



GS1 XML Message Architecture Implementation Guide

Issue 1, July-2009



Document Summary

Document Item	Current Value
Document Title	GS1 XML Message Architecture Implementation Guide
Date Last Modified	July-2009
Current Document Issue	Issue 1
Status	Approved
Document Description	This document is intended to clarify the function, structure and usage of the GS1 XML Message Layered Architecture. The document can be used by technical implementers that need to know how to implement the GS1 XML Message Architecture in their environment.

Contributors

Name	Organization
Ewa Iwicka	GS1
GSMP eTG Working Group members	GS1

Log of Changes in Issue 1

Issue No.	Date of Change	Changed By	Summary of Change
1	July 2009	Ewa Iwicka	1st Issue

Disclaimer

Whilst every effort has been made to ensure that the guidelines to use the GS1 standards contained in the document are correct, GS1 and any other party involved in the creation of the document HEREBY STATE that the document is provided without warranty, either expressed or implied, of accuracy or fitness for purpose, AND HEREBY DISCLAIM any liability, direct or indirect, for damages or loss relating to the use of the document. The document may be modified, subject to developments in technology, changes to the standards, or new legal requirements. Several products and company names mentioned herein may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

1. Introduction	4
1.1. Purpose.....	4
1.2. Scope	4
1.3. Target Audience.....	4
2. Message Architecture Principles	4
2.1. GS1 Message Architecture	4
2.2. Benefits	4
2.3. Overview of Message Layers.....	5
2.3.1. Transport and Routing Layer.....	5
2.3.2. Service Layer.....	6
2.3.3. Business Document Layer	9
3. Message Layer Details.....	9
3.1. Transport and Routing Layer Details	9
3.2. Service Layer Details	9
3.2.1. Message Component Details	9
3.2.2. Transaction Component Details	11
3.2.3. Document Command Component Details.....	13
3.3. Business Document Layer Details	15
3.3.1. Business Document Component Cetails.....	16
3.4. Batching	17
3.4.1. Batching of Transactions.....	18
3.4.2. Batching of Commands	19
3.4.3. Batching of Transactions and Commands	21
3.4.4. Batching of Documents	23
4. Appendix: Message Layer Usage in Business Scenario - Example	24

1. Introduction

1.1. Purpose

This document is intended to clarify the function, structure and usage of the GS1 XML Message Layered Architecture. The document can be used by technical implementers that need to know how to implement the GS1 XML Message Architecture in their environment.

1.2. Scope

This guide is based entirely on the GS1 XML Message Architecture used in the GS1 Business Message Standard Release 2.x. It does not explain the technical details of any earlier releases.

The implementation guidelines presented here apply to general Business Processes followed by the majority of the GS1 user companies. There are, however, special requirements related to specific implementation environment, like the Global Data Synchronisation Network – GDSN. Users involved in the GDSN should also refer to *GDSN 2.1 Operations Manual* which is located on the GS1 Website at:

http://www.gs1.org/docs/gsm/gdsn/GDSN_2.1_Operations_Manual_i1.pdf

Certain specific constraints may also apply to other business processes; therefore, it is advisable to contact the respective user group for further guidance (e.g. Global Upstream Supply Initiative).

1.3. Target Audience

This document is intended to serve as an implementation guide for business and technical people who will implement GS1 XML standards. It can also be used by the GS1 Member Organisation employees, to educate their user companies about the implementation of the GS1 XML standards.

2. Message Architecture Principles

2.1. GS1 Message Architecture

The main principle of the GS1 message architecture is separation of functionality layers, where each layer is wrapped inside the other. The three layers are:

- Transport and routing layer, containing information about routing and processing of the XML instance document
- Service layer, providing instructions (commands) for the receiving application about the actions to be performed on the business documents
- Business document layer, containing the actual business document or documents, e.g. Order, Despatch Advice, Invoice, etc.

All three layers combined constitute a GS1 XML message.

2.2. Benefits

The use of layered message architecture brings important benefits:

- Minimizes the overall message set needed for the GS1 System to perform an efficient exchange of business information. The same document can be used with different commands.

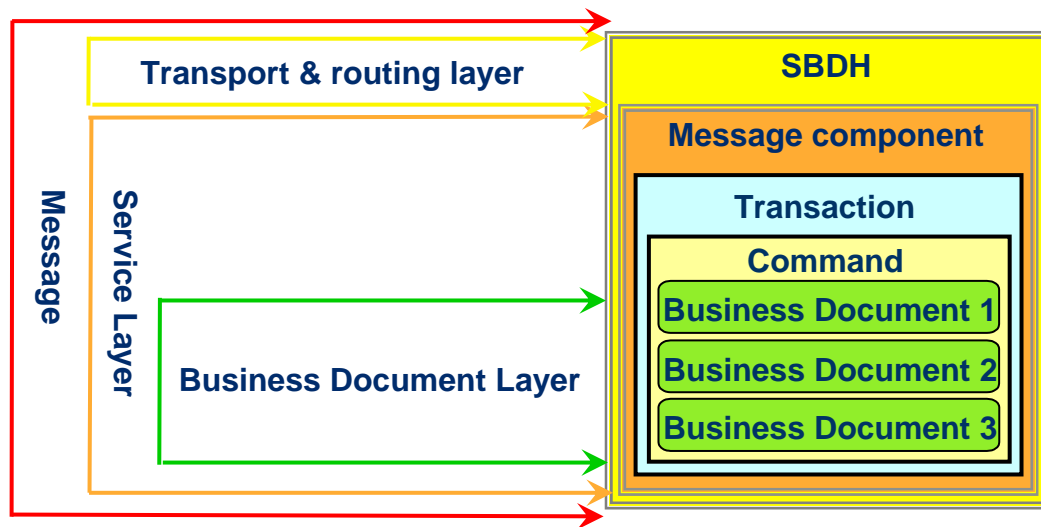
Hence, no separate documents like 'Add Order', 'Change Order' or 'Delete Order' are needed. The same 'Order' document can be sent with a relevant command, e.g. 'add', 'change' or 'delete'.

- Several documents can reuse the same commands, so adding new ones when needed is quick and easy, with no need to make any change to the existing document definitions (XML schemas). The new functionality can be added by creating a new value in a command list and used only by interested parties, not affecting the remaining users.

2.3. Overview of Message Layers

The diagram below illustrates the structure of the GS1 message layered architecture. An overview of every component is presented below and their detailed structure in Section 3.

Figure 2-1 GS1 Message architecture components



- ✓ **Note:** Because of the flexible design of the GS1 XML schemas (see Section 3), from a strictly technical standpoint each of these components (except for the interfaces linking the layers – see below) is optional. It is possible to create a valid XML instance document with all the components present or omit one or more of them. **It is very important to understand, however, that business applications do rely on information contained in every layer or their parts. Therefore, the functional cardinality of the message architecture components should always be considered in a specific business context.** The constraints for every message components are discussed in the Usage Notes in every section.

2.3.1. Transport and Routing Layer

GS1 System uses the Standard Business Document Header (SBDH) for routing and processing information in the GS1 XML standard.

The specifications for SBDH have been developed by the UN/CEFACT, the document is available at the GS1 website: www.gs1.org/services/gsm/kc/ecom/xml/xml_sbdh.html

The SBDH has been 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.

Using one common document header enables integration of documents between internal applications, enterprise applications, and business-to-business infrastructure by providing a consistent interface

between applications. It allows any application to determine the logical routing requirements and/or the logical processing requirements of a document based on the information contained in the SBDH.

SBDH contains information for communication applications about:

- **Document Routing** - identifies message sender and receiver; used by message exchange hubs, marketplaces, etc.
- **Document Identification** - used by the middleware to identify and route the message to the appropriate business application without opening or parsing it
- **Document Processing Context** - provides parameters for processing the business document in the context of a business choreography exchange
- **Payload** – a container for the business document

Detailed guidelines how to implement the SBDH with the GS1 XML standard have been documented separately in the *Standard Business Document Header (SBDH) Version 1.3 Technical Implementation Guide*, available at the GS1 website:

www.gs1.org/docs/gsmf/xml/sbdh/SBDH_v1.3_Technical_Implementation_Guide.pdf

Usage Notes

In GS1 XML standard, the SBDH is an integral part of the XML instance document and **its use is required**, although in an XML schema the SBDH layer is not defined as mandatory, due to the XML syntax limitations (see note in Section [2.3](#)). However, the business partners often rely on the information contained in this layer, e.g. for message routing.

2.3.2. Service Layer

This Layer contains the following components:

- Interface to the Transport and Routing layer (referred to as a message component)
- Transaction
- Command
- Interface to the Business Document Layer

It allows for combining complex commands and transactions into one message transmission. The information contained in this layer is transient. It means that the components are unpacked, processed, the instruction conveyed by each component is executed and then discarded. The only part of the message passed to the back end application is the business document.

2.3.2.1. Message Component

The Message component provides an interface linking the Service layer to the SBDH (Transport and Routing layer). It acts simply as a placeholder allowing for appending multiple transactions or commands in one SBDH.

The detailed structure of the Message component is explained in Section [3.2.1](#).

2.3.2.2. Transaction

A transaction provides functionality of submitting multiple commands simultaneously. It allows the users to process the group of business documents together. If one of them fails, all of them will be discarded. This functionality is also referred to as 'all or nothing' principle.

The transaction component can be considered as a flag indicating that if some of the documents nested within it fail the validation, the processing of all the remaining ones should stop. Obviously, this

functionality has to be pre-programmed into the processing application. It is important to note that this relates only to validation of an instance document against XML schema. It does not cover the business level validation, since it is not possible to verify the conformance to the business rules at the moment of 'unwrapping' of the transaction layer.

In GDSN (Global Data Synchronisation Network) it is also referred to as Unit of Work.

The detailed structure of the Transaction component is explained in Section [3.2.2](#).

Usage Notes

From a technical standpoint, Transaction component is not mandatory. However, the following restrictions apply in specific business context:

- Within GDSN (Global Data Synchronisation Network) the use of Transaction is required. For more information refer to the *GDSN 2.1 Operations Manual* which is located on the GS1 Website at: http://www.gs1.org/docs/gsmg/gdsn/GDSN_2.1_Operations_Manual_i1.pdf
- In transactional scenario (post data alignment) the Transaction should be used if 'all or nothing' principle is found useful. In a scenario where multiple business documents are being sent in one message, but their validation should not be coupled, the Transaction should not be used. In this case, even if one of the business documents fails the validation, the remaining ones will still be passed to the back end application.

Examples of messages with and without Transaction are presented in Section [3.4](#).

While the technical design allows for transmission of unlimited number of transactions in one message, the capabilities of processing applications and the bandwidth limitations should be taken into account.

- GDSN sets a limit of 100 Transactions wrapped in one message (see [GDSN 2.1 Operations Manual](#))
- In transactional scenario (post data alignment) there are no strict limitations, they depend solely on the capabilities of the applications and the bandwidth used to transport the message. Some eCom service providers acting as message exchange hubs may even set a limit of just one transaction in one message or prohibit the use of Transaction altogether.

2.3.2.3. Command

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

The only currently used command is the Document Command, wrapped around one or multiple Business Documents.


The Document Command provides a list of the following instructions:

Document Command	Description
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

Document Command	Description
CORRECT	<p>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.</p> <p>This instruction is typically used in GDSN.</p>
DELETE	The receiving application is instructed to delete the document or documents

The detailed structure of the Command component is explained in [Section 3.2.3](#).

A scenario illustrating the use of the above instructions in the business context is presented in the [Section 4 Appendix: Message Layer Usage in Business Scenario - Example](#).

-  **Note:** Originally, the GS1 XML standard provided two additional command constructs
- Document Identification Command – instructing the receiving application that a document referenced in the command by its identifier, that had been previously sent, should be deleted.
 - Link Command – instructing the receiving application to establish or discontinue parent-child or peer relationships between several entities referenced within this command.

Both Document Identification Command and Link Command are no longer used within the GS1 System and will not therefore be described in this document.

Usage Notes

From a technical standpoint (XML syntax limitations), the Command component is not mandatory (see note in [Section 2.3](#)). However, the GS1 business document choreography requires their use. The instructions provided in the Document Command perform the function of the Amend Document or Cancel Document, which are not provided as separate Business Documents. For example, there are no Order Change nor Order Cancellation documents. Therefore, in order to fully leverage the GS1 business process choreography, the users must implement the Command component.

The technical design of GS1 XML standards allows for transmission of unlimited number of commands wrapped in one message, but in practice it is limited by the capabilities of processing applications and the bandwidth limitations.

- GDSN sets a limit of 1 Command type wrapped in one Transaction (see [GDSN 2.1 Operations Manual](#))
- In transactional scenario (post data alignment) there are no strict limitations, they depend solely on the capabilities of the applications and the bandwidth used to transport the message. Some eCom service providers acting as message exchange hubs, may even set a limit of just one Command in one message.
- As a good practice, to avoid confusion with processing the message, it is advisable to use only one Command type in the message or at least one Command type in one Transaction, i.e. only ADD or only DELETE

2.3.2.4. Document

The Document provides an interface to the Business Document layer, containing the actual business document or documents, e.g. Order, Invoice, Despatch Advice.

The detailed structure of the Document component is explained in [Section 3.3.1](#).

Usage Notes

The Document interface allows wrapping of multiple documents within a Command. It is also possible to include multiple command / document pairs within one message. Users should select the method most suitable for their practice.

From a technical standpoint, the Document layer can contain unlimited number of business documents in one message. However, the capabilities of processing applications and the bandwidth limitations should be taken into account.

- GDSN sets a limit of 100 Documents in one message (see [GDSN 2.1 Operations Manual](#))
- In transactional scenario (post data alignment) there are no strict limitations, they depend solely on the capabilities of the applications and the bandwidth used to transport the message. Some eCom service providers acting as message exchange hubs, may even set a limit of just one document in one message.

The technical design of the GS1 XML allows inserting various business documents in one message, as long as they are a part of the GS1 XML standard. However, the business choreography or common practice may impose limitations.

- GDSN sets a limit of 1 document type within one message (see [GDSN 2.1 Operations Manual](#))
- In transactional scenario (post data alignment) there are no strict limitations, but in order to avoid confusion and complex routing, it is recommended to send only one document type within one message, e.g. only Invoices or only Orders.

2.3.3. Business Document Layer

Business Document layer contains the actual business documents. These documents are fully documented in GS1 Business Message Standards and will not be discussed in this document.

3. Message Layer Details

3.1. Transport and Routing Layer Details

As mentioned in Section [2.3.1](#), the Standard Business Document Header (SBDH) is used for providing the business partner the transport and routing information. The details how to implement the SBDH in GS1 standards are documented separately in the *Standard Business Document Header (SBDH) Version 1.3 Technical Implementation Guide* which is located on the GS1 website at:

http://www.gs1.org/docs/gsmf/xml/sbdh/SBDH_v1.3_Technical_Implementation_Guide.pdf

3.2. Service Layer Details

The general overview of the Service layer can be found in Section [2.3.2](#).

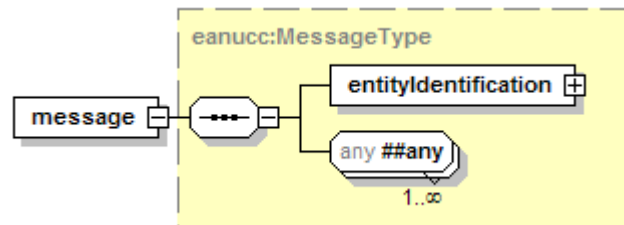
3.2.1. Message Component Details

3.2.1.1. Message Component Structure

Message component is defined in the Message.xsd schema. It links the Service layer to the Transport and Routing layer, as a placeholder for multiple Transactions or Commands in one SBDH.

The Message consists of two parts:

- Message element identification of *entityIdentification* type. It uniquely identifies each instance of the Message
- A placeholder for the next layer component: *##any* element. It can have multiple occurrence, so that many transactions or commands can be appended. Technically, any child component can be placed here (due to the XML syntax limitations), but the GS1 standard only allows here either Transaction (if used) or Document Command (if Transaction is not used). The XML 'any' element has a built-in attribute: *processContents* indicating how an XML processor should handle the validation of XML documents against the elements specified by the 'any' element. In the message component, the value of this attribute is set to *strict*, it means that the XML processor must obtain the schema for the required namespaces and validate any element from those namespaces.

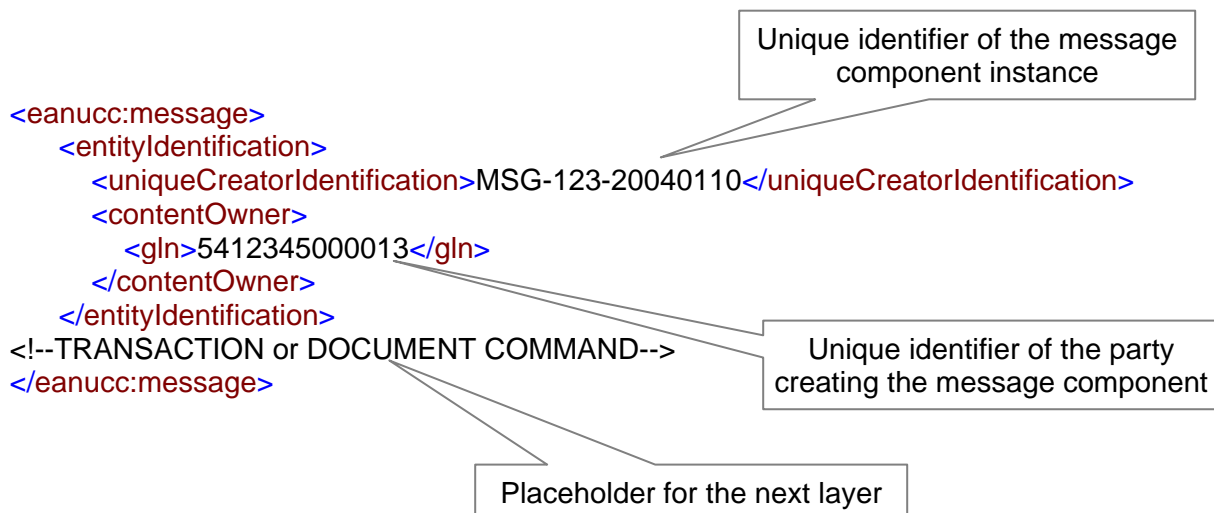
Figure 3-1 Message component structure


3.2.1.2. Message Component Tags

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
message	eanucc:MessageType	1..1	This element is used as a direct child of the <i>StandardBusinessDocumentHeader</i> element.
entityIdentification	eanucc:EntityIdentificationType	1..1	Uniquely identifies every instance of the <i>message</i> element by a combination of the unique document identifier assigned by its creator and the unique identification of that document creator (GLN and optionally, an additional identifier).
uniqueCreatorIdentification	xsd:string	1..1	A unique identification assigned by the creator for the message component. The best practice is that this value is unique for all the entities within the given <i>contentOwner</i> domain. However, some users may find it impossible due to the limited identifier capacity. If the <i>uniqueCreatorIdentification</i> values are ever re-used, the creator of the component should ensure that they are not repeated within the same message (transmission).
contentOwner	eanucc:PartyIdentificationType	1..1	Uniquely identifies the creator of the Message component. It may be the same party as the one creating other architecture components, including the business document or e.g. the party providing service of wrapping the business document into other message layers.
gln	eanucc:GlobalLocationNumberType	1..1	Global Location Number uniquely identifying the party creating the Message component.
additionalPartyIdentification	eanucc:AdditionalPartyIdentificationType	0..1	Additional Identifier of the Party. It can be used in addition to GLN - the primary identifier.
additionalPartyIdentificationValue	xsd:string	1..1	The additional Party identifier value.

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
additionalPartyIdentificationType	eanucc:AdditionalPartyIdentificationListType	1..1	The additional Party identifier type. The type values come from the enumerated list.
##any	-	1..∞	A placeholder for next level architecture layer components. In the instance document this will typically be either <i>transaction</i> or <i>documentCommand</i> elements.
processContents	-		The value of this built-in attribute is set to <i>strict</i> , instructing the validating tool that a schema must be obtained for the required namespaces and validate any element from those namespaces. These are the Transaction.xsd (if used) or DocumentCommand.xsd schemas.

3.2.1.3. Message Component Instance Example



3.2.2. Transaction Component Details

3.2.2.1. Transaction Component Structure

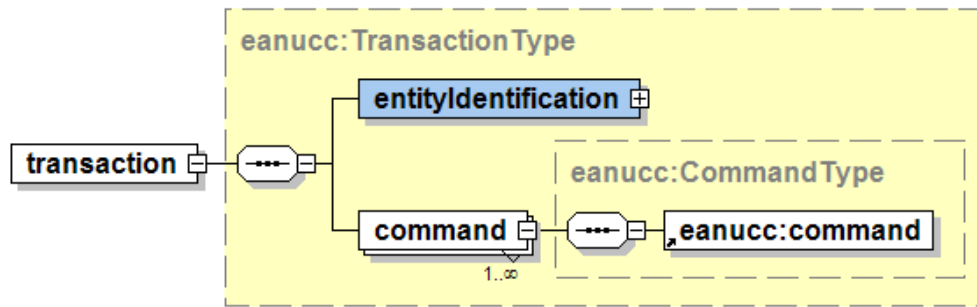
Transaction component is defined in the Transaction.xsd schema. It allows the processing of the group of documents together. If one of them fails, all of them will be discarded.

The Transaction consists of two parts:

- Transaction element identification of *entityIdentification* type. It uniquely identifies each instance of the Transaction
- A placeholder for the Commands. The command element is a container, where one or multiple commands that form one transaction can be inserted. The command element is of an abstract *CommandType* (defined in *Command.xsd* schema) that is extended by the *DocumentCommandType* concrete type.

Note: It was also extended by the *DocumentIdentificationCommandType* and *LinkCommandType*, defining the structure of the Document Identification Command and Link Command respectively, but these two constructs are no longer used in the GS1 XML standard.

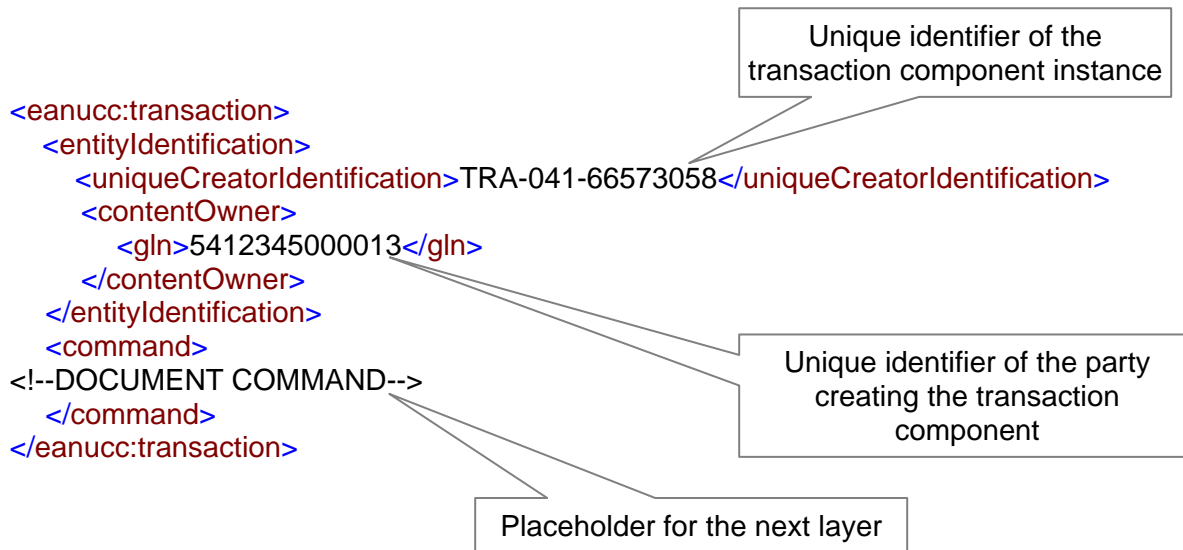
The *CommandType* references an abstract command element, substituted by the concrete *documentCommand* element.

Figure 3-2 Transaction Component Structure


3.2.2.2. Transaction Component Tags

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
transaction	eanucc:TransactionType	1..1	This element is used as a direct child of the <i>message</i> element.
entityIdentification	eanucc:EntityIdentificationType	1..1	Uniquely identifies every instance of the <i>transaction</i> element by a combination of the unique document identifier assigned by its creator and the unique identification of that document creator (GLN and optionally, an additional identifier).
uniqueCreatorIdentification	xsd:string	1..1	A unique identification assigned by the creator for the message component. The best practice is that this value is unique for all the entities within the given <i>contentOwner</i> domain. However, some users may find it impossible due to the limited identifier capacity. If the <i>uniqueCreatorIdentification</i> values are ever re-used, the creator of the component should ensure that they are not repeated within the same message (transmission).
contentOwner	eanucc:PartyIdentificationType	1..1	Uniquely identifies the creator of the transaction component. It may be the same party as the one creating other architecture components, including the business document or e.g. the party providing service of wrapping the business document into other message layers.
gln	eanucc:GlobalLocationNumberType	1..1	Global Location Number uniquely identifying the party creating the transaction component.
additionalPartyIdentification	eanucc:AdditionalPartyIdentificationType	0..1	Additional identifier of the Party. It can be used in addition to GLN - the primary identifier.
additionalPartyIdentificationValue	xsd:string	1..1	The additional Party identifier value.
additionalPartyIdentificationType	eanucc:AdditionalPartyIdentificationListType	1..1	The additional Party identifier type. The type values come from the enumerated list.
command	eanucc:CommandType	1..∞	A placeholder for command element
command	-	1..1	This is a referenced abstract element that cannot be directly instantiated, but must be substituted by the concrete <i>documentCommand</i> element.

3.2.2.3. Transaction Component Instance Example



3.2.3. Document Command Component Details

3.2.3.1. Document Command Component Structure

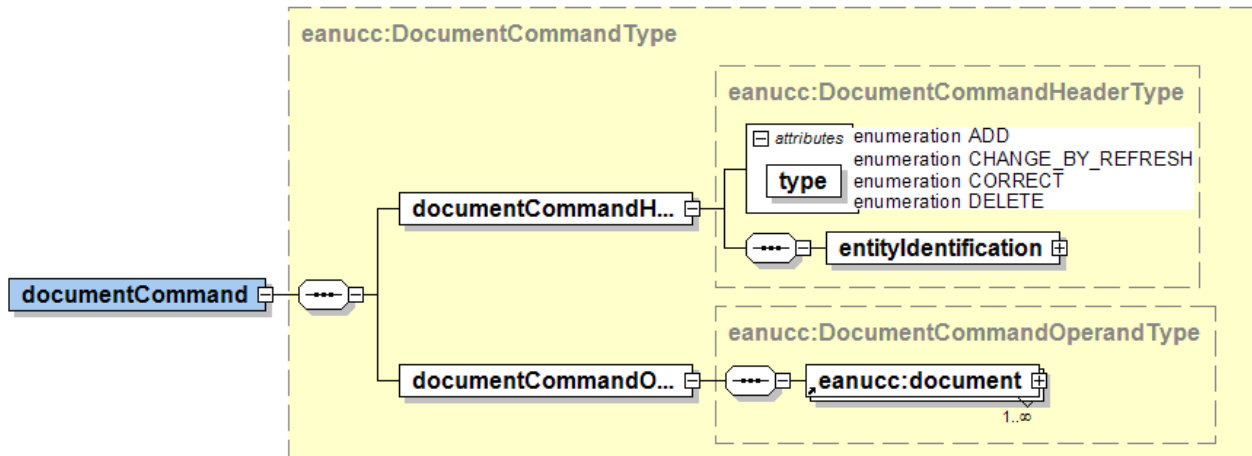
Document Command component is defined in the DocumentCommand.xsd schema. It is used to instruct the recipient of that command to perform a particular action on the documents within the command.

The Document Command substitutes the abstract Command. It contains two parts:

- Document command header specifies the action that should be performed on the given document. Those tasks, defined as the required attribute of the header element, include:
 - [ADD](#)
 - [CHANGE_BY_REFRESH](#)
 - [CORRECT](#)
 - [DELETE](#)

Document Command element identification of *entityIdentification* type. It uniquely identifies each instance of the Document Command.

- Document command operand containing placeholder for the business document layer component. It references an abstract *document* element that can be substituted by the concrete business document root elements, defined in the business document schemas. These are the documents upon which one of the actions defined in the header element should be performed.

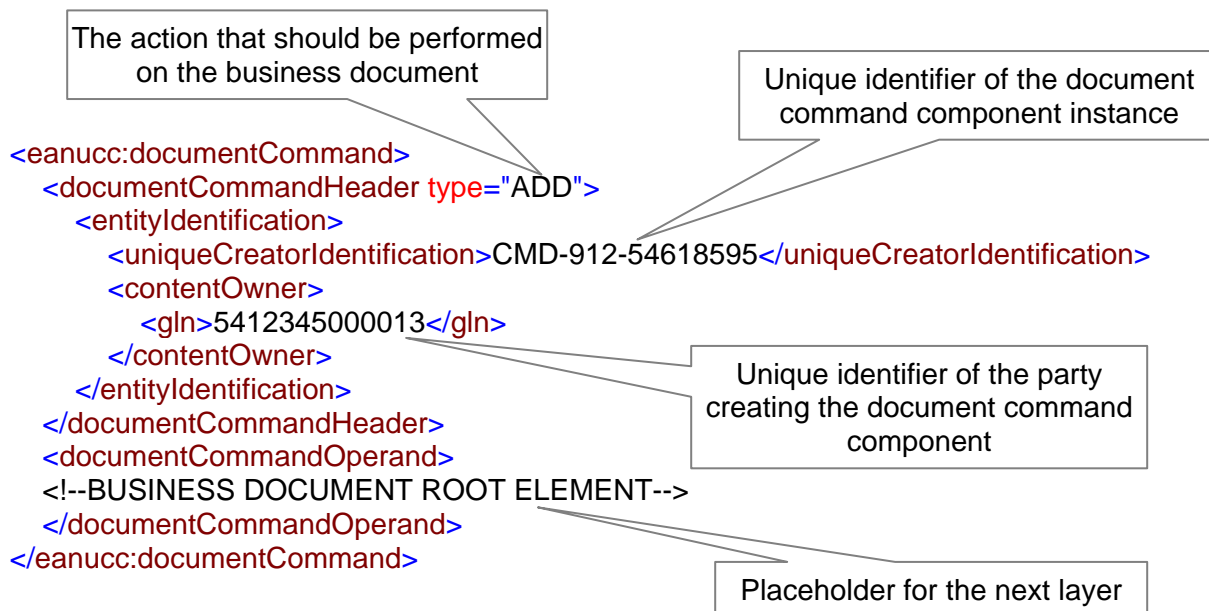
Figure 3-3 Document Command Structure


3.2.3.2. Document Command Tags

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
documentCommand	eanucc:DocumentCommandType	1..1	This element is used as a direct child of the <i>transaction</i> (if used) or <i>message</i> element (if transaction is not used).
documentCommandHeader	eanucc:DocumentCommandHeaderType		This element contains the command identification and specifies the action that should be performed on the given business document.
entityIdentification	eanucc:EntityIdentificationType	1..1	Uniquely identifies every instance of the document command element by a combination of the unique document identifier assigned by its creator and the unique identification of that document creator (GLN and optionally, an additional identifier).
uniqueCreatorIdentification	xsd:string	1..1	A unique identification assigned by the creator to the message component. The best practice is that this value is unique for all the entities within the given <i>contentOwner</i> domain. However, some users may find it impossible due to the limited identifier capacity. If the <i>uniqueCreatorIdentification</i> values are ever reused, the creator of the component should ensure that they are not repeated within the same message (transmission).
contentOwner	eanucc:PartyIdentificationType	1..1	Uniquely identifies the creator of the document command component. It may be the same party as the one creating other architecture components, including the business document or e.g. the party providing service of wrapping the business document into other message layers.
gln	eanucc:GlobalLocationNumberType	1..1	Global Location Number uniquely identifying the party creating the document command component.
additionalPartyIdentification	eanucc:AdditionalPartyIdentificationType	0..1	Additional identifier of the Party. It can be used in addition to GLN - the primary identifier.
additionalPartyIdentificationValue	xsd:string	1..1	The additional Party identifier value.

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
additionalPartyIdentificationType	eanucc:AdditionalPartyIdentificationListType	1..1	The additional Party identifier type. The type values come from the enumerated list.
type	eanucc:DocumentCommandListType	1..1	This attribute contains an enumeration list of the four possible actions that should be performed on the business document. The enumeration values are: ‘ADD’ ‘CHANGE BY REFRESH’ ‘CORRECT’ ‘DELETE’
documentCommandOperand	eanucc:DocumentCommandOperandType	1..1	A placeholder for <i>document</i> element
document	eanucc:DocumentType	1..∞	This is a referenced abstract element that cannot be directly instantiated, but must be substituted by the concrete business document root element (e.g. <i>order</i> , <i>invoice</i> , etc.). The list of the possible documents contains all the business messages defined within the given major release .

3.2.3.3. Document Command Instance Example



3.3. Business Document Layer Details

The general overview of the Business Document layer can be found in Section [2.3.3](#).

3.3.1. Business Document Component Details

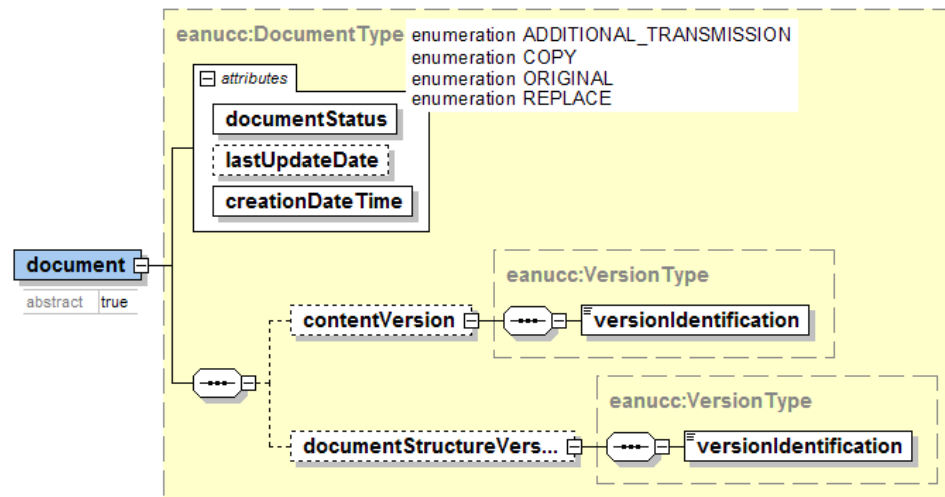
3.3.1.1. Business Document Component Structure

Document component is defined in the Document.xsd schema. It links the Business Document layer to the Service layer, as a placeholder for multiple business documents in one Document Command.

The Document consists of the following parts:

- Document versions – optionally specify the version of the structure and content of the document.
- Document status – the status values are defined as an enumeration list; they are not related to the XML message architecture functionality and therefore will not be discussed in detail in this document.
- Document dates – specify when was it created and (optionally) updated.

Figure 3-4 Document Component Structure

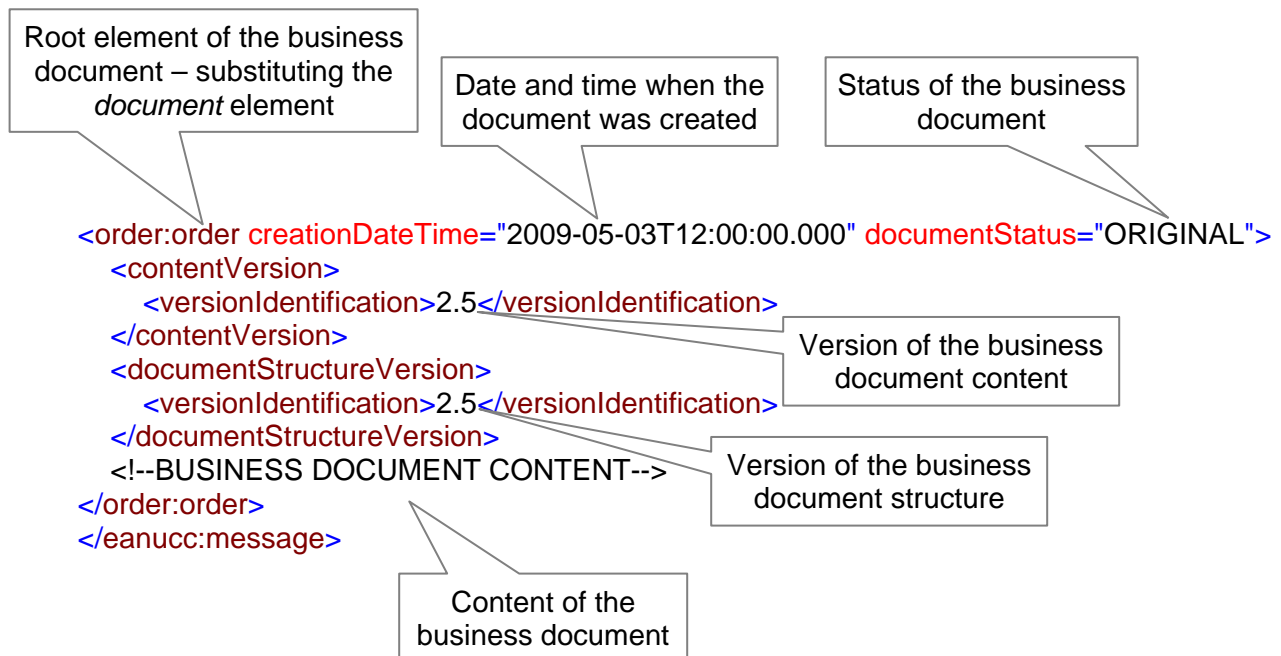


3.3.1.2. Document component tags

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
document	eanucc:DocumentType	1..1	This is an abstract element that cannot be directly instantiated, but must be substituted by the concrete business document root element (e.g. <i>order</i> , <i>invoice</i> , etc.). The list of the possible documents contains all the business messages defined within the given major release .
documentStatus	eanucc:DocumentStatusList Type	1..1	This attribute contains an enumeration list of the four possible status values of the document.
lastUpdateDate	xsd:date	1..0	Specifies the date when the document was last updated
creationDate Time	xsd:dateTime	1..1	An attribute used to specify the date and time when the document was created.
contentVersion	eanucc:VersionType	1..0	Provides information about the version number of the content of the document.

XSD Element / Attribute	XSD Type	Occ	Usage Guidelines
versionIdentification	xsd:string	1..1	Contains the actual value of the document content version.
documentStructureVersion	eanucc:VersionType	1..0	Reflects the complete version of the business document schema, e.g. for the Order.xml message, the document structure version reflects the version number of the Order.xsd schema.
versionIdentification	xsd:string	1..1	Contains the actual value of the document structure version.

3.3.1.3. Document Component Instance Example



3.4. Batching

The GS1 layered message architecture allows for batching various components – transmitting them as a part of one message. Depending on the business context, business practice adopted by the exchange partners and processing application requirements, batching can be done in different ways: with or without Transaction component, placing multiple documents in one command or multiple command and business document pairs.



Note: GS1 XML standard Implementer's Packets contain **proxy schemas**. They are published to help implementation using certain processing tools that do not support multiple schema location. They include and import all schemas that are necessary to validate the sample instance file also contained in the Implementer's Packets Proxy schemas are not normative, they do not have to be used and if used, their content can be modified. This modification is necessary if Transaction component is not used. More information about proxy schemas and their usage can be found at the GS1 website at:

http://www.gs1.org/ecom/xml/implementation/guide/12_Document_layer#Proxy_files

3.4.1. Batching of Transactions

The general overview of the Transaction component can be found in Section [2.3.2.2](#) and the detailed structure in Section [3.2.2](#).

Batching of Transactions allows multiple Transactions, each containing one or multiple commands and Business Documents be transmitted in one message.

When the Transaction batching is used, the GS1 XML message contains the following components wrapped around each other:

- 1 SBDH (Standard Business Document Header)
- 1 Message component
- Multiple Transaction components in one Message component
- One or multiple Commands in one Transaction component
- One or multiple Business Documents in one Command

Figure 3-5 Batching of Transactions in one message



In the example presented on [Figure 3-5](#), Business Documents 1-3 will be processed according to the instruction contained in Command 1, Business Documents 4-5 according to the instruction contained in Command 2 and Business Documents 6-8 according to the instruction contained in Command 3. However, if any of the documents 1-5 will be corrupted, all five of them will be rejected, but if the documents 6-8 are valid, they will still be passed into the back-end application. In the same way, if the

documents 1-5 are valid, they will be passed on for further processing, but if one of the documents 6-8 is corrupted, all three of them will be rejected.

The following architecture schemas will have to be used for Transaction batching (included also into a proxy schema if used):

- SBDH set of 6 schemas (not contained in the GS1 proxy)
- Message.xsd
- Transaction.xsd
- Command.xsd
- DocumentCommand.xsd
- DocumentCommandList.xsd

Below is an example of an XML instance containing one Transaction component, one Command and one Business Document – in this case the Order message:

```
<sh:StandardBusinessDocument">
  <sh:StandardBusinessDocumentHeader>
    <!--SBDH CONTENT-->
  </sh:StandardBusinessDocumentHeader>
  <eanucc:message>
    <!--MESSAGE COMPONENT IDENTIFICATION-->
    <eanucc:transaction>
      <!--TRANSACTION COMPONENT IDENTIFICATION-->
      <command>
        <eanucc:documentCommand>
          <documentCommandHeader type="ADD">
            <!--DOCUMENT COMMAND IDENTIFICATION-->
          </documentCommandHeader>
          <documentCommandOperand>
            <order:order creationDateTime="2003-11-03T11:00:00.000"
              documentStatus="ORIGINAL">
              <!--BUSINESS DOCUMENT CONTENT-->
            </order:order>
          </documentCommandOperand>
        </eanucc:documentCommand>
      </command>
    </eanucc:transaction>
  </eanucc:message>
</sh:StandardBusinessDocument">
```

3.4.2. Batching of Commands

The general overview of the Command component can be found in Section [2.3.2.3](#) and the detailed structure in Section [3.2.3](#).

Batching of Commands allows multiple Command plus Business Document pairs to be transmitted in one message.

When the Command batching is used, the GS1 XML message contains the following components wrapped around each other:

- 1 SBDH (Standard Business Document Header)

- 1 Message component
- One Command as a direct child of the Message component
- One Business Document in the Command
- Another Command as a direct child of the Message component
- One Business Document in the Command
- etc.

Figure 3-6 Batching of Commands in one message



In the example presented on [Figure 3-6](#), Business Document 1 will be processed according to the instruction contained in Command 1, Business Document 2 according to the instruction contained in Command 2, etc.

The Command – Document pairs are not wrapped in the Transaction component, therefore, if any of the five documents will be corrupted, the remaining four will be passed into the back-end application.

The following architecture schemas will have to be used for Command batching (included also into a proxy schema if used):

- SBDH set of 6 schemas (not contained in the GS1 proxy)
- Message.xsd
- DocumentCommand.xsd
- DocumentCommandList.xsd

Below is an example of an XML instance containing one Command and one Business Document – in this case the Order message:

```

<sh:StandardBusinessDocument">
  <sh:StandardBusinessDocumentHeader>
    <!--SBDH CONTENT-->
  </sh:StandardBusinessDocumentHeader>
  <eanucc:message>
    <!--MESSAGE COMPONENT IDENTIFICATION-->
    <eanucc:documentCommand>
      <documentCommandHeader type="ADD">
        <!--DOCUMENT COMMAND IDENTIFICATION-->
      </documentCommandHeader>
      <documentCommandOperand>
        <order:order creationDateTime="2003-11-03T11:00:00.000"
          documentStatus="ORIGINAL">
          <!--BUSINESS DOCUMENT CONTENT-->
        </order:order>
      </documentCommandOperand>
    </eanucc:documentCommand>
  </eanucc:message>
</sh:StandardBusinessDocument">
  
```

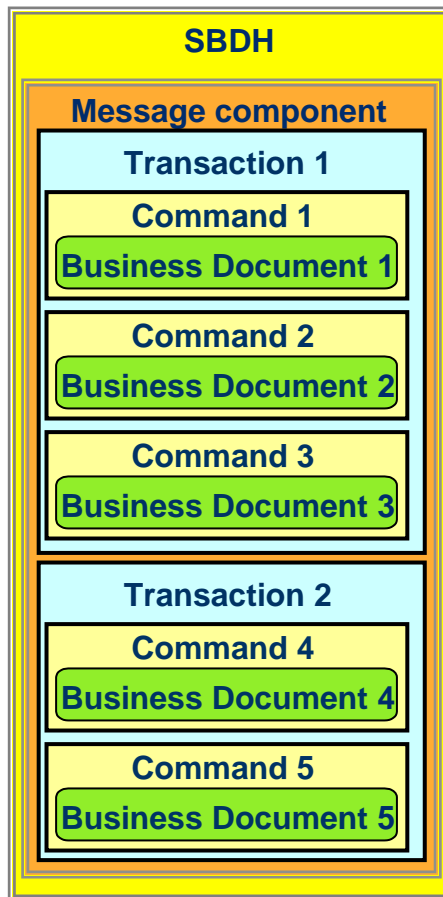
3.4.3. Batching of Transactions and Commands

The batching techniques presented in Sections [3.4.1](#) and [3.4.2](#) can be combined in one message.

Batching of Transactions and Commands allows multiple Transactions containing multiple Command plus Business Document pairs to be transmitted in one message.

When the Transactions and Command batching is used, the GS1 XML message contains the following components wrapped around each other:

- 1 SBDH (Standard Business Document Header)
- 1 Message component
- One Transaction as a direct child of the Message component
- One Command as a direct child of the Transaction component
- One Business Document in the Command
- Another Command as a direct child of the Transaction component
- One Business Document in the Command
- Another Transaction component
- etc.

Figure 3-7 Batching of Transactions and Commands in One Message


In the example presented on [Figure 3-7](#), Business Document 1 will be processed according to the instruction contained in Command 1, Business Document 2 according to the instruction contained in Command 2, etc. However, if any of the documents 1-3 will be corrupted, all three of them will be rejected, but if the documents 4-5 are valid, they will still be passed into the back-end application. The same way, if the documents 1-3 are valid, they will be passed on for further processing, but if one of the documents 4-5 is corrupted, both of them will be rejected.

The following architecture schemas will have to be used for Transaction and Command batching (included also into a proxy schema if used):

- SBDH set of 6 schemas (not contained in the GS1 proxy)
- Message.xsd
- Transaction.xsd
- Command.xsd
- DocumentCommand.xsd
- DocumentCommandList.xsd

An example of an XML instance containing one Transaction component, one Command and one Business Document is shown in [Section 3.4.1](#).

3.4.4. Batching of Documents

The general overview of the Document component can be found in Section [2.3.3](#) and the detailed structure in Section [3.3](#).

Batching of Documents allows multiple Commands to be wrapped around one or more Business Documents and transmitted in one message.

When the Command batching is used, the GS1 XML message contains the following components wrapped around each other:

- 1 SBDH (Standard Business Document Header)
- 1 Message component
- Multiple Commands in one Message component
- One or multiple Commands in one Message component
- One or multiple Business Documents in one Command

Figure 3-8 Batching of Documents in One Message



In the example presented on [Figure 3-8](#), Business Documents 1-3 will be processed according to the instruction contained in Command 1, Business Documents 4-5 according to the instruction contained in Command 2 and Business Document 6 according to the instruction contained in Command 3.

The Commands and Documents they contain are not wrapped in the Transaction component, therefore, if any of the six documents will be corrupted, the remaining five will be passed into the back-end application. The following architecture schemas will have to be used for batching of Documents (included also into a proxy schema if used):

- SBDH set of 6 schemas (not contained in the GS1 proxy)
- Message.xsd
- DocumentCommand.xsd
- DocumentCommandList.xsd

An example of an XML instance containing one Command and one Business Document is shown in Section [3.4.2](#).

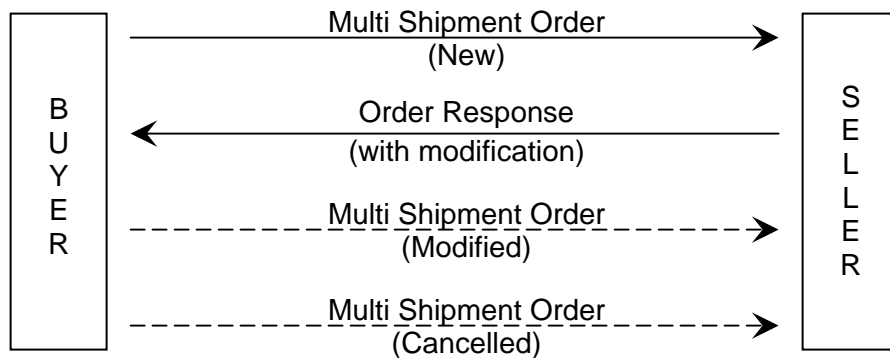
4. Appendix: Message Layer Usage in Business Scenario - Example

This section contains an example of using GS1 XML message architecture layers in a business scenario. The scenario described below presents one of the possible implementations of the GS1 Commands to send and subsequently change or cancel an Order document, in reaction to the Order Response received.

The actual choreography, however, would be a business decision, depending on the business practice agreed between the business partners.

The scenario involves two business partners: Buyer and Seller. Buyer sends to Seller the MultiShipmentOrder message. Seller responds to the message by sending the OrderResponse, suggesting modifications of the goods ordered. In reaction to the suggested modification, Buyer can either accept them in full, modify the original order or cancel the whole order.

Figure 4-1 Message Exchange Scenario



The example below was based on two business documents: MultiShipmentOrder rel. 2.5 and OrderResponse rel. 2.5. For simplicity, only mandatory elements and the ones required by the business scenario have been used.

STEP 1

Buyer sends a **Multi Shipment Order** message to Seller, ordering 2 items: Item1 times 15 and Item2 times 20. The Multi Shipment Order is sent with Document Command ADD.

Content of MultiShipmentOrder New (for the complete instance file for this example see [here](#)):

Data tag	Data value
MessageID => uniqueCreatorIdentification	MSG-123-34926
MessageID => contentOwner => GLN	8712345678920
documentCommandHeader (command) type	ADD
DocumentCommandID => uniqueCreatorIdentification	246810N
DocumentCommandID => contentOwner => GLN	8712345678920
MultiShipmentOrder => documentStatus	ORIGINAL
MultiShipmentOrder => creationDateTime	2009-08-07 T 09:00
orderIdentification => uniqueCreatorIdentification	4521560719U

Data tag	Data value
orderIdentification => contentOwner => GLN	8712345678920
Seller => GLN	8812345678934
Buyer => GLN	8712345678920
ShipTo => GLN	8712345678944
requestedDeliveryDate => date	2009-08-08
multiShipmentOrderLineItem => number	1
requestedQuantity => value	15
[Item1] GTIN	88123456798906
multiShipmentOrderLineItem => number	2
requestedQuantity => value	20
[Item2] GTIN	88123456798760

STEP 2

Seller sends an **Order Response** message, suggesting modification of order line item 1: substituting Item1 with Item3. Item2 can be delivered without modifications. Order Response references MultiShipmentOrder identification.

Content of OrderResponse (for the complete instance file for this example see [here](#)):

Data tag	Data value
MessageID => uniqueCreatorIdentification	MSG938544
MessageID => contentOwner => GLN	8812345678934
OrderResponse => responseStatusType	MODIFIED
OrderResponse => creationDateTime	2009-08-07 T 09:30
responseIdentification => uniqueCreatorIdentification	579135OR
responseIdentification => contentOwner => GLN	8812345678934
responseToOriginalDocument => referenceDocumentType	34
responseToOriginalDocument => referenceDateTime	2009-08-07 T 09:00
responseToOriginalDocument => referenceIdentification	4521560719U
Buyer => GLN	8712345678920
Seller => GLN	8812345678934
[Item3] SubstituteItemIdentification => GTIN	88123456798753
ModifiedOrderInformation => number	1
requestedQuantity => value	15
[Item1] GTIN	88123456798906

STEP 3a

In reaction to the suggested modification Buyer decides to change the original **Multi Shipment Order** and order 10 pieces of Item3 (instead of 15 pieces of Item1). Item 2 should be delivered like originally ordered.

Buyer re-sends the Multi Shipment Order with Document Command CHANGE_BY_REFRESH and the same Order number like in the original document (initial order). The Id of the Document Command is different from the one in the initial order.

Content of MultiShipmentOrder Modified (for the complete instance file for this example see [here](#)):

Data tag	Data value
MessageID => uniqueCreatorIdentification	MSG-123-34927
MessageID => contentOwner => GLN	8712345678920
documentCommandHeader (command) type	CHANGE_BY_REFRESH
DocumentCommandID => uniqueCreatorIdentification	246820M
DocumentCommandID => contentOwner => GLN	8712345678920
MultiShipmentOrder => documentStatus	ORIGINAL
MultiShipmentOrder => creationDateTime	2009-08-07 T 09:45
orderIdentification => uniqueCreatorIdentification	4521560719U
orderIdentification => contentOwner => GLN	8712345678920
Seller => GLN	8812345678934
Buyer => GLN	8712345678920
ShipTo => GLN	8712345678944
requestedDeliveryDate => date	2009-08-08
multiShipmentOrderLineItem => number	1
requestedQuantity => value	10
[Item3] GTIN	88123456798753
multiShipmentOrderLineItem => number	2
requestedQuantity => value	20
[Item2] GTIN	88123456798760

STEP 3b

In reaction to the suggested modification Buyer decides to cancel the entire order.

Buyer re-sends the initial **Multi Shipment Order** with Document Command DELETE and the same Order number like in the original document (initial order). The Id of the Document Command is different from the one in the initial order.

Content of MultiShipmentOrder Cancelled (for the complete instance file for this example see [here](#)):

Data tag	Data value
MessageID => uniqueCreatorIdentification	MSG-123-34928
MessageID => contentOwner => GLN	8712345678920
documentCommandHeader (command) type	DELETE
DocumentCommandID => uniqueCreatorIdentification	246830C
DocumentCommandID => contentOwner => GLN	8712345678920
MultiShipmentOrder => documentStatus	ORIGINAL
MultiShipmentOrder => creationDateTime	2009-08-07 T 09:45
orderIdentification => uniqueCreatorIdentification	4521560719U
orderIdentification => contentOwner => GLN	8712345678920
Seller => GLN	8812345678934
Buyer => GLN	8712345678920
ShipTo => GLN	8712345678944
requestedDeliveryDate => date	2009-08-08
multiShipmentOrderLineItem => number	1
requestedQuantity => value	15
[Item1] GTIN	88123456798906
multiShipmentOrderLineItem => number	2
requestedQuantity => value	20
[Item2] GTIN	88123456798760

- Note:** In the above example, both the original and modified Multi Shipment Order instances have the same Order Number (on the grounds that this is the same Order, just with some modification). But in practice this will depend on the agreement between the business partners. Another possible scenario would be to cancel the original order and send a new one, with a different Order Number. Again, this is to be agreed either bilaterally or within the user group.

The MultiShipmentOrder New instance file (Step 1 in the above scenario):

```
<?xml version="1.0" encoding="UTF-8"?>
<sh:StandardBusinessDocument
  xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
  xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader
  sbdh/StandardBusinessDocumentHeader.xsd urn:ean.ucc:2 MultiShipmentOrderProxy.xsd">
  <sh:StandardBusinessDocumentHeader>
```

```

<sh:HeaderVersion>1.0</sh:HeaderVersion>
<sh:Sender>
  <sh:Identifier Authority="EAN.UCC">8712345678920</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">8812345678934</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>2.5</sh:TypeVersion>
  <sh:InstanceIdentifier>100001</sh:InstanceIdentifier>
  <sh:Type>MultiShipmentOrder</sh:Type>
  <sh:MultipleType>>false</sh:MultipleType>
  <sh:CreationDateAndTime>2009-08-07T09:00:00.000</sh:CreationDateAndTime>
</sh:DocumentIdentification>
</sh:StandardBusinessDocumentHeader>
<eanucc:message>
  <entityIdentification>
    <uniqueCreatorIdentification>MSG-123-34926</uniqueCreatorIdentification>
    <contentOwner>
      <gln>8712345678920</gln>
    </contentOwner>
  </entityIdentification>
  <eanucc:documentCommand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2">
    <documentCommandHeader type="ADD">
      <entityIdentification>
        <uniqueCreatorIdentification>246810N</uniqueCreatorIdentification>
        <contentOwner>
          <gln>8712345678920</gln>
        </contentOwner>
      </entityIdentification>
    </documentCommandHeader>
    <documentCommandOperand>
      <order:multiShipmentOrder documentStatus="ORIGINAL" creationDateTime="2009-08-
      07T09:00:00Z">
        <orderIdentification>
          <uniqueCreatorIdentification>4521560719U</uniqueCreatorIdentification>
          <contentOwner>
            <gln>8712345678920</gln>
          </contentOwner>
        </orderIdentification>
      </order:multiShipmentOrder>
    </documentCommandOperand>
  </eanucc:documentCommand>
</eanucc:message>

```

```
<orderPartyInformation>
  <seller>
    <gln>8812345678934</gln>
  </seller>
  <buyer>
    <gln>8712345678920</gln>
  </buyer>
</orderPartyInformation>
<orderLogisticalInformation>
  <shipToLogistics>
    <shipTo>
      <gln>8712345678944</gln>
    </shipTo>
  </shipToLogistics>
  <orderLogisticalDateGroup>
    <requestedDeliveryDate>
      <date>2009-08-08</date>
    </requestedDeliveryDate>
  </orderLogisticalDateGroup>
</orderLogisticalInformation>
<multiShipmentOrderLineItem number="1">
  <requestedQuantity>
    <value>15</value>
  </requestedQuantity>
  <tradeItemIdentification>
    <gtin>88123456798906</gtin>
  </tradeItemIdentification>
</multiShipmentOrderLineItem>
<multiShipmentOrderLineItem number="2">
  <requestedQuantity>
    <value>20</value>
  </requestedQuantity>
  <tradeItemIdentification>
    <gtin>88123456798760</gtin>
  </tradeItemIdentification>
</multiShipmentOrderLineItem>
</order:multiShipmentOrder>
</documentCommandOperand>
</eanucc:documentCommand>
</eanucc:message>
</sh:StandardBusinessDocument>
```

The OrderResponse instance file (Step 2 in the above scenario):

- ✓ **Note:** Please note that the GS1 Response documents are not based on the Document, but on the Response component. Unlike the Document, the Response is not referenced in the Document Command and does not contain Service layer (the Response is a direct child of the Message component).

```
<?xml version="1.0" encoding="UTF-8"?>
<sh:StandardBusinessDocument
xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader
sbdh/StandardBusinessDocumentHeader.xsd urn:ean.ucc:2 OrderResponseProxy.xsd">
  <sh:StandardBusinessDocumentHeader>
    <sh:HeaderVersion>1.0</sh:HeaderVersion>
    <sh:Sender>
      <sh:Identifier Authority="EAN.UCC">8812345678934</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:Sender>
    <sh:Receiver>
      <sh:Identifier Authority="EAN.UCC">8712345678920</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:Receiver>
    <sh:DocumentIdentification>
      <sh:Standard>EAN.UCC</sh:Standard>
      <sh:TypeVersion>2.5</sh:TypeVersion>
      <sh:InstanceIdentifier>100002</sh:InstanceIdentifier>
      <sh:Type>OrderResponse</sh:Type>
      <sh:MultipleType>>false</sh:MultipleType>
      <sh:CreationDateAndTime>2009-08-07T09:30:00.000</sh:CreationDateAndTime>
    </sh:DocumentIdentification>
  </sh:StandardBusinessDocumentHeader>
  <eanucc:message>
    <entityIdentification>
      <uniqueCreatorIdentification>MSG938544</uniqueCreatorIdentification>
      <contentOwner>
        <gln>8812345678934</gln>
      </contentOwner>
    </entityIdentification>
  </eanucc:message>
</sh:StandardBusinessDocument>
```

```

<order:orderResponse responseStatusType="MODIFIED" creationDateTime="2009-08-
07T09:30:00Z" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:order="urn:ean.ucc:order:2">
  <responseIdentification>
    <uniqueCreatorIdentification>579135OR</uniqueCreatorIdentification>
    <contentOwner>
      <gln>8812345678934</gln>
    </contentOwner>
  </responseIdentification>
  <responseToOriginalDocument referenceDocumentType="34" referenceDateTime="2009-08-
07T09:00:00Z" referenceIdentification="4521560719U"/>
  <buyer>
    <gln>8712345678920</gln>
  </buyer>
  <seller>
    <gln>8812345678934</gln>
  </seller>
  <orderModification>
    <orderModificationLineItemLevel>
      <substituteItemIdentification>
        <gtin>88123456798753</gtin>
      </substituteItemIdentification>
      <modifiedOrderInformation number="1">
        <requestedQuantity>
          <value>15</value>
        </requestedQuantity>
        <tradeItemIdentification>
          <gtin>88123456798906</gtin>
        </tradeItemIdentification>
      </modifiedOrderInformation>
    </orderModificationLineItemLevel>
  </orderModification>
</order:orderResponse>
</eanucc:message>
</sh:StandardBusinessDocument>

```

The MultiShipmentOrder Modified instance file (Step 3a in the above scenario):

```

<?xml version="1.0" encoding="UTF-8"?>
<sh:StandardBusinessDocument
  xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
  xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader
sbdh/StandardBusinessDocumentHeader.xsd urn:ean.ucc:2 MultiShipmentOrderProxy.xsd">
  <sh:StandardBusinessDocumentHeader>
    <sh:HeaderVersion>1.0</sh:HeaderVersion>
    <sh:Sender>
      <sh:Identifier Authority="EAN.UCC">8712345678920</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">8812345678934</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>2.5</sh:TypeVersion>
  <sh:InstanceIdentifier>100003</sh:InstanceIdentifier>
  <sh:Type>MultiShipmentOrder</sh:Type>
  <sh:MultipleType>>false</sh:MultipleType>
  <sh:CreationDateAndTime>2009-08-07T09:45:00.000</sh:CreationDateAndTime>
</sh:DocumentIdentification>
</sh:StandardBusinessDocumentHeader>
<eanucc:message>
  <entityIdentification>
    <uniqueCreatorIdentification>MSG-123-34927</uniqueCreatorIdentification>
    <contentOwner>
      <gln>8712345678920</gln>
    </contentOwner>
  </entityIdentification>
  <eanucc:documentCommand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2">
    <documentCommandHeader type="CHANGE_BY_REFRESH">
      <entityIdentification>
        <uniqueCreatorIdentification>246820M</uniqueCreatorIdentification>
        <contentOwner>
          <gln>8712345678920</gln>
        </contentOwner>
      </entityIdentification>
    </documentCommandHeader>
    <documentCommandOperand>
      <order:multiShipmentOrder documentStatus="ORIGINAL" creationDateTime="2009-08-
      07T09:45:00Z">
        <orderIdentification>
          <uniqueCreatorIdentification>4521560719U</uniqueCreatorIdentification>
          <contentOwner>
            <gln>8712345678920</gln>
          </contentOwner>
        </orderIdentification>
        <orderPartyInformation>
          <seller>
            <gln>8812345678934</gln>
          </seller>
          <buyer>
            <gln>8712345678920</gln>
          </buyer>
        </orderPartyInformation>
        <orderLogisticalInformation>
          <shipToLogistics>
            <shipTo>

```



```

        <gln>8712345678944</gln>
      </shipTo>
    </shipToLogistics>
  </orderLogisticalDateGroup>
  <requestedDeliveryDate>
    <date>2009-08-08</date>
  </requestedDeliveryDate>
</orderLogisticalDateGroup>
</orderLogisticalInformation>
<multiShipmentOrderLineItem number="1">
  <requestedQuantity>
    <value>10</value>
  </requestedQuantity>
  <tradeItemIdentification>
    <gtin>88123456798753</gtin>
  </tradeItemIdentification>
</multiShipmentOrderLineItem>
<multiShipmentOrderLineItem number="2">
  <requestedQuantity>
    <value>20</value>
  </requestedQuantity>
  <tradeItemIdentification>
    <gtin>88123456798760</gtin>
  </tradeItemIdentification>
</multiShipmentOrderLineItem>
</order:multiShipmentOrder>
</documentCommandOperand>
</eanucc:documentCommand>
</eanucc:message>
</sh:StandardBusinessDocument>

```

The MultiShipmentOrder Cancelled instance file (Step 3b in the above scenario):

```

<?xml version="1.0" encoding="UTF-8"?>
<sh:StandardBusinessDocument
  xmlns:sh="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader"
  xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHeader
  sbdh/StandardBusinessDocumentHeader.xsd urn:ean.ucc:2 MultiShipmentOrderProxy.xsd">
  <sh:StandardBusinessDocumentHeader>
    <sh:HeaderVersion>1.0</sh:HeaderVersion>
    <sh:Sender>
      <sh:Identifier Authority="EAN.UCC">8712345678920</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">8812345678934</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>2.5</sh:TypeVersion>
  <sh:InstanceIdentifier>100003</sh:InstanceIdentifier>
  <sh:Type>MultiShipmentOrder</sh:Type>
  <sh:MultipleType>>false</sh:MultipleType>
  <sh:CreationDateAndTime>2009-08-07T09:45:00.000</sh:CreationDateAndTime>
</sh:DocumentIdentification>
</sh:StandardBusinessDocumentHeader>
<eanucc:message>
  <entityIdentification>
    <uniqueCreatorIdentification>MSG-123-34928</uniqueCreatorIdentification>
    <contentOwner>
      <gln>8712345678920</gln>
    </contentOwner>
  </entityIdentification>
  <eanucc:documentCommand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:eanucc="urn:ean.ucc:2" xmlns:order="urn:ean.ucc:order:2">
    <documentCommandHeader type="DELETE">
      <entityIdentification>
        <uniqueCreatorIdentification>246830C</uniqueCreatorIdentification>
        <contentOwner>
          <gln>8712345678920</gln>
        </contentOwner>
      </entityIdentification>
    </documentCommandHeader>
    <documentCommandOperand>
      <order:multiShipmentOrder documentStatus="ORIGINAL" creationDateTime="2009-08-
        07T09:45:00Z">
        <orderIdentification>
          <uniqueCreatorIdentification>4521560719U</uniqueCreatorIdentification>
          <contentOwner>
            <gln>8712345678920</gln>
          </contentOwner>
        </orderIdentification>
        <orderPartyInformation>
          <seller>
            <gln>8812345678934</gln>
          </seller>
          <buyer>
            <gln>8712345678920</gln>
          </buyer>
        </orderPartyInformation>
        <orderLogisticalInformation>
          <shipToLogistics>
            <shipTo>
              <gln>8712345678944</gln>
            </shipTo>
          </shipToLogistics>
          <orderLogisticalDateGroup>
            <requestedDeliveryDate>

```

```
        <date>2009-08-08</date>
      </requestedDeliveryDate>
    </orderLogisticalDateGroup>
  </orderLogisticalInformation>
  <multiShipmentOrderLineItem number="1">
    <requestedQuantity>
      <value>15</value>
    </requestedQuantity>
    <tradeItemIdentification>
      <gtin>88123456798906</gtin>
    </tradeItemIdentification>
  </multiShipmentOrderLineItem>
  <multiShipmentOrderLineItem number="2">
    <requestedQuantity>
      <value>20</value>
    </requestedQuantity>
    <tradeItemIdentification>
      <gtin>88123456798760</gtin>
    </tradeItemIdentification>
  </multiShipmentOrderLineItem>
</order:multiShipmentOrder>
</documentCommandOperand>
</eanucc:documentCommand>
</eanucc:message>
</sh:StandardBusinessDocument>
```