean.ucc/common/2.0/ DocumentCommand.xsd"/>

<xsd:include schemaLocation="../../../../1_XSD/ean.ucc/common/2.0/ AllowanceCharge.xsd"/>

<xsd:include schemaLocation="../../../../1_XSD/ean.ucc/common/2.0/ PaymentTerms.xsd"/>

<xsd:include schemaLocation="../../../../1_XSD/ean.ucc/common/2.0/ TradeItemIdentification.xsd"/>

<xsd:include schemaLocation="../../../../1_XSD/ean.ucc/common/2.0/ 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:eanu
```

```
cc="urn:ean.ucc:2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.unece.org/cefact/namespaces/ StandardBusinessDocumentHeader
../../../../1_XSD/sbdh/StandardBusinessDocumentHeader.xsd  urn:ean.ucc:2
../../../../2_XSD_PROXY/ean.ucc/pay/2.0/RequestForPaymentProxy.xsd"
xmlns:pay="urn:ean.ucc:pay:2">
```

4.10. Code Lists

Two types of code lists are found in the GS1 Standard: code lists maintained by external standards organizations and those maintain by EAN.

- Code lists developed and maintained outside of the GS1 System are referenced in the GS1 XML standards. Their content is not copied to the GS1 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 separate XML schemas. Each codelist can be versioned independently from the other schemas, so adding new codes is accomplished by changing the minor version of the schema. The list containing new codes is still backwardly compatible with earlier schemas within the same major version.

4.10.1. Codelists Maintained by External Organizations

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 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.

4.10.2. Codelists Maintained by GS1 in v2

Internal code lists are those developed and maintained within the GS1 System. Starting from release 2.0, those code lists are defined in separate schemas and are no longer embedded in the main schemas. This facilitates backward compatibility because any change to a code list schema does not trigger a new release of the entire set of GS1 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. Advantages for implementers from this approach include reduced maintenance costs, faster implementation of list changes, reduced versions of schemas and simpler publications. The handling of code lists in v2 is consistent with the similar handling of components.

Code lists are defined as xsd:enumeration. The lists are included or imported into the schema document, which uses it. In v2, Code list schemas follow the same principles as other GS1 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.

4.10.3. Versioning of GS1 Code Lists in v2

Each code list is defined in its own schema file, which is versioned separately. When a change is made to a code list schema in v2, 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.

Consistent with the rules defined in the XML Naming and Design Rules Document for v2, 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.

4.11. Responses in GDSN

In addition to the need for a business level Response – a functional Acceptance, Modification, or Rejection of business data, there is a need for the technical requirements and supporting processes for an application level acknowledgement of the receipt of a Message. Optional indication of detected errors are provided in the response and acknowledge BRD.(Please refer to the document, Business Requirements Document (BRD) for XML Application Receipt Acknowledgement And Or Error.) To ensure a reliable flow of information between companies, Business Managers must be assured that their trading partners receive GS1 Messages and are able to process them without error. This choreography or, conversation ensures that trading partners are aware that the process is progressing in a predictable fashion. A proper automated choreography allows trading partners to reduce expensive safeguards and manual checks, to recognize errors quickly, and smooth the flow of goods and services through the supply chain.

4.12. Null Dates

Due to errata in v1.3.2 the following work-around should be used to create an RCI CORRECT message containing a null date, reversing a previous cancellation or discontinuation. The BRD for Validation Rules for GDSN, rule # 106, indicates that one can pass a null value when using a CORRECT command to reset Discontinue Date or Cancel Date. A Change Request to fix this errata will make this work around no longer necessary in v2. The problem is that the current schema does not allow a null or an empty tag for date. Without this capability the feature cannot be passed to the registry or the recipient via RCI or CIN XML. The element should have been made nillable in the schema.

The Interim Solution is described in the following use case scenario:

In GS1 Global Registry implementation, the null dates are currently being handled this way:

1. The GS1 Global Registry receives the ADD RCI message. The item is now registered.
2. Next, the GS1 Global Registry receives the CHANGE/CORRECT RCI message with a cancel date populated.
3. If the supplier wants to rollback the cancel date, they send the CORRECT RCI message with no cancelDate tag populated in the message that will reset the cancel date back to null.

With this work around for version 1.3.2 the GDSN does not have to deal with a complex work around involving populating of hypothetical date or changes to the GS1 Global Registry. The Data Pools should adopt this approach for version 1.3.2. Version 2.0 – the next release of the standard – should allow for nil dates, rendering this workaround obsolete.

4.13. Standard on Transaction, Commands and Documents

When transmitting transactions, commands and documents n GDSN, a consistent convention in the usage of these 3 tags must be followed. There is a distinction between what is technically valid within a well-formed and validated GS1 XML instance file and the desired standard from the business point of view of the GDSN Task Group (TG). This section describes the standard for usage of transactions, commands and documents in the GDSN. In addition, the following restriction applies when sending messages within the GDSN:

- There is a limit of 10 Transactions per messages;
- There is a limit of 100 Documents per Transaction.

Technically it is feasible to create, validate and send various combinations of transactions, commands and documents in a single message “payload”. This technical ability is illustrated by the following hypothetical example, where in one message “envelope” 2 transactions are sent. This example is invalid in GDSN because it goes against the desired business outcome even though it is technically valid.

4.13.1. An Invalid Business Outcome

The first transaction contains one command of type ADD indicating the addition of 2 distinct documents. One document is of type Catalogue Item Notification (CIN) and the other of type Catalogue Item Registration Response (CIRR). The second transaction contains one CORRECT Command associated with one document of type Party and the another command of type DELETE associated with one document of type Catalogue Item Confirmation (CIC).

```
<!—The opening tag of the AS2 ‘envelope”. There is only one envelope tag in a GDSN message.  -->
<?xml version="1.0" encoding="UTF-8"?>
<eanucc:envelope …
<!-- The opening tag of the first transaction -->
<eanucc:transaction> …
<!-- The opening tag of the first command nested in the first transaction -->
<command><eanucc:documentCommand><documentCommandHeader type="ADD">
<!-- The opening tag of the first Catalogue Item Notification document nested in the first command nested in the first transaction. In this case the document type is Catalogue Item Notification. -->
<eanucc:catalogueItemNotification …
<!-- The opening tag of the second Catalogue Item Notification document nested in the first command nested in the first transaction-->
<eanucc:catalogueItemRegistrationResponse …
<!-- The opening tag of the second transaction -->
<eanucc:transaction>…
<!-- The opening tag of the first command nested in the second transaction -->
<command><eanucc:documentCommand><documentCommandHeader type="CORRECT">
<!-- The opening tag of the only document nested in the first command of the second transaction -->
<eanucc:Party  …
<!-- The opening tag of the second command nested in the second transaction -->
<command><eanucc:documentCommand><documentCommandHeader type="DELETE">
<!-- The opening tag of the only document nested in the second command of the second transaction -->
<eanucc:catalogueItemConfirmation
```

4.13.2. A Valid Business Outcome

The following business restrictions on the use of the transactions, commands and document types must be followed whenever an XML document with multiple transactions, commands and documents is created for transmission within the GDSN:

1. Limit 1 command type within 1 transaction.
2. Limit 1 document type within 1 message envelope.
3. Implementation of commands and documents could take the form of:
   a. Wrapping all documents with one command (lines 6-12 above)
   b. Listing the command/document pair multiple times (lines 14-21 above)

If a Data Pool receives a message that does not follow this standard, the receiving Data Pool may reject this message.

In the hypothetical example above, lines 15 and 19 within lines 13-21 are considered invalid and against standards because they mix different command types in one transaction. Additionally, all the document types MUST be of the same type, therefore this example is further invalid. Likewise, lines 17 and 21 within lines 13-21 are considered invalid and against standards because they mix different document types in one transaction.

The following hypothetical transmission is valid:

```
1  <!—The opening tag of the AS2 ‘envelope”. There is only one envelope tag in a GDSN message.  -->
2  <?xml version="1.0" encoding="UTF-8"?>
3  <eanucc:envelope …
4   <!-- The opening tag of the first transaction -->
5   <eanucc:transaction> …
6    <!-- The opening tag of the first command nested in the first transaction -->
7    <command><eanucc:documentCommand><documentCommandHeader type="ADD">
8     <!-- The opening tag of the first Catalogue Item Notification document nested in the first command nested in the first transaction. In this case the document type is Catalogue Item Notification. -->
9     <eanucc:catalogueItemNotification …
10     <!-- The opening tag of the second Catalogue Item Notification document nested in the first command nested in the first transaction-->
11     <eanucc: catalogueItemNotification …
12   <!-- The opening tag of the second transaction -->
13   <eanucc:transaction>…
14    <!-- The opening tag of the first command nested in the second transaction -->
15    <command><eanucc:documentCommand><documentCommandHeader type=" DELETE ">
16     <!-- The opening tag of the only document nested in the first command of the second transaction -->
17     <eanucc: catalogueItemNotification …
18    <!-- The opening tag of the second command nested in the second transaction -->
19    <command><eanucc:documentCommand><documentCommandHeader type="DELETE">
20     <!-- The opening tag of the only document nested in the second command of the second transaction -->
21     <eanucc: catalogueItemNotification
```

4.14. Message Identifiers in GDSN

GS1 response messages are required to pass the same message identifier of the original message. It would be beneficial in debugging efforts for data pools to receive response messages with the same message ID as the original message.

Version 2 schemas contain a tag called **CorrelationInformation RequestingDocumentInstanceIdentifier** in the Standard Business Document Header (see exhibit below for additional detail on how to use this tag in responding documents) which should be used in responding messages. This tag will contain the original instance identifier of the message that is being responded to.

The Task Group proposes that the following attributes, when taken together, must be unique in order to prevent duplicate documents and increase traceability within the GDSN:

- Data Pool GLN / DocumentIdentification InstanceIdentifier / Transaction ID / Command ID /Document ID.

For example:

1. If the same Document ID is used within the same Command the transaction can be failed by the receiving data pool.
2. If the same Command ID is used within the same transaction the transaction can be failed by the receiving data pool
3. If the same Transaction ID is used within the same message the message can be failed by the receiving data pool.
4. A receiving data pool can fail any messages from a sending data pool if the sending data pool has previously sent the receiving data pool a message using the same DocumentIdentification InstanceIdentifier..

4.15. Content Owner

Within the GDSN data pools need to be consistent in populating the *content owner* of a message at the document, transaction and command levels. Based on discussion at the Atlanta Physical Meeting in March 2005 the GDSN Task Group recommends that the content owner at the document, transaction and command levels is populated as the Supplier or Data Source. The rationale is that each document, command and transaction element contains entityIdentification and one or more command(s) and must be used to identify the transaction. The description of Entity Identification in the GS1 Business Message Standards for the Message Layer 1.1 June 2002 is:

Entity identification consists of a unique creator ID and the content owner.

- **UniqueCreatorIdentification -** This is a unique value assigned by the entity owner. UniqueCreatorIdentification must be used.
- **contentOwner** - uniquely identifies the party. It is of type PartyIdentification. ContentOwner must be used.

GDSN Task Group recommends that this definition should be applied in GDSN.

5. Use of Extensions

5.1. What are Extensions to the GS1 Standards and How Do They Work?

The business benefits of the concept of extensions in the GS1 XML Standards are that by defining core and extended parts of the standard schemas, the EDI approach is reversed. EDI utilizes an exhaustive catalogue of possible data elements. Trade organizations constrain the data elements into subsets – such as EANCOM – while users further refine the subsets into profiles.

In contrast, GS1 XML provides individual trading partners, trade organizations and other groups with a legal option – following the EAN-prescribed rules and guidelines for defining extensions - to extend the core components to meet the specific needs of individual countries or industries. The guidelines support the specialization requirements of distinct industries, geographies, usage, business practices and technologies. Extending existing standards creates new standards without having to create them from scratch.

5.2. Candidate Attributes in GDSN

The timely implementation and growth of the scope of GDSN requires a faster development track. To facilitate this, additions or changes to v1.3.2 and v2 can be introduced as a result of a special development process. All the new data components in both of the versions are the so called 'Candidate Attributes', implemented as extensions. The process of their developments, testing, review and approval is called Candidate Attributes. Candidate Attributes are defined as:

- One or more new attributes currently not in the standard that are requested for piloting or implementation purposes or
- An existing attribute that has been approved as a standard, but the standard has not yet been adopted by the GDSN.

The Candidate Attributes can be:

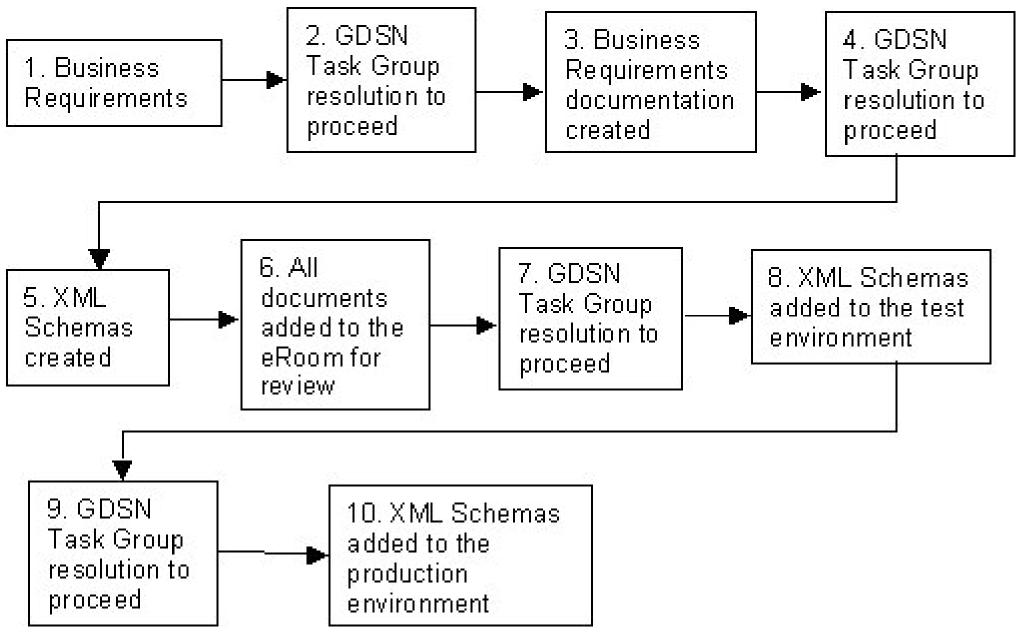
- New attributes currently not in the standard, requested for piloting/implementation purposes
- Existing attributes that have been approved as a standard, but the standard has not yet been adopted by the GDSN

The benefits of Candidate Attributes are that they are a means to:

- Develop business requirements without halting ongoing implementations that need them;
- Fulfill many requests for development;
- Expand the functions of the GDD;
- Or ensure that the global process meets market and retailer demands.

The detailed process for creating a Candidate Attribute is:

Figure 5-1 The Detailed Process for Developing Candidate Attributes


1. Business Requirements
2. GDSN Task Group resolution to proceed
3. Business Requirements documentation created
4. GDSN Task Group resolution to proceed
5. XML Schemas created
6. All documents added to the eRoom for review
7. GDSN Task Group resolution to proceed
8. XML Schemas added to the test environment
9. GDSN Task Group resolution to proceed
10. XML Schemas added to the production environment

1. A user submits their business requirements to the GDSN Task Group for review
2. The GDSN Task Group approves the creation of the documentation that supports the business requirements. In order to proceed a Change Request must be entered into GSMP to add the submitted requirements to the standards
3. GS1 staff works with the submitter and GDSN TG to create the proper documentation
4. The GDSN Task Group approves the documentation and agrees that it does meet the users needs
5. 5. XML schemas and sample instance files are developed to support the users' needs
6. 6. The documentation to support the user's needs and the schemas are posted to the GDSN Task Group's eRoom (Align Data Business Requirement Group (BRG) > GDSN Task Group > GDSN Documentation > Candidate Attributes > Step 1: Under Review) and a notification is sent to the team. This begins a 14 calendar day review period for the schemas.
7. 7. The GDSN Task Group approves the XML Schemas and confirms their placement into the test environment of the GDSN.
8. 8. The XML schemas are posted to the GDSN Task Group's eRoom (Align Data Business Requirement Group (BRG) > GDSN Task Group > GDSN Documentation > Candidate Attributes > Step 2: In the Test Environment) and a notification is sent to the team. This begins a 30-calendar day testing period for the schemas.
9. 9. The GDSN Task Group approves the XML Schemas and confirms their placement into the production environment of the GDSN. This is decided by the voting of the GDSN Task Group members. The voting process takes 14 days.
10. 10. The XML Schemas are posted to the GDSN Task Group's eRoom (Align Data Business Requirement Group (BRG) > GDSN Task Group > GDSN Documentation > Candidate Attributes > Step 3: In Production). The schemas are available for the implementation in GDSN.

The Change Request for Candidate Attributes has to contain the following information:

- Business Rationale for the addition of the attributes
- Commitment to implement from at least two trading partners and their Data Pool(s)

- Adequate documentation as to the definition and usage
- List of potential validation rules

The schemas for them are hosted on the GS1 Global Registry site. Once the Candidate Attributes are approved by the Align BRG they are upgraded from candidate attributes to standards. The use of Candidate Attributes in GDSN message exchanges is always optional.

5.3. Overview of the Extension Element in version 1.3.2 and v2

The extension element is introduced in GDSN v1.3.2 schemas and its use continues in v2.0. The extension element is a placeholder for context specific schemas and is optional. In GDSN v1.3.2, the extension contents are skipped over by the parser, however, in v2, if no schema definition for the extension components is found, the validator will not report an error. If the schema definition is found, the parser will perform validation. This is because the validation is upgraded to lax.

The content inside the extension element is subject to approval by the GSMP business and technical users following the approval process described above. Candidate Attribute Extension versions are independent of BMS versions since each will be enhanced independently. Use of an attribute within the extension to hold the version number is recommended. Recipients need to reference this version number when multiple versions of the same extension are sent on the same message (to preserve compatibility with multiple data pools migrating at different speeds. Consequently, the menu path at the GS1 Global Registry hosted site for the extensions should be independent of the menu path for the BMS schema (since the versions are independent). A few examples are provided:

www.gdsregistry.org:eanucc:extensions:itemIdentification barcodeextension.xsd

www.gdsregistry.org:eanucc:extensions:itemIdentification:1 otherextension.xsd

www.gdsregistry.org:eanucc:extensions:itemIdentification:2 barcodeextension.xsd

5.3.1. Sample of the Hardlines Extension Schema of v1.3.2

Note the differences between the schemas for hardlines extension v1.3.2 and v2 (highlighted in bold).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ean-ucc.org/eanucc/extensions/hardlines/0"
xmlns:eanucc="http://www.ean-ucc.org/schemas/1.3.2/eanucc" xmlns="http://www.ean-
ucc.org/eanucc/extensions/hardlines/0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
 <xsd:include schemaLocation="../../../1.3.2/Components.xsd"/>
 <xsd:import namespace="http://www.ean-ucc.org/schemas/1.3.2/eanucc"
schemaLocation="../../../1.3.2/TradeItemComponents.xsd"/>
 <xsd:element name="hardlinesTradeItemExtension" type="HardlinesTradeItemExtensionType"/>
 <xsd:complexType name="HardlinesTradeItemExtensionType">
  <xsd:sequence>
   <xsd:element name="isTradeItemRecalled" type="xsd:boolean"/>
   <xsd:element name="modelNumber" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="piecesPerTradeItem" type="MeasurementType" minOccurs="0"/>
   <xsd:element name="nestingInformation" type="NestingInformationType" minOccurs="0"
maxOccurs="unbounded"/>
```

```
   <xsd:element name="outOfBoxInformation" type="OutOfBoxInformationType" minOccurs="0"
maxOccurs="unbounded"/>
   <xsd:element name="tradeItemFinish" type="TradeItemFinishType" minOccurs="0"/>
   <xsd:element name="rightOfReturnForNonSoldTradeItem" type="RightOfReturnForNonSoldTradeItemType"/>
   <xsd:element name="securityTagInformation" type="SecurityTagInformationType" minOccurs="0"/>
   <xsd:element name="tradeItemImportIdentification" type="TradeItemImportIdentificationType" minOccurs="0"
maxOccurs="unbounded"/>
   <xsd:element name="warrantyInformation" type="WarrantyInformationType" minOccurs="0"/>
   <xsd:element name="orderingAndSellingUnitOfMeasure" type="OrderingAndSellingUnitOfMeasureType"
minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="version" type="xsd:string" use="required" fixed="0"/>
 </xsd:complexType>
 <xsd:complexType name="NestingInformationType">
  <xsd:sequence>
   <xsd:element name="nestingIncrement" type="MeasurementType"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="OrderingAndSellingUnitOfMeasureType">
  <xsd:sequence>
   <xsd:element name="orderingUnitOfMeasure" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="sellingUnitOfMeasure" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="OutOfBoxInformationType">
  <xsd:sequence>
   <xsd:element name="depth" type="MeasurementType"/>
   <xsd:element name="height" type="MeasurementType"/>
   <xsd:element name="width" type="MeasurementType"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="RightOfReturnForNonSoldTradeItemType">
  <xsd:sequence>
   <xsd:element name="returnGoodsPolicy" type="RightOfReturnForNonSoldTradeItemCodeList"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="SecurityTagInformationType">
  <xsd:sequence>
   <xsd:element name="securityTagLocation" type="SecurityTagLocationCodeTypeList" minOccurs="0"/>
   <xsd:element name="securityTagType" type="SecurityTagTypeList" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="TradeItemFinishType">
  <xsd:sequence>
   <xsd:element name="tradeItemFinishDescription" type="DescriptionType"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="TradeItemImportIdentificationType">
  <xsd:sequence>
   <xsd:element name="importClassificationType" type="ImportClassificationTypeList" minOccurs="0"/>
   <xsd:element name="importClassificationValue" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="11"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="WarrantyInformationType">
  <xsd:sequence>
   <xsd:element name="urlForWarranty" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="1000"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="warrantyDescription" type="eanucc:LongTextDescriptionType" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:simpleType name="ImportClassificationTypeList">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="CUSTOMS_TARIFF_NUMBER"/>
   <xsd:enumeration value="HARMONIZED_COMMODITY_DESCRIPTION_AND_CODING_SYSTEM"/>
   <xsd:enumeration value="HARMONIZED_TARIFF_SCHEDULE_OF_THE_US"/>
   <xsd:enumeration value="INTRASTAT"/>
   <xsd:enumeration value="INTRASTAT_COMBINED_NOMENCLATURE"/>
   <xsd:enumeration value="NETHERLANDS"/>
   <xsd:enumeration value="TARIF_INTEGRE_DE_LA_COMMUNAUTE"/>
  </xsd:restriction>
 </xsd:simpleType>
 <xsd:simpleType name="RightOfReturnForNonSoldTradeItemCodeList">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="CALL_FOR_AUTHORIZATION"/>
   <xsd:enumeration value="DESTROY_FOR_CREDIT"/>
   <xsd:enumeration value="HOLD_FOR_INSPECTION"/>
   <xsd:enumeration value="RETURN_FOR_CREDIT"/>
  </xsd:restriction>
```

```
</xsd:simpleType>
<xsd:simpleType name="SecurityTagLocationCodeTypeList">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="CONCEALED_INSIDE_THE_TRADE_ITEM"/>
  <xsd:enumeration value="INTEGRATED_INSIDE_OF_TRADE_ITEM"/>
  <xsd:enumeration value="ON_OUTSIDE_OF_TRADE_ITEM"/>
 </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="SecurityTagTypeList">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="ACOUSTO_MAGNETIC_EAS_TAG"/>
  <xsd:enumeration value="ELECTRO_MAGNETIC_EAS_TAG"/>
  <xsd:enumeration value="INK_OR_DYE_EAS_TAG"/>
  <xsd:enumeration value="MICROWAVE_EAS_TAG"/>
  <xsd:enumeration value="RADIO_FREQUENCY_EAS_TAG"/>
 </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
```

5.3.2. Sample of the Hardlines Extension Schema of v2

Note the differences between the schemas for hardlines extension v1.3.2 and v2 (highlighted in bold).

1. Namespace differs from v1.3.2 to v2
2. Measurement and Description have been removed to their own physical files in v2
3. ImportClassificationTypeList**,** RightOfReturnForNonSoldTradeItemCodeList, SecurityTagLocationCodeTypeList andSecurityTagTypeCodeList have been removed to their own physical files in v2

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:ean.ucc:hardlines:1"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:hl="urn:ean.ucc:hardlines:1"
xmlns:align="urn:ean.ucc:align:2" xmlns:eanucc="urn:ean.ucc:2"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
 <xsd:import namespace="urn:ean.ucc:2"
schemaLocation="../../common/2.0/Measurement.xsd"/>
 <xsd:import namespace="urn:ean.ucc:2"
schemaLocation="../../common/2.0/Description.xsd"/>
 <xsd:include schemaLocation="ImportClassificationTypeCodeList.xsd"/>
 <xsd:include schemaLocation="RightOfReturnForNonSoldTradeItemCodeList.xsd"/>
 <xsd:include schemaLocation="SecurityTagLocationCodeTypeCodeList.xsd"/>
 <xsd:include schemaLocation="SecurityTagTypeCodeList.xsd"/>
 <xsd:element name="hardlinesTradeItemExtension" type="hl:HardlinesTradeItemExtensionType"/>
 <xsd:complexType name="HardlinesTradeItemExtensionType">
  <xsd:sequence>
   <xsd:element name="hardlinesTradeItem" type="hl:HardlinesTradeItemType"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="HardlinesTradeItemType">
  <xsd:sequence>
   <xsd:element name="isTradeItemRecalled" type="xsd:boolean"/>
   <xsd:element name="modelNumber" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="piecesPerTradeItem" type="eanucc:MeasurementType" minOccurs="0"/>
   <xsd:element name="nestingInformation" type="hl:NestingInformationType" minOccurs="0"
maxOccurs="unbounded"/>
   <xsd:element name="outOfBoxInformation" type="hl:OutOfBoxInformationType" minOccurs="0"
maxOccurs="unbounded"/>
   <xsd:element name="tradeItemFinish" type="hl:TradeItemFinishType" minOccurs="0"/>
   <xsd:element name="rightOfReturnForNonSoldTradeItem"
type="hl:RightOfReturnForNonSoldTradeItemType"/>
   <xsd:element name="securityTagInformation" type="hl:SecurityTagInformationType"
minOccurs="0"/>
   <xsd:element name="tradeItemImportIdentification" type="hl:TradeItemImportIdentificationType"
minOccurs="0" maxOccurs="unbounded"/>
   <xsd:element name="warrantyInformation" type="hl:WarrantyInformationType" minOccurs="0"/>
   <xsd:element name="orderingAndSellingUnitOfMeasure"
type="hl:OrderingAndSellingUnitOfMeasureType" minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="NestingInformationType">
  <xsd:sequence>
   <xsd:element name="nestingIncrement" type="eanucc:MeasurementType"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="OrderingAndSellingUnitOfMeasureType">
  <xsd:sequence>
   <xsd:element name="orderingUnitOfMeasure" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="sellingUnitOfMeasure" minOccurs="0">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:minLength value="1"/>
      <xsd:maxLength value="70"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
```

```
<xsd:complexType name="OutOfBoxInformationType">
 <xsd:sequence>
  <xsd:element name="depth" type="eanucc:MeasurementType"/>
  <xsd:element name="height" type="eanucc:MeasurementType"/>
  <xsd:element name="width" type="eanucc:MeasurementType"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RightOfReturnForNonSoldTradeItemType">
 <xsd:sequence>
  <xsd:element name="returnGoodsPolicy"
type="hl:RightOfReturnForNonSoldTradeItemCodeListType"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SecurityTagInformationType">
 <xsd:sequence>
  <xsd:element name="securityTagLocation" type="hl:SecurityTagLocationCodeTypeListType"
minOccurs="0"/>
  <xsd:element name="securityTagType" type="hl:SecurityTagTypeListType" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TradeItemFinishType">
 <xsd:sequence>
  <xsd:element name="tradeItemFinishDescription" type="eanucc:DescriptionType"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TradeItemImportIdentificationType">
 <xsd:sequence>
  <xsd:element name="importClassificationType" type="hl:ImportClassificationTypeListType"
minOccurs="0"/>
  <xsd:element name="importClassificationValue" minOccurs="0">
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:minLength value="1"/>
     <xsd:maxLength value="11"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="WarrantyInformationType">
 <xsd:sequence>
  <xsd:element name="urlForWarranty" minOccurs="0">
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:minLength value="1"/>
     <xsd:maxLength value="1000"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
  <xsd:element name="warrantyDescription" type="eanucc:LongTextDescriptionType"
minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

5.4. Context

Extension mechanisms are required to extend core components to handle conflicts in user requirements. Context provides a means to classify differences in requirements based on specified processes, industries, product classifications, geopolitical regions and other criteria. It is then used to classify extension schemas based on a context classification.

EbMethodology identifies the following context classifications within the current context classification scheme:

- Business Process: business process as described in the GS1 reference model
- Product Classification: based on the types of goods or services being exchanged
- Industry Classification: the industry or sub-industry in which a business process takes place.
- Geopolitical: related to region, nationality, or geographically based cultural factors
- Official Constraints: legal and governmental influences
- Business Process Role: the actors conducting a particular business process
- Supporting Role: requirements of non-partner roles
- System Capabilities: limitations of systems

When a context specific requirement is discovered by an analyst, a context set will be created (for example CPG North America). This context set will be then applied to the requirements and in some cases the resulting schema extension.

In the GS1 XML standards, namespaces are used for reflecting context. In the GS1 System it is mandatory to use the standard namespace prefixes, in order to support the global interoperability of the standards.

5.5. GS1 Core Schemas in v2

The namespaces in GS1 XML V2 reflect the context and 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:ean.ucc:_____:_____:_____:_____

BP IC GP Major Version

where:

- BP – Business Process
- IC – Industry Classification
- GP – Geopolitical Context

In the case of extensions, the version number reflects the major version of the extension and is not tied to the version number of the release. This principle is the same for v1.3.2 and v2.

5.5.1. GS1 Namespace Prefix

GS1 XML core or non-extension schemas do not use a 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:

- "align" for ALIGN process, e.g.: **align**="urn:ean.ucc:align:2".

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

Naming conflicts occur when two entities are given the same name. For example, if both a simple and complex W3C type are given the same name, there will be a conflict. To avoid naming conflicts, the EAN.UCC XML Standard uses the following guidelines:

- Use the same component name from different modules for a schema made up of several schema modules.
- Place each named element into separate namespaces.

An <any> element enables XML documents to add new elements. It allows the XML Instance designer to add new elements “on-demand” rather than by consensus of the standard body. Extensibility is controlled by setting maxOccurs. The <any> element allows a designer to dynamicaly declare the namespace in the namespace placeholder, to postpone decisions until implementation of the XML Instance document and to dynamically add elements to a namespace to meet future needs. The importance of this is illustrated by the following example: the XML Designer can describe the Item as both a “hardlines” type of item and an “fmcg” type of item in one instance.

In v1.3.2 the CatalogueItemNotification and in v2 all EAN.UCC XML schemas contain the optional extension element that acts as a placeholder for non-generic, standard business domain–specific attributes. These are implemented as an optional, repeating XML schema <any> element within the GS1 Document layer acting as a placeholder for context specific elements. The schema is implemented using an xsd:any element with a namespace attribute set to *##any* and a *processContents* attribute set to *lax (skip in version 1.3.2)*.

Example:

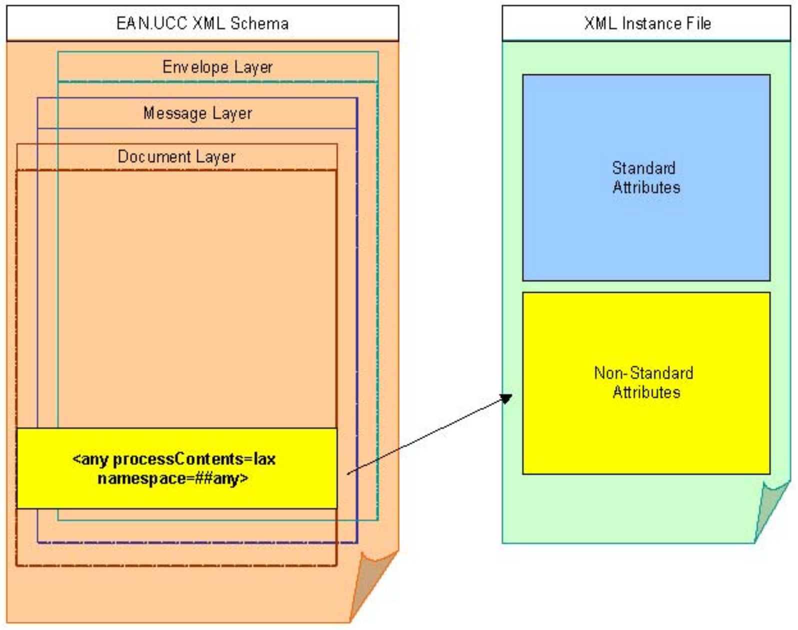
```
<xsd:complexType name="ExtensionType"><xsd:sequence><xsd:any namespace="##any"
processContents="lax" minOccurs="0"

maxOccurs="unbounded"/></xsd:sequence>

</xsd:complexType>
```

If no domain-specific attributes are expected in the partner-to-partner message, the extension tag will bypass XML schema validation. The approach is shown in the following diagram:

Figure 5-2 Extensions in GS1 schemas and XML



The following example is of the v1.3.2 CatalogueItemNotificationSample.xml Instance Document snippet and shows 2 separate extensions – bar code and hardlines extensions:

```
<extension>
 <iid:barCodeExtension xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:iid="http://www.ean-ucc.org/eanucc/extensions/itemidentification/1"
xsi:schemaLocation="http://www.ean-ucc.org/eanucc/extensions/itemidentification/1
barCodeExtension.xsd" version="1">
  <isBarCodeDerivable>false</isBarCodeDerivable>
  <eanuccCode type="UN">12345678912343</eanuccCode>
  <eanuccCode type="UP">123456789012</eanuccCode>
 </iid:barCodeExtension>
 <hl:hardlinesTradeItemExtension xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:hl="http://www.ean-ucc.org/eanucc/extensions/hardlines/0"
xsi:schemaLocation="http://www.ean-ucc.org/eanucc/extensions/hardlines/0 hardlines.xsd"
version="0">
  <isTradeItemRecalled>false</isTradeItemRecalled>
  <modelNumber>123</modelNumber>
  <piecesPerTradeItem>
   <measurementValue unitOfMeasure="CAS">10</measurementValue>
  </piecesPerTradeItem> ...
 </hl:hardlinesTradeItemExtension>...
</extension>
```

5.6. Validation of 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 non-standard information; therefore it is important for interoperability to follow GS1 Standards directives. It is recommended to use only the extension attributes approved within the GSMP process.

In v1.3.2 the parser is set to lax validation of extensions. This means that no validation takes place. In v2, because the processContents attribute is set to 'lax', the validation of the extension element is optional: a 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.

5.7. Qualification of 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 recognizable in the instance document.

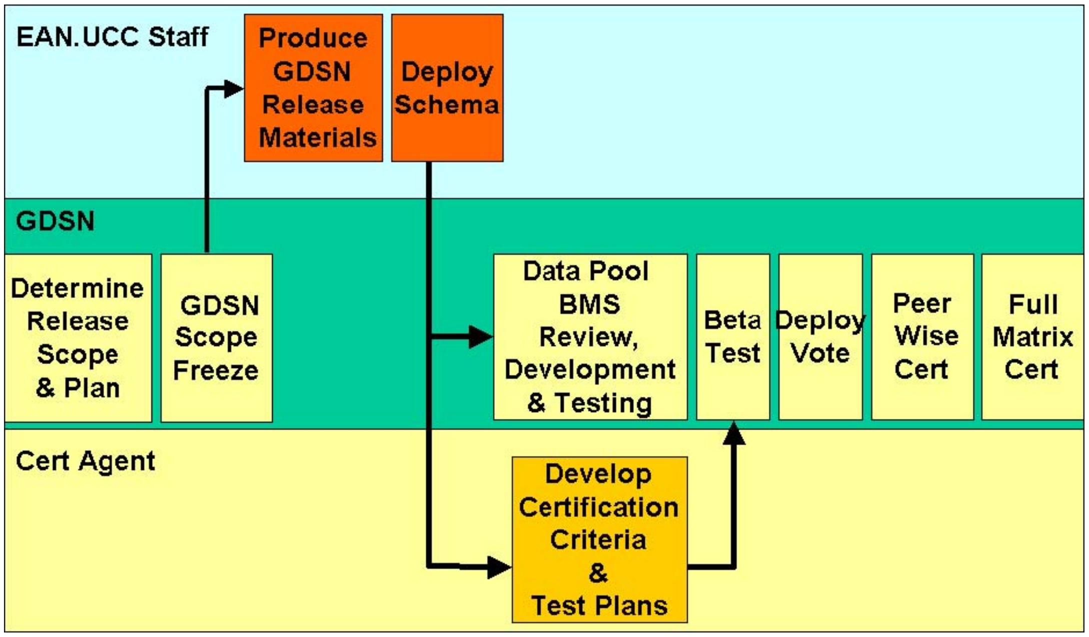
6. Change Management

The change management process for GDSN standards will entail the following steps:

- Determine Release Management
- Freeze Project Scope
- Produce GDSN Release Materials
- Deploy Schema
- Data pool and GS1 Global Registry Internal Development/Testing
- Development of Certification Criteria and Plans
- Beta Test
- Deployment Vote
- Peer Wise Certification
- Full Matrix Certification

The process is depicted in Figure 6-1:

Figure 6-1 The Change Management Process


EAN.UCC Staff
Produce GDSN Release Materials
Deploy Schema
GDSN
Determine Release Scope & Plan
GDSN Scope Freeze
Data Pool BMS Review, Development & Testing
Beta Test
Deploy Vote
Peer Wise Cert
Full Matrix Cert
Cert Agent
Develop Certification Criteria & Test Plans

Each of these steps has associated timelines, processes and documentation outputs.

Determine Release Scope and Create Release Plan

This phase involves the following steps:

- **Determine Release Content/Scope:**

  GDSN TASK GROUP DETERMINES THE FUNCTIONALITY THAT IS TO BE INCLUDED WITHIN THE NEXT RELEASE. THIS FUNCTIONALITY CAN BE DEVELOPED WITHIN THE GDSN TASK GROUP, OR CAN BE DELIVERED TO THE GDSN FOR INCORPORATION INTO THE STANDARD FROM OTHER BRGS (FOR EXAMPLE RELATIONSHIP DEPENDENT DATA FOR PRICE SYNCHRONISATION.

- **Determine Release Impact (Major/Minor):**

  Based on the functionality targeted for the next release, the release will be labelled major or minor. The following criteria determines that the release is major:

  - Attribute changes that effect the core message
  - Changes to existing GDSN choreography.

  The following criteria determines that a release is minor:

  - New choreography that does not impact existing choreography.
  - New functionality that does not raise restraint of trade issues.
  - New extensions that do not affect the core message.

  A detailed impact analysis should also be drafted determining the impact of the release on existing messages and processes.

- Determine Release Date

  The release date is determined based on whether the release is deemed major or minor and other external factors and dependencies (resources, effect on other projects, industry need).

- Determine Version Number:

Determine the release's version number based on whether the release is deemed major or minor.

- Develop Release Milestones/Timeline:

Release milestones to be determined include the planned certification period, date of Go/No Go meeting (to measure readiness of release process), date of Production cut over, Sunset date of old versions/end of concurrent version support period, and Trading Partner Migration Period.

GDSN Scope Freeze

After determining release content, remaining standards development work will be completed. Once completed, work is frozen and packaged for publication. Changes introduced after a scope freeze are not certified until the "next + 1" certification event.

Development and Testing Process

There are 3 periods to the implementation window.

The sub windows are:

Implementation Window	Description
Development Window	The period of time when Data Pools, GS1 Global Registry or Trading Partners build out their systems in preparation for testing.
Testing Window	The period of time when Data Pools, GS1 Global Registry or Trading Partners interface their systems by sending standard messages to each other in an end-to-end or system test.
Implementation Window	The period of time when the Data Pools, GS1 Global Registry or Trading Partners bring the new CR or version into production and quiesce the older version. During this time two or more versions may be in production simultaneously.

During the Implementation Windows development and Testing, the Data Pools and GS1 Global Registry will use that time to make announcements to their communities and educate the communities on how to use the new feature or version.

For each specific implementation event, the time periods for each sub-window will vary based on release type (Major or Minor).

6.1. Develop Certification Criteria and Test Matrix

Test plans are developed by a certification agent based upon latest approved standard and determination of test instances by GDSN Task Group. Process starts after new standard materials are created.

6.2. Beta Test

Approved functionality is beta tested between data pools and a Beta Test Results Report is issued. The Beta Test Result Report details the learnings from the Beta Test and will result in input to the standard, certification test instances and implementation guides.

6.3. Certification

Still to be completed.

6.4. Strategy for Versions 1.3.2 and 2

Upgrades of releases or versions in GDSN need to be considered. Currently, GDSN is standardizing on version 1.3.2 while GS1 is working on publishing version 2. The upgrade to a new version raises many questions.

6.4.1. Proposed Versioning Mechanism and Backward Compatibility

GS1 schemas will institute the following versioning mechanisms:

- Separation of individual XML schema versions from release naming.
- Incorporating a major and minor versioning strategy.
- Design of schemas such that all minor versions within a major version are backwardly compatible with each other.

The definition of backward compatibility defined by ITRG is:

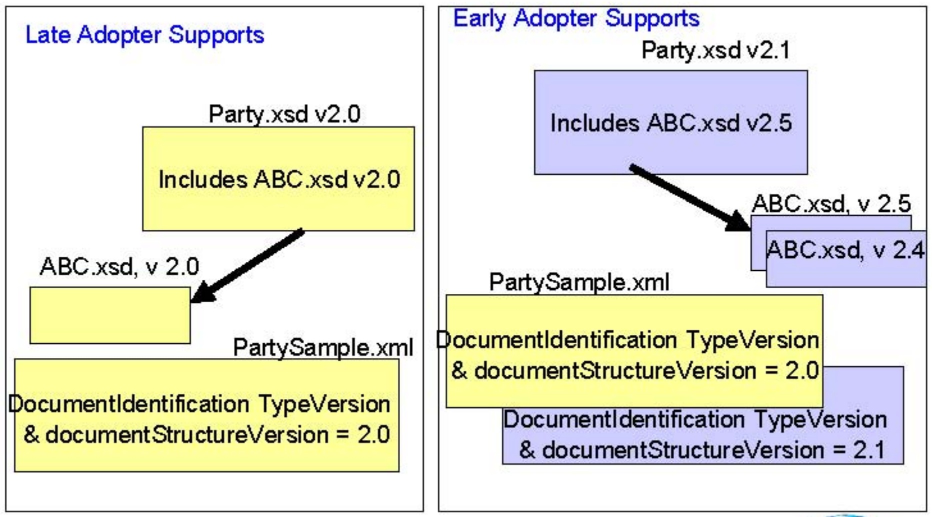
Backward compatibility is defined as when a message sender can create a Business Document based on an old schema, dispatch it to a recipient, who will validate it against a new schema without it being failed by the recipient’s schema validator.

Note: GS1 schemas will not support forward compatibility. They will only support backward compatibility.

Forward Compatibility Definition: The ability to design schemas such that even the older schema can validate the instance documents created according to the newer version called *forward compatibility*.]

Versions 1.3.2 are not backwardly compatible with version 2. Starting with v2, all versions in the GS1 XML standards are reflected either as major or minor versions. All minor versions are backward compatible within the same major version. The impact on BRDs and Schemas will be that there may be an interval before the schemas match the BRD 100%. An early Adopter can validate both XML instances, 2.0 and 2.1, as shown in the next figure.

Figure 6-2 Early and Late Adopters of Minor Versions.


Late Adopter Supports
Party.xsd v2.0
Includes ABC.xsd v2.0
ABC.xsd, v 2.0
PartySample.xml
DocumentIdentification TypeVersion & documentStructureVersion = 2.0
Early Adopter Supports
Party.xsd v2.1
Includes ABC.xsd v2.5
ABC.xsd, v 2.5
ABC.xsd, v 2.4
PartySample.xml
DocumentIdentification TypeVersion & documentStructureVersion = 2.0
DocumentIdentification TypeVersion & documentStructureVersion = 2.1

6.4.1.1. 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 the 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.

6.4.1.2. Version Numbers in XML Instance Business Documents

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.

GS1 business documents use the "documentStructureVersion" attribute defined in the DocumentType complex type and the SBDH Document Identification TypeVersion to reflect the complete version of the business document schema, e.g. for the Order.xml message, the " documentStructureVersion and the SBDH Document Identification TypeVersion" 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.

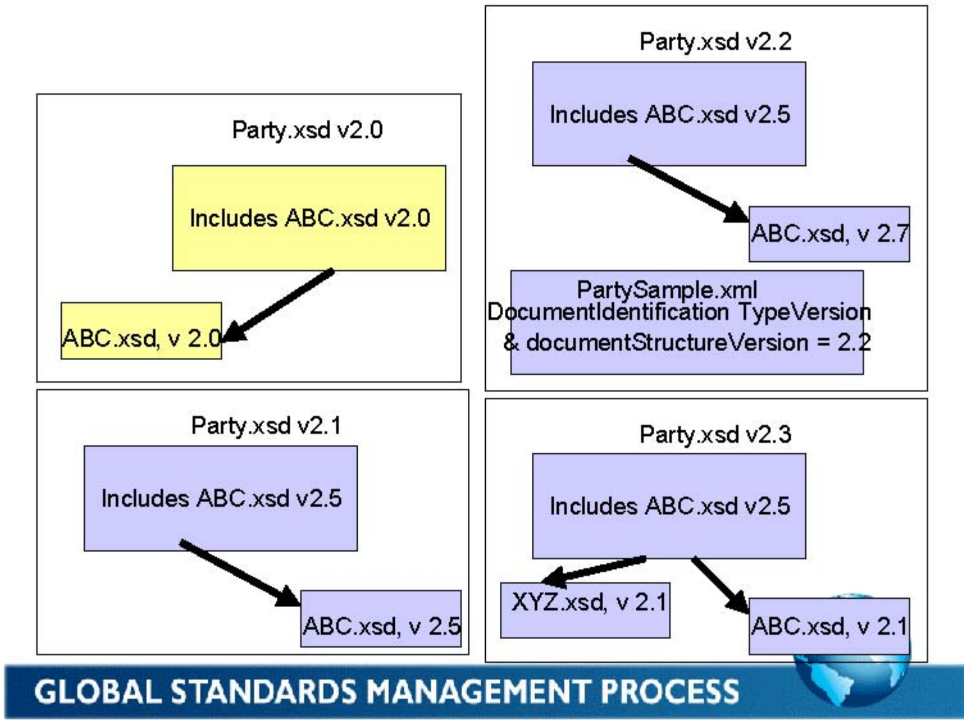
This type is defined as:

```
<xsd:complexType name="DocumentType" abstract="true">
<xsd:attribute name="contentVersion" type="eanucc:VersionType" default="2.0"/>
<xsd:attribute name="documentStructureVersion" type="eanucc:VersionType"/>
 <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.

The minor schema modules will be created as specified by the Business Requirements Analysis Document (BRAD) or changes to the BRAD This is shown in the next figure. In this case, 4 BRADs or BRAD changes would be created resulting in 4 distinct Implementer's Packets. The main modules of each would have a different major.minor version number because the main module has been changed by referencing a new or modified included or imported schema. Thus the figure depicts Party.xsd versions 2.0, 2.1, 2.2, and 2.3. The included schema, ABC.xsd likewise carries a different major.minor version number. These 4 minor versions of the Party.xsd would be delivered to the web site or to the hosted site in its entirety. The Implementer's Packet that conforms to the BRAD complying with the user's need will be selected by the user and the user will embed those schemas into their solution.

Figure 6-3 Versioning of Minor Schema Modules and Main Schemas



How will a receiver know whether the sender is an early or late adopter? Knowing this has implications on the choreography of message flows during the window when 2 (or more) minor versions are supported. The receiver of a message will know which version to respond to the sender by checking the incoming DocumentIdentification TypeVersion & documentStructureVersion. It is the responsibility of the sender to update the tags to be an identical match to the major.minor version of the main schema module in use.

6.4.1.3. Version Numbers in Business Documents and SBDH

The SBDH 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.

7. References

- AS2 Transport Communications Guide for the GS1 GDSN Community" (version 1.1 refers to v1.3.2 schemas and version 2.0 refers to the version 2.0 schemas).
- Business Message Standards, version, 1.3.1.

- Business Requirements Document (BRD) For XML Application Receipt Acknowledgement & Error, Draft, Date: 2004-09-10.
- GS1 Business Message Standards for the Message Layer 1.1, June 2002
- ML Architectural Guide 2.0, October 2004 (version 1.1 refers to v1.3.2 schemas and version 2.0 refers to the version 2.0 schemas).
- GS1 Global Registry Data Validation Rules.
- UN/CEFACT Standard Business Document Header Technical Specification Version 1.3.
- XML Design Rules for EAN.UCC, Version 2.0, August 2004.
- GDSN – The way to go NOW!, October 2004