

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

STANDARDS RELATED DOCUMENT

ATDLP-7.04

FRAMEWORK DOCUMENT FOR THE REPRESENTATION OF TACTICAL DATA LINK (TDL) STANDARDS IN EXTENSIBLE MARKUP LANGUAGE (XML)

**EDITION A VERSION 1
JUNE 2021**



NORTH ATLANTIC TREATY ORGANIZATION

ALLIED TACTICAL DATA LINK PUBLICATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)
© NATO/OTAN

NATO UNCLASSIFIED

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

INTENTIONALLY BLANK

NATO UNCLASSIFIED

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

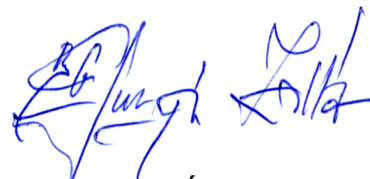
NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

1 June 2021

1. The enclosed Standards Related Document, ATDLP-7.04, Edition A, Version 1, FRAMEWORK DOCUMENT FOR THE REPRESENTATION OF TACTICAL DATA LINK (TDL) STANDARDS IN EXTENSIBLE MARKUP LANGUAGE (XML), which has been approved in conjunction with ATDLP-5.16 by the nations in the Consultation, Command and Control Board (C3B), is promulgated herewith.
2. ATDLP-7.04, Edition A, Version 1 is effective upon receipt and supersedes the XTDL FRAMEWORK DOCUMENT, dated February 2009, which shall be destroyed in accordance with the local procedure for the destruction of documents.
3. ATDLP-7.04, Edition A, Version 1 incorporates the following changes to the original xTDL Framework Document:
 - Specifics about the representation of the Link 16 Specification in XML into ATDLP-7.04 (see Annex C);
 - Updates to the URN of the xTDL Generic Schemas;
 - XML Schema for NATO Implementation Codes and Rules (NICR); and
 - Alignment of security related terminology with STANAG 4774; as well as general editorial and technical corrections.
4. This NATO standardization document is issued by NATO. In case of reproduction, NATO is to be acknowledged. NATO does not charge any fee for its standardization documents at any stage, which are not intended to be sold. They can be retrieved from the NATO Standardization Document Database (<https://nso.nato.int/nso/>) or through your national standardization authorities.
5. This publication shall be handled in accordance with C-M(2002)60.



Zoltán GULYÁS
Brigadier General, HUNAF
Director, NATO Standardization Office

NATO UNCLASSIFIED

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

INTENTIONALLY BLANK

NATO UNCLASSIFIED

TABLE OF CONTENTS

<u>Chapter</u>		<u>Page</u>
1.	INTRODUCTION.....	1-1
2.	BACKGROUND.....	2-1
3.	PURPOSE.....	3-1
4.	APPROACH.....	4-1
5.	THE XTDL FRAMEWORK.....	5-1
6.	NAMING CONVENTION FOR THE REPRESENTATION OF TDL STANDARDS IN XML	6-5
6.1	xTDL Naming Convention Design Considerations	6-5
6.1.1	Use of the Period (".") Character	6-5
6.1.2.	XML Element and Type Name Uniqueness	6-5
6.1.3.	Avoiding Duplicatively Named Element and Type Names	6-6
6.1.4.	Use of Abbreviations and Acronyms	6-6
6.2	xTDL Naming Convention Rules	6-7
6.3	xTDL Namespace Naming Convention	6-9
7	CONFIGURATION MANAGEMENT OF TDL CAT CI DOCUMENTS IN XML.....	7-1
7.1	Representation of the structure and content of TDL CaT CI documents in XML	7-1
7.1.1	Logical Model capturing the structure of TDL CaT CI documents in XML	7-1
7.1.2	Physical Model capturing Metadata of TDL CAT CI documents in XML.....	7-1
7.1.2.1	The “ BaselineInfo ” element.....	7-2
7.2	Logical Model capturing the components required for the CM of TDL CaT CIs	7-6
7.3	Physical Model capturing the components required for the CM of TDL CaT CIs	7-8
8	REPRESENTATION OF THE TDL DATA ELEMENT DICTIONARY IN XML.....	8-1
8.1	Prerequisites for the usage of a generic TDL Data Element Dictionary	8-1

<u>Chapter</u>	<u>Page</u>
8.2	Logical Model capturing the components of the generic TDL Data Element Dictionary 8-3
8.3	Specification of Data Element values 8-5
8.4	The Generic xTDL Data Element Dictionary schema 8-7
8.4.1	The “DataElements” element 8-7
8.4.2	The “Dui” element 8-9
8.4.3	The “CodingSwitch” element 8-12
8.4.4	The “Enum” element 8-15
8.5	Description of Types and Elements used within the Generic xTDL Data Element Dictionary schema 8-17
8.6	Relationship of the Generic xTDL DED schema with other XML schemas 8-21
8.7	Considerations for the usage of the Generic xTDL Data Element Dictionary schema 8-22
9	REPRESENTATION OF THE TDL MESSAGE STRUCTURE IN XML 9-1
9.1	Prerequisites for the usage of a generic TDL message structure 9-1
9.2	Logical Model capturing the components of a generic TDL message structure 9-3
9.3	The Generic xTDL Message Structure schema 9-5
9.3.1	The “MessageStructure” element 9-5
9.3.2	The “MessageCatalogue” element 9-6
9.3.3	The “Word” element 9-8
9.3.4	The “StructureSwitch” element 9-10
9.3.5	Description of Types and Elements used within the Generic xTDL Message Structure schema 9-13
9.4	Relationship of the Generic xTDL Message Structure schema with other XML schemas 9-15
9.5	Considerations for the usage of the Generic xTDL Message Structure schema 9-16
10	REPRESENTATION OF TDL PROCESSING IN XML 10-1
10.1	Transactional TDL Processing 10-1
10.1.1	Logical Model capturing transactional TDL Processing 10-1
10.1.2	Logical Model capturing the components required for TDL Transactions 10-3
10.2	TDL Processing based on Transmit and Receive Rules 10-5
11	REPRESENTATION OF DATA FORWARDING IN XML 11-1
11.1	Logical Model capturing the High-Level TDL Forwarding Rule Concepts 11-1

<u>Chapter</u>		<u>Page</u>
12	BUSINESS RULES.....	12-1
13	REPRESENTATION OF MINIMUM IMPLEMENTATION / IMPLEMENTATION REQUIREMENTS IN XML	13-1
14	TDL MESSAGE INSTANCES.....	14-1
15	REPRESENTATION OF THE TDL SYSTEM IMPLEMENTATION IN XML	15-1
15.1	Prerequisites for the usage of a generic TDL System Implementation.....	15-1
15.2	Logical Model capturing the components of the generic TDL SID Schema.....	15-4
15.3	Transmit/receive values	15-6
15.4	Switches.....	15-6
15.4.1	Structure switches.....	15-7
15.4.2	Implementation switches	15-7
15.4.3	Coding switches	15-7
15.5	Implementation based on several STANAG/ATDLP versions	15-8
15.6	The Generic TDL System Implementation Data schema	15-9
15.6.1	SID Top-level structure	15-9
15.6.2	The PlatformInfo element	15-10
15.6.3	SID Classification Markings.....	15-11
15.6.4	The SID ImplementationCodes and Message element	15-13
15.6.5	The SID Word element.....	15-14
15.6.6	The SID DataElementDictionary with DFI and DUI elements.....	15-16
15.7	Rules enforcement	15-17
15.8	SID XML Instance Verification.....	15-17

<u>Chapter/Annex</u>	<u>Page</u>
16 ANNEXES.....	16-1
A. SPECIFICS ABOUT THE REPRESENTATION OF LINK 1 IN XML	A-1
B. SPECIFICS ABOUT THE REPRESENTATION OF LINK 11/11B IN XML	B-1
C. SPECIFICS ABOUT THE REPRESENTATION OF LINK 16 IN XML	C-1
C.1 General	C-1
C.2 xTDL Data Element Dictionary	C-1
C.3 xTDL Message Structure	C-1
C.4 xTDL Processing.....	C-1
C.5 Data Forwarding.....	C-1
C.6 Business Rules	C-2
C.7 Implementation Requirements (IMP REQ)	C-2
D. SPECIFICS ABOUT THE REPRESENTATION OF LINK 22 IN XML	D-1
E. DATA FORWARDING IN XML	E-1
F. EXAMPLE OF AN XTDL MESSAGE INSTANCE	F-1
G. OVERVIEW OF THE TDL GOVERNANCE NAMESPACE NMRR FOLDER STRUCTURE	G-1
G.1 Overview of the TDL Governance Namespace NMRR Folder Structure	G-1
H. OVERVIEW OF AUTHORITATIVE XTDL AND RELATED XML SCHEMAS	H-1
H.1 List of Authoritative xTDL Schemas	H-1
H.2 List of related XML Schemas	H-1
I. DESCRIPTION OF THE UML MODEL SYMBOLOGY USED WITHIN THE XTDL UML MODELS.....	I-1
I.1 UML Model Symbology used within the xTDL UML Models.....	I-1

LIST OF FIGURES

Figure	Page
5-1	Graphical view of the xTDL Framework components..... 5-1
5-2	Relationship between xTDL artefacts and the integrity checking process 5-3
7-1	xTDL schema depicting the element BaselineInfo as root element with its related child elements as container for TDL CaT CI related metadata 7-3
7-2	Logical Model capturing the components required for the CM of TDL CaT CIs..... 7-6
7-3	Physical Model capturing the components required for the CM of TDL CaT CIs 7-8
8-1	Logical Model capturing the components of a generic xTDL Data Element Dictionary 8-4
8-2	xTDL schema depicting the element “ DataElements ” as root element with its related child elements as part of the generic TDL Data Element Dictionary 8-8
8-3a	xTDL schema depicting the element “ Dui ” with its related child elements as part of the generic TDL Data Element Dictionary..... 8-10
8-3b	xTDL schema depicting the element “ Dui ” with its related child elements as part of the generic TDL Data Element Dictionary..... 8-11
8-4	xTDL schema depicting the element “ CodingSwitch ” with its related child elements as part of the generic TDL Data Element Dictionary 8-13
8-5	xTDL schema depicting the element “ Enum ” as root element with its related child elements as part of the generic TDL Data Element Dictionary 8-16
9-1	Logical Model capturing the components of a generic TDL message structure..... 9-4
9-2	xTDL schema depicting the element “ MessageStructure ” as root element with its related child elements..... 9-5
9-3	xTDL schema depicting the element “ MessageCatalogue ” with its related child elements as part of the generic TDL message structure 9-7
9-4	xTDL schema depicting the element “ Word ” with its related child elements as part of the generic TDL message structure 9-9

LIST OF FIGURES (continued)

Figure	Page
9-5	xTDL schema depicting the element “ StructureSwitch ” with its related child elements as part of the generic TDL message structure 9-11
10-1	Logical Model capturing the components required for transactional TDL Processing 10-2
10-2	Logical Model capturing the components required for TDL Transactions..... 10-4
11-1	Logical Model capturing the components required for the representation of High-Level TDL Forwarding Rule Concepts 11-2
15-1	Logical Model of the components of System Implementation Data 15-5
15-2	SID “ CodingSwitch ” element 15-8
15-3	Top level structure of SID Schema..... 15-10
15-4	SID “ PlatformInfo ” element..... 15-11
15-5	SID Implementation Codes and “ Message ” element 15-13
15-6	SID “ Word ” element..... 15-15

Figure	Page
G-1	TDL Governance Namespace NMRR Folder Structure, Level 1G-2
G-2	TDL Governance Namespace NMRR Folder Structure, Level 2G-3
I-1	UML Model Class Diagram Description I-1
I-2	Description of the UML Model symbols used within the xTDL UML Models I-2

LIST OF TABLES

Table	Page
6-1 Examples of xTDL Tag Names	6-4
6-2 xTDL URN segments	6-5
6-3 Examples for xTDL URNs	6-6
7-1 Description of the Types used within the xTDL schema depicting the element “ BaselineInfo ” as root element with its related child elements	7-4
7-2 Description of the Elements used within the xTDL schema depicting the element “ BaselineInfo ” as root element with its related child elements	7-5
7-3 Description of the Classes used within the Logical Model capturing the components required for the CM of TDL CaT CIs	7-7
7-4 Description of the Classes used within the Physical Model capturing the components required for the CM of TDL CaT CIs	7-9
8-1 Generic concepts for the Generic TDL DED	8-2
8-2 Description of Elements used within the Logical Model capturing the components of a generic xTDL Data Element Dictionary	8-4
8-3 Examples of Data Element value representation	8-6
9-1 Generic concepts for the generic TDL message structure	9-2
9-2 Description of Elements used within the Logical Model capturing the components of a generic TDL message structure	9-4
9-3 Description of Types used within the Generic xTDL Message Structure schema	9-13
10-1 Description of Classes used within the Logical Model capturing the components required for transactional TDL Processing	10-2
10-2 Description of Classes used within the Logical Model capturing the components required for TDL Transactions	10-4
11-1 Description of Classes used within the Logical Model capturing the High-Level TDL Forwarding Rule Concepts	11-3
15-1 Generic concepts for the Generic TDL SID Schema	15-2
15-2 Description of Elements used within the Logical Model capturing the components of System Implementation Data	15-6

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

LIST OF TABLES (continued)

Table	Page
H-1 List of Authoritative xTDL Schemas	H-1
H-2 List of related XML Schemas	H-1

1. INTRODUCTION

The goal of the Tactical Data Links (TDLs) in eXtensible Mark-Up Language (XML) initiative is to develop representations of TDL-related Standards in XML, in order to:

- enhance the current Configuration Management (CM) process for TDL-related publications and documentation used in NATO, e.g. by improving the evaluation and publishing process,
- increase the efficiency and functionality to end users, by harmonizing the information exchange standards and procedures, and by providing a capability for automated generation of system implementations to facilitate the development of compliant TDL systems which improves interoperability,
- open TDL specifications for integration into an NATO Network Enabled Capability (NNEC) environment by using open, recommended and widely-used standards with expected widespread growth in the area of information management.

With the transformation into XML, the existing capability for binary exchanges shall be preserved while providing an additional capability for data exchanges in XML.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

2. BACKGROUND

At the end of 2005, as a result of an analysis of evolving technologies, the NATO TDL Community of Interest (COI), represented in the Data Link Working Group (DLWG) - WG/1 under the Information Services Sub Committee (ISSC) – SC/5, took the decision to transform its Configuration Item (CI) documents into XML.

In February 2006, a TDL-XML Syndicate (TDLXMLS) was established as a temporary forum under governance of the DLWG, which was superseded by the Tactical Data Link Capability Team (TDL CaT) under the Communication and Information Capability Panel (CIS CaP, CP/1) in November 2011, in order to:

- analyse the developmental steps required to transform TDL CaT CI documents into XML,
- provide advice to the TDL CaT in subjects related to XML, and to
- coordinate the TDL-XML transformation process.

As a first step, the syndicate identified the requirement for the development of an xTDL Framework, xTDL Best Practice Guidelines, and a common xTDL Vocabulary to support the overall transformation process.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

NATO UNCLASSIFIED

3. PURPOSE

This “Framework Document for the Representation of TDL Standards in XML” provides a set of guidelines, articulated both in terms of design and XML implementation, that will promote a common approach across several potential contributors to the transformation process.

The document shall record the method of work and products developed during the transformation process. It contains a set of Unified Modelling Language (UML) models and references to associated XML schemas (XSD) supporting TDL CaT CIs.

ATDLP-7.04 is predominately designed for use by individuals who have a background on the architecture of TDL standards and TDL-related procedures, which are e.g.

- configuration managers for TDL specifications,
- system architects,
- system implementers, and
- Information Exchange Requirement (IER) standard developers.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

NATO UNCLASSIFIED

4. APPROACH

In order to define the different components associated with the

- Configuration Management (CM),
- message formatting (Binary and XML), forwarding and processing, and
- documentation (Main Body and Annexes),

and to allow the analysis of their relationship, the TDLXMLS decided to use UML 2.1 models, describing the components as classes and attributes. The UML Model symbology used within this document is described in Annex I.

All datum types that exist in the TDL standards are expressed as class diagrams, including the explicit associations between the classes. The models provide different views of the information. Each UML model is supported by a table providing a brief description of the classes being used.

TDL message structures are developed in response to IERs for an operational domain and to imbed metadata at TDL design time. Exposing this data explicitly will enable developers, implementers, and future users to understand how the data was envisaged to be used.

As a first incremental step, ATDLP-7.04 will focus on the representation of the Link 16 (J-series message) Standard in XML, starting with STANAG 5516 Edition 5.

In further steps, additional standard specifications, operating procedures, data forwarding protocols and technical documentation under CM of the TDL CaT will be addressed.

xTDL components developed within the TDL COI are expected to be registered/published in the NATO Meta Data Registry and Repository (NMRR) as soon as they have reached a certain maturity level, which needs to be defined.

Terminology used within this document adheres to terminology defined and recommended by the World Wide Web Consortium (W3C) and by the NATO XML Management Services Working Group (XMLSWG), unless otherwise noted.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

NATO UNCLASSIFIED

5. THE XTDL FRAMEWORK

The xTDL Framework comprises those components described in the different TDL specifications (domains), which are required to develop an authoritative suite of XML representations that would encompass the types of data currently configuration managed through the formal TDL CaT CM process (see Figure 5-1 below).

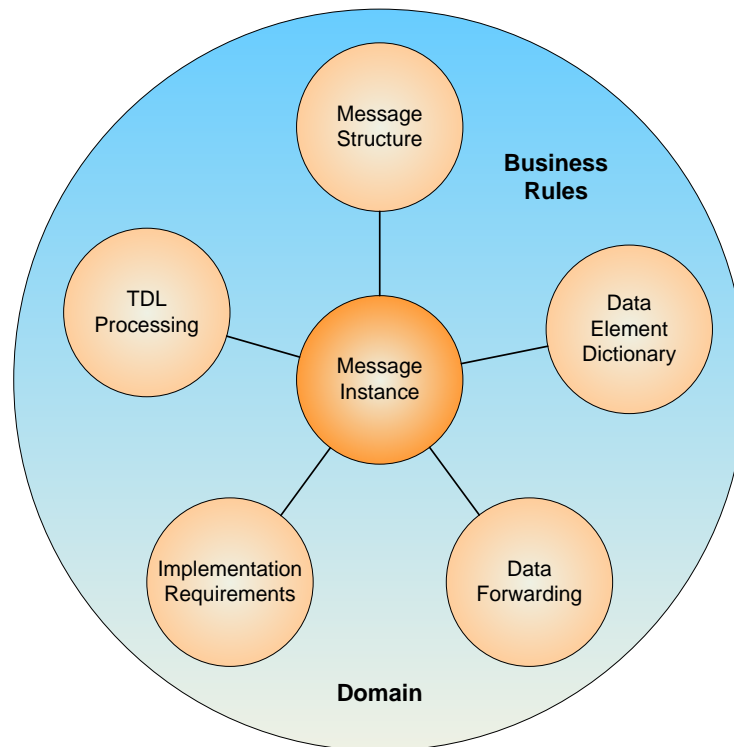


Figure 5-1: Graphical view of the xTDL Framework components

xTDL representations will be developed to characterize the following components, which will be described in more detail in the following chapters:

- **MESSAGE STRUCTURE**, providing the definition of the structure of a TDL message and which data elements each can contain;
- **DATA ELEMENT DICTIONARY (DED)**, containing a dictionary of allowed data elements, which includes all information to encode and decode between its representations;
- **TDL PROCESSING**, describing the transactions, transmit and receive rules and all corresponding tables;

- **MINIMUM IMPLEMENTATION (MIN IMP) / IMPLEMENTATION REQUIREMENTS (IMP REQ)**, identifying the data exchange requirements that must be implemented by platforms according to particular functional areas in which they participate on TDLs;
- **DATA FORWARDING**, providing the definition on how to report information that is exchanged between different TDL standards;
- **MESSAGE INSTANCE**, providing information on how an instance of a message, either the binary form or in XML (literal or integer), should be constituted;
- **BUSINESS RULES**, defining rules necessary to tie the previous components together and are applicable to the specific domains.

Apart from the core elements identified above, TDL standards also contain other text, which is not captured within the elements above, however needs to be taken into consideration for a representation in XML (see C 7.1.1).

The approach in the development of the UML models in support of the framework was predicated on establishing a logical model illustrative of the family of TDLs. This approach was explicitly taken to preclude the continued independent development of unique solutions for each TDL standard.

To achieve this goal, these models are initially founded on using, as tasked by the former DLWG, the terminology and structures contained in the Link 16 J-series message standard, with the aim to develop a suite of generic xTDL representations and models covering TDL specifications for Link 1, Link 11/11B, Link 16, Link 22, and future TDLs in the long term.

As a first step, based on the models, several XML schemas were developed, that each cover one of the above mentioned elements of Link 16, but in such a generic way, that they also support further TDL standards. Each of these XML schemas will have its own namespace in the sense of a Uniform Resource Identifier (URI) reference within an XML artefact ¹. To support these schemas, XML instance documents are required to represent each of these elements. To ease CM and increase reusability, the TDL standard is represented by several modular XML instance documents instead of only one large one. The set of documents together are regarded to be the xTDL standard representation.

The validity of the separate XML instance documents is verified using their respective schema. However, to test the integrity between elements

¹ The term "namespace" also is used as a container of items within the NMRR, in the context of this document referred to as "governance namespace".

in the separate XML documents, an additional process as depicted in Figure 5-2 below is required, because an XML schema with “**key/keyref**” constructs cannot be used across documents. Applying this procedure will be part of the CM process and the process itself will not be part of the CI document.

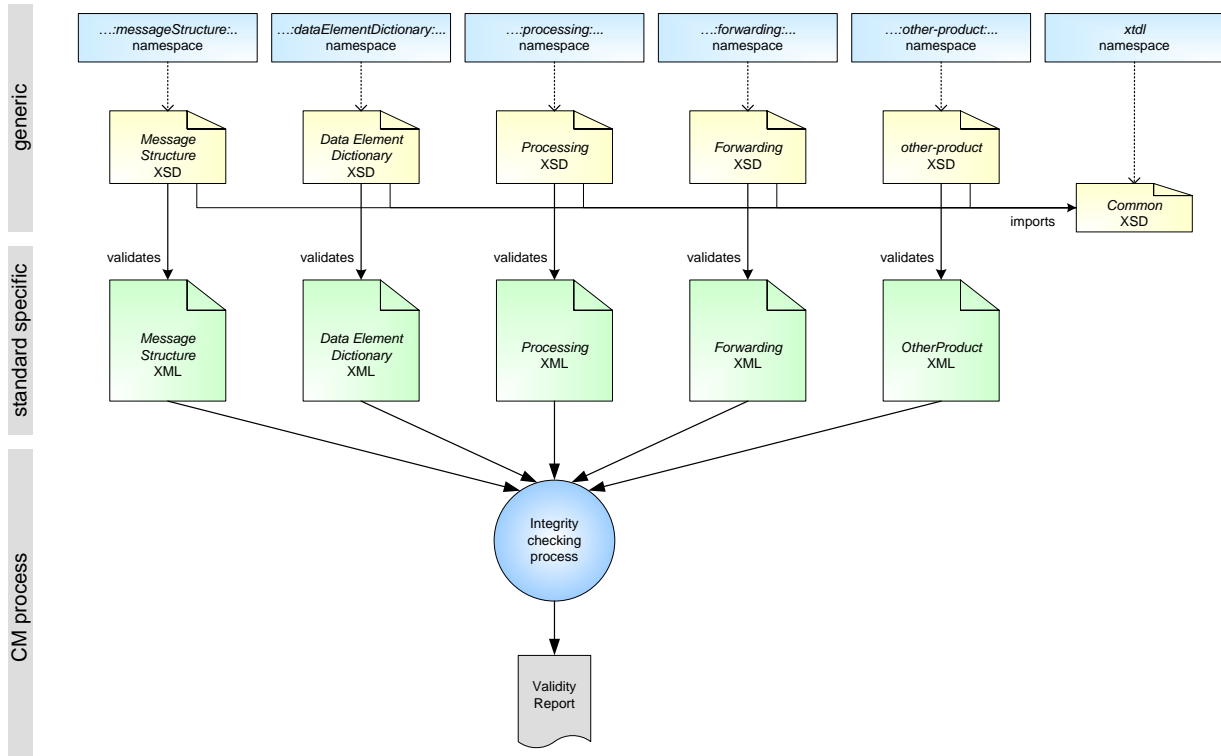


Figure 5-2: Relationship between xTDL artefacts and the integrity checking process

The main body of this document describes the generic UML models and XML schemas required to represent the TDL standards in XML, while specifics referring to the various TDL standards can be found in the annexes.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

NATO UNCLASSIFIED

6. NAMING CONVENTION FOR THE REPRESENTATION OF TDL STANDARDS IN XML

This chapter describes the naming convention for the representation of TDLs in XML (xTDL NC), also used within the related UML models, in particular for xTDL schemas, to which all xTDL objects shall comply.

This naming convention is deemed necessary for consistent physical implementation and interpretation of metadata objects in a large scale distributed environment. It is derived from best practices and lessons learned from a number of NATO XML development and implementation activities. Applying the xTDL NC to both the XML schema and the UML models means that the same component may have two different renderings. For example, an attribute shown in an xTDL physical model may be represented as an element in XML. However, traceability between the models and the XML schemas will be easy as the difference is minor.

6.1 xTDL Naming Convention Design Considerations

The xTDL NC adheres to the W3C XML 1.0 specification and to ISO/IEC 11179-5, Specification and Standardization of Data Elements – Naming and Identification Principles for Data Elements. It in general complies with the conventions described in the “Guidance for XML Naming and Design within NATO” (GXND).

Further considerations are:

6.1.1 Use of the Period (".") Character

The xTDL NC disallows the use of the period (".") character. Modern programming languages utilize the period (".") character to separate a class from its methods; for example, the class "file" can have methods "file.open" and "file.close". When using these XML names as objects in a programming language, the period cannot be used as part of a name.

6.1.2 XML Element and Type Name Uniqueness

Similar to programming languages, XML has a hierarchical structure that supports independent use of local and global declarations within a single namespace. W3C XML schema design rules do not allow duplicative global XML elements and type names when their declarations are different, which occurs when there are multiple XML elements with the same name at the same hierarchical level. Therefore, the xTDL NC design guidelines help ensure that only unique XML names are used within a single XML namespace.

6.1.3. Avoiding Duplicatively Named Element and Type Names

The W3C XML schema language allows both an XML element and an XML type ("**complexType**" and "**simpleTypes**") to have the same name. While legal in XML schema, the use of the same name for an element and its supporting type definition adds ambiguity. A simple device to achieve uniqueness between XML elements and XML type names is to append an underscore "_" and the word "Type" to the type name, e.g. "**TargetType_Type**", which would have the advantage of being easy to automate. It should also be noted, that the XML type names do not appear in an XML instance document, so that human readability should not be an issue.

6.1.4. Use of Abbreviations and Acronyms

The use of abbreviations and acronyms may be meaningful inside a COI, but when supporting the NNEC design goal of discovery, acronyms and abbreviations take on an ambiguity that is difficult to resolve with automated tools. Therefore, it is best to avoid the use of abbreviations and acronyms, where possible. When it is not possible to avoid the use of abbreviations and acronyms, each abbreviation or acronym is to be treated as a word; for example: "TDL Naming Convention" would become "**TdlNamingConvention**". If an abbreviation or acronym is used, an annotation should provide the full name. While annotations only appear in the (documentation) schema, they may be utilized by an application that can render this information into a GUI for a user, for example, in a help menu.

6.2 xTDL Naming Convention Rules

The following rules shall be applied to generate a valid XML tag name from a character string designated for use as the value of an xTDL schema element name. Examples can be found further down below.

- (1) Apply the Upper Camel Case (UCC) naming convention for xTDL elements and element types, Lower Camel Case (LCC) for xTDL attributes, attribute types, and attribute groups. In all cases, acronyms and abbreviations are to be treated as words;
- (2) Replace each single occurrence or sequence of non-alphanumeric characters with a single underscore '_'. If the filtered element name ends in an underscore '_', remove the final underscore. This prevents the occurrence of a double underscore '__' when the corresponding type is developed. For example, "DEPTH INDICATOR (SONOBUOY)" becomes "**DepthIndicator_Sonobuoy**".
- (3) If the first character of the resultant string is a digit, then prepend a single underscore '_'.
- (4) In instances where the resultant name will be duplicative within a namespace, then the following additional steps must be taken in order to ensure name uniqueness. Note that duplicative names are not meant as separate instances of the same element or attribute, but rather instances where a single name is being associated to different semantic concepts or syntactic declarations.
 - A. appended an underscore followed by an integer that is incremented up from 1 for each duplicative term to the end of the name;
 - B. assign a digit to all instances of the name.
- (5) For xTDL types, append "**_Type**".
- (6) For xTDL attributeGroups, apply the attribute NC and append "**_Group**".

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

Examples:

Original Name	xTDL Element Name	Rules applying
LAND TRACK PPLI MESSAGE	"LandTrackPpliMessage"	(1)
1ST PRIORITY/CRITICAL	"_1stPriority_Critical"	(1), (2), (3)
136 FT	"_136Ft"	(1), (3)
PPLI IFF/SIF INDICATOR	"Pplilff_SifIndicator"	(1), (2)
LATITUDE, 0.0051 MINUTE	"Latitude_0_0051Minute"	(1), (2)
LABEL, J-SERIES	"Label_J_Series"	(1), (2)
SUBSURFACE (MARITIME) SPECIFIC TYPE	"Subsurface_Maritime_SpecificType"	(1), (2)
DEPTH 15 METERS	"Depth15Meters"	(1)
SPARE 1 SPARE 2	"Spare_1" "Spare_2"	(1), (4)
Original Name	xTDL Attribute Name	Rules applying
Baseline	"baseline"	(1)
TDL Encoding	"tdlEncoding"	(1)
Original Name	xTDL Type Name	Rules applying
SUBSURFACE (MARITIME) SPECIFIC TYPE	"Subsurface_Maritime_SpecificType_Type"	(1), (2), (5)
DEPTH 15 METERS	"Depth15Meters_Type"	(1), (5)
TDL Encoding	"tdlEncoding_Type"	[attribute] (1), (5)
Original Name	xTDL Group Name	Rules applying
LOCATION	"location_Group"	[attribute] (1), (6)
MANAGED GROUP	"managedGroup_Group"	[attribute] (1), (6)

Table 6-1: Examples of xTDL Tag Names

6.3 xTDL Namespace Naming Convention

All xTDL components that require a namespace will have to conform to the following uniform xTDL Namespace Naming Convention (xTDL Namespace NC).

The following URN structure is to be used for xTDL components:

“urn:nato:tdl:<specification>:<component>:<version>:<release date>:<status>”

“xTDL URN segment”	Description																		
“urn:nato:tdl”	requisite prefix that will be applied to all xTDL components developed by the TDL COI in accordance with the GXND;																		
“<specification>”	Identifies the specification that the schema represents, which may be a STANAG, ATDLP, DLCP, or any other applicable standard document. The term “generic” will be used when representing the generic xTDL schema;																		
“<component>”	<p>Identifies the conceptual component being defined by the schema. The acceptable values for this field, as defined by the components identified in ATDLP-7.04, are:</p> <table> <tr> <th>Component</th><th>“xTDL URN component”</th></tr> <tr> <td>Message Structure</td><td>“messageStructure”</td></tr> <tr> <td>Data Element Dictionary</td><td>“dataElementDictionary”</td></tr> <tr> <td>Processing</td><td>“processing”</td></tr> <tr> <td>Forwarding</td><td>“forwarding”</td></tr> <tr> <td>Implementation Requirements</td><td>“implementationRequirements”</td></tr> <tr> <td>Minimum Implementation</td><td>“minimumImplementation”</td></tr> <tr> <td>Message Instance</td><td>“messageInstance”</td></tr> <tr> <td>System Implementation Data</td><td>“systemImplementationData”</td></tr> </table>	Component	“xTDL URN component”	Message Structure	“messageStructure”	Data Element Dictionary	“dataElementDictionary”	Processing	“processing”	Forwarding	“forwarding”	Implementation Requirements	“implementationRequirements”	Minimum Implementation	“minimumImplementation”	Message Instance	“messageInstance”	System Implementation Data	“systemImplementationData”
Component	“xTDL URN component”																		
Message Structure	“messageStructure”																		
Data Element Dictionary	“dataElementDictionary”																		
Processing	“processing”																		
Forwarding	“forwarding”																		
Implementation Requirements	“implementationRequirements”																		
Minimum Implementation	“minimumImplementation”																		
Message Instance	“messageInstance”																		
System Implementation Data	“systemImplementationData”																		
“<version>”	Uniquely identifies a specific release of an xTDL component. This corresponds to the edition, revision, issue, change or other similar identifying indicator for STANAGs, ATDLPs, DLCPs, and other specific products. For generic components, this is a consecutive, sequential integer assigned and maintained by the Data Link Support Staff (DLSS) of the NATO Headquarters C3 Staff (NHQC3S).																		

"xTDL URN segment"	Description
"<release date>"	Identifies the date of product release. The date is specified in the format: "YYYYMMDD". Releases may take place after each TDL CaT meeting (e.g. following the MG agreement for movement of DLCs to Supplement Section "Foxtrot"). This may occur more frequently than the baseline/edition release cycle, thus necessitating the specification of a release date. The release authorization body has yet to be determined.
"<status>"	Identifies the status of a component as either " draft " or " final ". Draft components are considered developmental in nature and should be treated as such. Final components are considered agreed and vetted in nature and can be relied upon as such.

Table 6-2: xTDL URN segments

Examples for xTDL URNs are:

Example xTDL component	Corresponding xTDL URN
The final issue of version 1 of the generic xTDL schema defining the TDL message structure concept released on 21 March 2005.	"urn:nato:tdl:generic:messageStructure:1:20050321:final"
A draft issue of version 4 of the generic xTDL schema defining the DED concept released on 25 December 2009.	"urn:nato:tdl:generic:dataElementDictionary:4:20091225:draft"
A draft issue of the STANAG 5516 Edition 5 Message Instance schema released on 22 September 2008.	"urn:nato:tdl:stanag5516:messageInstance:5:20080922:draft"

Table 6-3: Examples for xTDL URNs

7 CONFIGURATION MANAGEMENT OF TDL CAT CI DOCUMENTS IN XML

The TDL CaT is tasked with the execution of CM for TDL-related publications and documentation (TDL CaT CI documents) used within NATO, which are:

- STANAGs – or ATDLPs where applicable, describing the technical specifications for TDL standards and the data forwarding protocols,
- ATDLP Standard Related Documents (SRDs), describing the Standard Operating Procedures (SOPs) for the employment of the TDL Standards, and in addition to that,
- ATDLP Standard Related Documents (SRDs), providing supporting information.

7.1 Representation of the structure and content of TDL CaT CI documents in XML

The TDL CaT must manage all aspects of a TDL CaT CI document, the structure of the document, which consists of one or more covering pages, a main body, and appendices, annexes and/or volumes, if applicable, and the content of the document, comprising text, tables, and figures that support the explanation of the TDL formatted messages and supporting information.

7.1.1 Logical Model capturing the structure of TDL CaT CI documents in XML

The structure of a TDL CaT CI document must comply with agreed NATO directives for the development and production of NATO documentation, e.g. AAP-3, applicable for the development and production of STANAGs and Allied Publications (APs). This requires the development of a logical model capturing the common structure of NATO documents at a higher level, which then, when available, will have to be applied by the TDL COI for the representation of TDL CaT CI documents in XML.

7.1.2 Physical Model capturing Metadata of TDL CAT CI documents in XML

For the representation of the content and structure in XML of a TDL CaT CI document metadata is required, which includes the title, version, and classification as mandatory elements.

7.1.2.1 The “**BaselineInfo**” element

The metadata necessary to represent a TDL CaT CI document in XML is captured in the element “**BaselineInfo**”, for which the Type is defined in the Common XSD (see Chapter 5, Figure 5-2). The XML schema for this Type is depicted in Figure 7-1 below.

The set of XML elements to convey the classification is placed in its own namespace, which for now is xTDL specific, but will be aligned with a common NATO security specification in XML when available.

Each XML instance document that characterizes a component of the TDL standard will contain the “**BaselineInfo**” element and the various XML instance documents that together comprise a specific edition or version of a TDL standard should each have the same “**BaselineInfo**” element contents.

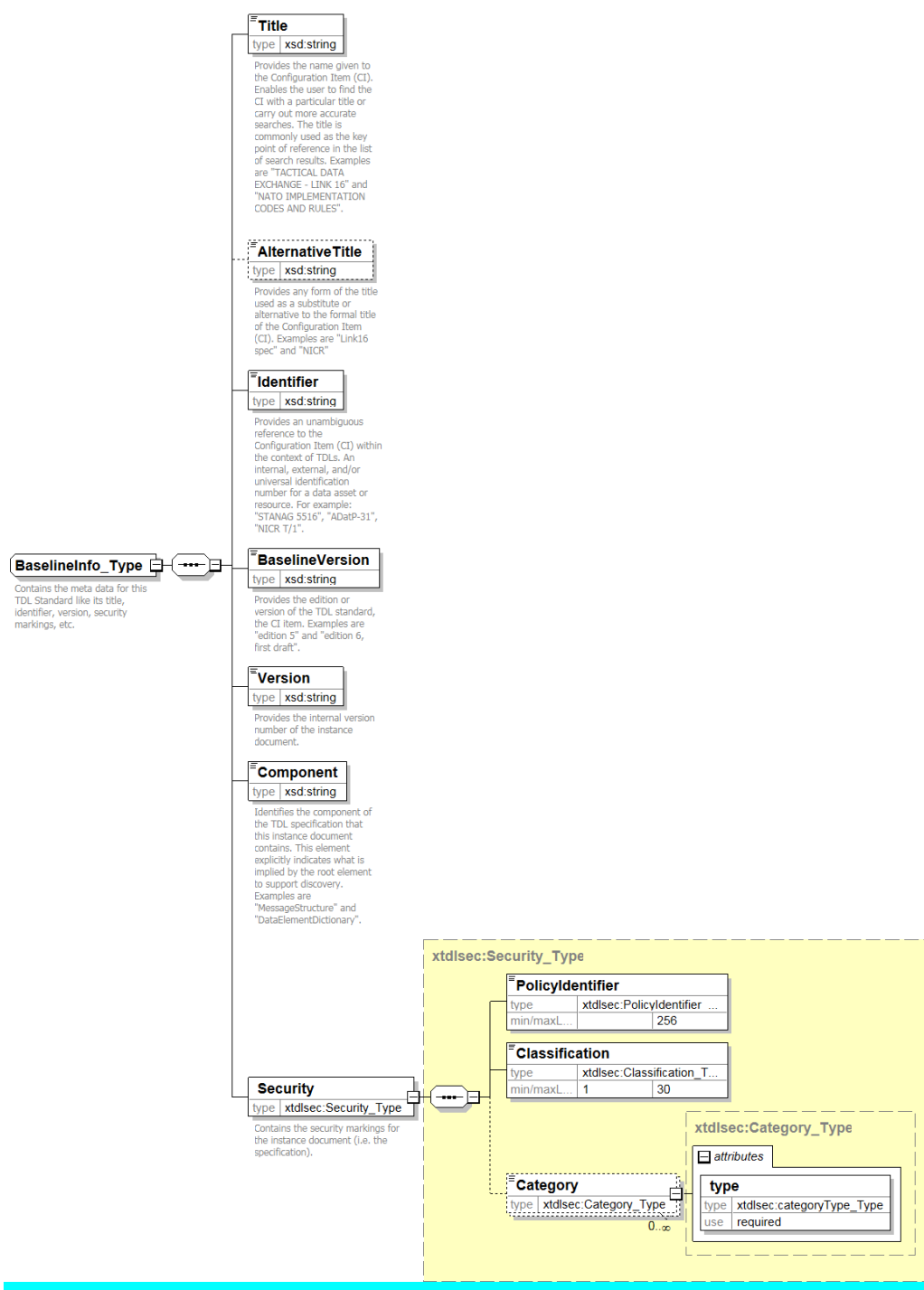


Figure 7-1: xTDL schema depicting the element “**BaselineInfo**” as root element with its related child elements as container for TDL CaT CI related metadata

An example of an XML instance corresponding to the XML schema above is shown in Section 9.3.1, where it is included in a Message Structure instance document.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

The tables below provide the description of the types and elements used in the above xTDL schema (automatically extracted from the schema's annotations).

"Type"	Description
"BaselineInfo_Type"	Contains the metadata for this TDL standard like its title, identifier, version, classification markings, etc.
"Category_Type"	<p>Provides an indication of an additional, specific sensitivity, or a dissemination control, or an informational marking on which no automated access control is performed (see reference: EAPC(AC/322-SC/5)N(2006)0008).</p> <p>Special category designators include: ATOMAL, CRYPTO, SIOP, SIOP ESI;</p> <p>Dissemination Limitation Markings include: EXCLUSIVE, INTELLIGENCE, LOGISTICS, OPERATIONS;</p> <p>Release categories include: RELEASABLE TO xxxxx, RELEASABLE FOR xxxxx (e.g. RELEASABLE TO ISAF, RELEASABLE TO ALBANIA/CROATIA).</p> <p>Administrative markings include: MANAGEMENT, STAFF, PERSONAL, MEDICAL, COMMERCIAL.</p>
"Classification_Type"	<p>Provides classification markings that indicate the sensitivity level of the information (see reference: EAPC(AC/322-SC/5)N(2006)0008).</p> <p>Examples as defined in AC/322-D(2004)0021 and in the "Guidance on the use of metadata element descriptions for use in NDMS": are: UNMARKED, UNCLASSIFIED, RESTRICTED, CONFIDENTIAL, SECRET, and COSMIC TOP SECRET.</p>
"PolicyIdentifier_Type"	<p>Identifies the nation or organization responsible for creating, maintaining, and implementing the security or information management policy to be applied to the information.</p> <p>The security or information management policy is understood as a set of rules for protecting information against unauthorized disclosure, while maintaining authorized access, and preventing loss of unauthorized modification. The policy bodies of different security or information management domains must agree on a common understanding of the handling requirements for information of a particular sensitivity.</p> <p>After the understanding exists, mappings from one security or information management policy to another can be created (see reference: EAPC(AC/322-SC/5)N(2006)0008), for example: NATO, NATO/EAPC, NATO/PFP, NATO/EU, NATO/RUSSIA, NATO/UKRAINE. National use includes: e.g. USA, FRA, GBR, NLD, etc.</p>
"Security_Type"	Provides specific Information Assurance (IA) metadata for data objects; supports typical existing labels to express policy, classification and dissemination attributes.

Table 7-1: Description of the Types used within the xTDL schema depicting the element **"BaselineInfo"** as root element with its related child elements

"Element"	Type	Description
"AlternativeTitle"	xsd:string	Provides any form of the title used as a substitute or alternative to the formal title of the Configuration Item (CI). Examples are "Link16 spec" and "NICR".
"BaselineVersion"	xsd:string	Provides the edition or version of the TDL standard, the TDL CaT CI. Examples are "Edition 5 Original" and "Edition 6 First Draft".
"Component"	xsd:string	Identifies the component of the TDL specification that this instance document contains. This element explicitly indicates what is implied by the root element to support discovery. Examples are "MessageStructure" and "DataElementDictionary" .
"Identifier"	xsd:string	Provides an unambiguous reference to the CI within the context of TDLs, using an internal, external, and/or universal identification number for a data asset or resource. For example: "STANAG 5516", "ATDLP-7.31".
"Security"	xtdlsec:Security_Type	Contains the security classification and/or administrative markings for the instance document (i.e. the specification).
"Title"	xsd:string	Provides the name given to the CI. Enables the user to find the CI with a particular title or carry out more accurate searches. The title is commonly used as the key point of reference in the list of search results. Examples are "TACTICAL DATA EXCHANGE - LINK 16" and "NATO IMPLEMENTATION CODES AND RULES".
"Version"	xsd:string	Provides the internal version number of the instance document.

Table 7-2: Description of the Elements used within the xTDL schema depicting the element **"BaselineInfo"** as root element with its related child elements

7.2 Logical Model capturing the components required for the CM of TDL CaT CIs

The logical model provided below (Figure 7-2) contains an overview of the TDL CaT CI information objects required for the successful execution of the CM process implemented to manage the TDL specification. The model provides an abstract, implementation independent view, based on the TDL CaT Handbook.

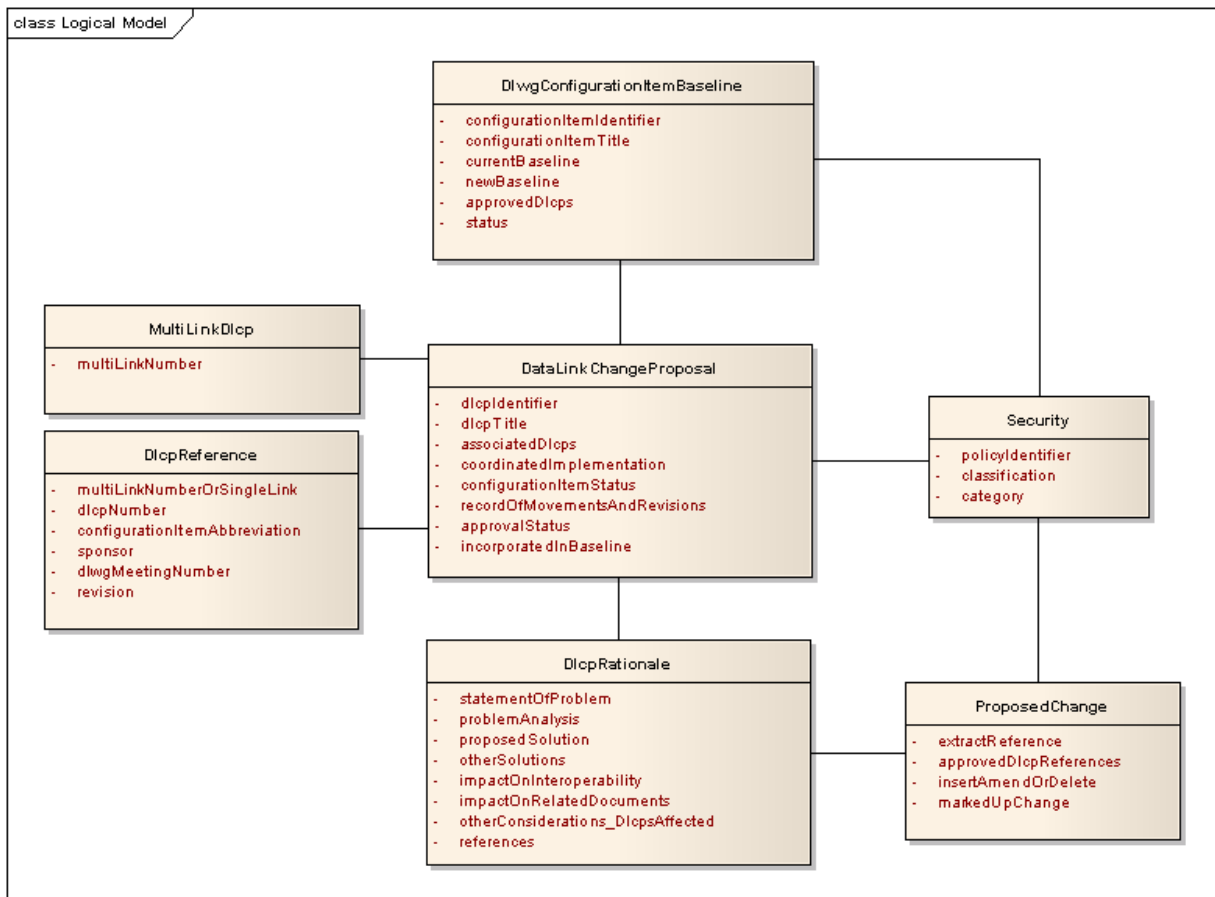


Figure 7-2: Logical Model capturing the components required for the CM of TDL CaT CIs

"Class"	Description
"DlwgConfigurationItemBaseline"	Identifies the TDL CaT CI and its proposed new baseline (version) based upon a set of approved DLCPs agreed for incorporation into the current baseline.
"DataLinkChangeProposal"	Details a set of proposed changes to a CI, together with supporting analysis and rationale.
"DlcpRationale"	Provides information about the problem identified, possible impacts, and references
"DlcpReference"	Identifies a DLCP according to its corresponding CI, sponsor nation or Strategic Command (SC), meeting submitted and latest revision.
"MultiLinkDLCP"	Associates a set of DLCPs that address related changes affecting more than one CI.
"ProposedChange"	Identifies the extracted portion(s) of the CI, together with any relevant previously approved DLCP(s), where the changes, insertions and/or deletions are proposed.
"Security"	Provides information and instructions how the DLCP as a NATO document should be administered.

Table 7-3: Description of the Classes used within the Logical Model
capturing the components required for the CM of TDL CaT CIs

7.3 Physical Model capturing the components required for the CM of TDL CaT CIs

The different components involved in the CM process for TDL CaT CIs are captured in the physical model below (Figure 7-3), depicted as “**Classes**”. The associated Table 7-3 provides a brief description of the different “**Classes**” used within the model.

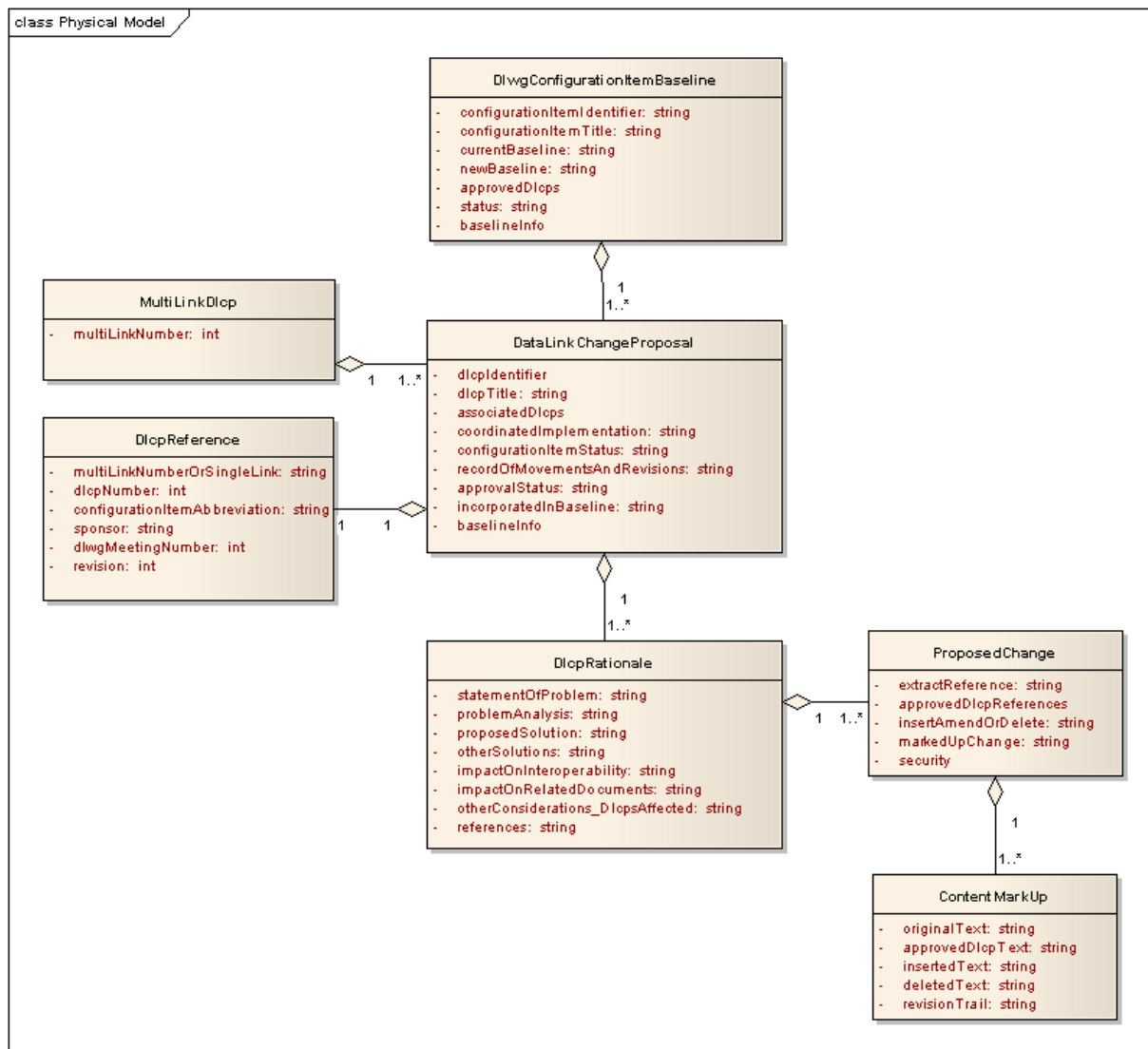


Figure 7-3: Physical Model capturing the components required for the CM of TDL CaT CIs

"Class"	Description
"ContentMarkUp"	Provides the substance of the proposed changes, marked up in accordance with the TDL CaT Handbook conventions.
"DataLinkChangeProposal"	Details a set of proposed changes to a CI, together with supporting analysis and rationale.
"DlcpRationale"	Provides information about the problem identified, possible impacts, and references
"DlcpReference"	Identifies a DLCP according to its corresponding CI, sponsor nation or Strategic Command (SC), meeting submitted and latest revision.
"ProposedChange"	Identifies the extracted portion(s) of the CI, together with any relevant previously approved DLCP(s), where the changes, insertions and/or deletions are proposed.
"DlwgConfigurationItemBaseline"	Identifies the TDL CaT CI and its proposed new baseline (version) based upon a set of approved DLCPs agreed for incorporation into the current baseline.
"MultiLinkDlcp"	Associates a set of DLCPs that address related changes affecting more than one CI.

Table 7-4: Description of the Classes used within the Physical Model capturing the components required for the CM of TDL CaT CIs

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

NATO UNCLASSIFIED

8 REPRESENTATION OF THE TDL DATA ELEMENT DICTIONARY IN XML

This chapter provides a description of the representation of a TDL Data Element Dictionary (DED) in XML. The DED describes the possible data elements for a given TDL standard which are the building blocks of actual data to construct messages. Each data element can be uniquely identified and provides information to translate between its binary representation and other, more meaningful representations.

The XML representation of the TDL DED as described in this chapter is generic in that it can be applied to any bit-based data element format. As with the generic TDL message structure in Chapter 9 this generic TDL DED is applicable and identical for all supported TDL standards and is defined in an XML schema. For each version (edition, baseline, etc.) of a TDL CaT CI document a specific XML instance document is created which defines the DED for that specific version. This XML instance document can be used to support (semi-)automatic platform implementation.

Section 8.1 describes the prerequisites for the usage of a generic TDL DED by providing a non-exhaustive list of requirements to which a generic TDL DED has to comply. Section 8.2 provides the logical model capturing the generic TDL DED components. Section 8.3 provides some background on the way the value of a data element can be represented. Section 8.4 contains the proposed XML schema for the generic TDL DED with examples of the representation in action followed by a description of all relevant types and elements used in the Generic xTDL DED schema. Section 8.5 indicates which relation the Generic xTDL DED schema has with other XML schemas in this document. Finally, in Section 8.6 some considerations for the usage of the Generic xTDL DED schema in its current state are provided, reflecting on “loose ends” and possible action to be taken for improvement.

8.1 Prerequisites for the usage of a generic TDL Data Element Dictionary

This section provides a non-exhaustive list of requirements to which a generic TDL DED has to comply for any bit-based message format. Link 16 has been taken as first focus after which the characteristics of the other TDLs are included in the DED definition. Some background on the way the value of a data element can be represented is provided in Section 8.3.

A number of generic concepts will be used which are introduced in Table 8-1 below. The xTDL name indicates the name used within this document and within the XML schemas.

Concept	"xTDL name"	Definition
data element	"DataElement"	A basic unit of information having a unique meaning and distinct units or values. For example: Identity, Altitude, Callsign. For more information on data elements see Chapter 8.
coding switch	"CodingSwitch"	A construct used to indicate that different coding information applies for a data element, depending on the value of another data field. For example, depending on the value of a 'depth indicator' data field, the actual depth is either reported in multiples of 3, 30 or 300 meter.
enumeration	"Enum"	A mapping from a value or a range of values to a string representation. For example, value '1' for the "Environment/Category" data element maps to 'SPACE'.

Table 8-1: Generic concepts for the Generic TDL DED

The Generic TDL DED shall provide support for the following aspects:

- A **"DataElement"** shall be uniquely identifiable by a key consisting of two numbers: its Data Field Identifier (DFI) and Data Use Identifier (DUI);
- A **"DataElement"** shall have a name, a length (in bits), a bit-coding, and support other meta-data (like a description);
- The bit-coding of a **"DataElement"** shall indicate how the binary value should be translated to an integer value and vice-versa. The default bit-coding is unsigned and possible other bit-codings are twos-complement, ones-complement, sign-plus-magnitude and other that are in use in the TDL COI;
- The field descriptor is defined in the current STANAG - or ATDLP where applicable - and is used in the Word Maps and its length is limited by the size of the **"DataElement"** (number of bits). For backwards compatibility, a **"DataElement"** shall have a **"FieldDescriptor"** which defaults to its name;

- Each “**DataElement**” shall provide information to decode and encode between its integer value and its meaning;
- Each “**DataElement**” shall support a list of one or more “**Enums**” each with an optional explanation;
- For each “**DataElement**” it shall be able to define a formula which can be used to transform the decimal value to another value. This value can be of different types like integer, floating point, boolean, string, etc.;
- A “**Formula**” shall have the possibility to define optional parameters depending on the “**DataElement**”;
- A “**DataElement**” shall support the “**CodingSwitch**” construct to indicate different decoding/encoding information which includes different “**Enums**”, types and formulas;
- A “**CodingSwitch**” shall refer to the controlling “**DataField**”;
- All possible values for the controlling “**DataField**” in a “**CodingSwitch**” shall be specified using a catch-all case as appropriate;
- The controlling “**DataField**” shall be able to be part of a well-defined context within a TDL specification;
- The controlling “**DataField**” shall be able to be the same “**DataElement**” as it is contained in (see e.g. Link 16 DFI/DUI 417/016);
- A “**CodingSwitch**” shall support nesting.

8.2 Logical Model capturing the components of the generic TDL Data Element Dictionary

The following logical model shows the components that directly support the generic TDL DED. The attributes shown in the classes denote relevant information that needs to be captured on the classes or indicate a relation between classes. Figure 8-1 below shows the high-level components needed to support the generic TDL DED, e.g. to support the “**DataElements**” for Link 16 J-series messages.

From this logical model, the actual XML schema is derived by including all components (element and attributes) that are required to fully model the generic TDL DED.

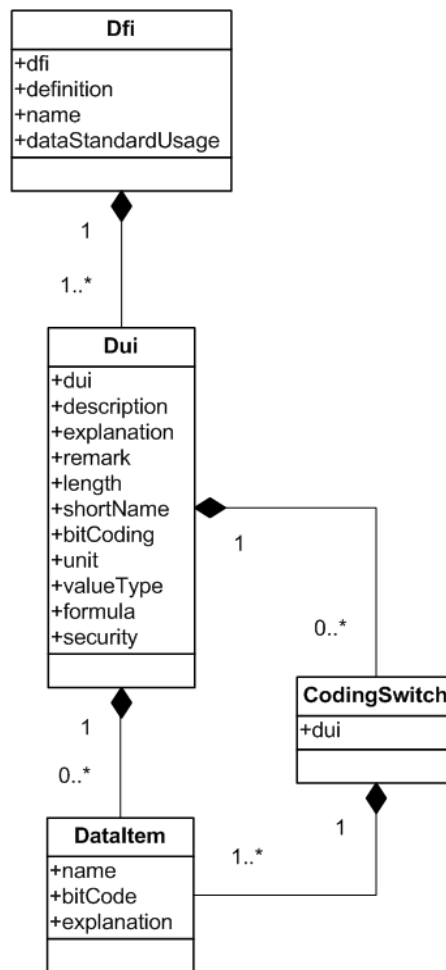


Figure 8-1: Logical Model capturing the components of a generic xTDL Data Element Dictionary

"Elements"	Description
"Dfi"	Describes a single concept and is the generic representation of the DUIs grouped under it.
"Dui"	Is representative of the corresponding DFI concept. It denotes the actual Data Element and contains the Data Items used to compose the Data Element. The combination of a DFI and a DUI uniquely define a Data Element definition.
"DataItem"	Provides the description and/or decoded value of an enumeration.
"CodingSwitch"	Indicates that different coding information applies for a data element, depending on the value of another data field.

Table 8-2: Description of Elements used within the Logical Model capturing the components of a generic xTDL Data Element Dictionary

8.3 Specification of Data Element values

This section provides some background on the way the value of a data element can be specified and how it applies to the TDL standards and the XML representation thereof.

Each data element in a binary TDL standard has a number of characteristics that are used to determine its value, e.g. its length (number of bits), bit-coding (unsigned, two's-complement, etc.), value type (integer, double/real, boolean, etc.). Each value is referenced in the standard as a data Item. This data item can be represented in a number of ways each of which has its specific usage. The possible representations are explained below (assuming basic computer science knowledge) with references to the examples in Table 8-2 below.

- (1) The binary value: the actual 0's and 1's as transmitted on the TDL in their proper endianness (each bit represented using the '0' and '1' ASCII character). E.g. '00110' (example [1]) or '10000' (example [2]) for a data element of 5 bits.
- (2) The unsigned integer value by converting the binary value straight to its integer value. E.g. respectively '6' (example [3]) and '16' (example [4]) for the two binary values above assuming most significant bit (MSB) first. ATDLP-5.11 and ATDLP-5.16 use this notation to represent the bit code of each data item of a data element.
- (3) The actual integer value by converting the binary value to its integer value respecting the bit-coding associated with the data element. The bit-coding indicates whether negative numbers can be represented

as a binary value and, if so, how that is done. Examples of bit-codings are unsigned, ones'-complement, two's-complement, or sign + magnitude (the previous representation is in fact the unsigned bit-coding). E.g. '6' and '-16' for the two binary values above assuming two's-complement, or '6' and '-15' for ones' complement. This value is the one that is normally used within a system for numerical data elements.

- (4) The meaning as defined in the TDL standard. E.g. if the data element associated with the above two binary values is for holding the Weapon system, the meaning could be 'ANTISUBMARINE WARFARE MISSILE' (example [5]) and 'DECEPTION JAMMER'. If, on the other hand, the data element is for holding a Sensor channel set, the meaning could be 'CHANNEL SET 1 THROUGH 31' for both (example [6]). If the data element represents a numerical value, like an altitude or a latitude, normally not all values are spelled out and instead the range is included, e.g. for a latitude it provides as meaning for value 745016 (or any other number in the range 524289 through 1048575) the text "-90 THROUGH -90/524,287 DEGREES SOUTH" (example [7]).
- (5) The actual meaning by interpreting the definition in the TDL standard. E.g. for the data element above for holding the Sensor channel set, the meaning could be '6' and '16' (example [8]) as channel set (interpretation of the use is required whether the result should be a number or a string). If the data element represents a numerical value, like an altitude or a latitude, the binary value 10110101111000111000 (20 bits) represents the two's complement decimal number -303560 which represents the real number -52.10962697911 (example [9]) as latitude (giving the latitude definition of the previous representation). This value is the one that is normally used within a system when performing real mathematical computations.
- (6) The actual meaning as in the previous representation but then presented in a human-readable format. E.g. the actual meaning of a latitude data element can be -52.10962697911 but represented in a format that humans prefer might show 52° 06' 34.39' S (example [10]). Obviously there are multiple choices here like the type of coordinate representation and the accuracy.

Data Element	Binary	Un-signed integer	Actual integer	Meaning	Actual meaning	Human-readable form of actual meaning
WEAPON SYSTEM (5-bit, unsigned)	[1] 00110	[3] 6	6	[5] ANTISUBMARINE WARFARE MISSILE	ANTISUBMARINE WARFARE MISSILE	ANTISUBMARINE WARFARE MISSILE
SENSOR CHANNEL SET (5-bit, unsigned)	[2] 10000	[4] 16	16	[6] CHANNEL SET 1 THROUGH 31	[8] 16	CHANNEL SET 16
LATITUDE (20-bit, two's complement)	10110101111000111000	745016	-303560	[7] -90 THROUGH -90/524,287 DEGREES SOUTH	[9] -52.10962697911	[10] 52° 06" 34.39' S

Table 8-3: Examples of Data Element value representation

All these various representations have their use in a system or the standard's documentation and for each there should be support in the XML representation of the data element. Of course, defaults can be used for representations that do not pose special cases, e.g. the Data Items of the data element 'Weapon System' do not need further definitions to define their actual meanings.

8.4 The Generic xTDL Data Element Dictionary schema

Based on the prerequisites for the usage of a generic xTDL DED described in Section 8.1 and on the logical model described in Section 8.2, an XML schema has been developed. This XML schema for a generic TDL DED is shown and explained in the following sections. In each section, a high-level, generic description is provided after which a diagram is provided explaining that particular part of the schema. Included also is an example of a partial instance document demonstrating the XML schema in action. A description of the major types and elements used within the Generic xTDL DED schema is provided in Section 8.4.

8.4.1 The "DataElements" element

Figure 8-2 shows the element "**DataElements**" as root element of the Generic xTDL DED schema, with its child elements "**BaselineInfo**" and "**DataElementDictionary**". The "**BaselineInfo**" element is the same container as for the TDL message structure described in Chapter 9, containing the metadata required to describe the TDL standard specification itself (see Chapter 7, Section 7.1).

The "**DataElementDictionary**" element contains a list of "**Dfi**" elements, each of which has one or more "**Dui**" elements.

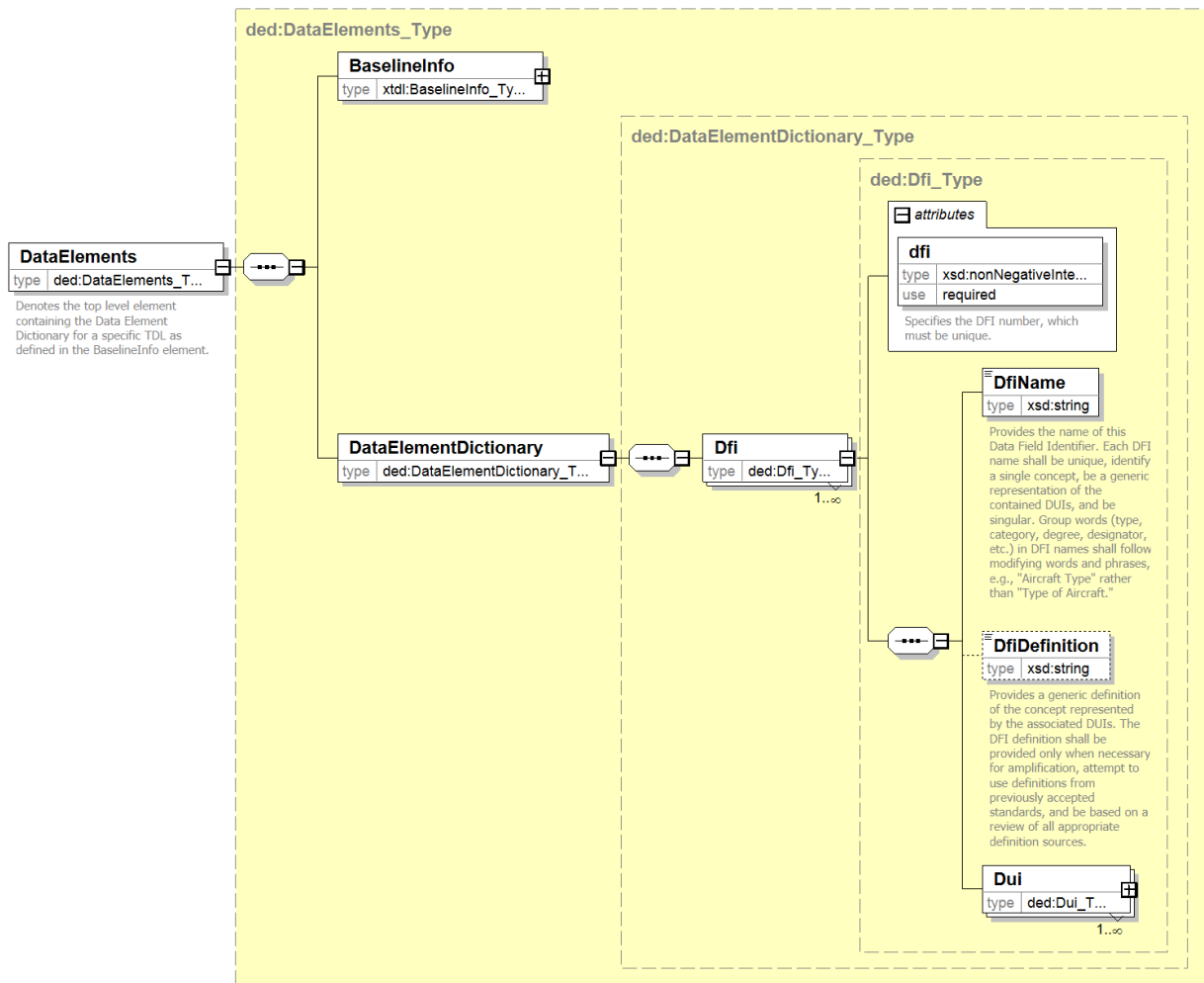


Figure 8-2: xTDL schema depicting the element “DataElements” as root element with its related child elements as part of the generic TDL Data Element Dictionary

An example of an XML instance document of the DED for STANAG 5516 corresponding to the XML schema above is shown below:

```
<ded:DataElements
  xmlns:ded="urn:nato:tdl:generic:dataElementDictionary:1:20140301:draft"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:nato:tdl:generic:dataElementDictionary:1:20140301:draft xTDL-
    DataElementDictionary.xsd">

  <BaselineInfo>
    <Title>TACTICAL DATA EXCHANGE - LINK 16</Title>
    <Identifier>STANAG 5516</Identifier>
    <BaselineVersion>edition 3</BaselineVersion>
    <Version>2009-02</Version>
    <Component>DataElementDictionary</Component>
    <!-- Actual structure of the classification markings is still under discussion. Namespace
    should also be updated. Placeholder. -->
    <Security>
      <PolicyIdentifier>NATO</PolicyIdentifier>
      <Classification>UNCLASSIFIED</Classification>
      <Category type="permissive">RELEASABLE FOR INTERNET TRANSMISSION</Category>
    </Security>
  </BaselineInfo>

  <DataElementDictionary>
    <Dfi dfi="281">
      <DfiName>LATITUDE</DfiName>
      <DfiDefinition>THE ANGULAR DISTANCE NORTH OR SOUTH FROM THE EQUATOR TO A POINT ON THE
      EARTH'S SURFACE, MEASURED IN DEGREES, FROM 0 DEGREES AT THE EQUATOR UP TO, BUT NOT EXCEEDING,
      THE 90 DEGREE ANGLES NORTH AND SOUTH BETWEEN THE EQUATOR AND THE POLES.</DfiDefinition>
      <Dui dui="014" type="data">
        <!-- See next section -->
      </Dui>
    </Dfi>
  </DataElementDictionary>
```

8.4.2 The “Dui” element

Figure 8-3 shows the “Dui” element as root element with all its child elements. The elements describe characteristics of this “DataElement”, information on how to decode the “DataElement” and its enumerations. A “Formula” can be specified for the decoding of the value to a meaningful value or human-readable format. The “Formula” is defined by its name and optional Parameters. The actual algorithm for the “Formula” is currently not part of the XML schema. The actual decoding information and the enumeration can depend on another “DataElement” which is captured in a “CodingSwitch” which is explained in the next section.

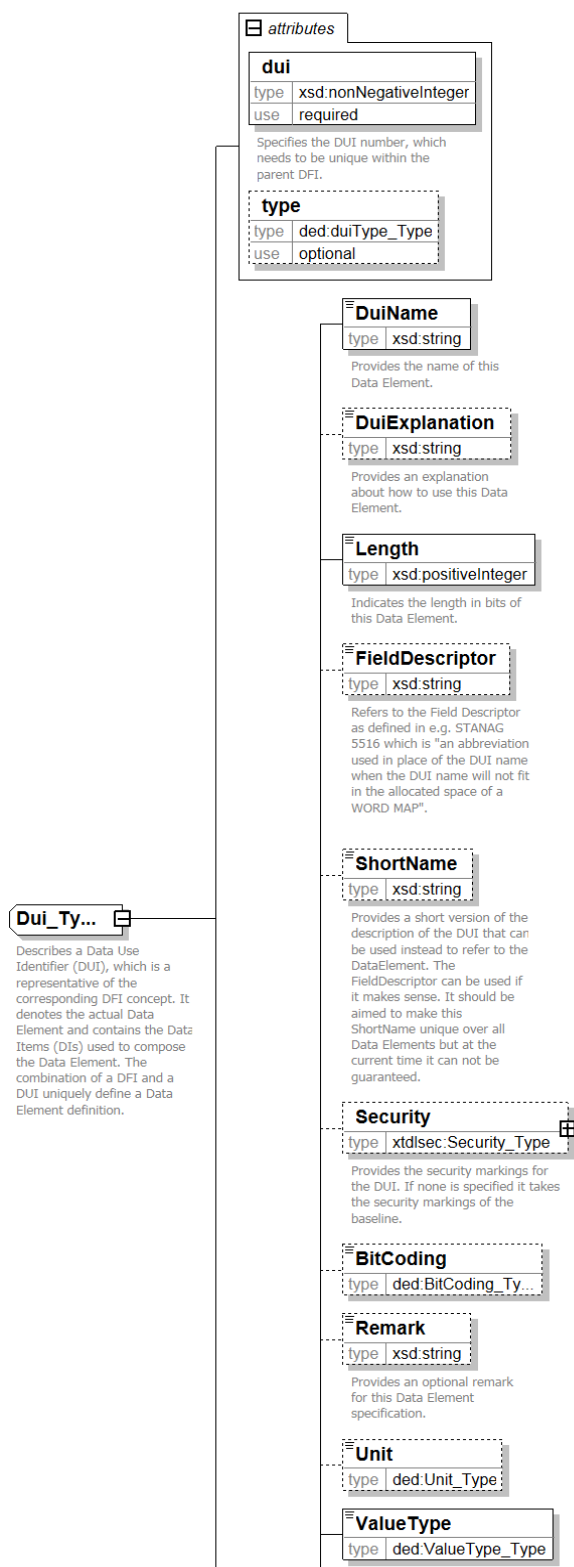


Figure 8-3a: xTDL schema depicting the element “Dui” with its related child elements as part of the generic TDL Data Element Dictionary (continued on next page)

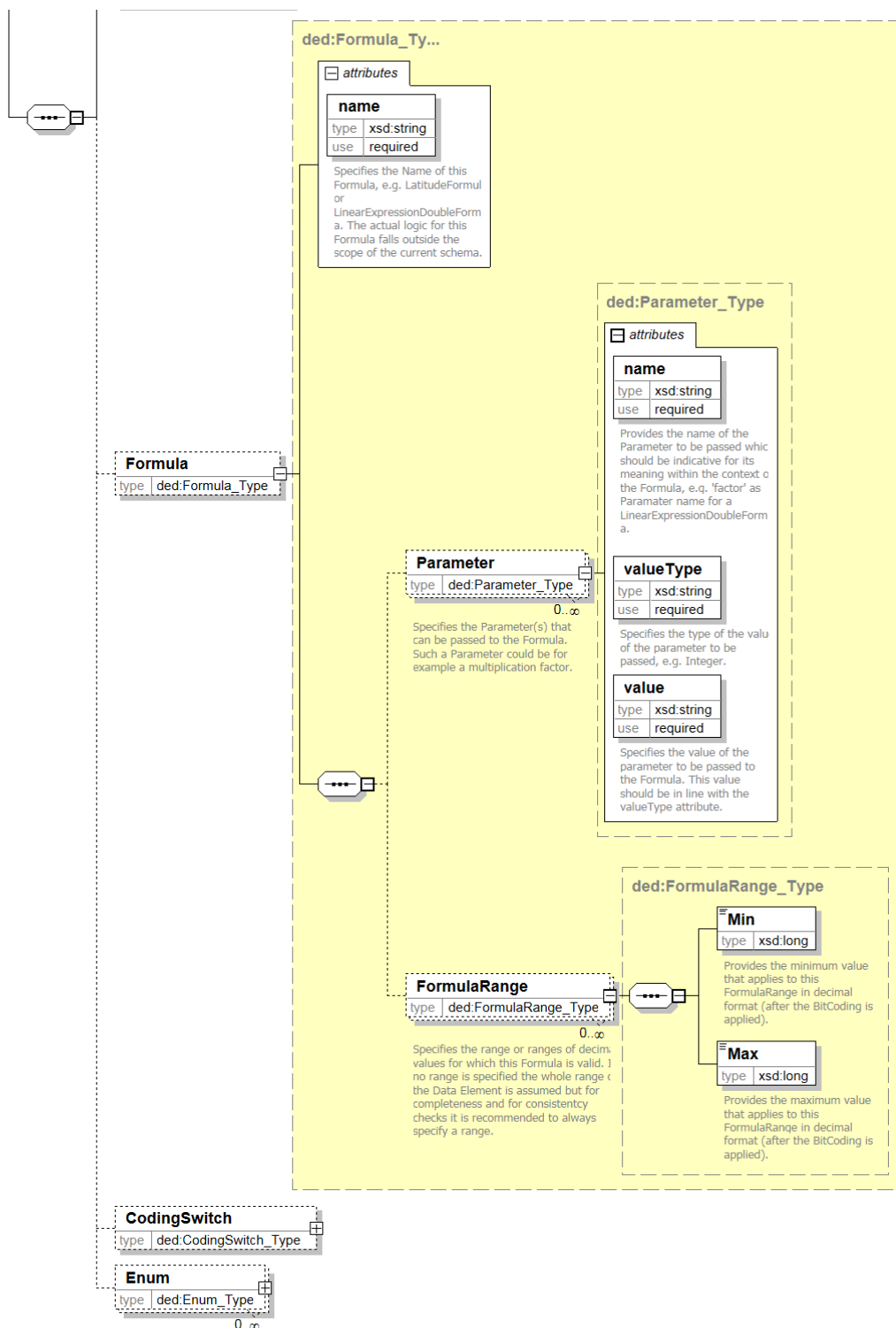


Figure 8-3b: xTDL schema depicting the element “Dui” with its related child elements as part of the generic TDL Data Element Dictionary (continued from previous page)

An example of a DUI for ATDLP-5.16 is shown below:

```
<Dui dui="014" type="data">
  <DuiName>LATITUDE, 0.0051 MINUTE</DuiName>
  <DuiExplanation>THE PRECISION IS APPROXIMATELY 31 FEET.</DuiExplanation>
  <Length>21</Length>
  <BitCoding>twosComplement</BitCoding>
  <Unit>DEGREE</Unit>
  <ValueType>Double</ValueType>
  <Formula name="LatitudeFormula">
    <!-- See next section -->
  </Formula>
  <Enum>
    <!-- See next section -->
  </Enum>
</Dui>
```

8.4.3 The “CodingSwitch” element

Figure 8-4 shows the “**CodingSwitch**” element which is used to capture the fact that the decoding and the enumeration of one “**DataElement**” depends on the current value of another “**DataElement**”. An example is a scale indicator which specifies whether the altitude is reported in 100 feet or 500 feet increments.

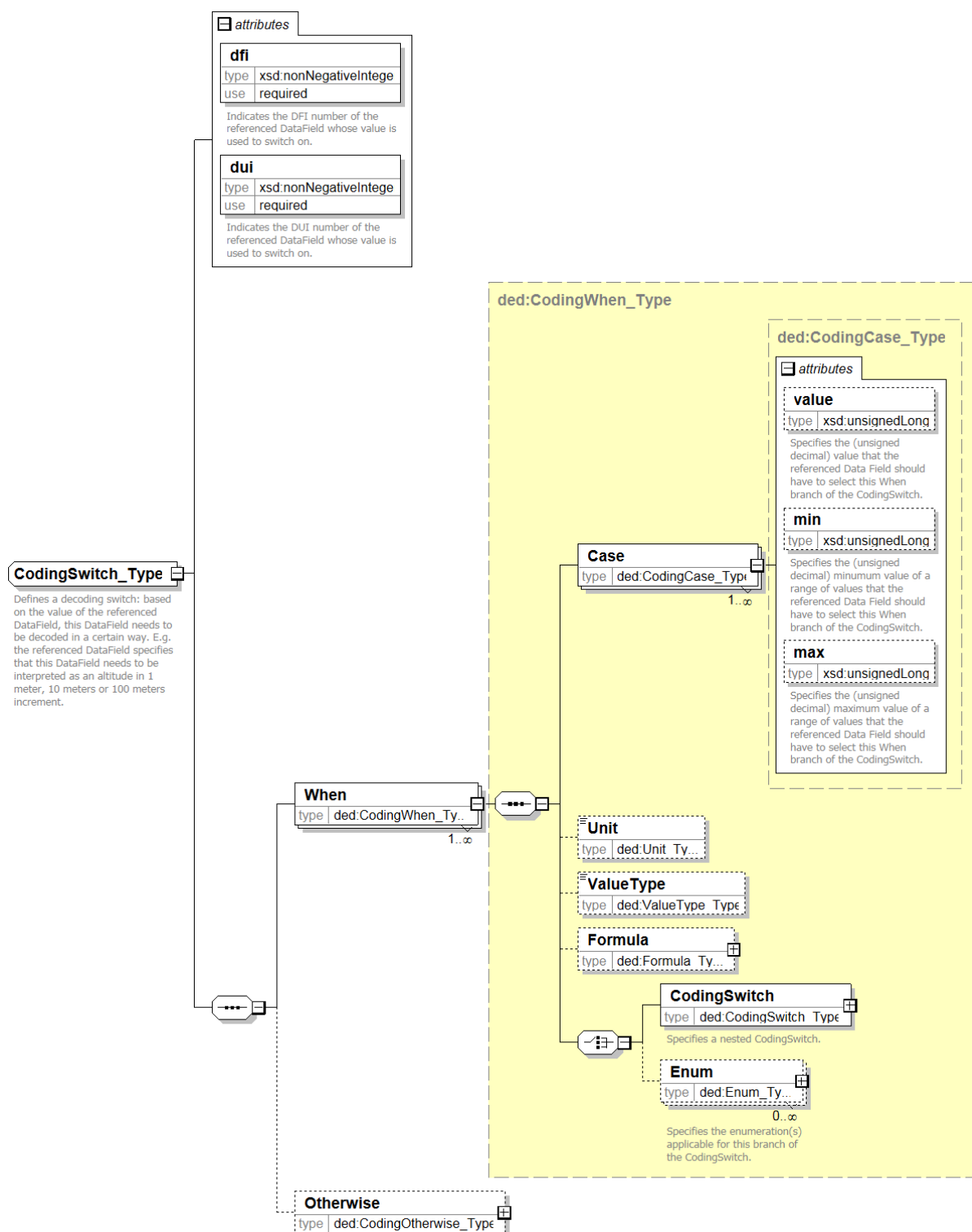


Figure 8-4: xTDL schema depicting the element “**CodingSwitch**” with its related child elements as part of the generic TDL Data Element Dictionary

An example of an XML instance for a “**CodingSwitch**” for DFI/DUI 366/013 for ATDLP-5.16 corresponding to the schema above is shown below. Basically, it describes that the actual depth in meter is decoded by multiplying the value in the “**DataField**” by either 3, 30, or 300 depending on the value of the depth indicator (DFI/DUI 366/012). It also shows the use of a “**Formula**” with a range where the formula is valid. Outside that range the formula is not valid and the information on the value is provided by the “**Enum**” elements (e.g. 0 = NO STATEMENT, or 12 = UNDEFINED).

```
<Dui dui="013">
  <!-- skipped -->
  <ValueType>Enumeration</ValueType>
  <CodingSwitch dfi="366" dui="012">
    <When>
      <Case value="1" />
      <Unit>METER</Unit>
      <ValueType>Integer</ValueType>
      <Formula name="LinearExpressionIntegerFormula">
        <Parameter name="factor" value="3" />
        <FormulaRange>
          <Min>1</Min>
          <Max>9</Max>
        </FormulaRange>
      </Formula>
      <Enum>
        <DataItem type="NO STATEMENT">NO STATEMENT</DataItem>
        <BitCode>0</BitCode>
        <Explanation />
      </Enum>
      <Enum>
        <DataItem>DEPTH (METERS X DEPTH INDICATOR)</DataItem>
        <BitCodeRange>
          <Min>1</Min>
          <Max>9</Max>
        </BitCodeRange>
        <Explanation />
      </Enum>
      <Enum>
        <DataItem type="UNDEFINED">UNDEFINED</DataItem>
        <BitCodeRange>
          <Min>10</Min>
          <Max>15</Max>
        </BitCodeRange>
        <Explanation />
      </Enum>
    </When>
    <When>
      <Case value="2" />
      <Unit>METER</Unit>
      <ValueType>Integer</ValueType>
      <Formula name="LinearExpressionIntegerFormula">
        <Parameter name="factor" value="30" />
        <FormulaRange>
          <Min>1</Min>
          <Max>9</Max>
        </FormulaRange>
      </Formula>
      <Enum>
        <DataItem>NO STATEMENT</DataItem>
        <BitCode>0</BitCode>
        <Explanation />
      </Enum>
      <Enum>
        <DataItem>DEPTH (METERS X DEPTH INDICATOR)</DataItem>
        <BitCodeRange>
          <Min>1</Min>
```

```

        <Max>9</Max>
      </BitCodeRange>
      <Explanation />
    </Enum>
    <Enum>
      <DataItem>UNDEFINED</DataItem>
      <BitCodeRange>
        <Min>10</Min>
        <Max>15</Max>
      </BitCodeRange>
      <Explanation />
    </Enum>
  </When>
</When>
<Case value="3" />
<Unit>METER</Unit>
<ValueType>Integer</ValueType>
<Formula name="LinearExpressionIntegerFormula">
  <Parameter name="factor" value="300" />
  <FormulaRange>
    <Min>1</Min>
    <Max>9</Max>
  </FormulaRange>
</Formula>
<Enum>
  <DataItem>NO STATEMENT</DataItem>
  <BitCode>0</BitCode>
  <Explanation />
</Enum>
<Enum>
  <DataItem>DEPTH (METERS X DEPTH INDICATOR)</DataItem>
  <BitCodeRange>
    <Min>1</Min>
    <Max>9</Max>
  </BitCodeRange>
  <Explanation />
</Enum>
<Enum>
  <DataItem>UNDEFINED</DataItem>
  <BitCodeRange>
    <Min>10</Min>
    <Max>15</Max>
  </BitCodeRange>
  <Explanation />
</Enum>
</When>
</CodingSwitch>
</Dui>

```

8.4.4 The “Enum” element

Figure 8-5 shows the “**Enum**” element which holds a specific “**DataItem**”, i.e. a value of the “**DataElement**”. The enumeration can apply to one specific value (“**BitCode**”) or to a range of values (“**BitCodeRange**”). The “**DataItem**” holds the meaning which, in case of data elements that specify a numerical quantity (e.g. a latitude), might be a description of the range (e.g. “0 through 90 degrees north”).

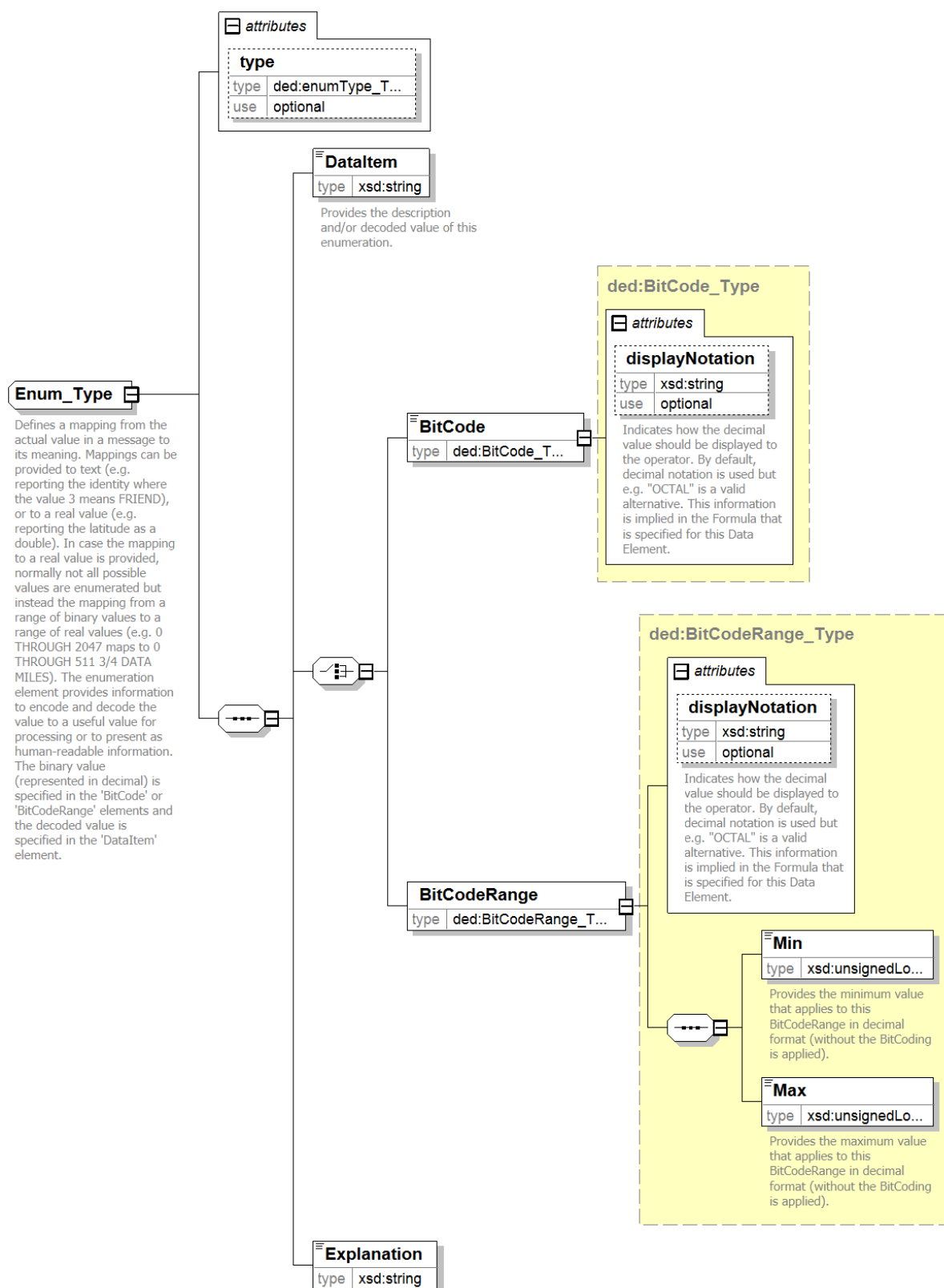


Figure 8-5: xTDL schema depicting the element “**Enum**” as root element with its related child elements as part of the generic TDL Data Element Dictionary

An example of an “Enum” for DFI/DUI 366/013 for STANAG 5516 Edition 3 is shown in the previous section.

8.5 Description of Types and Elements used within the Generic xTDL Data Element Dictionary schema

The following tables provide a description of types and elements used in the text and diagrams in the sections above. These descriptions are automatically extracted from the XML schema (contained in the xsd:annotation). Only the main types and elements are included, for all others the reader is referred to the actual schema (see Annex H). Furthermore, the description of an element and its type often overlap as they both describe the same thing. In fact, the element should describe the particular usage of an element of that type in its context while the type should describe it in more generic terms.

“Type”	Description
“BitCode_Type”	Specifies the value that defines this enumeration in its decimal unsigned representation. This means that the binary value as exchanged on the link is converted to the corresponding unsigned integer value, i.e. the Bit Coding associated with this Data Element is not applied.
“BitCodeRange_Type”	Specifies the range of values that define this enumeration with the values in their decimal unsigned representation. This means that the binary value as exchanged on the link is converted to the corresponding unsigned integer value, i.e. the “BitCoding” associated with this Data Element is not applied.
“BitCoding_Type”	Provides a specification on the way the value for this Data Element is encoded in binary in case this Data Element represents a numerical value (e.g. two's complement, ones' complement). The default value is 'unsigned'.
“CodingCase_Type”	Defines for which value a specific coding applies. This is either indicated with a single value or a range of values.
“CodingOtherwise_Type”	Encapsulates a specific coding for the Data Element which is chosen if none of the When branches is selected.
“CodingSwitch_Type”	Defines a decoding switch: based on the value of the referenced Data Field, this Data Field needs to be decoded in a certain way. E.g. the referenced Data Field specifies that this Data Field needs to be interpreted as an altitude in 1 meter, 10 meters or 100 meters increment.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

"Type"	Description
"CodingWhen_Type"	Encapsulates a specific coding for the Data Element. The enclosed "Case" element(s) indicate for which value(s) of the referenced Data Field this coding should be chosen.
"DataElementDictionary_Type"	A dictionary of all Data Elements used in the Messages. Each Data Element is identified by a DFI and a DUI and contains metadata like length and type, and information for decoding the bit-value. All Data Elements should be referenced from a Word.
"DataElements_Type"	Contains all information on the Data Elements for a specific TDL. The DED describes all Data Elements keyed via a DFI and a DUI. The DED describes the metadata for each Data Element and Coding information to map the bit code value to its meaning (either a text or a value like a number).
"Dfi_Type"	Describes a DFI which includes a single concept and is the generic representation of the DUIs grouped under it.
"Dui_Type"	Describes a DUI, which is representative of the corresponding DFI concept. It denotes the actual Data Element and contains the Data Items used to compose the Data Element. The combination of a DFI and a DUI uniquely define a Data Element definition.
"duiType_Type"	Provides metadata on the nature of the Data Element: whether it's spare, disused, to identify the label or structure of a message or word, or actual data. Further values might be defined.
"Enum_Type"	Defines a mapping from the actual value in a message to its meaning. Mappings can be provided to text (e.g. reporting the identity where the value 3 means FRIEND), or to a real value (e.g. reporting the latitude as a double). In case the mapping to a real value is provided, normally not all possible values are enumerated but instead the mapping from a range of binary values to a range of real values (e.g. 0 THROUGH 2047 maps to 0 THROUGH 511 3/4 DATA MILES). The "Enum" element provides information to encode and decode the value to a useful value for processing or to present as human-readable information. The binary value (represented in decimal) is specified in the "BitCode" or "BitCodeRange" elements and the decoded value is specified in the "DataItem" element.
"enumType_Type"	Provides metadata to specify the type of a specific Data Item, e.g. "NO STATEMENT" or "UNDEFINED". See also ATDLP-5.16, Annex B.4.1.4.4 Generic Data Items Entries.

"Type"	Description
"Formula_Type"	Specifies the " Formula " needed to decode the decimal value of a Data Element to a meaningful value or human-readable format. The " Formula " is defined by its name and optional Parameters. The actual algorithm for the formula is currently not part of the schema. The name for name should be descriptive to indicate its logic (e.g. " OctalFormula " for converting a decimal value to its octal string representation, " LinearExpressionDoubleFormula " for multiplying the decimal value with a given factor as specified as Parameter). If the coding for a Data Element utilizes a " CodingSwitch " (i.e. the coding depends on another " DataElement "), the " Formula " can also be different for different coding variants. In that case the " Formula " should be specified within the " CodingSwitch ".
"FormulaRange_Type"	Specifies a range of decimal values for which a " Formula " is valid. The decimal values are provided with the " BitCoding " of the corresponding Data Element (DUI) applied, e.g. the range can be from -128 to +127.
"Parameter_Type"	Specifies a parameter that can be passed to a " Formula ".
"Unit_Type"	Specifies the measurement unit for this Data Element, e.g. METERS, DEGREES, FEET. The possible units are specific for a TDL although preferably units should be used that are defined in standards. If no unit is specified, the value is without unit which is true for all pure enumerations. If the coding for this Data Element utilizes a " CodingSwitch " (i.e. the coding depends on another Data Element), the unit can be different for different coding variants. In that case the Unit should be specified within the " CodingSwitch ".
"ValueType_Type"	Specifies the specific kind of value that is represented, e.g. Double, Integer or Enumeration. The current list of types can be extended if required. If the coding for this Data Element utilizes a " CodingSwitch " (i.e. the coding depends on another Data Element), the value type can also be different for different coding variants. In that case the " ValueType " should be specified within the " CodingSwitch ".

Table 8-4: Description of Types used within the Generic xTDL Data Element Dictionary schema

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

"Element"	Type	Description
"DataElements"	ded:DataElements_Type	Denotes the top level element containing the DED for a specific TDL as defined in the "BaselineInfo" element.
"DataItem"	xsd:string	Provides the description and/or decoded value of this enumeration.
"DuiExplanation"	xsd:string	Provides an explanation about how to use this Data Element.
"DuiName"	xsd:string	Provides the name of this Data Element.
"Explanation"	xsd:string	Provides an additional explanation for this Data Item only when necessary for amplification, it shall attempt to use explanations from previously accepted standards, it shall be based upon a review of all appropriate sources, and it shall not be a restatement of the name unless it is spelling out an acronym. When a Data Item value is "NUMERIC", the "Explanation" must contain a description of what the DI represents. In expressing numerical quantities, the "Explanation" must indicate how to derive the actual value from the "BitCode" (e.g. an "Explanation" of "DISTANCE IN 1/4 DATA MILE INCREMENTS" for a "BitCodeRange" of Min=0 and Max=2047 and a Data Item of "0 THROUGH 511 3/4 DATA MILES"). This information shall also be captured in the "Formula" of the Data Element.
"FieldDescriptor"	xsd:string	Refers to the Field Descriptor as defined in e.g. ATDLP-5.16 which is "an abbreviation used in place of the DUI name when the DUI name will not fit in the allocated space of a WORD MAP".
"FormulaRange"	ded:FormulaRange_Type	Specifies the range or ranges of decimal values for which this "Formula" is valid. If no range is specified the whole range of the Data Element is assumed but for completeness and for consistency checks it is recommended to always specify a range.
"Length"	xsd:positiveInteger	Indicates the length in bits of this Data Element.
"Max"	xsd:long	Provides the maximum value that applies to this "FormulaRange" in decimal format (after the "BitCoding" is applied).

"Element"	Type	Description
"Max"	xsd:unsignedLong	Provides the maximum value that applies to this "BitCodeRange" in decimal format (without the "BitCoding" is applied).
"Min"	xsd:long	Provides the minimum value that applies to this "FormulaRange" in decimal format (after the "BitCoding" is applied).
"Min"	xsd:unsignedLong	Provides the minimum value that applies to this "BitCodeRange" in decimal format (without the "BitCoding" is applied).
"Parameter"	ded:Parameter_Type	Specifies the "Parameter(s)" that can be passed to the "Formula" . Such a Parameter could be for example a multiplication factor.
"Remark"	xsd:string	Provides an optional remark for this Data Element specification.
"Security"	xtdlsec:Security_Type	Provides the security classification and/or administrative markings for the DUI. If none is specified it takes the markings of the baseline.
"ShortName"	xsd:string	Provides a short version of the description of the DUI that can be used instead to refer to the "DataElement" . The "FieldDescriptor" can be used if it makes sense. It should be aimed to make this "ShortName" unique over all Data Elements but at the current time it cannot be guaranteed.

Table 8-5: Description of Elements used within the Generic xTDL Data Element Dictionary schema

8.6 Relationship of the Generic xTDL DED schema with other XML schemas

The Generic xTDL DED schema, together with the instance document for a DED, is directly related to the XML schemas of XML message instance documents as described in Chapter 14, where the latter can be automatically derived from the XML instance documents for the TDL message structure and DED.

8.7 Considerations for the usage of the Generic xTDL Data Element Dictionary schema

The described XML schema for the generic TDL DED still has some “loose ends” that are not captured yet and need to be addressed in the future:

- The logic behind a “**Formula**” is not represented in XML and is therefore still open for interpretation by developers etc. Alternatives are defining standard “**Formulas**” (stored in a catalogue) which can be referenced from the data elements. The standard “**Formula**” can use XML elements to describe e.g. simple mathematical operations (e.g. multiplication with a certain factor). More complex operations (e.g. for positional information like latitude and longitude) will require more work or maybe even external references.
- The “**Unit**” of a “**DataElement**” (DUI) is defined as a simple string (e.g. “METER”, “SECOND”, “DATAMILE”) without any restriction or coupling to external standards. Whenever there is a standard defining such unit there should be a way to link to that.

9 REPRESENTATION OF THE TDL MESSAGE STRUCTURE IN XML

This chapter describes the XML characterization of the structure of the messages used within TDL standards. The TDL message structure prescribes what kind of data can be put in a message and the structure to which it should adhere.

This message structure is generic, as it is assessed to be applicable and identical for multiple TDL standards. This generic TDL message structure is defined in an XML schema.

For each version (edition, baseline, etc.) of a TDL CaT CI document a specific XML instance document will be created which defines the TDL message structure for that specific version. This XML instance document will be used to generate the relevant parts of the TDL standard and can be used to support (semi-)automatic platform implementation.

Section 9.1 explains the prerequisites for the usage of a generic TDL message structure, what kind of constructs and conditions the generic TDL message structure has to support. Section 9.2 provides the logical model capturing the generic TDL message structure components, which links to the logical model capturing the TDL DED components described in Chapter 8, Section 8.2. After that, in Section 9.3, the XML schema for the generic TDL message structure is provided with examples to illustrate the usage of the various constructs, followed by the description of all relevant types and elements. Section 9.4 indicates which relation the Generic xTDL Message Structure schema has with other XML schemas in this document. Finally, in Section 9.5 some considerations for the usage of the Generic xTDL Message Structure schema in its current state are provided, reflecting on “loose ends” and possible actions for improvement.

9.1 Prerequisites for the usage of a generic TDL message structure

This section provides a non-exhaustive list of requirements to which a generic TDL message structure has to comply for any bit-based TDL message format. Link 16 has been taken as first focus after which the characteristics of the other TDL standards are included in the structure definition.

A number of generic concepts will be used which have already been introduced in Chapter 8, Table 8-1, which are supplemented with those concepts in Table 9-1 described below. The xTDL name indicates the name used within this document and within the XML schemas.

Concept	“xTDL name”	Definition
data field	“ DataField ”	The instantiation or use of a data element.
word	“ Word ”	<p>A structured collection of one or more data fields used to report on a specific aspect.</p> <p>For example, the J3.1I word reports on the basic information for an emergency point, while J3.1E0 reports the position and J3.1C1 provides the IFF/SIF codes.</p>
structure switch	“ StructureSwitch ”	<p>A construct used to specify overlaid sets of data fields where the value of another, referenced data field defines which set is present in a word.</p> <p>For example, if the environment/category indicates AIR then the word contains the Air Platform and the Air Platform Activity data fields.</p>
message	“ Message ”	<p>A structured collection of one or more words to report a particular set of information.</p> <p>For example, the J3.2 message for reporting (the state of) an air track can contain the J3.2I, J3.2E0, and J3.2C1 words.</p>

Table 9-1: Generic concepts for the generic TDL message structure

The generic TDL message structure shall provide support for the following aspects:

- A “**Message**” shall have a unique name and support other metadata (e.g. a title and a purpose);
- A “**Message**” shall be able to hold one or more “**Words**”. Depending on the TDL standard under development, this restriction can be made stricter e.g. to allow only one “**Word**” per “**Message**” (e.g. for Link 1);
- A “**Word**” shall have a unique name and support other metadata (e.g. a title);
- A “**Word**” shall consist of “**DataFields**” which will hold the actual data;
- A “**DataField**” shall uniquely refer to a “**DataElement**”. In other words, a “**DataField**” is an instance of a “**DataElement**”. See Chapter 8 for more details on the DED;
- A “**DataField**” shall have a start position in the “**Word**”. Together with the length of the “**DataElement**” it defines the start and end position of the “**DataField**”;

- A “**DataField**” shall be able to have an optional fixed value which must be one of the legal values for the “**DataElement**”. This is used to specify the value for the message label “**DataField**” of a “**Message**”;
- The structure definition of a “**Word**” shall support the “**StructureSwitch**” construct to allow for the definition of overlaid sets of “**DataFields**”;
- A “**StructureSwitch**” shall refer to the controlling “**DataField**”;
- All possible values for the controlling “**DataField**” in a “**StructureSwitch**” shall be specified using a catch-all case as appropriate;
- The controlling “**DataField**” shall be able to be part of a well-defined context within a TDL specification, e.g. the same “**Word**” or the same “**Message**”;
- A “**StructureSwitch**” shall support nesting;
- “**DataFields**” shall be unique with respect to their DFI/DUI in a “**Word**” with the exception of SPARE and DISUSED.

9.2 Logical Model capturing the components of a generic TDL message structure

This logical model shows the components that directly support the generic TDL message structure. The attributes shown in the classes denote relevant information that needs to be captured on the classes or indicate a relation between classes (e.g. “**DuiRef**”). Figure 9-1 below shows the high-level components needed to support the generic message structure to specify TDL messages.

From this logical model, the actual XML schema is derived by including all components (elements and attributes) that are required to fully model the message structure.

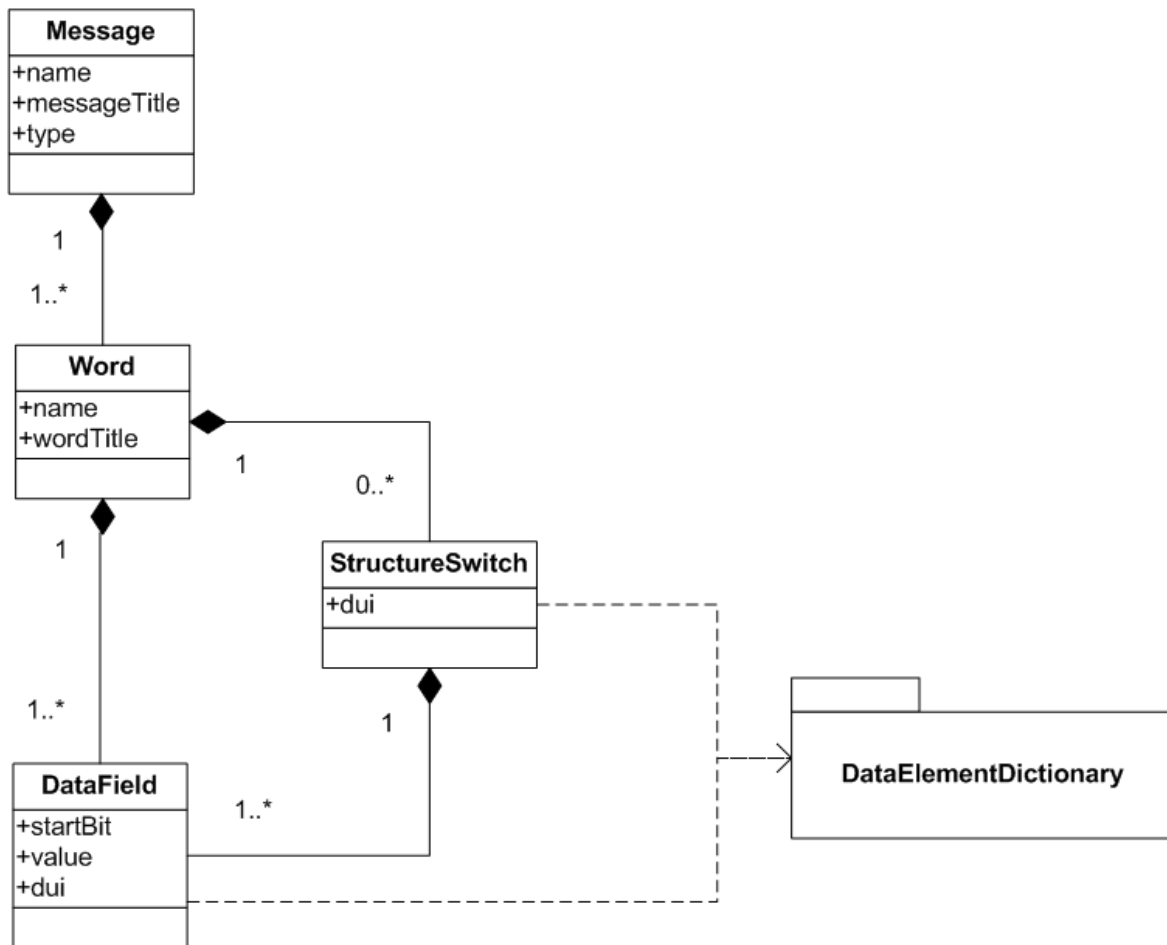


Figure 9-1: Logical Model capturing the components of a generic TDL message structure

"Elements"	Description
"Message"	Is a structured collection of one or more words to report a particular set of information.
"Word"	Is a structured collection of one or more data fields used to report on a specific aspect.
"DataField"	Is the instantiation or use of a data element.
"StructureSwitch"	Is a construct used to specify overlaid sets of data fields where the value of another, referenced data field defines which set is present in a word.
"DataElementDictionary"	Is a dictionary of all Data Elements used in the Messages, Each Data Element is identified by a DFI and a DUI and contains metadata like length and type, and information for decoding the bit-value. All Data Elements should be referenced from a Word.

Table 9-2: Description of Elements used within the Logical Model capturing the components of a generic TDL message structure

9.3 The Generic xTDL Message Structure schema

Based on the prerequisites for the usage of a generic TDL message structure described in Section 9.1 and on the logical model described in Section 9.2, an XML schema has been developed. This XML schema for a generic TDL message structure is shown and explained in the following sections. In each section, a high-level, generic description is provided after which a diagram is provided explaining that particular part of the schema. Included also is an example of a part of an instance document demonstrating the schema in action. A description of the major types and elements used within the Generic xTDL Message Structure schema is provided in Section 9.3.5.

Although the TDL message structure and the DED are captured in separate XML schemas, integrity checking between the XML instance documents is required and conducted for the references from “**DataFields**” and “**StructureSwitch**” in the generic TDL message structure to “**DataElements**” in the generic TDL DED as described in Chapter 5.

9.3.1 The “MessageStructure” element

Figure 9-2 shows the element “**MessageStructure**” as root element of the Generic xTDL Message Structure schema, with its child elements “**BaselineInfo**” and “**MessageCatalogue**”. The “**BaselineInfo**” element is the same container as for the generic xTDL DED described in Chapter 8, containing the metadata required to describe the TDL standard specification itself (see Section 7.1).

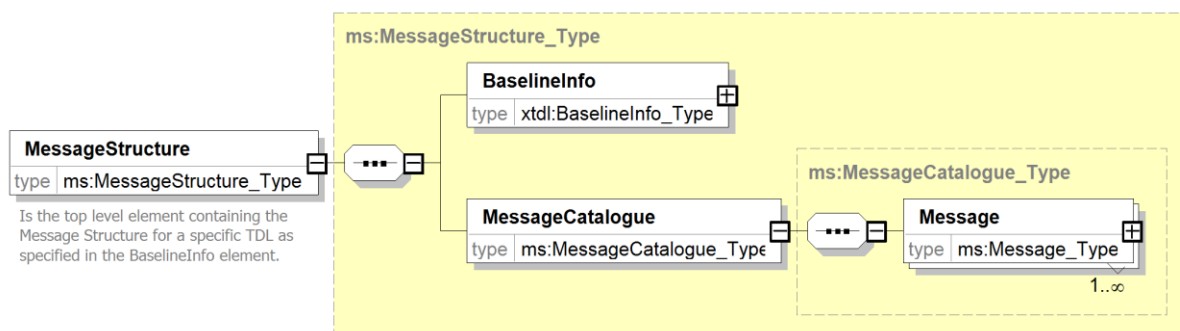


Figure 9-2: xTDL schema depicting the element “**MessageStructure**” as root element with its related child elements

An example of an XML instance for STANAG 5516 corresponding to the schema above is shown below:

```
<ms:MessageStructure
  xmlns:ms="urn:nato:tdl:generic:messageStructure:1:20140301:draft"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    urn:nato:tdl:generic:messageStructure:1:20140301:draft xTDL-MessageStructure.xsd">

  <BaselineInfo>
    <Title>TACTICAL DATA EXCHANGE - LINK 16</Title>
    <Identifier>STANAG 5516</Identifier>
    <BaselineVersion>edition 3</BaselineVersion>
    <Version>2009-02</Version>
    <Component>MessageStructure</Component>
    <!-- Actual structure of the classification markings is still under discussion.
      Namespace should also be updated. Placeholder. -->
    <Security>
      <PolicyIdentifier>NATO</PolicyIdentifier>
      <Classification>UNCLASSIFIED</Classification>
      <Category type="permissive">RELEASABLE FOR INTERNET TRANSMISSION</Category>
    </Security>
  </BaselineInfo>

  <MessageCatalogue>
    <!-- see next section -->
  </MessageCatalogue>
</ms:MessageStructure>
```

9.3.2 The “MessageCatalogue” element

Figure 9-3 shows the “**MessageCatalogue**” element, which includes a list of “**Message**” elements with their child elements that specify some metadata of the “**Message**” (e.g. the title) and the “**Word**” elements that hold the actual data.

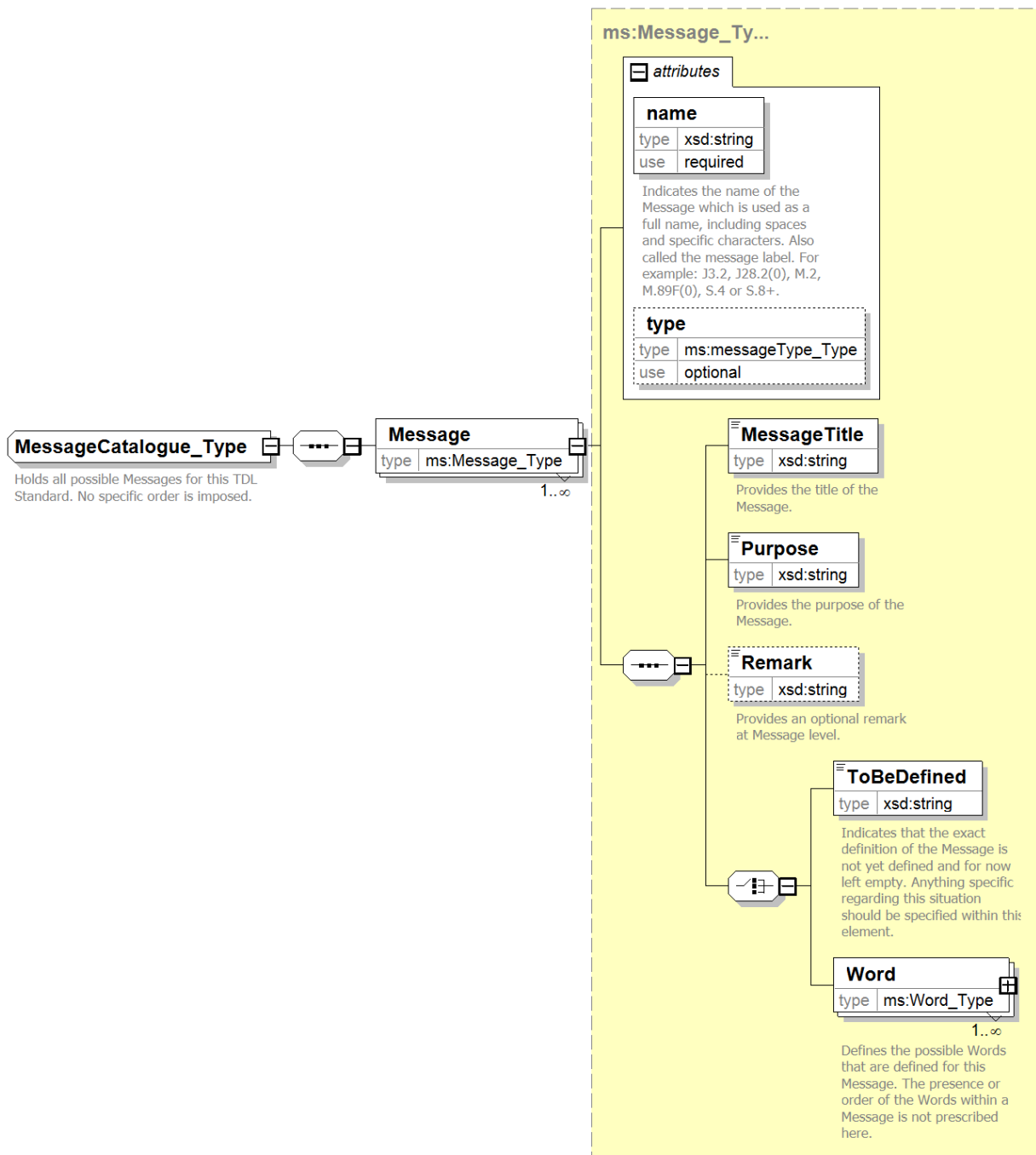


Figure 9-3: xTDL schema depicting the element “**MessageCatalogue**” with its related child elements as part of the generic TDL message structure

An example of an XML instance for three “**Messages**” from ATDLP-5.16 corresponding to the schema above is shown below:

```
<MessageCatalogue>
  <Message name="J3.2">
    <MessageTitle>Air Track Message</MessageTitle>
    <Purpose>The J3.2 Air Track message is used to exchange information on air
tracks.</Purpose>
    <Remark />
    <Word name="J3.2I">
      <!-- see next section -->
    </Word>
    <Word name="J3.2E0">
      <!-- see next section -->
    </Word>
    <Word name="J3.2C1">
      <!-- see next section -->
    </Word>
  </Message>
  <Message name="J3.3">
    <MessageTitle>Surface (Maritime) Track Message</MessageTitle>
    <Purpose>The J3.3 Surface (Maritime) Track message is used to exchange information on
surface (maritime) tracks.</Purpose>
    <Remark />
    <Word name="J3.3I">
      <!-- see next section -->
    </Word>
    <Word name="J3.3E0">
      <!-- see next section -->
    </Word>
    <Word name="J3.3C1">
      <!-- see next section -->
    </Word>
  </Message>
  <Message name="J13.4">
    <MessageTitle>Subsurface (Maritime) Platform and System Status Message</MessageTitle>
    <Purpose>The J13.4 Subsurface (Maritime) Platform and System Status message provides
the current status of a subsurface (maritime) platform to include operational status and on
board systems' status.</Purpose>
    <Remark />
    <ToBeDefined>To be defined.</ToBeDefined>
  </Message>
</MessageCatalogue>
```

9.3.3 The “Word” element

The “**Word**” element as depicted in Figure 9-4 contains “**DataField**” and/or “**StructureSwitch**” elements. The “**DataField**” holds the actual data referring to a “**DataElement**” that defines how that data is held.

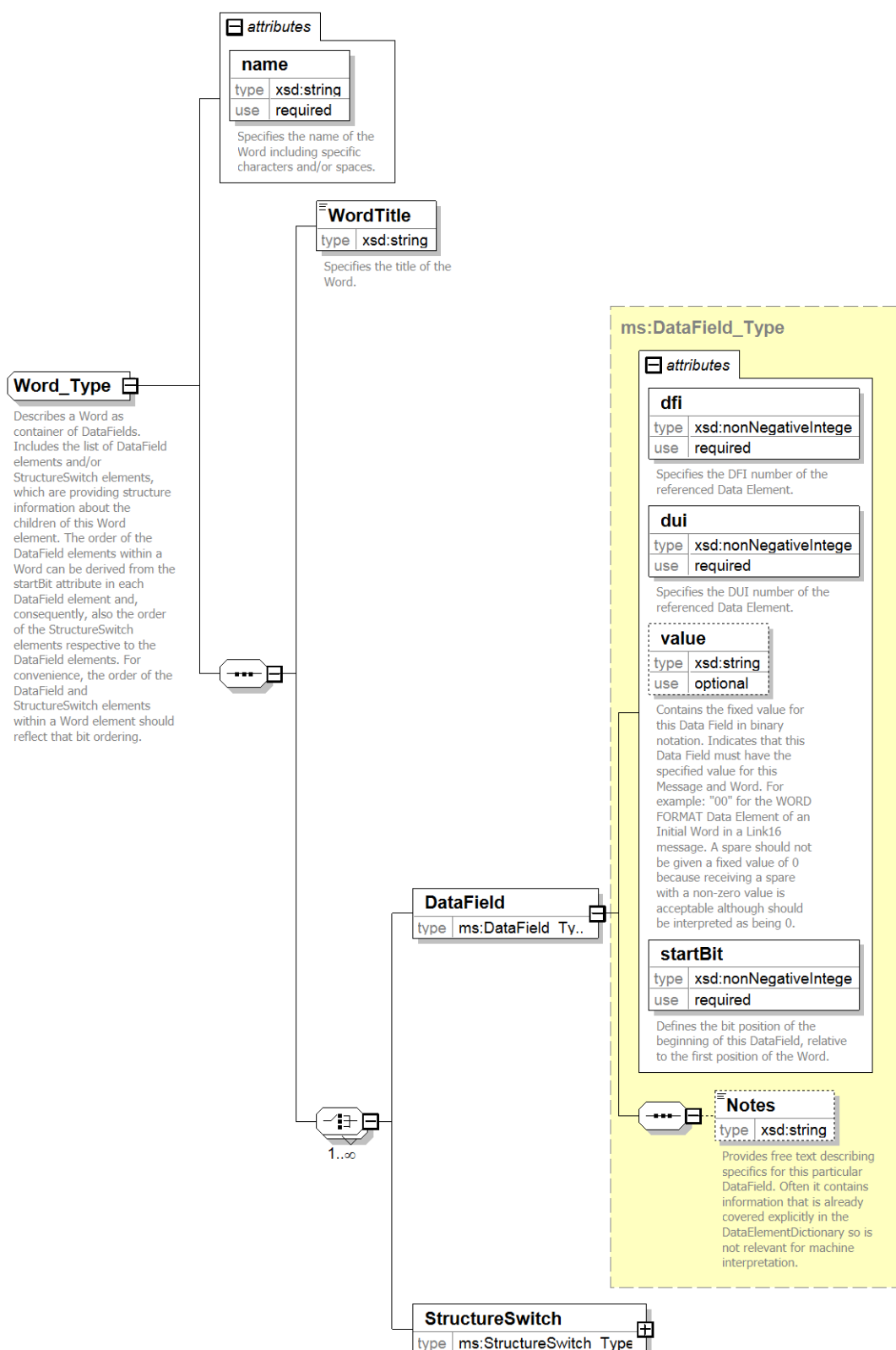


Figure 9-4: xTDL schema depicting the element “**Word**” with its related child elements as part of the generic TDL message structure

An example of an XML instance for two “**Words**” from ATDLP-5.16 corresponding to the schema above is shown below. The information contained in the “**Notes**” element is not essential for the processing of the information but reflects content from the current ATDLP document.

```
<Word name="J3.1I">
  <WordTitle>EMERGENCY POINT INITIAL WORD</WordTitle>
  <DataField dfi="1550" dui="001" startBit="0" value="00" />
  <DataField dfi="270" dui="004" startBit="2" value="00011" />
  <DataField dfi="271" dui="005" startBit="7" value="001" />
  <DataField dfi="800" dui="001" startBit="10">
    <Notes>0 NO ADDITIONAL WORDS. 1-7 NUMBER OF ADDITIONAL WORDS.</Notes>
  </DataField>
  <DataField dfi="385" dui="003" startBit="13" />
  <DataField dfi="756" dui="003" startBit="14" />
  <DataField dfi="292" dui="002" startBit="17" />
  <DataField dfi="1604" dui="001" startBit="18" />
  <DataField dfi="769" dui="002" startBit="19" />
  <DataField dfi="1643" dui="001" startBit="38" />
  <DataField dfi="756" dui="005" startBit="42" />
  <DataField dfi="1641" dui="001" startBit="47" />
  <DataField dfi="769" dui="018" startBit="51" />
</Word>

<Word name="J3.7C5">
  <WordTitle>ELECTRONIC WARFARE PRODUCT INFORMATION CONTINUATION WORD 5</WordTitle>
  <DataField dfi="1550" dui="001" startBit="0" value="01" />
  <DataField dfi="1551" dui="001" startBit="2" value="00101" />
  <StructureSwitch dfi="275" dui="004">
    <!-- see next section -->
  </StructureSwitch>
  <DataField dfi="756" dui="051" startBit="19" />
</Word>
```

9.3.4 The “StructureSwitch” element

The “**StructureSwitch**” element in Figure 9-5 provides support for overlaid sets of “**DataFields**” in a “**Word**” using branches for specific value(s) (via the “**When**” elements) and an optional default branch (via the “**Otherwise**” element) which is taken if none of the When clauses apply. The “**StructureSwitch**” element refers to a “**DataElement**” whose value is used to select a branch. Each “**When**” element specifies for which value(s), via the “**Case**” elements, the branch should be taken. As can be seen, each “**When**” or “**Otherwise**” itself holds “**DataFields**” and provides support for nested switching via another “**StructureSwitch**” element.

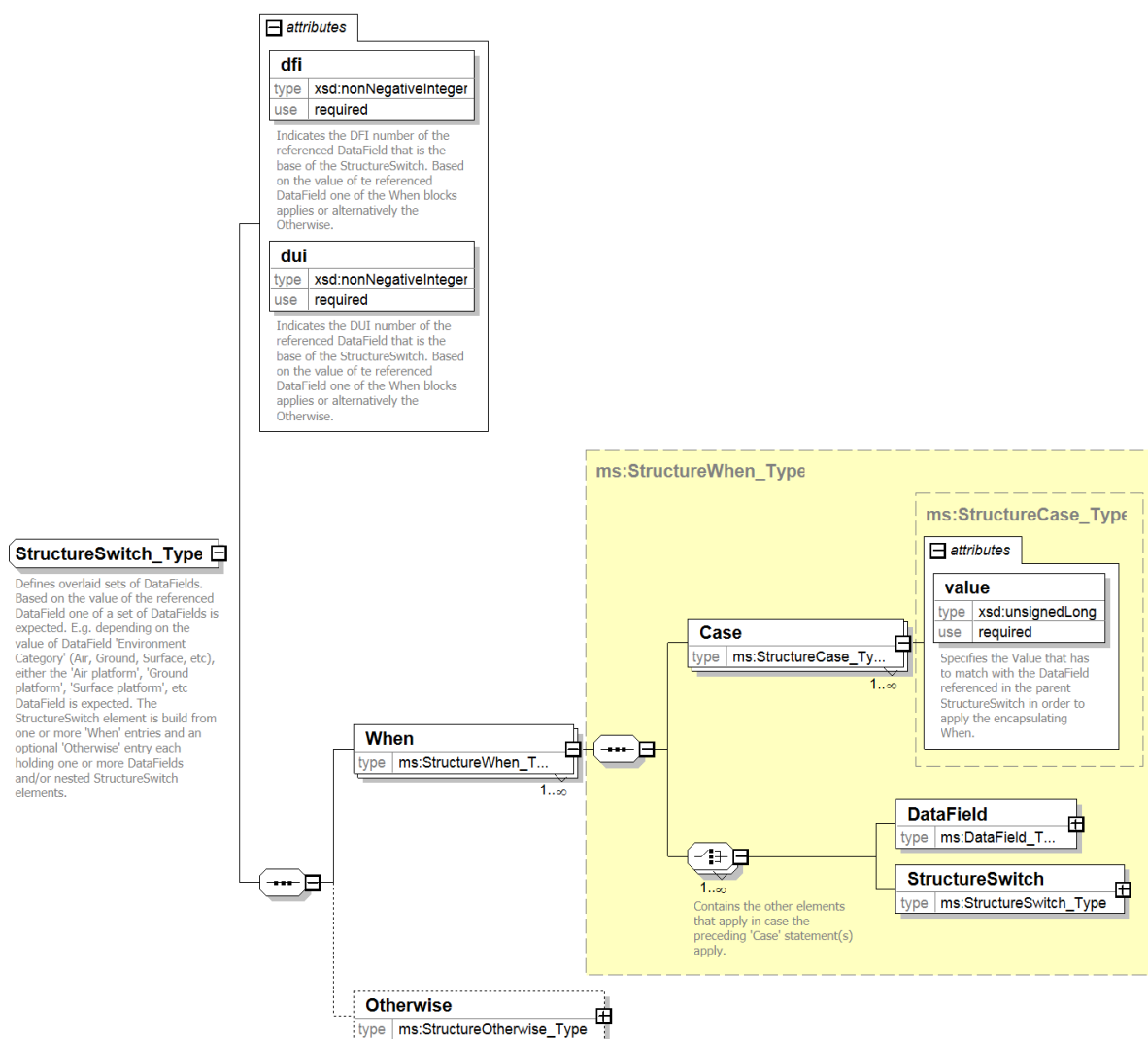


Figure 9-5: xTDL schema depicting the element “**StructureSwitch**” with its related child elements as part of the generic TDL message structure

An example of an XML instance for three “**StructureSwitches**” in the J3.7C5 word from ATDLP-5.16 corresponding to the schema above is shown below:

```
<StructureSwitch dfi="275" dui="004">
  <When>
    <Case value="1" />
    <DataField dfi="749" dui="002" startBit="7" />
  </When>
  <When>
    <Case value="2" />
    <DataField dfi="804" dui="001" startBit="7" />
  </When>
  <When>
    <Case value="3" />
    <DataField dfi="808" dui="001" startBit="7" />
  </When>
  <When>
    <Case value="4" />
  </When>
</StructureSwitch>
```

```

    <DataField dfi="809" dui="001" startBit="7" />
  </When>
  <When>
    <Case value="5" />
    <DataField dfi="810" dui="001" startBit="7" />
  </When>
  <!-- The ATDLP does not specify anything for the other values
       which are NO STATEMENT and UNDEFINED. In normal situations
       these indeed should not happen but for completeness they
       should be specified.
  <Otherwise>
    <DataField dfi="756" dui="012" startBit="7" />
  </Otherwise>
-->
</StructureSwitch>

```

This following example, based on the J3.0E0 word, shows a nested “**StructureSwitch**”:

```

<StructureSwitch dfi="363" dui="003">
  <When>
    <Case value="0" />
    <StructureSwitch dfi="379" dui="002">
      <When>
        <Case value="4" />
        <DataField dfi="1768" dui="001" startBit="2" />
      </When>
      <Otherwise>
        <DataField dfi="756" dui="003" startBit="2" />
      </Otherwise>
    </StructureSwitch>
  </When>
  <Otherwise>
    <DataField dfi="756" dui="003" startBit="2" />
  </Otherwise>
</StructureSwitch>
<DataField dfi="281" dui="017" startBit="5" />
<StructureSwitch dfi="363" dui="003">
  <When>
    <Case value="0" />
    <StructureSwitch dfi="379" dui="002">
      <When>
        <Case value="4" />
        <DataField dfi="1767" dui="001" startBit="25" />
      </When>
      <Otherwise>
        <DataField dfi="756" dui="002" startBit="25" />
      </Otherwise>
    </StructureSwitch>
  </When>
  <Otherwise>
    <DataField dfi="756" dui="002" startBit="25" />
  </Otherwise>
</StructureSwitch>

```

This example taken from the J7.1I word shows a “**StructureSwitch**” that demonstrates that a when-clause can be chosen for more than one value (in this case value ‘1’ or ‘2’). It also shows that each when-clause can hold a different number of “**DataFields**”, as long as the total length in bits of each branch is the same:

```

<StructureSwitch dfi="1606" dui="002">
  <When>
    <Case value="0" />
    <DataField dfi="756" dui="019" startBit="19" />
    <DataField dfi="1675" dui="001" startBit="38" />
  </When>
  <Case value="1" />
  <DataField dfi="756" dui="019" startBit="19" />
  <DataField dfi="1675" dui="001" startBit="38" />
  <Case value="2" />
  <DataField dfi="756" dui="019" startBit="19" />
  <DataField dfi="1675" dui="001" startBit="38" />
</StructureSwitch>

```

```

<DataField dfi="1675" dui="002" startBit="39" />
<DataField dfi="1675" dui="003" startBit="40" />
<DataField dfi="1675" dui="004" startBit="41" />
<DataField dfi="1675" dui="009" startBit="42" />
<DataField dfi="1675" dui="005" startBit="43" />
<DataField dfi="1675" dui="006" startBit="44" />
<DataField dfi="1675" dui="007" startBit="45" />
<DataField dfi="1675" dui="008" startBit="46" />
</When>
<When>
  <Case value="1" />
  <Case value="2" />
  <DataField dfi="769" dui="002" startBit="19"/>
  <DataField dfi="756" dui="009" startBit="38" />
</When>
<Otherwise>
  <DataField dfi="756" dui="019" startBit="19" />
  <DataField dfi="756" dui="009" startBit="38" />
</Otherwise>
</StructureSwitch>

```

9.3.5 Description of Types and Elements used within the Generic xTDL Message Structure schema

The following tables provide a description of types and elements used in the text and diagrams in the sections above. These descriptions are automatically extracted from the XML schema (contained in the xsd:annotation). Only the main types and elements are included, for all others the reader is referred to the actual schema (see Annex H).

"Type"	Description
"DataField_Type"	Describes a "DataField" within a Word and holds the actual binary data. A "DataField" refers to a Data Element via the DFI and DUI. The "DataField" specifies its starting position in the "Word" while the Data Element specifies the length of the "DataField" . Optionally a "DataField" can have a fixed value.
"Message_Type"	Defines the structure information for a particular Message. A Message has some metadata (like a Name) and consists of "Word" elements, which contain "DataField" elements that can hold the actual data. If required "StructureSwitch" elements can be used.
"MessageCatalogue_Type"	Holds all possible Messages for this TDL standard. No specific order is imposed.
"MessageStructure_Type"	Contains all information for a specific TDL on the syntactic and semantic meaning of the messages. The Message Structure is generic to fit the bit-oriented TDL specifications for Link 1, Link 11/Link 11B, Link 16, Link 22 and future TDLs in the long term. Structuring is composed of a "Message" level, a "Word" level and a "DataField" level. If a TDL standard does not use the "Word" level, a dummy "Word" level is inserted which covers the whole Message. This dummy "Word" level should not be exposed to end users. "DataField" s refer

"Type"	Description
	to Data Elements identified by a DFI and a DUI. In case a TDL prescribes alternative sets of "DataField" s to occur in a Word depending on the value of another "DataField" , so-called switching is used. See for more information the documentation of the respective elements.
"messageType_Type"	Denotes the nature of the message: whether it is a test message, blank, for management or containing actual (tactical) data. Further values might be defined.
"StructureCase_Type"	Specifies the value for the referenced "DataField" for which the enclosing "When" element is selected and therefore the following "DataField" s and/or nested "StructureSwitch" es.
"StructureOtherwise_Type"	Contains a set of Data Fields or nested "StructureSwitch" (es) that all apply in case none of the preceding "When" elements was applied.
"StructureSwitch_Type"	Defines overlaid sets of "DataField" s. Based on the value of the referenced "DataField" one of a set of "DataField" s is expected. E.g. depending on the value of "DataField" 'Environment Category' (Air, Ground, Surface, etc.), either the 'Air platform', 'Ground platform', 'Surface platform', etc. "DataField" is expected. The "StructureSwitch" element is built from one or more 'When' entries and an optional 'Otherwise' entry each holding one or more "DataField" s and/or nested "StructureSwitch" elements.
"StructureWhen_Type"	Defines an overlaid set of one or more "DataField" s. The enclosed "Case" element(s) indicate for which value(s) of the referenced "DataField" this set should be chosen.
"Word_Type"	Describes a Word as container of "DataField" s. Includes the list of "DataField" elements and/or "StructureSwitch" elements, which are providing structure information about the children of this "Word" element. The order of the "DataField" elements within a Word can be derived from the "startBit" attribute in each "DataField" element and, consequently, also the order of the "StructureSwitch" elements respective to the "DataField" elements. For convenience, the order of the "DataField" and "StructureSwitch" elements within a "Word" element should reflect that bit ordering.

Table 9-3: Description of Types used within the Generic xTDL Message Structure schema

"Element"	Type	Description
"MessageStructure"	ms:MessageStructure_Type	Is the top level element containing the Message Structure for a specific TDL as specified in the "BaselineInfo" element.
"MessageTitle"	xsd:string	Provides the title of the Message.
"Notes"	xsd:string	Provides free text describing specifics for this particular "DataField" . Often it contains information that is already covered explicitly in the "DataElementDictionary" so is not relevant for machine interpretation.
"Purpose"	xsd:string	Provides the purpose of the Message.
"Remark"	xsd:string	Provides an optional remark at Message level.
"ToBeDefined"	xsd:string	Indicates that the exact definition of the Message is not yet defined and for now left empty. Anything specific regarding this situation should be specified within this element.
"Word"	ms:Word_Type	Defines the possible Words that are defined for this Message. The presence or order of the "Word" s within a "Message" is not prescribed here.
"WordTitle"	xsd:string	Specifies the title of the "Word" .

Table 9-4: Description of Elements used within the Generic xTDL Message Structure schema

9.4 Relationship of the Generic xTDL Message Structure schema with other XML schemas

The Generic xTDL Message Structure schema uses some types common to the xTDL Standard representation which are defined in a separate schema.

A TDL message contains units of data called data elements. The available data elements of a specific TDL standard are defined in a TDL DED which is described in Chapter 8. Integrity checking between the XML instance documents for the TDL message structure and DED is achieved as previously described in Chapter 5.

The generic XML schemas together with the XML instance document is directly related to the XML schema of XML message instance documents as described in Chapter 14.

9.5 Considerations for the usage of the Generic xTDL Message Structure schema

The described XML schema for the generic TDL message structure still has some “loose ends” that are not captured yet and need to be addressed in the future:

- TDLs with a more complex structure cannot be represented with the current “**Message**” / “**Word**” / “**DataField**” structure. E.g. NILE (the encapsulating protocol for Link 22) or VMF requires more nesting support. A possible extension of the current structure would be to allow for nesting of the Word elements.
- The current structure does not yet support TDLs that define messages of variable length by including optional contents (e.g. ASTERIX and VMF), but can be enhanced to serve this purpose.

10 REPRESENTATION OF TDL PROCESSING IN XML

This chapter describes the representation of the TDL Processing in XML which is either composed of transactions or transmit and receive (Tx/Rx) rules, each with their corresponding tables, and the supporting XML schema. Section 10.1 will describe the TDL Processing in its Transactional form while Section 10.2 will address the TDL Processing based on Transmit and Receive Rules.

The transactional format will first be introduced with STANAG 5516 Edition 6 whereas Tx/Rx rules are used with the TDL standards for Link 1 (ATDLP-5.01), Link 11/11B (ATDLP-5.11), Link 22 (ATDLP-5.22), and previous editions of STANAG 5516/ATDLP-5.16.

10.1 Transactional TDL Processing

This section describes in a first iteration only the Logical Model for Transactional TDL Processing. After a first draft STANAG 5516 Edition 6 has been produced the Physical Model and the related XML schema will be developed and incorporated into this section.

10.1.1 Logical Model capturing transactional TDL Processing

The logical model below depicts the components that directly support transactional TDL Processing.

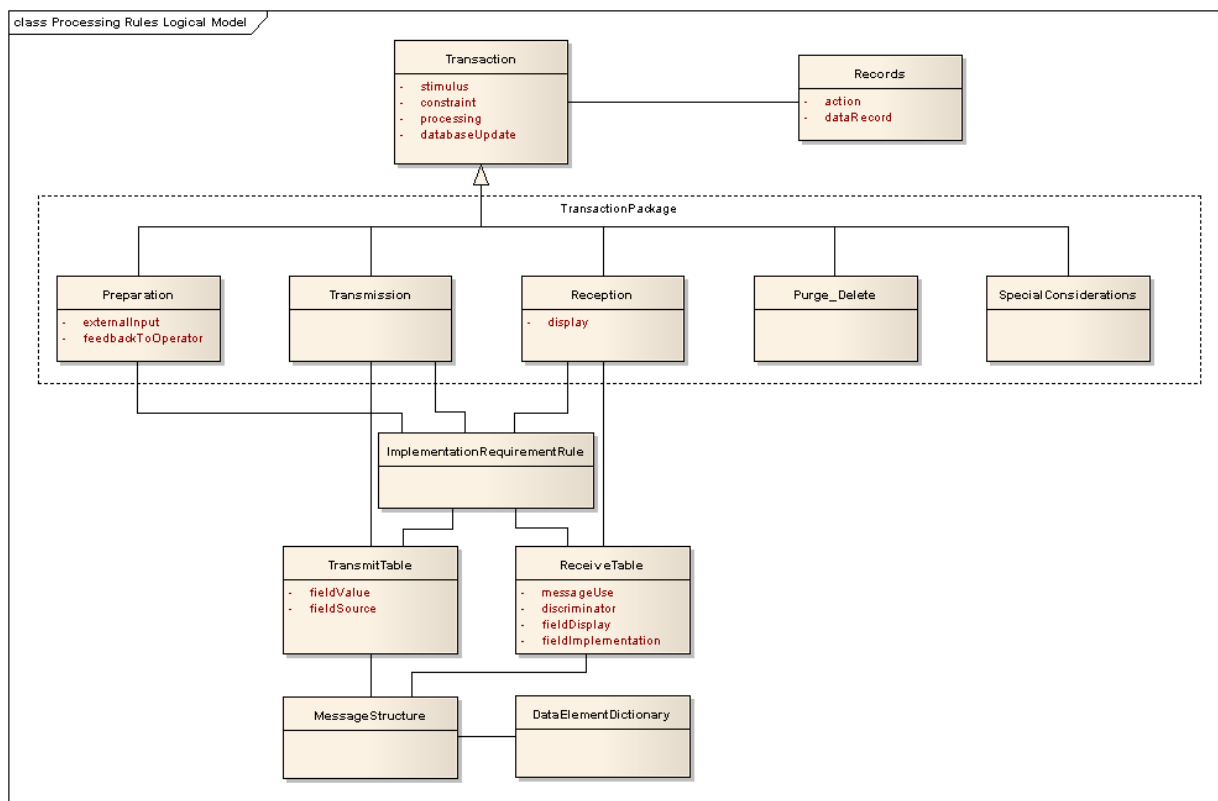


Figure 10-1. Logical Model capturing the components required for transactional TDL Processing

“Class”	Description
“ImplementationRequirementRule”	Defines the rule set that is a minimum to implement a system compliant to a TDL standard (e.g. Link 16).
“MessageStructure”	Prescribes which data elements can be placed in which order in a message and the structure to which it should adhere.
“DataElementDictionary”	Describes the possible data elements for a given TDL standard which are the building blocks of actual data to construct messages.
“Preparation”	Defines the transaction within the transaction package describing what capability the host system shall provide the operator or system/platform to allow the initiation or modification of messages on the TDL.
“Purge_Delete”	Defines the transaction within the transaction package describing the processing required to remove a record from the host system database.

"Class"	Description
"ReceiveTable"	Defines how each message received will be discriminated by message use, and for each message use, which transaction(s) will be stimulated and what the requirements are for the data field processing and display.
"Reception"	Defines the transaction within the transaction package describing the processing the host system is required to perform when a message is received.
"Records"	Store data related to the system/platform and the reception/transmission of messages.
"SpecialConsiderations"	Defines the transaction within the transaction package describing the processing which is not covered by Preparation, Transmission, Reception, or Purge/Delete.
"Transaction"	Describes a specific set of processing steps which are executed when initiated by event(s) (" stimuli ") under pre-defined conditions (" constraints ").
"Transmission"	Defines the transaction within the transaction package describing the processing requirements for transmitting messages.
"TransmitTable"	Defines how the system/platform will construct messages for transmission.

Table 10-1: Description of Classes used within the Logical Model capturing the components required for transactional TDL Processing

10.1.2 Logical Model capturing the components required for TDL Transactions

The logical model depicts a decomposition of the "**Transaction**" class from Figure 10-1. The "**Transaction**" logical model is presented in Figure 10-2. This model exposes the stimuli and constraints controlling TDL Processing.

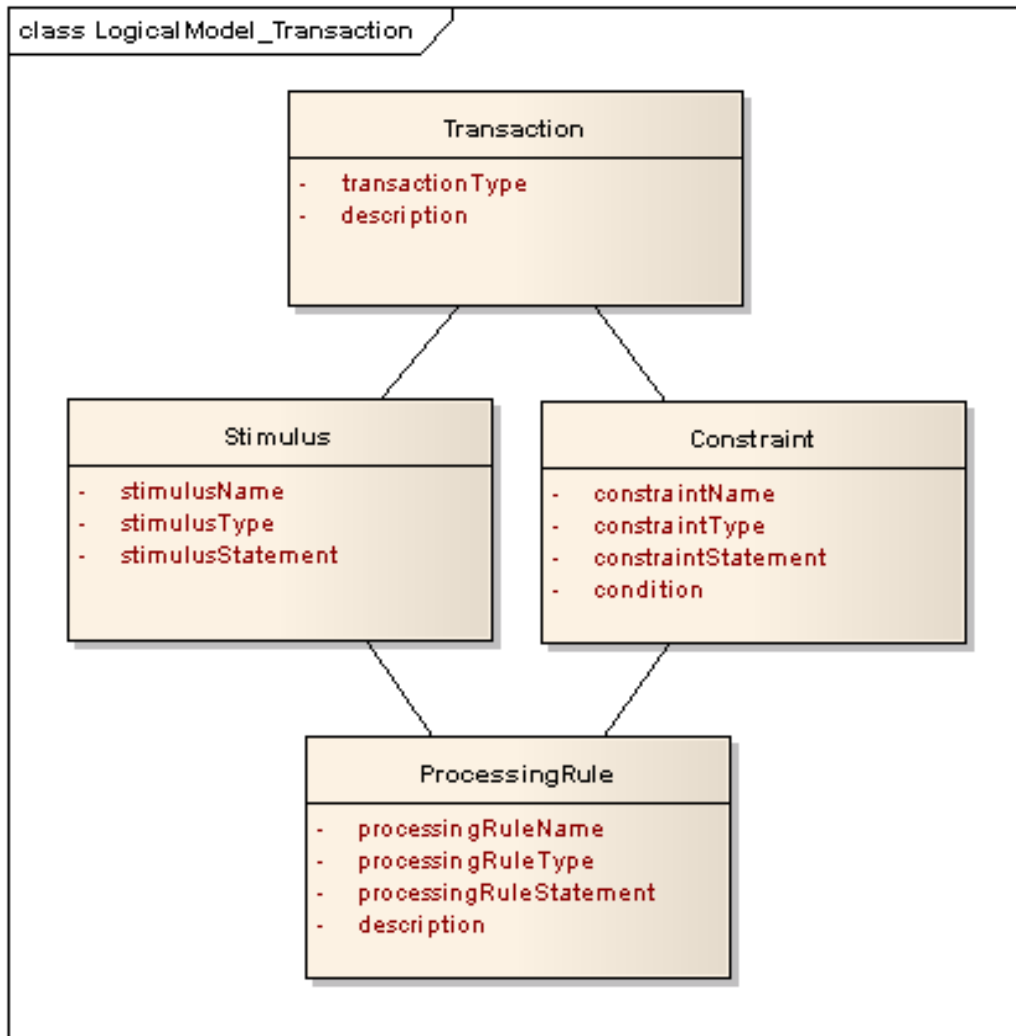


Figure 10-2. Logical Model capturing the components required for TDL Transactions

"Class"	Description
"Transaction"	Describes a specific set of processing steps which are executed when initiated by event(s) (" stimuli ") under pre-defined conditions (" constraints ").
"Stimulus"	Is an event (a message reception, system/operator event, periodic event, or result from another transaction) that triggers a transaction.
"Constraint"	Is a condition, verified before the processing rules are executed, that may cause an operator alert, may prevent processing from starting, or may stimulate other transactions.
"ProcessingRule"	Describes the behaviour of the transaction.

Table 10-2: Description of Classes used within the Logical Model capturing the components required for TDL Transactions

10.2 TDL Processing based on Transmit and Receive Rules

Models for TDL standards, which do not yet exist in transactional format but employ Transmit and Receive Rules, will be developed and incorporated into ATDLP-7.04 as appropriate.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

11 REPRESENTATION OF DATA FORWARDING IN XML

This chapter describes the representation of data forwarding rules in XML. The information has been derived from STANAG 5616 Edition 5 Volume 1, Annex B.

11.1 Logical Model capturing the High-Level TDL Forwarding Rule Concepts

The UML based model shown below is a mix of a logical and physical class model. This format was selected in order to show specific relationships between different components, represented in the model as classes and attributes that will help describe relationships between forwarding rule components that would not appear in a simpler logical model. Because this is a combined logical and physical view, the content has not been normalized and so some fields may be duplicated, but this was deliberate in order to keep the model relatively simple and to provide clarity. The intent of the model is to show those components that directly support the forwarding rules for the messages, words, data elements, and values. The table below the model provides definitions for the classes and attributes that appear in the model.

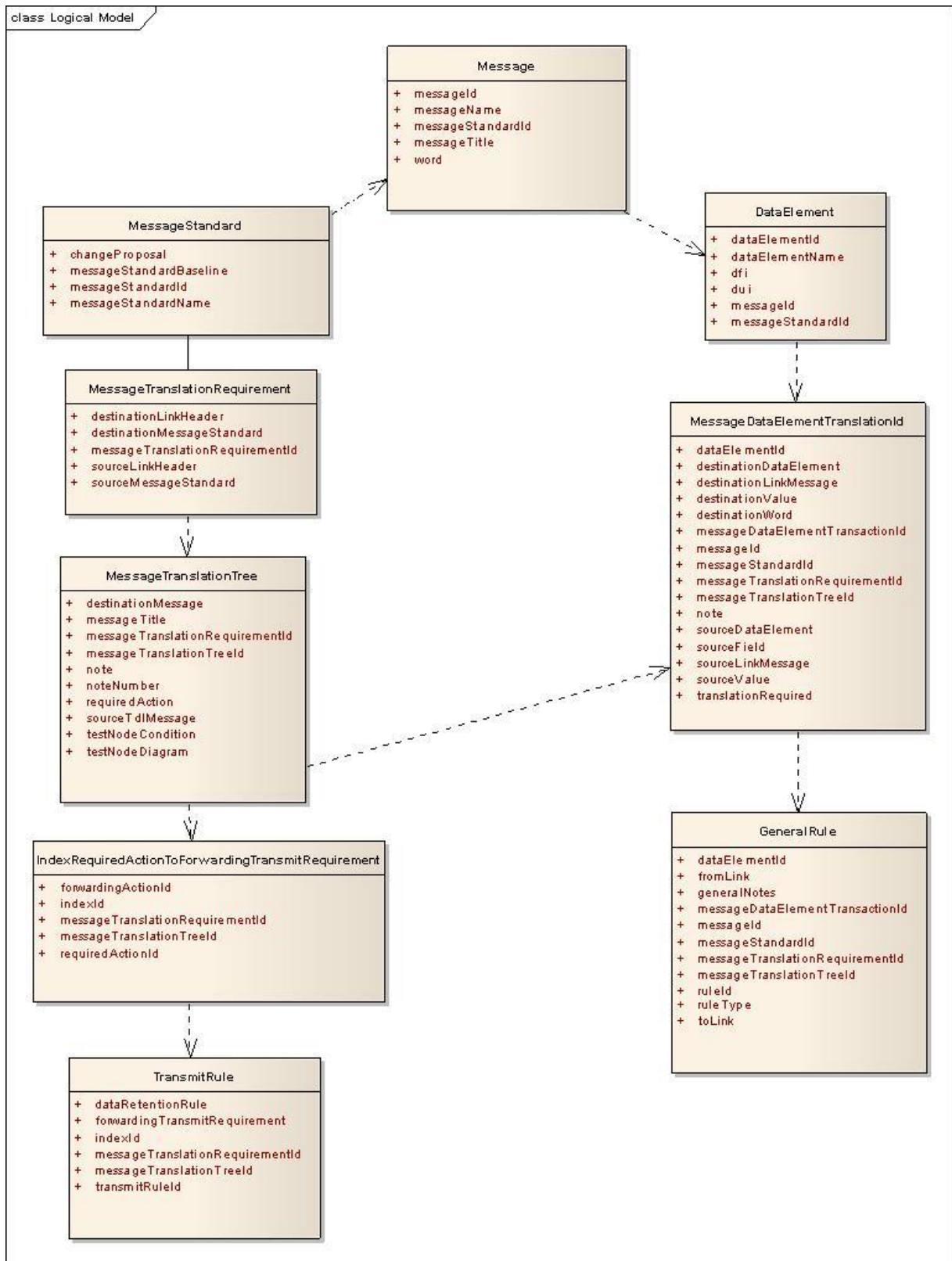


Figure 11-1: Logical Model capturing the components required for the representation of High-Level TDL Forwarding Rule Concepts

"Classes"	Description
"MessageStandard"	Specifies a set of messages, including the structure, format, data, metadata and associated rules.
"Message"	Is a structured collection of one or more words to report a particular set of information.
"DataElement"	Is the instantiation or use of a data element.
"MessageTranslationRequirement"	Describes an evaluation of messages or message sequences, message contents, link protocols and message exchange rules for each translatable message or message sequence.
"MessageTranslationTree"	Is a logical set of conditions depicting how a received message or message sequence is to be tested to determine the appropriate translation and action to be taken.
"MessageDataElementTranslation"	These tables are a data element by a data element depiction of the message to be generated with an indication of the source of the data to be used in the data element.
"IndexRequiredActionTo ForwardingTransmitRequirement"	Is a notional physical table that was added to show that there is relationship specifically between the rules documented in the field "Required Action" and a relate rule in the "Forwarding Action".
"TransmitRule"	Describes a rule for transmitting a message between TDL standards.
"GeneralRule"	Describes a rule that applies to more than one message, word, data element, or value.

Table 11-1: Description of Classes used within the Logical Model capturing the High-Level TDL Forwarding Rule Concepts

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

12 BUSINESS RULES

Business Rules are described within Standard Operating Procedures (SOP) covering one or more TDL standard(s), e.g. ATDLP-7.12, ATDLP-7.31 and ATDLP-7.33. A representation of these NATO documents in XML will have to be addressed by the TDL COI when appropriate.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

13

**REPRESENTATION OF MINIMUM IMPLEMENTATION /
IMPLEMENTATION REQUIREMENTS IN XML**

The representation of Minimum Implementation (MIN IMP) and Implementation Requirements (IMP REQ) in XML is subject to further analysis and will be incorporated when appropriate.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

14 TDL MESSAGE INSTANCES

An xTDL Message Instance document is an XML representation (i.e., an XML "instance") of a TDL message. It will include all required W3C XML information, message and data element attributes, and TDL specific elements.

The schemas for validating xTDL message instance documents will be automatically derived from the XML instance documents for the xTDL message structure and DED.

For an example of a TDL Message Instance in XML see Annex F.

The representation of TDL Message Instances in XML is subject to further analysis and will have to be reviewed when appropriate.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

15 REPRESENTATION OF THE TDL SYSTEM IMPLEMENTATION IN XML

To assess interoperability between platforms at an early stage, an analysis of their TDL implementation should be performed using their respective System Implementation Documents (SID) based on a common and open standard such as the NATO Implementation Codes and Rules (NICR T/1) document. To facilitate making the comparison as automated as possible, a machine-interpretable XML format is provided in this chapter.

The SID Schema described in this chapter applies the same generic concept as is defined for the Data Element Dictionary (Chapter 8) and Message Structure (Chapter 9). This means that the Schema defines, e.g., a generic **“Message”** element which has a **“name”** attribute to indicate the name of the message. The resulting XML representation for, e.g., ATDLP-5.16, will therefore have a **“Message”** XML element with **“J3.2”** as value for the **“name”** attribute.

The generic approach implies that the same Schema can be used for different TDLs, providing benefits to the implementation of supporting tools and results in consistency across the TDLs.

Section 15.1 describes the prerequisites for the definition of a generic TDL SID by providing a non-exhaustive list of requirements to which a generic TDL SID Schema has to comply. Section 15.2 provides the logical model capturing the generic TDL SID components. The following sections 15.3, 15.4 and 15.5 provide descriptions of common concepts which are referenced in section 15.6, where each of the Schema's relevant XML elements is described. In Section 15.7, the use of Schematron to verify the NICR T/1 Rules is explained and in Section 15.8 the overall Verification process of an XML SID instance document is described.

15.1 Prerequisites for the usage of a generic TDL System Implementation

This section provides a non-exhaustive list of requirements to which a generic TDL SID Schema has to comply for any bit-based message format.

A number of generic concepts will be used which are introduced in Table 15-1 below. These are in addition to the ones defined for the DED (Table 8-1) and Message Structure (Table 9-1). The **“xTDL name”** refers to the name as used within this document and within the XML schemas.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

Concept	"xTDL name"	Definition
System implementation data	" SID "	<p>The set of data describing the level of support of a platform for transmission and reception of the elements of a TDL information exchange.</p> <p>For example, an SID will indicate whether a platform will process a particular message on reception but will not transmit the message itself. Also, it can indicate that only specific values of a Data Field are transmitted.</p>
Platform information	" PlatformInfo "	<p>The information defining a platform, like its name, the STANAG/ATDLP and edition it supports including zero or more DLCPs and other relevant meta data.</p> <p>For example, the platform's name is "TBD" and it implements STANAG 5516 Ed5 plus DLCP X, Y and Z.</p>
Codes	" Codes "	<p>A set of letters, defined in NICR T/1, to indicate the level of support for transmission and reception.</p> <p>For example, if transmit code "T" is used for message J3.2, it indicates that the platform has support for the transmission of this message, while a reception code "DM" indicates that the message will be discarded (Discard Message) on reception.</p>
Rules	" Rules "	<p>The rules describe the hierarchical dependency of the implementation codes at each level, i.e., which codes are valid at each level and the relations between the codes used at the various levels.</p> <p>For example, if transmit code "T" is used for message J3.2, the Initial Word must also have transmit code "T". Also, if receive code "R" is used for a Word, one or more included Data Fields must indicate a code of "R".</p>

Implementation switch	"ImplementationSwitch"	<p>A construct used to indicate that different implementation support applies for one or more Data Fields, depending on the value of another Data Field (the so called Action value as defined in NICR T/1).</p> <p>For example, in a J7.0 Track Management Message, the value of the Action data field defines the purpose (Message Use) of the overall message. Therefore, depending on the Action value, implementation support for, e.g., the Reference Track Number field may be different.</p> <p>Similarly, in a J3.0 Reference Point Message, the Point Type can result in different support of other fields or words within the message.</p> <p>For more details, see section 15.4.2.</p>
-----------------------	-------------------------------	--

Table 15-1: Generic concepts for the Generic TDL SID Schema

The generic TDL SID Schema shall support the following requirements:

- SID-001 The platform for which the SID is captured shall be identified by a name, implemented STANAG/ATDLP and edition, incorporated DLCPs, classification marking, and other relevant metadata.
- SID-002 The System Implementation Data Schema shall use the **"Codes"** as defined in the NICR T/1 to indicate the platform's support for each element for transmission and reception.
- SID-003 The System Implementation Data Schema shall use the **"Rules"** as defined in the NICR T/1 to verify the consistency between the used **"Codes"**.
- SID-004 The System Implementation Data Schema shall allow the **"Rules"** to be automatically validated.
- SID-005 The System Implementation Data Schema shall follow the structure of the Messages of the supported edition of the STANAG/ATDLP while allowing for alteration as a result of DLCPs incorporated in the platform's implementation.
- SID-006 The System Implementation Data Schema shall follow the definitions of the Data Elements of the supported edition of the

STANAG/ATDLP while allowing for alteration as a result of incorporated DLCPs.

- SID-007 The System Implementation Data Schema shall allow different implementation support for one or more Data Fields within a Word, depending on the value of another Data Field, to support the Action value concept as defined in the NICR T/1.
- SID-008 The System Implementation Data Schema shall allow different classification markings for individual Messages, Words, Data Fields and Values, which will override the one defined for the overall platform.

15.2 Logical Model capturing the components of the generic TDL SID Schema

The logical model shows the components that directly support the generic TDL SID Schema. The attributes shown in the classes denote relevant information that needs to be captured on the classes or indicate a relation between classes. Figure 15-1 below shows the high-level components needed to support the generic TDL SID Schema, e.g., to support capturing the transmission and reception requirements for Link 16 J-series messages.

From this logical model, the actual XML schema is derived by including all components (element and attributes) that are required to fully model the generic TDL SID Schema.

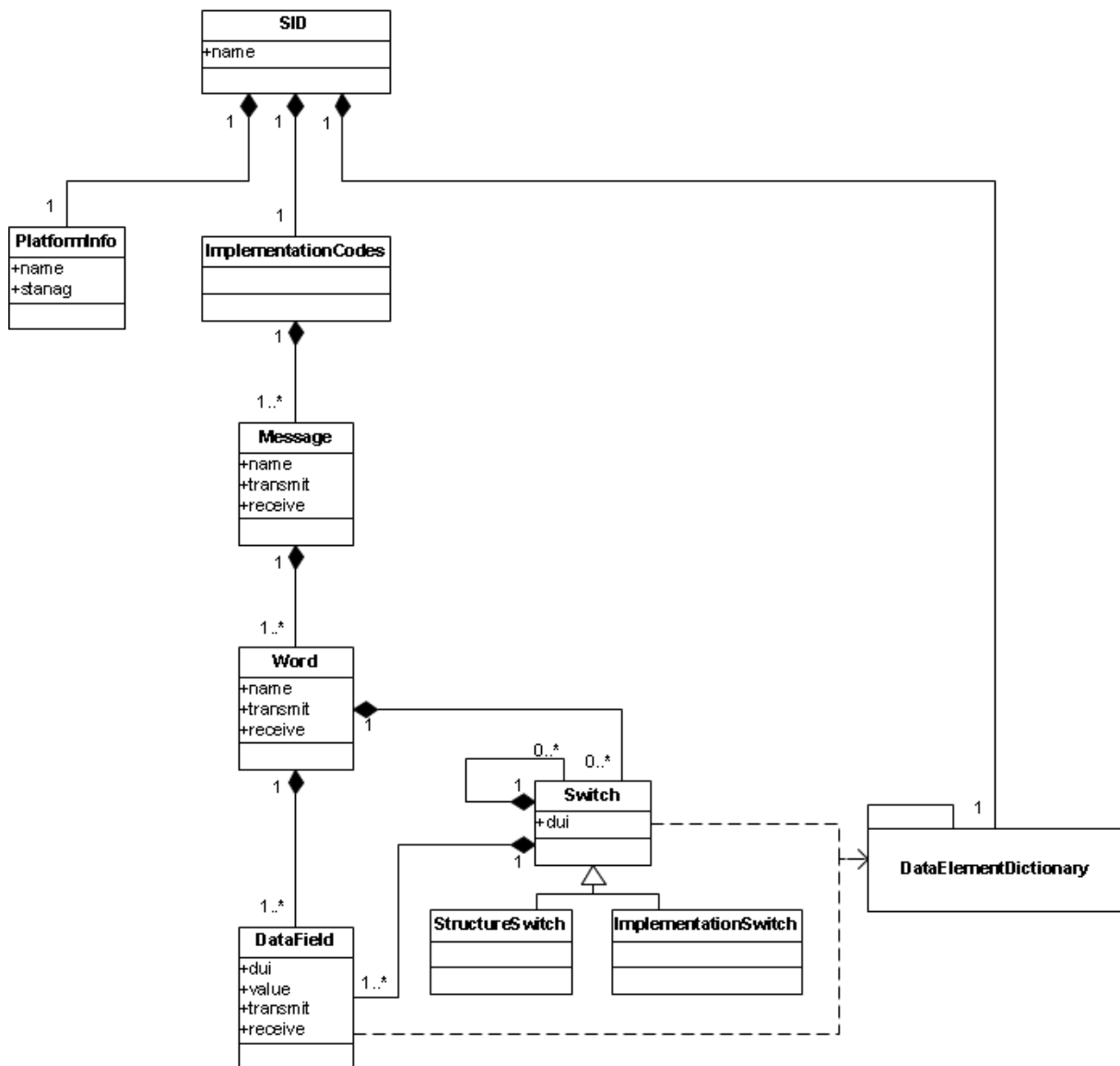


Figure 15-1: Logical Model of the components of System Implementation Data

"Elements"	Description
"SID"	Is the set of data defining the platform and the level of support the platform has implemented for the various aspects of a particular datalink.
"PlatformInfo"	Identifies the platform for which the SID is captured including metadata such as name, implemented STANAG/ATDLP, classification, incorporated DLCPs, etc.
"ImplementationCodes"	Provides per aspect (Message, Word, Field, Value) the support by the platform using the codes defined in the NICR T/1 and following the associated rules. It follows

"Elements"	Description
	the same structure as the MessageCatalogue element defined in Chapter 9.
"Message"	Is a structured collection of one or more words to report a particular set of information.
"Word"	Is a structured collection of one or more data fields used to report on a specific aspect.
"DataField"	Is the instantiation or use of a data element.
"StructureSwitch"	Is a construct used to specify overlaid sets of data fields where the value of another, referenced data field defines which set is present in the word.
"ImplementationSwitch"	Is a construct used to specify different implementation support for a set of data fields. The value of another, referenced data field (also referred to as the Action Value) defines what the implementation support is for each of the fields within the set.
"DataElementDictionary"	Is a dictionary of all Data Elements used in the Messages, identical in structure to the Data Element Dictionary for a particular STANAG/ATDLP. Each Data Element is identified by a DFI and a DUI and contains metadata like length and type, and information for decoding the bit-value.

Table 15-2 Description of Elements used within the Logical Model capturing the components of System Implementation Data

15.3 Transmit/receive values

The transmit and receive values that can be set on the various elements are the **"Codes"** as defined in NICR T/1. The Schema has a restriction on the allowed value for each level (Message, Word, Field and Value) in accordance with NICR T/1.

15.4 Switches

To fulfil the requirements for capturing the System Implementation Data, the Schema defines three types of switches: structure switches, implementation switches and coding switches. These three types of switches are introduced and described below and referenced in the specific sections.

15.4.1 Structure switches

The StructureSwitch has the same purpose as defined in the MessageStructure schema (see section 9.3.4) while containing slightly different content, so therefore has the same name but in a different namespace. In the StructureSwitch, the different Data Fields with their transmit and receive codes will be specified separately for specific values of the referenced data field.

15.4.2 Implementation switches

Situations exist where the level of support for a Word or set of DataFields is different if a particular DataField has a specific value (typically the Action Value as per NICR T/1). For example, a system might not have support for handling areas via the J3.0 message, so in the case that the POINT TYPE field has value 5 or 6 (indicating an area), the platform might discard the whole message. This functionality is supported via the ImplementationSwitch which has the same structure as the StructureSwitch but is meant specifically for those situations where the structure does not change among the different switch cases but only the implementation support differs.

15.4.3 Coding switches

The Coding Switch is the same concept as defined in the DataElementDictionary (see 8.4.3) and allows a different coding of a DataElement, depending on the value of another, defining Data Element (for instance, a Data Element may specify an altitude on a 1 meter, 10 meter or 100 meter scale, with the scale being defined by another data field).

In the case of the SID Schema, the Coding Switch is used in the Data Fields within the ImplementationCodes element to specify different transmit and receive values for each case in a specific context. It differs from the CodingSwitch element within the DataElementDictionary (section 8.4.3), in that further details about decoding the data element (e.g., Unit, ValueType and Formula) are not required at the DataField level and instead are included in the DataElementDictionary provided by the SID. Figure 15-2 shows the structure of the CodingSwitchImpl.

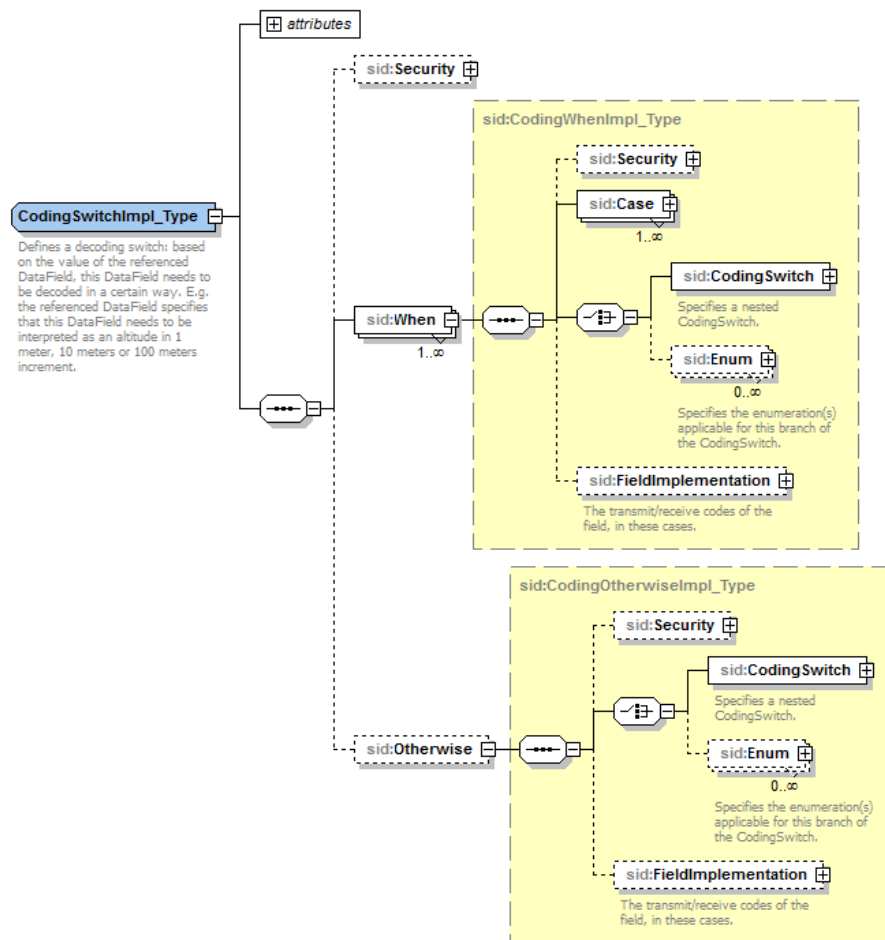


Figure 15-2 SID CodingSwitch element

15.5 Implementation based on several STANAG/ATDLP versions

A platform typically does not implement the Messages, Words and Data Elements exactly as defined by a single STANAG/ATDLP edition, but will often employ a mixture of different (interim) editions because of the introduction of specific DLCPs. Basically, this is supported in the Schema by allowing it to define the Message Structure and Data Element Dictionary as applicable for the specific platform. Technically it is accomplished by having the ImplementationCodes element define the Messages, Words and their Data Fields itself, so that it can be adapted based on incorporated DLCP changes. For traceability, the base STANAG/ATDLP edition is as stated in the PlatformInfo element and then overridden at Message and Word level (via a stanagEdition attribute).

Similarly, for the Data Element Dictionary, the XML Schema incorporates its own instance in the SID, thereby allowing it to be adapted for the specific platform. At the Schema level, it re-uses the existing Data

Element Dictionary Schema. The traceability of incorporated DLCPs cannot be traced directly within these elements and should instead be handled in the PlatformInfo element by enumerating the incorporated DLCPs.

15.6 The Generic TDL System Implementation Data schema

15.6.1 SID Top-level structure

The top-level structure of the XML Schema is depicted in Figure 15-3 and shows how an SID consists of the following three generic elements:

- The PlatformInfo element contains generic information on the captured platform, such as its name, which STANAG/ATDLP and edition it implements including any incorporated DLCPs and the overall sensitivity level of the SID.
- The ImplementationCodes element contains the transmit and receive codes for each message, word, data field and data value indicating the level of support by the platform.
- The DataElementDictionary element contains all data elements and values as required by the SID.

Note that the DataElementDictionary structure is the one as defined in Chapter H. An SID includes its own version of the DED as it applies to the implementation of the platform which can differ from a specific edition of the STANAG/ATDLP.

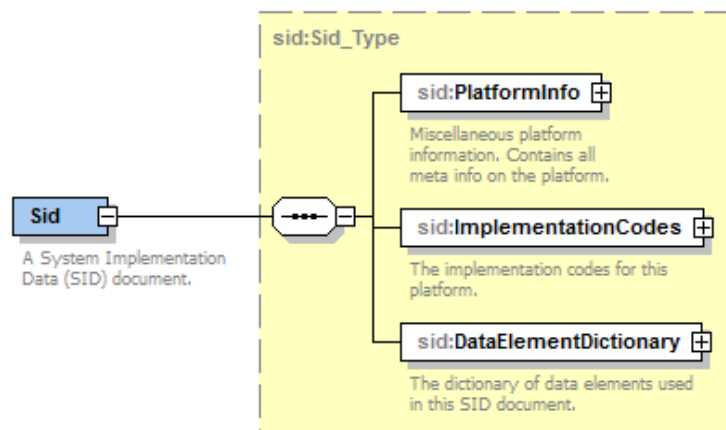


Figure 15-3 Top level structure of SID Schema

An SID in XML will contain a lot of information on the implementation of the platform which often is a copy of what is defined in the STANAG/ATDLP and in fact, can be derived from the XML representation of the STANAG/ATDLP. Implementers are encouraged to provide automated means to compare the platform's SID in XML against the indicated STANAG/ATDLP edition (plus DLCs).

15.6.2 The PlatformInfo element

The overall structure of the PlatformInfo element is shown in Figure 15-4 below with some of its detailed elements hidden.

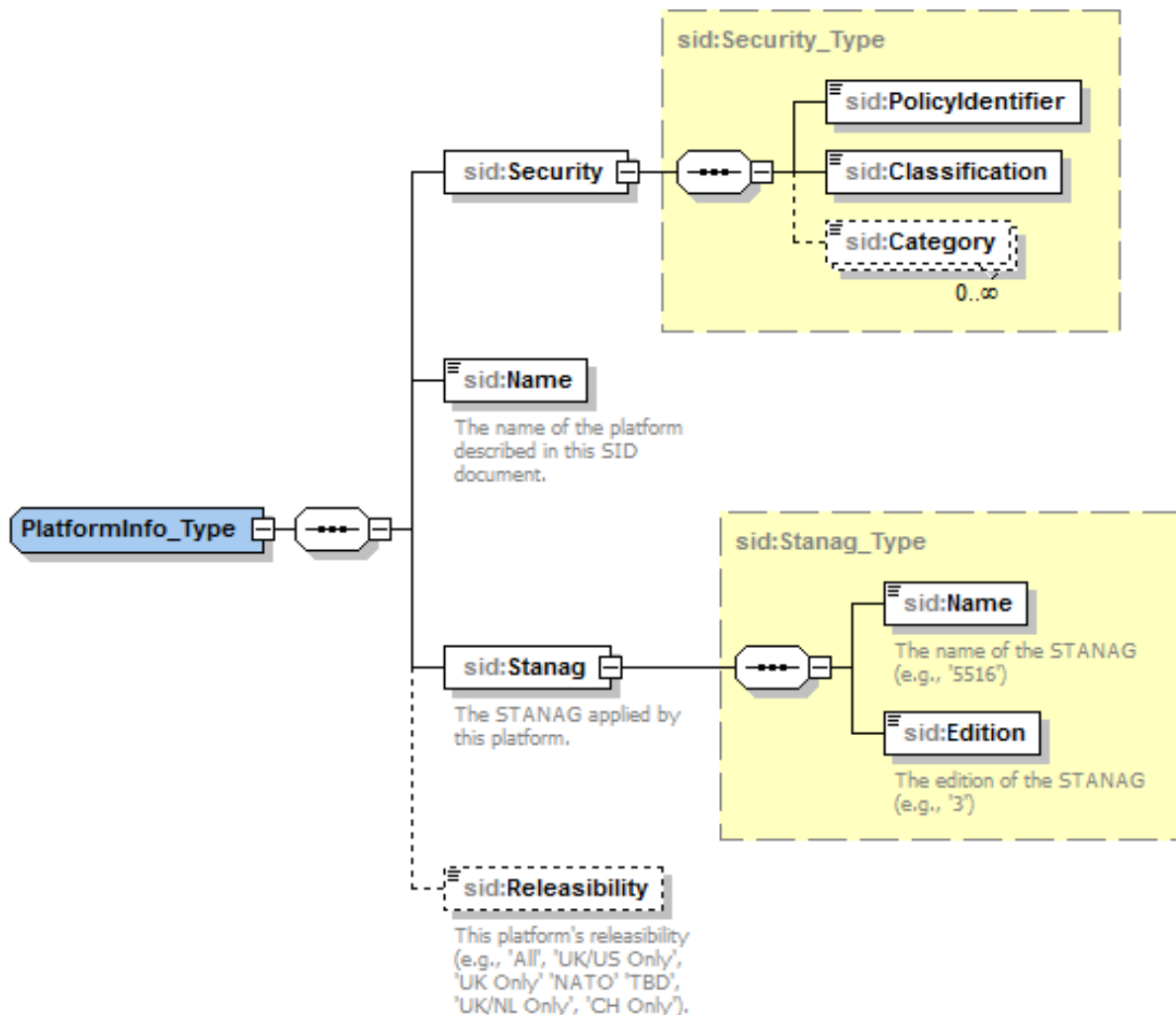


Figure 15-4 SID PlatformInfo element

An example of the PlatformInfo element is shown below.

```
<sid:Sid xmlns:sid="urn:nato:tdl:generic:systemImplementationData:0:20160127:final">
  <sid:PlatformInfo>
    <sid:Security>
      <sid:PolicyIdentifier>NATO</sid:PolicyIdentifier>
      <sid:Classification>UNCLASSIFIED</sid:Classification>
    </sid:Security>
    <sid:Country>TBD</sid:Country>
    <sid:Name>DemoPlatformLink16</sid:Name>
    <sid:SoftwareVersion>1.0</sid:SoftwareVersion>
    <sid:Service>Air Force</sid:Service>
    <sid:EnvironmentCategory>AIR</sid:EnvironmentCategory>
    <sid:Date>2014-01-01</sid:Date>
    <sid:JuType>C2</sid:JuType>
    <sid:Stanag>
      <sid:Name>STANAG 5516</sid:Name>
      <sid:Edition>edition 6</sid:Edition>
    </sid:Stanag>
    <sid:Dlcps>
      <sid:Dlcp>SL-123-5516-XYZ-M99-R5</sid:Dlcp>
      <sid:Dlcp>ML-999-5516-ACO-M88-R1</sid:Dlcp>
    </sid:Dlcps>
    <sid:C2Mode>C2</sid:C2Mode>
  </sid:PlatformInfo>
  <!-- ... -->
</sid:Sid>
```

15.6.3 SID Classification Markings

An SID captures specifics of a platform and therefore normally has a specific classification, which is captured as the overall classification in the “**PlatformInfo**” element.

The overall classification of the SID may be relaxed in certain sections. This can be modelled in the XML representation. When overriding the classification, only relaxations over higher-level elements are allowed (though this restriction is not enforced automatically yet), and a relaxation applies to the element itself and all its children (unless the classification is further relaxed).

The example listed below shows an SID that is classified NATO CONF*ENTIAL at top-level; the security constraints for the J3.0 transmit and receive codes are, however, considered NATO RESTR*CTED. Message J3.1 does not override the classification. For that reason, the transmit/receive codes for this message are considered NATO CONF*ENTIAL.

```
<sid:Sid xmlns:sid="urn:nato:tdl:generic:systemImplementationData:0:20160127:final">
  <sid:PlatformInfo>
    <sid:Security>
      <sid:PolicyIdentifier>NATO</sid:PolicyIdentifier>
      <sid:Classification>CONF*ENTIAL</sid:Classification>
    </sid:Security>
    <sid:Name>Example SID</sid:Name>
    <sid:Stanag>
      <sid:Name>STANAG 5516</sid:Name>
      <sid:Edition>6</sid:Edition>
    </sid:Stanag>
    <!-- ... -->
  </sid:PlatformInfo>
  <sid:ImplementationCodes>
    <sid:Message name="J3.0" ...>
      <sid:Security>
        <sid:PolicyIdentifier>NATO</sid:PolicyIdentifier>
        <sid:Classification>RESTR*CTED</sid:Classification>
      </sid:Security>
      <!-- ... -->
    </sid:Message>
    <sid:Message name="J3.1" ...>
      <!-- ... -->
    </sid:Message>
  </sid:ImplementationCodes>
  <sid:DataElementDictionary>
    <!-- ... -->
  </sid:DataElementDictionary>
</sid:Sid>
```

The Classification Marking in the SID re-uses the same XML type definitions as in the BaselineInfo (see 7.1.2.1) as also used in the Data Element Dictionary and Message Structure.

15.6.4 The SID ImplementationCodes and Message element

The ImplementationCodes element contains the list of all the Messages, as defined for the implemented STANAG/ATDLP, specifying for each Message the level of support. It follows the same structure as the MessageCatalogue element within the MessageStructure as defined in 9.3.2. The structure of the ImplementationCodes element is shown in Figure 15-5 below. Each Message has a transmit and a receive attribute to hold the NICR T/1 compliant codes to indicate the level of support.

Note that for each Message, the edition of the implemented STANAG/ATDLP and the classification can be overridden.

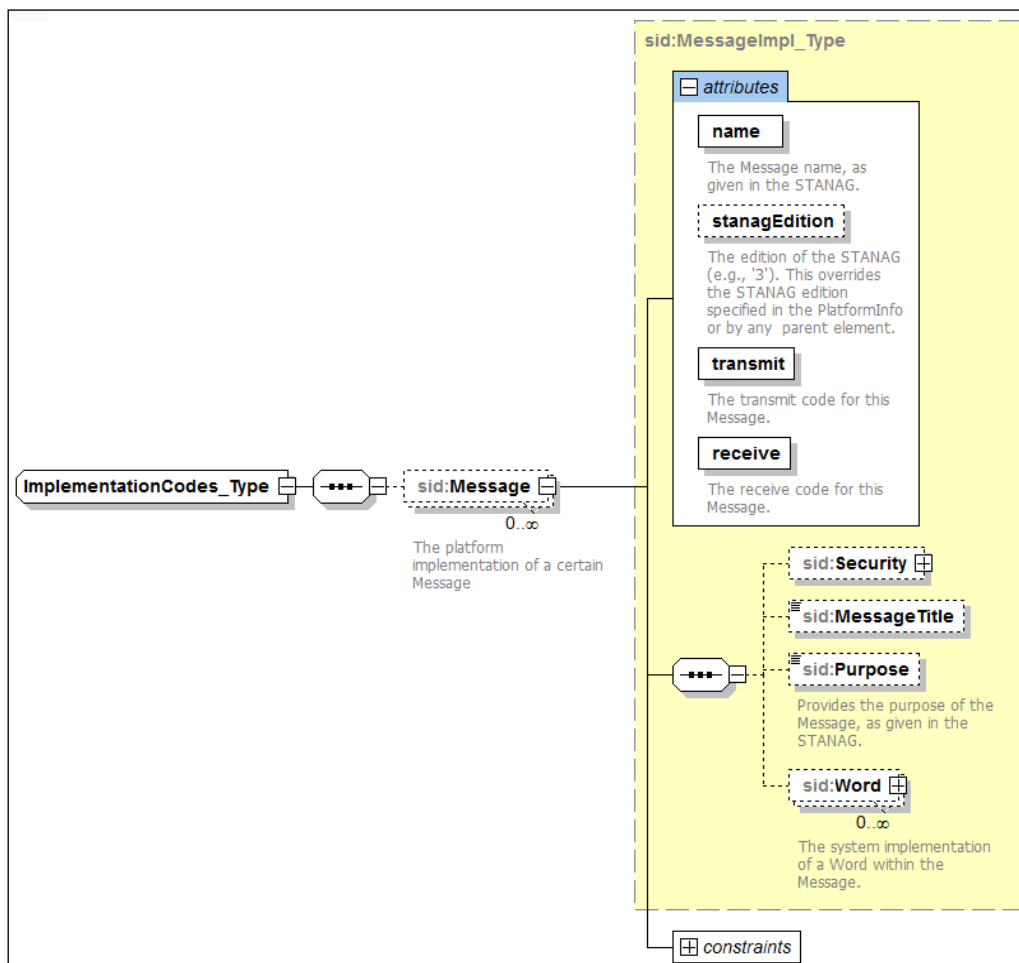


Figure 15-5 SID ImplementationCodes and Message element

An example of the ImplementationCodes element is shown below with a number of Messages and their enclosed Words. Note that some of the elements, like MessageTitle and Purpose, are optional and are typically omitted unless they differ from what is defined in the STANAG/ATDLP (an uncommon event). The example also shows how the J3.1 indicates that its structure is based on ed7 instead of the one defined in the PlatformInfo element.

```
<sid:ImplementationCodes>
  <sid:Message name="J3.0" receive="R" transmit="T">
    <sid:MessageTitle>Reference Point</sid:MessageTitle>
    <sid:Word name="J3.0I" receive="R" transmit="T">
      <sid:WordTitle>REFERENCE POINT INITIAL WORD</sid:WordTitle>
      <!-- ---->
    </sid:Word>
    <sid:Word name="J3.0E0" receive="R" transmit="T">
      <!-- ---->
    </sid:Word>
    <!-- ---->
  </sid:Message>
  <sid:Message name="J3.1" receive="R" transmit="T" stanagEdition="ed7">
    <sid:Word name="J3.1I" receive="R" transmit="T">
      <!-- ---->
    </sid:Word>
    <sid:Word name="J3.1E0" receive="R" transmit="T">
      <!-- ---->
    </sid:Word>
  </sid:Message>
  <sid:Message name="J3.2" receive="R" transmit="T">
    <sid:Word name="J3.2I" receive="R" transmit="T">
      <!-- ---->
    </sid:Word>
    <!-- ---->
  </sid:Message>
</sid:ImplementationCodes>
```

15.6.5 The SID Word element

The structure of the Word element is shown in Figure 15-6 below. It basically follows the same structure as the one in the MessageStructure as a container for the enclosed DataFields, but metadata elements like WordTitle, etc., are optional. The StructureSwitch (see section 15.4.1) allows the same support for overlaid DataFields as in the MessageStructure, but now to provide the transmit and receive support for the overlaid DataFields. Also the ImplementationSwitch can be used (as described in 15.4.2).

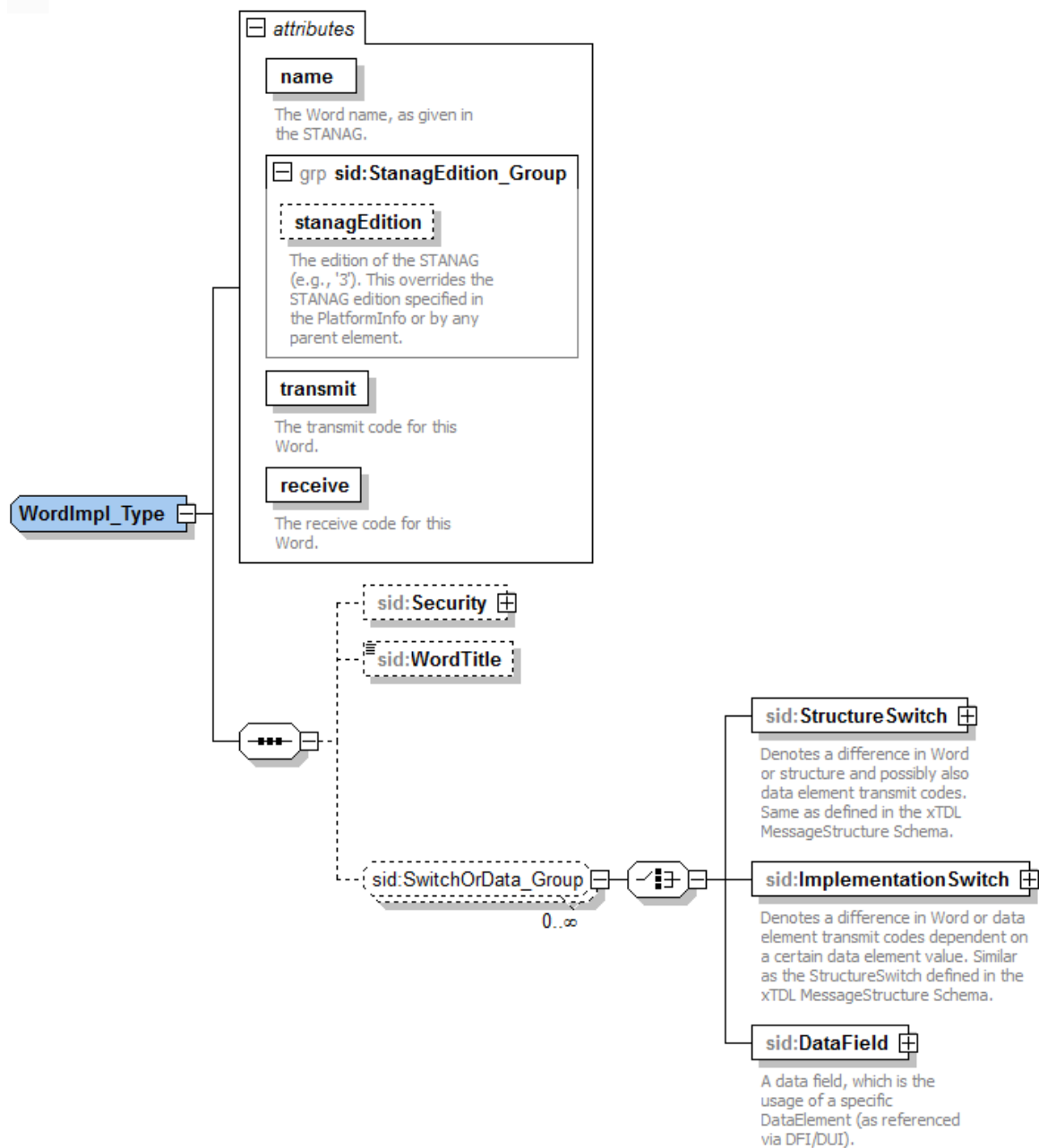


Figure 15-6 SID Word element

An example of the Word element is shown below with a number of enclosed DataFields, each with their own transmit and receive code.

As the level of support for the Data Items can vary among the Messages, the Data Items (via the Enum element) are inlined in the Word (contrary to the MessageStructure schema). For example, this allows a Data Element to be transmitted in the context of one Message, while in another

Message the Data Element is not supported, or only a specific number of Data Items.

Note that only the BitCode (and BitCodeRange) element is included while the actual Data Item (i.e., the meaning) is the same over all uses and therefore stored in the DataElementDictionary included in the SID (see 15.6.6).

```

<sid:Word name="J3.0" receive="R" transmit="T">
  <!-- ...-->
  <sid:DataField dfi="385" dui="3" startbit="13" receive="R" transmit="T">
    <sid:Enum receive="R" transmit="T">
      <sid:BitCode>0</sid:BitCode>
    </sid:Enum>
    <sid:Enum receive="DM" transmit="NT">
      <sid:BitCode>1</sid:BitCode>
    </sid:Enum>
  </sid:DataField>
  <!-- ...-->
  <sid:DataField dfi="354" dui="2" startbit="15" receive="R" transmit="T">
    <sid:Enum receive="R" transmit="T">
      <sid:BitCode>0</sid:BitCode>
    </sid:Enum>
    <sid:Enum receive="R" transmit="T">
      <sid:BitCode>1</sid:BitCode>
    </sid:Enum>
  </sid:DataField>
  <sid:DataField dfi="756" dui="1" startbit="16" receive="R0"
transmit="T0">
    <sid:Enum receive="R" transmit="T">
      <sid:BitCode>0</sid:BitCode>
    </sid:Enum>
    <sid:Enum receive="R" transmit="NT">
      <sid:BitCode>1</sid:BitCode>
    </sid:Enum>
  </sid:DataField>
  <!-- ... -->
</sid:Word>

```

15.6.6 The SID DataElementDictionary with DFI and DUI elements

Each SID contains its own DataElementDictionary, which is specific to the platform, allowing the capturing of differences between what a platform has implemented and the STANAG/ATDLP edition (plus DLCPs). Obviously, the number of differences should be kept to a minimum and automatic means can be applied to compare the DataElementDictionary against the one from the implemented edition of the ATDLP. The structure of the SID's DataElementDictionary is provided in Chapter 8.

The DataElementDictionary element contains the definition of the Data Elements that are referenced, via the DFI and DUI, in the ImplementationCodes element. The reference to a Data Element in the ImplementationCodes element represents the Data Field and reflects the platform's support for it in a specific context, i.e., usage within a Word.

Each Data Element may be used as a Data Field more than once in the ImplementationCodes element, and platform support may differ among the occurrences.

Note that the DataElementDictionary itself does not contain any transmit or receive codes: these are all specific to the context of the Data Element's use within a message and not to the definition of the Data Elements.

15.7 Rules enforcement

XML Schemas can only enforce a limited set of constraints on an XML instance document's structure and content. Specifically, the NICR T/1 hierarchical dependencies of the implementation codes cannot be effectively enforced using an XML Schema. To be able to express (and automatically check) these constraints, the SID XML Schema uses Schematron rules.

Schematron – “a rule-based validation language for making assertions about the presence or absence of patterns in XML” - is defined in a publicly available ISO standard (ISO/IEC 19757-3:2006 Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron). It allows for more flexible constraints on an XML document's structure and content.

The SID XML Schema embeds the Schematron rules for checking the NICR T/1 hierarchical dependencies in an appinfo annotation at the top of the schema. These rules can be extracted using an XSD-to-Schematron converter and subsequently executed.

15.8 SID XML Instance Verification

The process of verifying an SID XML file shall follow the following steps:

- (1) Validate against the SID XML Schema.
- (2) Run the Schematron rules to check the correctness of the hierarchical implementation code dependencies.
- (3) Check the consistency between the SID XML and the STANAG/ATDLP XML (several, if the implementation uses multiple editions) by means of a yet-to-be-implemented process, as also depicted in Figure 5-2 in chapter 5.

All of these steps can be fully automated.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

ATDLP-7.04

INTENTIONALLY BLANK

16 ANNEXES

Annex A	Specifics about the representation of Link 1 in XML
Annex B	Specifics about the representation of Link 11/11B in XML
Annex C	Specifics about the representation of Link 16 in XML
Annex D	Specifics about the representation of Link 22 in XML
Annex E	Data Forwarding in XML
Annex F	Example of an xTDL Message Instance
Annex G	Overview of the TDL Governance Namespace NMRR Folder Structure
Annex H	Overview of Authoritative xTDL and related XML schemas
Annex I	Description of the UML Model Symbolology used in the xTDL UML Models

INTENTIONALLY BLANK

A. SPECIFICS ABOUT THE REPRESENTATION OF LINK 1 IN XML

[tbd.]

INTENTIONALLY BLANK

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX B TO
ATDLP-7.04**

B. SPECIFICS ABOUT THE REPRESENTATION OF LINK 11/11B IN XML

[tbd.]

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX B TO
ATDLP-7.04**

INTENTIONALLY BLANK

C. SPECIFICS ABOUT THE REPRESENTATION OF LINK 16 IN XML

C.1 General

The generic conceptual models and schemas present in the main body of the ATDLP-7.04 are applicable to ATDLP-5.16 editions.

C.2 xTDL Data Element Dictionary

The generic DED schema captures the specification of the Data Elements of Link 16 with the following remark(s):

- Link 16 employs the concepts of DFI and DUI which map directly on the concepts in the schema.

C.3 xTDL Message Structure

The generic Message Structure schema captures the structure of Link 16 messages with the following remark(s):

- Link 16 contains Messages, encompassing one or more Words where each Word contains one or more Data Fields. Each Message contains one Initial Word, zero or more Extension Words and zero or more Continuation Words.

C.4 xTDL Processing

The TDL Processing Rules capture the processing of Link 16 with the following remark(s):

- The rules for ATDLP-5.16 editions are developed in transactional format (see Section 10.1).
- The occurrence and order of the required words in the message are defined by the Transmit and Receive Tables contained in the Link 16 Standard.

C.5 Data Forwarding

The conceptual model in Chapter 11 and corresponding XML representation complies with ATDLP-6.16 editions.

C.6 Business Rules

The Link 16 XML representation will comply with the business rules expressed in Chapter 12, once represented in XML.

C.7 Implementation Requirements (IMP REQ)

The conceptual model in Chapter 13 and corresponding XML representation, once developed will comply with the Link 16 IMP REQ.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX D TO
ATDLP-7.04**

D. SPECIFICS ABOUT THE REPRESENTATION OF LINK 22 IN XML

[tbd.]

INTENTIONALLY BLANK

E. DATA FORWARDING IN XML

[tbd.]

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX E TO
ATDLP-7.04**

INTENTIONALLY BLANK

F. EXAMPLE OF AN XTDL MESSAGE INSTANCE

[tbd.]

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX F TO
ATDLP-7.04**

INTENTIONALLY BLANK

NATO UNCLASSIFIED

**G. OVERVIEW OF THE TDL GOVERNANCE NAMESPACE NMRR
FOLDER STRUCTURE**

**G.1 Overview of the TDL Governance Namespace NMRR Folder
Structure**

In this Annex the folder structure for the TDL Governance Namespace is described as assessed suitable for the registration of xTDL components in the NATO Metadata Registry and Repository (NMRR).

TDL Governance Namespace NMRR Folder Structure

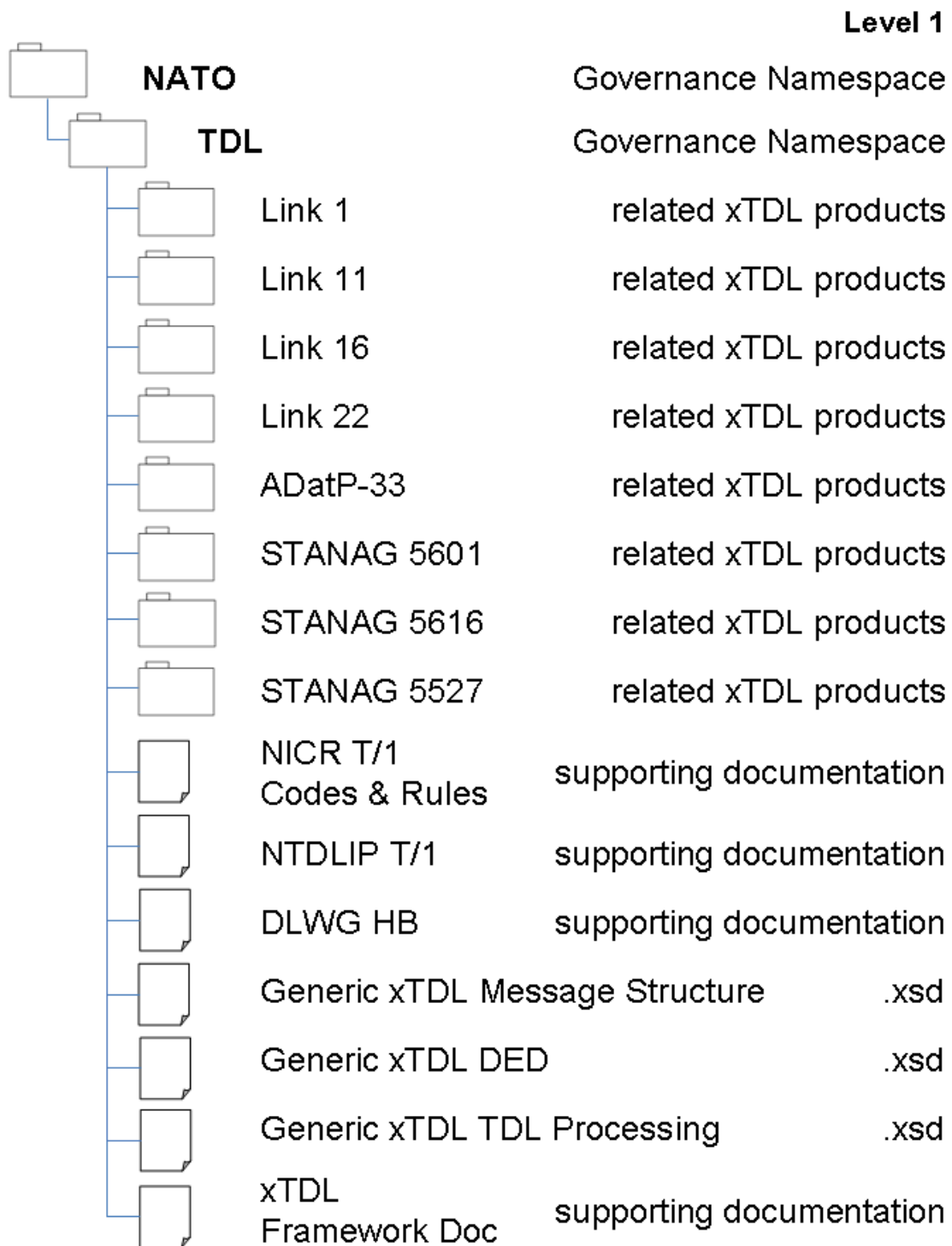
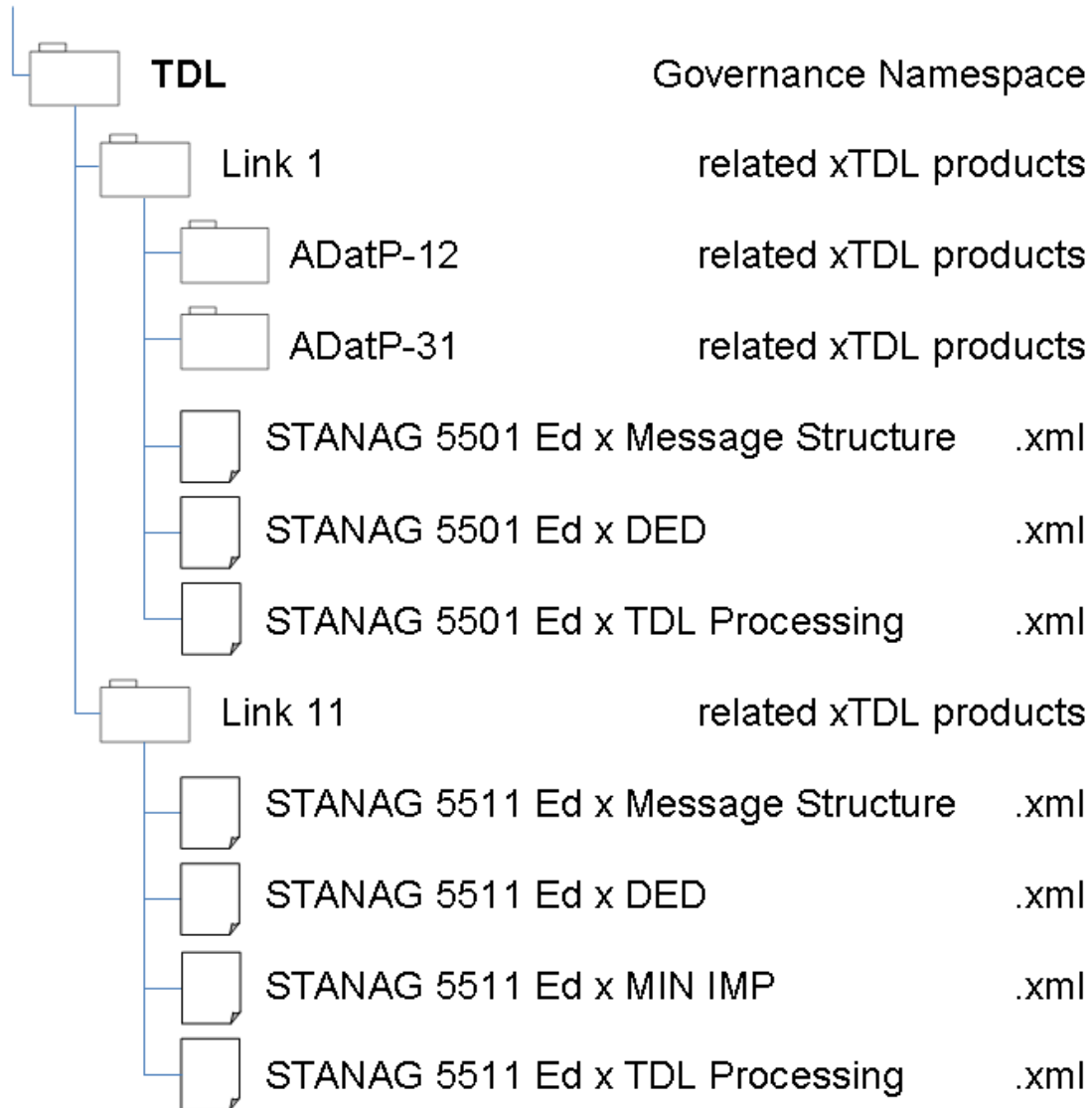


Figure G-1: TDL Governance Namespace NMRR Folder Structure

TDL Governance Namespace NMRR Folder Structure

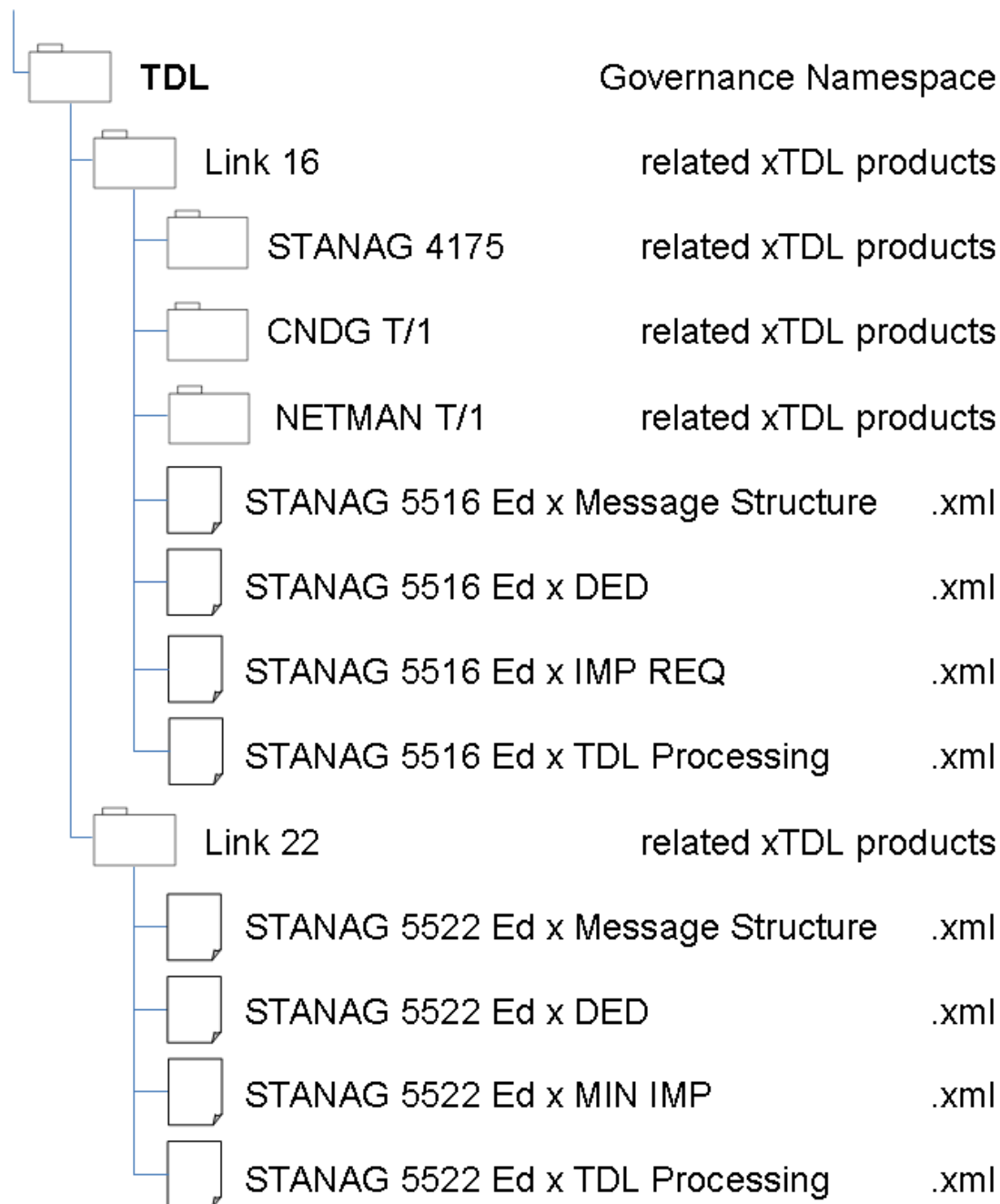
Level 2



TDL Governance Namespace NMRR Folder Structure

Level 2

(contd.)



TDL Governance Namespace NMRR Folder Structure

Level 2
(contd.)

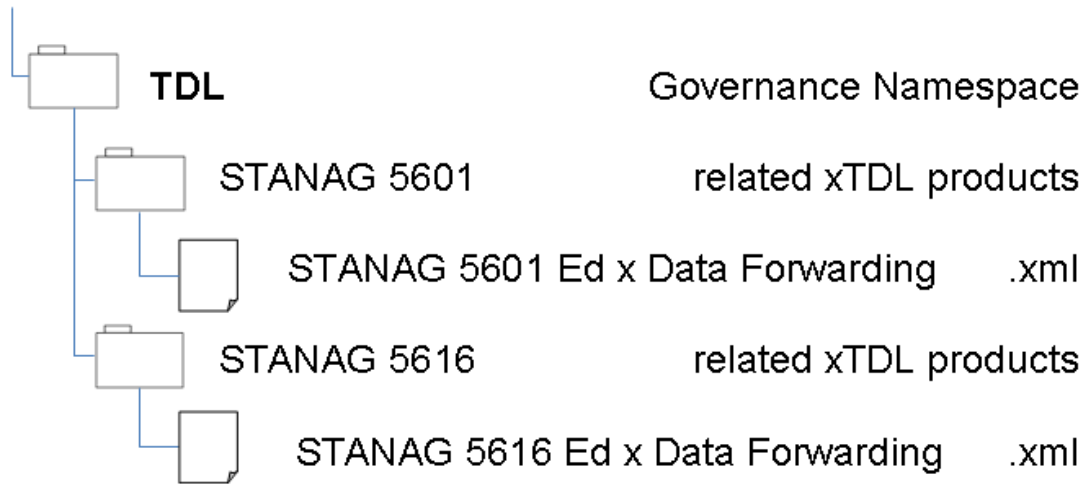


Figure G-2: TDL Governance Namespace NMRR Folder Structure

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX G TO
ATDLP-7.04**

INTENTIONALLY BLANK

H. OVERVIEW OF AUTHORITATIVE XTDL AND RELATED XML SCHEMAS

H.1 List of Authoritative xTDL Schemas

For the latest versions of XML schemas listed below refer to the TDL Governance Namespace within the NMRR^{note1}.

Schema Name	“URN”; Description
Generic xTDL Message Structure	“urn:nato:tdl:generic:messageStructure:1:20140301:draft”
Generic xTDL Data Element Dictionary	“urn:nato:tdl:generic:dataElementDictionary:1:20140301:draft”
Generic xTDL System Implementation Data	“urn:nato:tdl:generic:systemImplementationData:0:20160127:final”

Table H-1: List of Authoritative xTDL Schemas

H.2 List of related XML Schemas

For the latest versions of related XML schemas listed below refer to the NMRR^{note1}.

Schema Name	“URN”; Description
[tbd.]	[tbd.]

Table H-2: List of related XML Schemas

Note 1: The interim NATO Metadata Registry and Repository (NMRR) has been handed over to the NATO HQ C3 Staff (Ref. NHQC3S(DIR)0037-2015) on 4 June 2015. The intent is to continue provision of the NMRR service as an interim solution that can be used for production products until industrialisation under NATO Capability Package CP 9C0150 (Core Information Services for C2)/Project Mandate (PM) 94.

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland

**ANNEX H TO
ATDLP-7.04**

INTENTIONALLY BLANK

I. DESCRIPTION OF THE UML MODEL SYMBOLOGY USED WITHIN THE XTDL UML MODELS

I.1 UML Model Symbology used within the xTDL UML Models

The symbols depicted in all models are specified in UML. The following two figures provide a short description of the symbology used in the models:

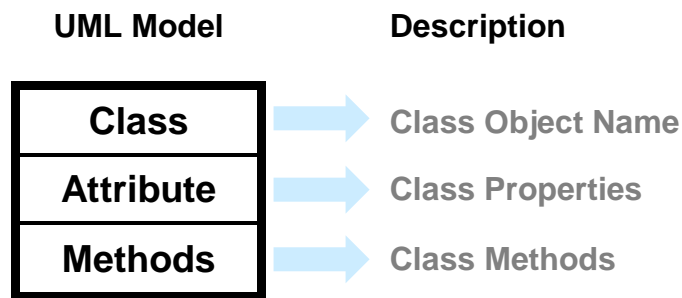


Figure I-1: UML Model Class Diagram Description


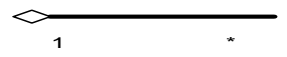

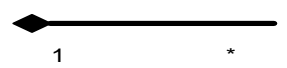
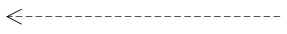
Symbol	Description
	<u>Association</u> Establishes an association relationship which may be named. Multiplicities must be assigned. If an arrow is present, it establishes the direction of the relationship.
	<u>Aggregation</u> Establishes a special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.
	<u>Generalization</u> Establishes a SuperClass – SubClass relationship. The subclass inherits all attributes and features. The arrow always points to the SuperClass.
	<u>Composition</u> Establishes a composite aggregation which is a strong form of aggregation that requires a part instance be included in at most one composite at a time. The diamond always points to the container.
	<u>Dependency</u> Establishes a dependency between classes. A keyword may be added to specify the dependency.
1..*	<u>Multiplicity</u> Defines the number of relationships between instances of connected classes.

Figure I-2: Description of the UML Model symbols used within the xTDL UML Models

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland
ATDLP-7.04

INTENTIONALLY BLANK

Edition A Version 1

NATO UNCLASSIFIED

NATO UNCLASSIFIED

Releasable to Australia, Austria, Finland, New Zealand, Sweden and Switzerland
ATDLP-7.04

ATDLP-7.04(A)(1)

Edition A Version 1

NATO UNCLASSIFIED