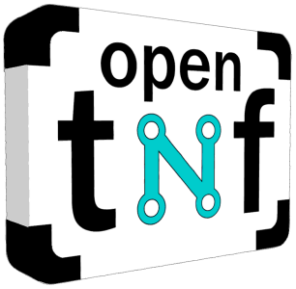
OpenTNF™

White Paper

Version 1.0



An open and efficient solution for exchange of Transport Network data.

Acknowledgement

Consistent high-quality specifications can only be created and maintained with the co-operation and dedication of a number of experts. The editors of this paper would like to thank all those who have donated the hours necessary to review, evaluate, discuss and comment on this paper.

We are especially grateful to the following distinguished persons for their important contributions:

Linda Therese Støeng, Statens vegvesen
John Mikalsen, Statens vegvesen
Knut Jetlund, Statens vegvesen
Eivind Stalheim, Vianova Systems AS
Johnny Jensen, Vianova Systems AS
Per Isaksson, Trafikverket
Leif Johansson, Trafikverket
Tomas Norlin, Trafikverket

Contents

1 Introduction 5

1.1 Common terms and abbreviations 6

1.2 Use of UML 6

1.3 References 9

1.4 Common assumptions and definitions 9

1.4.1 DateTime 9

2 Concepts and content 11

2.1 Identities and version 11

3 Logical data format specification 13

3.1 Identifiers 13

3.2 Model for a transport network 13

3.2.1 Overview 13

3.2.2 TNF_LINK_SEQUENCE (OPTIONAL) 15

3.2.3 TNF_LINK 16

3.2.4 TNF_NODE (OPTIONAL) 19

3.2.5 TNF_NETWORK (OPTIONAL) 20

3.2.6 TNF_NETWORK_CONNECTION (OPTIONAL) 21

3.2.7 TNF_NETWORK_ELEMENT_IN_CONNECTION (OPTIONAL) 22

3.2.8 <<union>>TNF_NetworkElementRef 23

3.2.9 <<union>>TNF_ LinearElementRef 23

3.3 Generic model for attribution and features related to the transport network 23

3.3.1 Overview 23

3.3.2 TNF_PROPERTY_OBJECT 25

3.3.3 TNF_PROPERTY 26

3.3.4 TNF_NETWORK_REFERENCE 29

3.3.5 TNF_DIRECT_LOCATION_REFERENCE (OPTIONAL) 38

3.4 Generic model for transport property object catalogues 40

3.4.1 Overview 40

3.4.2 TNF_CATALOGUE 42

3.4.3 TNF_PROPERTY_OBJECT_TYPE 42

3.4.4 TNF_PROPERTY_OBJECT_TYPE_VALID_FOR_TYPE_OF_TRANSPORT (OPTIONAL) 45

3.4.5 TNF_PROPERTY_OBJECT_PROPERTY_TYPE 46

3.4.6 TNF_VALUE_DOMAIN 47

3.4.7 TNF_STRUCTURED_VALUE_DOMAIN_PROPERTY_TYPE 50
3.4.8 TNF_VALID_VALUE 51
3.4.9 TNF_ SECONDARY_LRS 52
3.4.10 TNF_SECONDARY_LRS_IDENTITY 54
3.4.11 Examples 55
3.5 Model for metadata 58
3.5.1 TNF_METADATA 58
3.6 Model for update transactions 59
3.6.1 TNF_CHANGE_TRANSACTION (CONDITIONAL) 60
3.6.2 TNF_CHANGE (CONDITIONAL) 61
4 SUPPLEMENT 1 – Specializations for Swedish/Norwegian national road databases 64
4.1 Model for transport network 64
4.1.1 TNF_LINK_SEQUENCE (OPTIONAL) 65
4.1.2 TNF_LINK 65
4.1.3 TNF_NODE (OPTIONAL) 68
4.1.4 TNF_CONNECTION_PORT (OPTIONAL) 68
4.1.5 TNF_TOPOLOGY_LEVEL (OPTIONAL) 69
4.2 Generic model for attribution and features related to the road network 70
4.2.1 TNF_NETWORK_REFERENCE 70
4.3 Generic model for transport property object catalogues 71
4.3.1 TNF_PROPERTY_OBJECT_TYPE 72

1 Introduction

This document describes and specifies OpenTNF™ - Open Transport Network Format, an open specification for interoperability and exchange of data on Transport Networks. The format enables users to read and write OpenTNF™ data files using either self-written or Open Source software. The OpenTNF™ format should be seen as an open complement to international standards such as INSPIRE DS TN, GDF and ISO 19100 and national standards such as SOSI and SS63700x.

When appropriate The OpenTNF™ specification has borrowed model concepts and terms from the *Generic network model* in "INSPIRE Generic Conceptual Model, Version 3.1" [INSPIRE GCM] and "INSPIRE Data Specification on Transport Networks" [INSPIRE DS TN].

The mission for OpenTNF™ is

> "*Create and maintain, as a community, an international, open and efficient solution for exchange of Transport Network data. The solution will enable easier usage of data on multiple platforms.*"

The OpenTNF™ format is specified as a set of classes that in principal may be implemented as data tables in any relational database engine. An OpenTNF™ dataset may contain the following types of data:

- Transport network including the definition of linear referencing, geometric, topological and temporal aspects.
- Attribution and features, here called transport properties, related to the geography and/or transport network using linear referencing mechanisms. OpenTNF™ uses a generic approach where all types use the same schema where the types are defined in a separate catalogue data structure.
- A transport property type catalogue defining the various transport property types. This catalogue may also contain definitions of secondary linear referencing systems.
- Metadata, also according to any XML compliant standard such as ISO 19139.
- Update (transaction) information for the case that the dataset represents a set of updates

The technical goal of the OpenTNF™ is to specify an open standard for exchange of:

- Potentially large amounts of transport network data in an efficient and standardized way.
- Datasets according to "INSPIRE Data Specification on Transport Networks" [INSPIRE DS TN]
- Datasets to and from the National Road databases.

Usage of this format instead of more or less system dependent implementations of "binary" exchange formats makes systems integration and development easier over time since systems can evolve more independent of the exchange-format.

Compared to XML- and GML based data representations, OpenTNF™ shall be viewed as a complement, for example when an efficient exchange of large amounts of data is required.

This format is specified as a set of classes that in principal may be implemented as data tables in any relational database engine.

A reference implementation using OGC GeoPackage (www.geopackage.org) and SQLite (www.sqlite.org) are available at www.opentnf.org. OGC GeoPackage and SQLite are choosen because they are open, free and allow for a single file for storage and exchange. Furthermore, SQLite does not imply any limitations in dataset size (at least not in practice) which makes it suitable for very large datasets.

The goal of OpenTNF™ is wide-scale adoption by the industry at large. OpenTNF™ is therefore proposed as an open standard in an Open Source framework. It shall be usable for anyone dealing with exchange of transport network data.

While OpenTNF™ initially was developed and maintained by Triona AB (www.triona.se), everyone is invited to contribute to its further development and maintenance. We encourage you to read, review, evaluate, use, discuss and comment this initiative and document.

This document describes the basic concepts and content of the OpenTNF™ standard and the logical data format.

1.1 Common terms and abbreviations

The following tables explain common terms and abbreviations used in this document and in the context of data exchange on Transport Networks.

Term	Description
LRS	Linear referencing system
Linear referencing system	set of **Linear Referencing Methods** and the policies, records and procedures for implementing them [ISO 19148]
Linear referencing method	manner in which measurements are made along (and optionally laterally offset from) a linear element [ISO 19148]
Linear element	1-dimensional object that serves as the axis along which linear referencing is performed NOTE Also known as curvilinear element. [ISO 19148]

Table 1 Explanation of common terms and abbreviations

1.2 Use of UML

Throughout the document, the concepts/types are illustrated using UML. Since the document focuses on exchange of data, the simplest possible notation is used. Each type is modelled in such a way that it easily may be represented in a relational database table. The table below explains the principles for using UML in this document.

Concept	Relational concept	UML	Comment
Type	Table or set of columns	Class	Each type/table is modelled as a UML class. Stereotype <<optional>> is used to specify that the class describes an optional type/table. Stereotype <<datatype>> indicates that the class defines a “pattern” of attributes that are reused in other class definitions. There will never exist any separate instances of such a type. Sterotype <<union>> for a class indicates a set of columns/attributes where each row/instance shall use only one column/attribute.
Property of a type	Column	Attribute	Stereotypes <<optional>> and <<conditional> are used to specify that the attribute describes optional or conditional properties/columns. Default (no stereotype) means mandatory.
Relation between types/tables	Column/columns as primary key in target table and column/columns as foreign key in source table	Association/composition	An ordinary association by-reference is modelled as a UML association. The role name and direction of the association indicates which type/table that shall instantiate the association and the name of the property/column that contains the foreign key.

			A composition by-value is modelled as a UML composition. This means that lifetime shall be shared between the parent and its children. The role name and direction of the composition indicates which type/table that shall instantiate the association and the name of the property/column that contains the foreign key. An association or composition may be declared optional or conditional by using stereotypes <<optional>> or <<conditional>> the same way as an attribute. Default (no stereotype) means mandatory.
Multiplicity restrictions for relations		Multiplicity	Multiplicity declarations for the association/composition roles specify restrictions regarding the number of instances that are required or allowed for each role of a relationship between classes.

Datatype	Pre-defined datatype	According to ISO 19103	The primitive data types used for all properties/columns are the primitive types defined in ISO/TS 19103 – Conceptual schema language. For any given database engine or other exchange format, each primitive type has to be mapped to the respective specific type system.
Geometry	Pre-defined datatype	A spatial property of a type is modelled as an attribute of type Geometry qualified by any restriction regarding type.	It is assumed that for each database engine or exchange format there is a native way to represent spatial characteristics, e.g. OGC Simple Feature Access – Part2: SQL option.

1.3 References

[INSPIRE GCM] INSPIRE Generic Conceptual Model, Version 3.1

[INSPIRE DS TN] INSPIRE Data Specification on Transport Networks – Guidelines, Version 3.0

[ISO 19103] ISO/TS 19103:2005, Geographic information -- Conceptual schema language

[ISO 19139] ISO/TS 19139:2007, Geographic information -- Metadata -- XML schema implementation

[ISO 19148] ISO/DIS 19148, Geographic information -- Linear referencing

[ISO 17572-3] ISO 17572-3:2008, Intelligent transport systems (ITS) -- Location referencing for geographic databases -- Part 3: Dynamic location references (dynamic profile)

1.4 Common assumptions and definitions

1.4.1 DateTime

The Gregorian Calendar shall be used as a reference system for date values, and the Universal Time Coordinated (UTC) or the local time including the time zone as an offset from UTC shall be used as a reference system for time values.

2 Concepts and content

The OpenTNF™ format is specified as a set of classes that in principal may be implemented as data tables in any relational database engine.

To align to this white paper an implementation shall implement all mandatory and conditional elements as specified. Optional elements may be implemented and if they are implemented they shall be implemented as specified. Finally, an implementation may add additional elements and extend this specification provided that all mandatory, conditional and optional elements are not affected. This means that implementations are free to add classes (tables) or attributes (columns).

To make things as simple as possible an OpenTNF™ dataset may contain only one primary linear referencing system (LRS). Zero or many secondary linear referencing system can be defined using a set of mechanisms that facilitate the use of transport properties. These mechanisms are described later in this paper.

An OpenTNF™ dataset can for instance be mapped to datasets according to INSPIRE DS TN, the Swedish NVDB-systems and Norwegian NVDB-systems. It can also be mapped to the formal Swedish and ISO standards.

2.1 Identities and version

The specification supports the exchange of complete datasets as well as incremental updates.

Exchanging complete datasets means that it is the complete state of a source dataset that is exchanged. It is of course possible to exchange only parts of a complete dataset according to agreements regarding spatial, temporal and thematic restrictions or perhaps restrictions regarding access rights.

Exchanging incremental updates assumes that at some point in time, an initial and complete (in some sense) dataset has been exchanged. From that point in time, only the data that is updated in the source dataset is exchanged. The types used to specify the incremental update primitives are specified in the chapter *Model for update transactions*.

To support the exchange of incremental updates each instance of a type is required to have a globally unique id. This id is the only way to securely identify a certain instance.

Each update of an instance results in a new version of that instance, whether or not the update was made due to real world changes or in order to represent the real world in a more correct way.

Optionally, to enable identification of a specific version of an instance, a version id is included in the specification. The version id is also globally unique, i.e. an instance updated independently at different sources shall not have the same version id. The version id is intended to enable a more secure handling of conflicts in incremental updates:

- Enable detection of errors in the chain of updates concerning a single instance
    a. Each update may contain a reference to the previous version of the instance
- Enable detection of multiple independent updates
    a. Same mechanism as the bullet point above

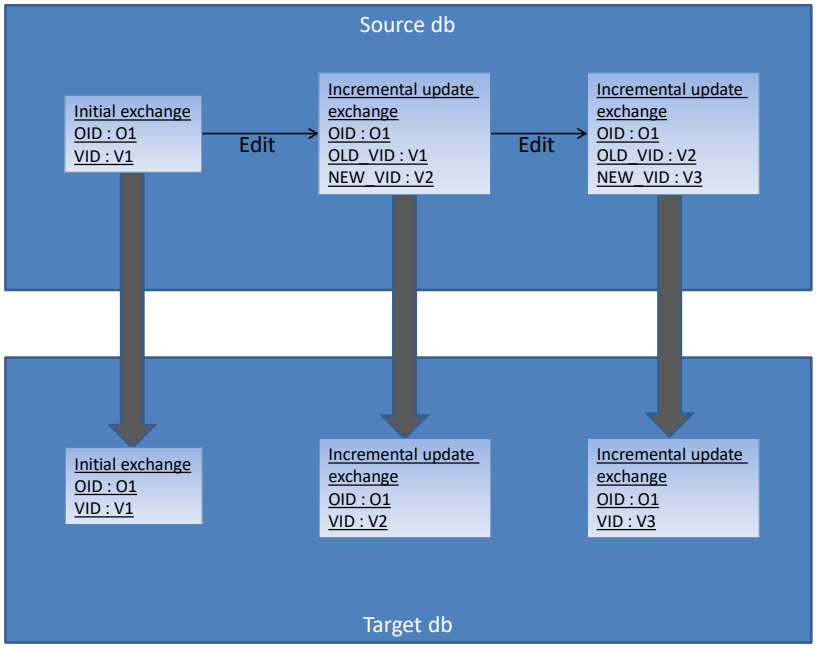


Figure 1 - Exchange of incremental updates

3 Logical data format specification

3.1 Identifiers

In previous draft versions of this specification, the datatype TNF_Identifier was defined where an identifier always consisted of two parts, NAMESPACE and LOCAL_ID according to the INSPIRE specifications. This identifier was used everywhere a globally unique identifier was needed to uniquely identify an object.

When testing this specification from a practical viewpoint, the suggested solution was considered too complex. The specification also posed a specific view on identification for the implementations, which is not always true or applicable. Therefore, the specification was simplified so that each identity instead is implemented as a CharacterString. In case a more complex scheme, such as INSPIRE [INSPIRE GCM], is required, we suggest that conventions are introduced for the use of CharacterString identities such as using a URI according to e.g. https://tools.ietf.org/html/rfc3986

3.2 Model for a transport network

3.2.1 Overview

As illustrated by Figure 3, many of the important mechanisms that are needed to properly describe transport networks are included in OpenTNF™:

- The relationships between Nodes, Links and Link Sequences
- The mechanism for cross-border and intermodal connections
- Collection of network elements in a transport networks

Linear referencing is a technique to position phenomena along a linear element using a measure from the beginning of the linear element. When linear referencing is used in OpenTNF, the position on linear elements shall be expressed as measures along the supplied geometry of the underlying linear object(s).

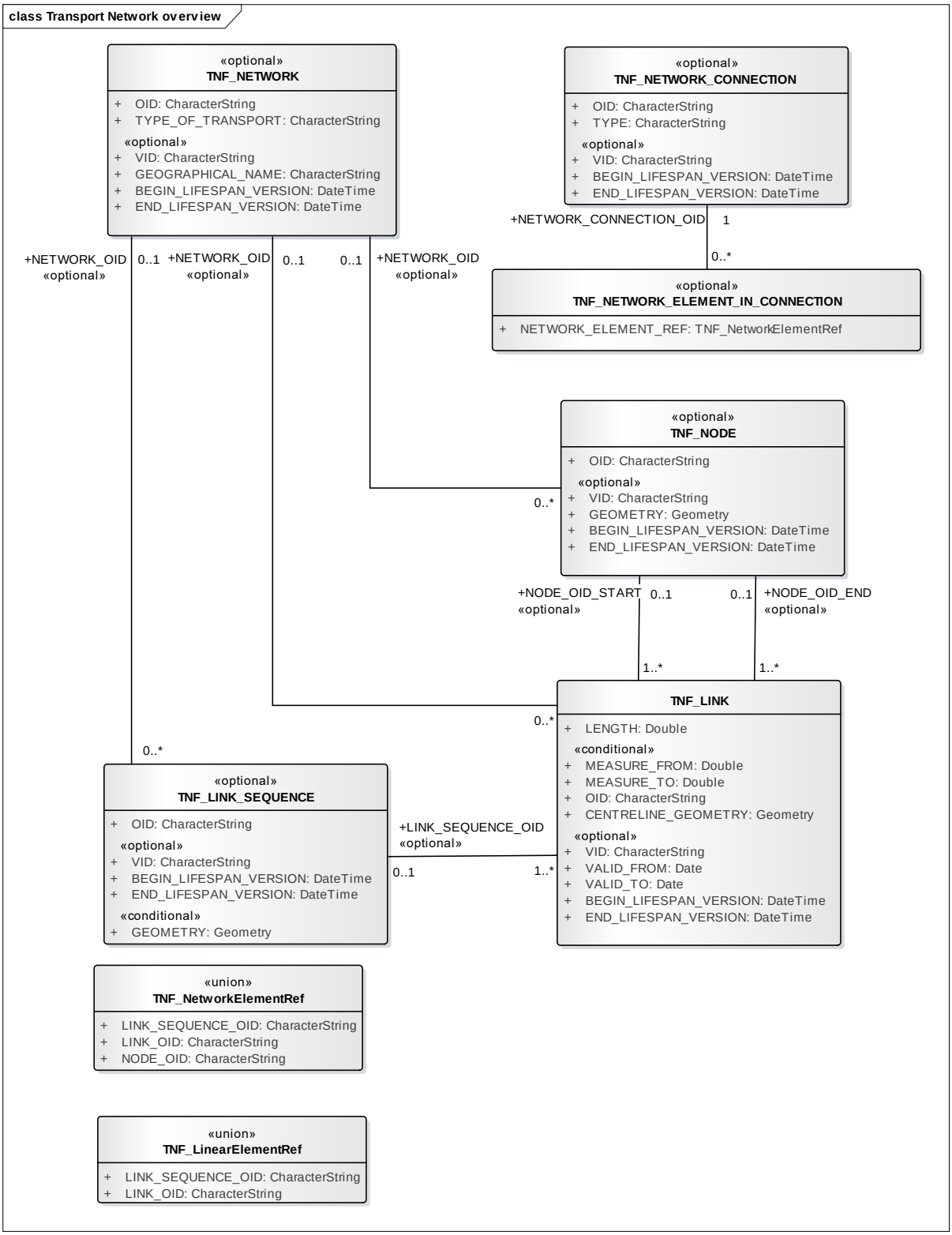


Figure 2 - Transport Network overview

3.2.2 TNF_LINK_SEQUENCE (OPTIONAL)

A linear spatial object composed of an ordered collection of transport links, which represents a continuous path in the transport network without any branches. The link sequence has a defined beginning and end.

This type represents a linear network element in the primary linear referencing system (LRS) for a network dataset.

REQUIREMENT 1: The direction of the transport links that constitute the sequence shall agree with the direction of the link sequence.

REQUIREMENT 2: Every position on the transport link sequence shall be identifiable with one single parameter such as length.

REQUIREMENT 3: The measures for a transport link belonging to a link sequence shall not overlap the measures in any of the other transport links in the same sequence.

REQUIREMENT 4: The ordered collection of links in a link sequence is defined by sorting MEASURE_FROM values in ascending order.

EXAMPLE 1: A continuous road section has been opened for traffic at a certain point in time. The geographical extent of this road section stays constant over time, even if connected road sections are added or parts of the road section are terminated. Representing this path in the network with a LINK_SEQUENCE is very stable over time and is therefore very suitable as a target for linear referencing.

Name	Type	Comment
OID Primary Key	CharacterString	Globally unique object id. *Note: This id is used during the entire lifespan of the object.*
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
NETWORK_OID Foreign Key (TNF_NETWORK)	CharacterString (OPTIONAL)	OID for the transport network
GEOMETRY	Geometry (Line) (CONDITIONAL)	The centreline geometry for this link sequence. The coordinates shall be at least 3 dimensional (x, y and z) and may include m-values as a fourth dimension. Coordinates shall be interpreted according to the

		coordinate reference system specified in metadata. *Note: A z value of -99999 indicates that z is unknown.* *Note: If m values are present the first coordinate shall have an m value of 0.0 and the last coordinate shall have an m value of 1.0.* **Condition:** Must exist if TNF_LINK does not have the CENTRELINE_GEOMETRY attribute.
BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.
END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

The figure below shows an example of a link sequence composed of four transport links.

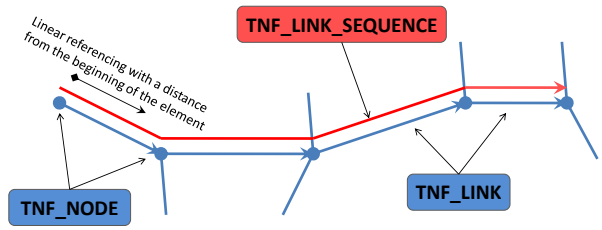


Figure 3 - A link sequence example

3.2.3 TNF_LINK

A link represents a linear spatial object that describes the geometry and connectivity of a transport network between two points in the transport network. It can also function as a linear element referenced from a TNF_LinearElementRef.

REQUIREMENT 1: For a link with measures the MEASURE_FROM shall be less than MEASURE_TO.

Name	Type	Comment

OID Primary Key	CharacterString (CONDITIONAL)	Globally unique object id. Shall be set when instances of TNF_LINK need to be uniquely identified, such as if they are referenced as linear elements in a linear referencing context. *Note: This id is used during the entire lifespan of the object.*
NETWORK_OID Foreign Key (TNF_NETWORK)	CharacterString (OPTIONAL)	OID for the transport network
LENGTH	Real (Double precision)	The agreed length (may be the geometric length) for this transport link. *Note: This attribute is mandatory primary to avoid misunderstandings regarding the agreed length. Even if the length of the geometry is used, there may be discrepancies between parties in how z values or missing z values are used.*
CENTRELINE_GEOMETRY	Geometry (Line) (CONDITIONAL)	The centreline geometry for this transport link. The coordinates shall be at least 3 dimensional (x, y and z) and may include m-values as a fourth dimension. Coordinates shall be interpreted according to the coordinate reference system specified in metadata. *Note: A z value of -99999 indicates that z is unknown.* *Note: If m values are present they shall correspond to from- and to-measure given for this transport link.* **Condition:** Must exist if TNF_LINK_SEQUENCE does not have the GEOMETRY attribute.
MEASURE_FROM	Real (Double precision) (CONDITIONAL)	Length measure value for the linear element at the start point for this transport link. The value shall be set if LINK_SEQUENCE_OID is not null or if the link is being referenced by a *TNF_LinearElementRef.LINK_OID*

MEASURE_TO	Real (Double precision) (CONDITIONAL)	Length measure value for the linear element at the end point for this transport link. The value shall be set if LINK_SEQUENCE _OID is not null or if the link is being referenced by a *TNF_LinearElementRef.LINK_OID*.
VID	CharacterString (OPTIONAL)	Globally unique version id. A VID shall only be set if the OID value is set for the same instance. *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
LINK_SEQUENCE_OID Foreign Key (TNF_LINK_SEQUENCE)	CharacterString (OPTIONAL)	OID for the linear element to which this transport link belongs.
VALID_FROM	Date (OPTIONAL)	The date from which this transport link is valid in the real world.
VALID_TO	Date (OPTIONAL)	The date from which this transport link is no longer valid in the real world. *Note: A null value indicates that the to date is unknown and that the object is valid if the current date >= VALID_FROM*
NODE_OID_START Foreign Key (TNF_NODE)	CharacterString (OPTIONAL)	OID for the start node *Note: This attribute is optional, since explicit topology is not mandatory in a lot of applications. However, it is strongly recommended that the use of explicit topology is used consequently in a dataset.*
NODE_OID_END Foreign Key (TNF_NODE)	CharacterString (OPTIONAL)	OID for the end node *Note: This attribute is optional, since explicit topology is not mandatory in a lot of applications. However, it is strongly recommended that the use of explicit topology is used consequently in a dataset.*

BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.
END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

3.2.4 TNF_NODE (OPTIONAL)

A node represents a point spatial object which is used for connectivity. Nodes are found at either end of the transport link.

The use of nodes is optional, but it is strongly recommended that the use of explicit topology is consequent in a dataset.

Name	Type	Comment
OID Primary Key	CharacterString	Globally unique object id. *Note: This id is used during the entire lifespan of the object.*
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
NETWORK_OID Foreign Key (TNF_NETWORK)	CharacterString (OPTIONAL)	OID for the transport network
GEOMETRY	Geometry (point) (OPTIONAL)	The geometry for this node. The coordinates shall be 3 dimensional (x, y and z). Coordinates shall be interpreted according to the coordinate reference system specified in metadata. *Note: A z value of -99999 indicates that z is unknown.* *Note: The geometry shall correspond exactly to the start- or end vertex of the geometry for the transport links to which the node is*

		connected depending on the role (start- or end node) of the node for the transport link.
BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.
END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

3.2.5 TNF_NETWORK (OPTIONAL)

A network represents a collection of network elements that belong to a single mode of transport. Road, rail, water, cable, and air transport are always considered separate transport modes. Even within these four categories, multiple modes of transport can be defined. E.g. all road transport can be considered one mode of transport for some applications. For other applications, it might be necessary to distinguish between different public road transport networks.

A network element can belong to zero, one or many networks defined for different applications.

Name	**Type**	**Comment**
OID Primary Key	CharacterString	Globally unique object id.
TYPE_OF_TRANSPORT	CharacterString	Type of transport network, based on the type of infrastructure the network uses. *Note: Can be set to one of the following "air", "cable","rail","road", "water"*
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
GEOGRAPHICAL_NAME	CharacterString (OPTIONAL)	A geographical name that is used to identify the transport network object in the real world.
BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.

END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

The figure below shows a simple example with two networks.

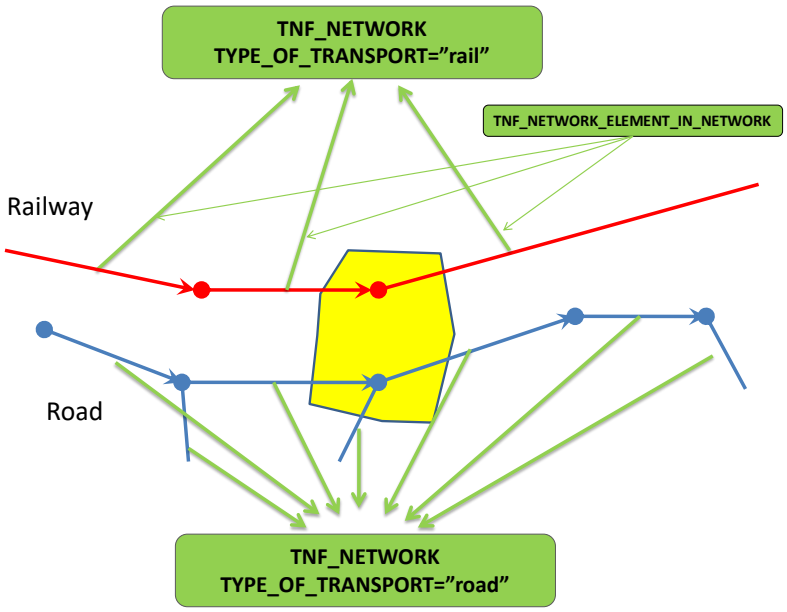


Figure 4 - A network example

3.2.6 TNF_NETWORK_CONNECTION (OPTIONAL)

A network connection represents a logical connection between two or more network elements in different networks.

Name	Type	Comment
OID Primary Key	CharacterString	Globally unique object id.
TYPE	CharacterString	Type of connection *Note: Can be set to one of the following "crossBorderConnected", "crossBorderIdentical","intermodal"*
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is*

		changed due to real world changes or corrections of the data.
BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.
END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

The figure below shows an example of an intermodal network connection.

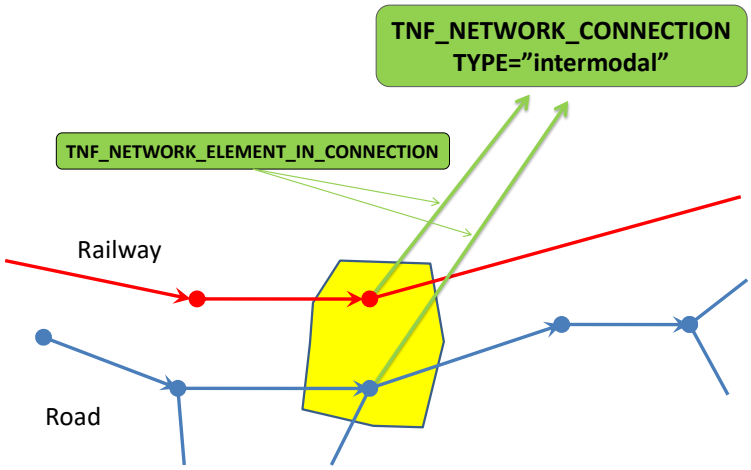


Figure 5 - A network connection example

3.2.7 TNF_NETWORK_ELEMENT_IN_CONNECTION (OPTIONAL)

A TNF_NETWORK_ELEMENT_IN_CONNECTION represents one connection between a network connection and a network element.

Name	Type	Comment
NETWORK_CONNECTION_OID Primary Key Foreign Key (TNF_NETWORK_CONNECTION)	CharacterString	Globally unique object id.

NETWORK_ELEMENT_REF Primary Key Foreign Key (TNF_LINK_SEQUENCE, TNF_LINK, TNF_NODE)	TNF_NetworkElementRef	OID-reference to a network element (either a link sequence, link or node)

3.2.8 <<union>>TNF_NetworkElementRef

A NetworkElementRef is used to represent exactly one reference to a link, a link sequence or a node.

3.2.9 <<union>>TNF_ LinearElementRef

A LinearElementRef is used to represent exactly one reference to a link or a link sequence.

3.3 Generic model for attribution and features related to the transport network

3.3.1 Overview

OpenTNF defines a generic model for attribution and features related to the transport network. Attribution in this context relates to the representation of characteristics of the individual elements in the network (or parts of the individual elements). Features relates to abstractions of real world phenomena that exists on their own right but has a physical or logical relationship with locations in the network.

In OpenTNF, the same generic model is used to represent both attribution and features which are related to the network. This means that the same set of classes is used for things such as speed limit, accident and traffic sign. In OpenTNF this is collected in the concept of a property object. Furthermore, every property object has to be uniquely identified. The primary reason for this is to enable the OpenTNF mechanism for incremental updates, where attribution (e.g. speed limit) may be updated independently of the underlying transport network.

Since an xml representation is used to represent the characteristics of a property object, it is possible to represent very complex data structures including geometry (using GML). The characteristics for each property object type is defined using the model in chapter

Generic model for transport property object catalogues.

The only characteristics of a property object that are explicitly modelled are the temporal characteristics and the actual references to the transport network location. The network location may be represented both statically by using explicit references to network elements and dynamically (or map agnostic) using AGORA or OPENLR. The primary use of map-agnostic location referencing such as AGORA or OPENLR is when the parties do not have the same representation of the network but still wants to exchange property objects.

Each property object (TNF_PROPERTY_OBJECT) may carry nformation on the temporal validity of the real world characteristic or feature that it represents. Furthermore, since each property object may

contain more than one property (TNF_PROPERTY) it is possible to keep track on how a single real world characteristic or feature has changed over time.

EXAMPLE A speed limit changed value from 70 to 80 at 2011-01-01. This is represented by a single TNF_PROPERTY_OBJECT ({“STA”,”aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa”}). The TNF_PROPERTY_OBJECT contains two instances of TNF_PROPERTY ({“STA”,”bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb”} and {“STA”,”cccccccc-cccc-cccc-cccc-cccccccccccc”}) representing the two different states. The table below shows the data needed to instantiate the two TNF_PROPERTY objects.

Attribute	Value	Comment
PROPERTY_OBJECT_OID	{“STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa”}	
OID	{“STA:bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb”}	
VALID_FROM	2000-01-01	
VALID_TO	2011-01-01	
ATTRIBUTE_VALUES	<tnf:SimpleAttribute attributeType="SpeedLimit"> <tnf:values>70</tnf:values> </tnf:SimpleAttribute>	The value in this example is simplified to save space. A more complete example is illustrated in 3.3.3.
PROPERTY_OBJECT_OID	{“STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa”}	The same as for OID {“STA:bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb”} above.
OID	{“STA:cccccccc-cccc-cccc-cccc-cccccccccccc”}	
VALID_FROM	2011-01-01	
VALID_TO	<NULL>	
ATTRIBUTE_VALUES	<tnf:SimpleAttribute attributeType="SpeedLimit"> <tnf:values>80</tnf:values> </tnf:SimpleAttribute>	The value in this example is simplified to save space. A more complete example is illustrated in 3.3.3.

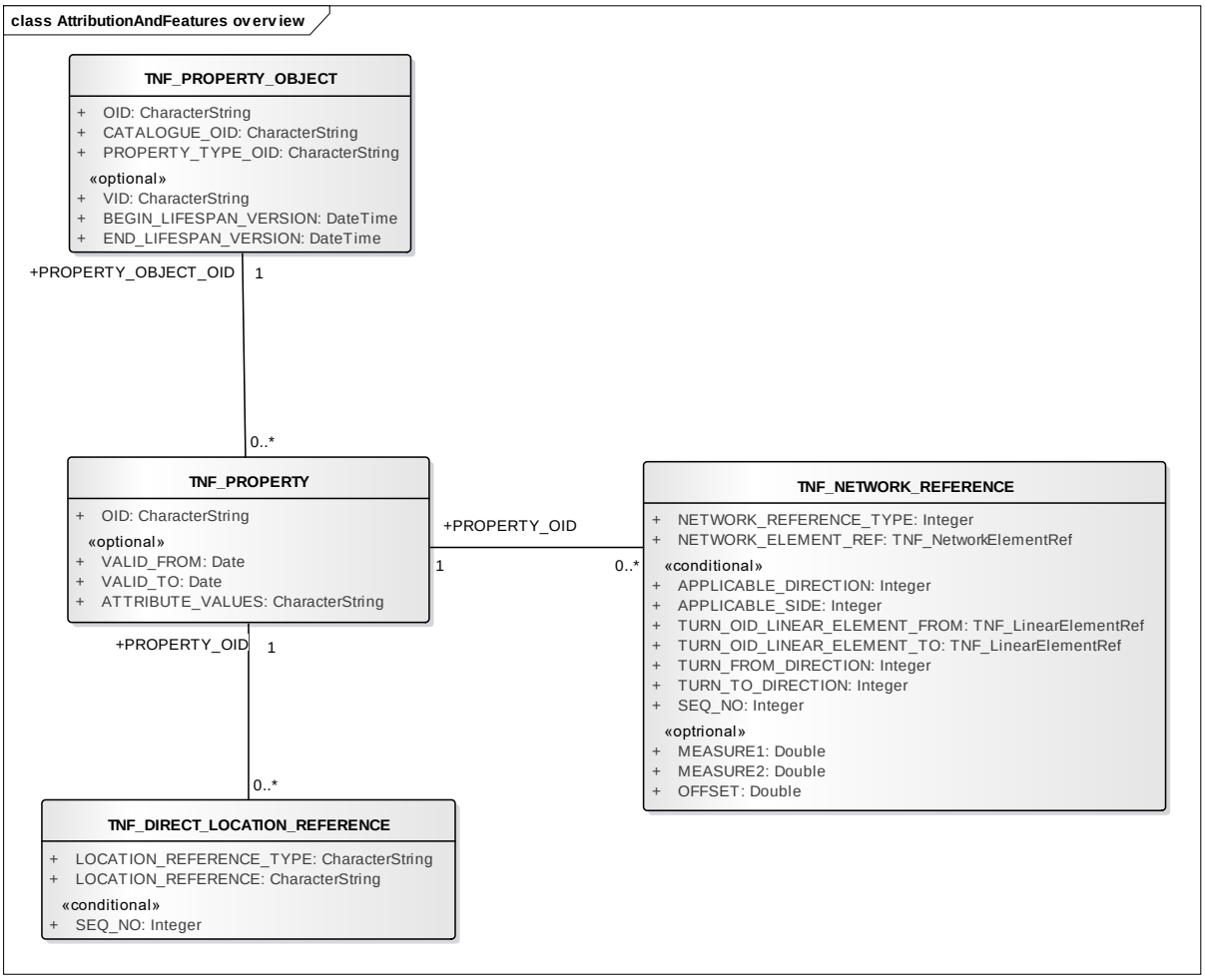


Figure 6 - Property objects overview

3.3.2 TNF_PROPERTY_OBJECT

A property object may represent any type of property of the transport network.

Name	Type	Comment
OID Primary Key	CharacterString	Globally unique object id. *Note: This id is used during the entire lifespan of the object.*
CATALOGUE_OID Foreign Key (TNF_CATALOGUE)	CharacterString	The identity of the feature catalogue where the property object type is defined.
PROPERTY_OBJECT_TYPE_OID	CharacterString	The identity of the property object type of which this property object is an instance

Foreign Key (TNF_PROPERTY_OBJECT_TYPE)		
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
BEGIN_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was inserted or changed in the source data set.
END_LIFESPAN_VERSION	DateTime (OPTIONAL)	Specifies the date and time at which this version of the object was superseded or retired in the source data set.

3.3.3 TNF_PROPERTY

A property represents the state of a property object which is valid during a specified period in the real world.

Name	Type	Comment
OID Primary Key	CharacterString	OID for the property.
PROPERTY_OBJECT_OID Foreign Key (TNF_PROPERTY_OBJECT)	CharacterString	OID for the property object to which this property belongs.
VALID_FROM	Date (OPTIONAL)	The date from which this property is valid. *Note: A null value means that the from date is unknown and that the object is valid if the current date < VALID_TO*
VALID_TO	Date (OPTIONAL)	The date from which this property is no longer valid.

		Note: A null value means that the to date is unknown and that the object is valid if the current date >= FROM_DATE
ATTRIBUTE_VALUES	CharacterString (OPTIONAL)	An XML representation of the entire set of attributes including structured and multiple attributes. Format and schema for the encoding and decoding of this data is specified in the feature catalogue for the corresponding property object. Allowed formats at the moment is: - xml/text - xml/gzip/base64 The xml is encoded according to http://schemas.opentnf.org/1.0/tnf_attr.xsd which is also shown below. *Note: Associations encode the associated property object OID as a character string.* *Note: Spatial attribute values shall be encoded according to the TNF_SPATIAL_ATTRIBUTE_ENCODING metadata key.*

Tnf_attr.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tnf="http://www.opentnf.org "
xmlns:gml="http://www.opengis.net/gml" targetNamespace="http://www.opentnf.org"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
    <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="http://www.isotc211.org/schemas/2005/gml/gml.xsd"/>
    <xs:element name="Attributes" type="tnf:AttributesType"/>
    <xs:complexType name="AttributesType">
        <xs:sequence>
            <xs:element ref="tnf:Attribute" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="catalogueOID" type="xs:string" use="required"/>
        <xs:attribute name="propertyObjectTypeOID" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="Attribute" type="tnf:AttributeType" abstract="true"/>
    <xs:complexType name="AttributeType" abstract="true">
        <xs:attribute name="attributeType" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:element name="SimpleAttribute" type="tnf:SimpleAttributeType" abstract="false"
substitutionGroup="tnf:Attribute"/>
    <xs:complexType name="SimpleAttributeType" abstract="false">
        <xs:complexContent>
            <xs:extension base="tnf:AttributeType">
```

```
                <xs:sequence>
                    <xs:element name="values" type="xs:anyType" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="StructuredAttribute" type="tnf:StructuredAttributeType" abstract="false"
substitutionGroup="tnf:Attribute"/>
    <xs:complexType name="StructuredAttributeType" abstract="false">
        <xs:complexContent>
            <xs:extension base="tnf:AttributeType">
                <xs:sequence>
                    <xs:element ref="tnf:Attribute" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

EXAMPLE 1: SpeedLimit with GML ATTRIBUTE_VALUES for a TNF_PROPERTY of type "SpeedLimit" having the maximum speed limit defined as 70 and a conditional speed limit (the actual condition such as time or vehicle type not being defined in the example) defined as 30. The speed limit also has a geometry associated which is represented by a gml:LineString element.

```
<?xml version="1.0" encoding="UTF-8"?>
<tnf:Attributes xmlns:tnf="http://www.triona.se/tnf" xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:gco="http://www.isotc211.org/2005/gco" xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:gss="http://www.isotc211.org/2005/gss"
xmlns:gts="http://www.isotc211.org/2005/gts" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.opentnf.org/1.0/tnf_attr.xsd" catalogueOID="NVDB_DK"
propertyObjectTypeOID="SpeedLimit">
    <tnf:SimpleAttribute attributeType="MaximumSpeedLimit">
        <tnf:values>70</tnf:values>
    </tnf:SimpleAttribute>
    <tnf:SimpleAttribute attributeType="Geometry">
        <tnf:values>
            <gml:LineString gml:id="i1">
                <gml:posList>123.456 234.567 11.11 456.678 567.89 12.34</gml:posList>
            </gml:LineString>
        </tnf:values>
    </tnf:SimpleAttribute>
    <tnf:StructuredAttribute attributeType="ConditionalSpeedLimit">
        <tnf:SimpleAttribute attributeType="SpeedLimit">
            <tnf:values>30</tnf:values>
        </tnf:SimpleAttribute>
    </tnf:StructuredAttribute>
</tnf:Attributes>
```

EXAMPLE 2: SpeedLimit with WKT ATTRIBUTE_VALUES for a TNF_PROPERTY of type "SpeedLimit" having the maximum speed limit defined as 70 and a conditional speed limit (the actual condition such as time or vehicle type not being defined in the example) defined as 30. The speed limit also has a geometry associated which is represented by a WKT (Well-known Text) LineString value.

```
<?xml version="1.0" encoding="UTF-8"?>
<tnf:Attributes xmlns:tnf="http://www.triona.se/tnf xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.opentnf.org/1.0/tnf_attr.xsd" catalogueOID="NVDB_DK"
propertyObjectTypeOID="SpeedLimit">
    <tnf:SimpleAttribute attributeType="MaximumSpeedLimit">
```

```
        <tnf:values>70</tnf:values>
    </tnf:SimpleAttribute>
    <tnf:SimpleAttribute attributeType="Geometry">
        <tnf:values>
            LineStringZ(123.456 234.567 11.11, 456.678 567.89 12.34)
        </tnf:values>
    </tnf:SimpleAttribute>
    <tnf:StructuredAttribute attributeType="ConditionalSpeedLimit">
        <tnf:SimpleAttribute attributeType="SpeedLimit">
            <tnf:values>30</tnf:values>
        </tnf:SimpleAttribute>
    </tnf:StructuredAttribute>
</tnf:Attributes>
```

3.3.4 TNF_NETWORK_REFERENCE

A network reference specifies the extent of a transport property, using the primary linear referencing system.

Name	Type	Comment
PROPERTY_OID Foreign Key (TNF_PROPERTY)	CharacterString	Reference to the property to which this network reference belongs.
NETWORK_REFERENCE_TYPE	Integer	Specifies the type of network reference: 1 : Node 4 : PointOnLinearElement 8 : SegmentOnLinearElement 64 : Turn 512 : RailwayTurn *Note: The reason for specifying the type for each network reference instance is that different types of references may be mixed for a single property object type.*
NETWORK_ELEMENT_REF Foreign Key (TNF_LINK_SEQUENCE, TNF_LINK, TNF_NODE)	TNF_NetworkElementRef	Reference to the network element (link sequence, link or node) to which this network reference refers.

APPLICABLE_DIRECTION	Integer (CONDITIONAL)	Specifies the direction of the network reference with regards to the underlying linear element (TNF_LINK_SEQUENCE or TNF_LINK). This value shall be specified if the NETWORK_REFERENCE _TYPE={4,8,64} and the corresponding transport property object type specifies that the type is direction dependent. The possible values are: 1 = Same as LRS -1 = Opposite to LRS 0 = Direction independent or both
APPLICABLE_SIDE	Integer (CONDITIONAL)	Specifies the side of the network reference with regards to the underlying linear element (TNF_LINK_SEQUENCE or TNF_LINK). This value shall be specified if the NETWORK_REFERENCE _TYPE={4,8} and the corresponding transport property object type specifies that the type is side dependent. The possible values are: -1 = Left 0 = Mid 1 = Right 2 = Left and right 3 = Crossing

SEQ_NO	Integer (CONDITIONAL)	A sequence number to record the order of the network references within one TNF_PROPERTY. The SEQ_NO attribute must be set for TNF_PROPERTY_OBJECT:s whose TNF_PROPERTY_OBJECT_TYPE specifies ORDERED_NETWORK_REFERENCES = True
TURN_OID_LINEAR_ELEMENT_FROM Foreign Key (TNF_LINK_SEQUENCE, TNF_LINK)	TNF_LinearElementRef (CONDITIONAL)	Specifies the linear element where the turn starts. Shall be specified if NETWORK_REFERENCE_TYPE={64}
TURN_FROM_DIRECTION	Integer (CONDITIONAL)	Specifies the direction of the turn in relation to the underlying linear element specified with TURN_OID_FROM. Shall be specified when NETWORK_REFERENCE_TYPE={64}. Possible values are: 1 = Same as linear element -1 = Opposite to linear element
TURN _OID_LINEAR_ELEMENT_TO Foreign Key (TNF_LINK_SEQUENCE, TNF_LINK)	TNF_LinearElementRef (CONDITIONAL)	Specifies the linear element where the turn ends. Shall be specified if NETWORK_REFERENCE_TYPE={64}

TURN_TO_DIRECTION	Integer (CONDITIONAL)	Specifies the direction of the turn in relation to the underlying linear element specified with TURN_OID_TO. Shall be specified when NETWORK_REFERENCE _TYPE={64}. Possible values are: 1 = Same as LRS -1 = Opposite to LRS
MEASURE1	Real (double precision) (OPTIONAL)	A measure within the measures of the linear element (transport link sequence or transport link) that specifies the start of the network reference (for linear network references) or the location of the extent (for point network references). If omitted, the start of the element is assumed. This value may only be specified for NETWORK_REFERENCE _TYPE={4, 8}
MEASURE2	Real (double precision) (OPTIONAL)	A measure within the measures of the linear element (transport link sequence or transport link) that specifies the end of the network reference. If omitted, the end of the element is assumed. This value may only be specified for NETWORK_REFERENCE _TYPE={8}
OFFSET	Real (double precision) (OPTIONAL)	A positive value specifying a lateral offset from the linear element. The side relative to the positive direction of the linear element is specified by “APPLICABLE_SIDE” above.

IS_PREFERRED	Boolean (CONDITIONAL)	True if the network reference represents the preferred turn through the node. This value shall be specified if the NETWORK_REFERENCE _TYPE={512}.

3.3.4.1 Example on NETWORK_REFERENCE_TYPE = PointOnLinearElement

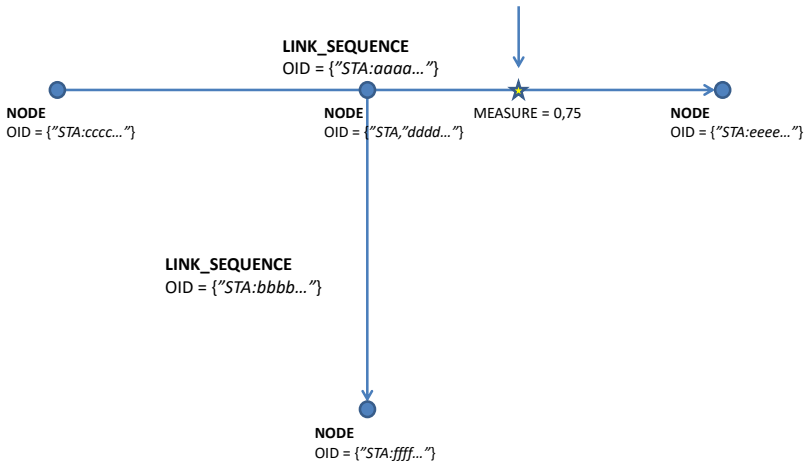


Figure 7 - A PointOnLinearElement example

To define the above point on linear element to the definitions in TNF_NETWORK_REFERENCE, the following attributes shall be set:

Attribute	Value	Comment
NETWORK_REFERENCE_TYPE	4	PointOnLinearElement
NETWORK_ELEMENT_REF.LINK_SEQUENCE_OID	{"STA:aaaa…"}	
MEASURE1	0,75	
APPLICABLE_SIDE	2	Could be used to indicate a side in relation to the LRS
APPLICABLE_DIRECTION	1	Could be used to indicate that the network reference (logically) belongs to a

		certain direction in relation to the LRS
OFFSET	5	Could be used to indicate an offset distance from the LRS

3.3.4.2 Example on NETWORK_REFERENCE_TYPE = SegmentOnLinearElement

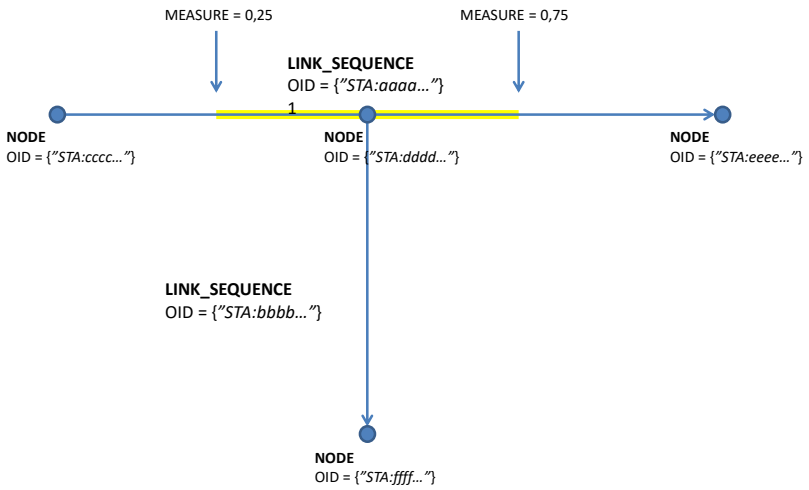


Figure 8 - SegmentOnLinearElement example

To define the above segment on linear element to the definitions in TNF_NETWORK_REFERENCE, the following attributes shall be set:

Attribute	Value	Comment
NETWORK_REFERENCE_TYPE	8	SegmentOnLinearElement
NETWORK_ELEMENT_REF.LINK_SEQUENCE_OID	{"STA:aaaa…"}	
MEASURE1	0,25	
MEASURE2	0,75	
APPLICABLE_SIDE	2	Could be used to indicate a side in relation to the LRS
APPLICABLE_DIRECTION	1	Could be used to indicate a direction in relation to the LRS

OFFSET	5	Could be used to indicate an offset distance from the LRS

3.3.4.3 Example on NETWORK_REFERENCE_TYPE = Turn

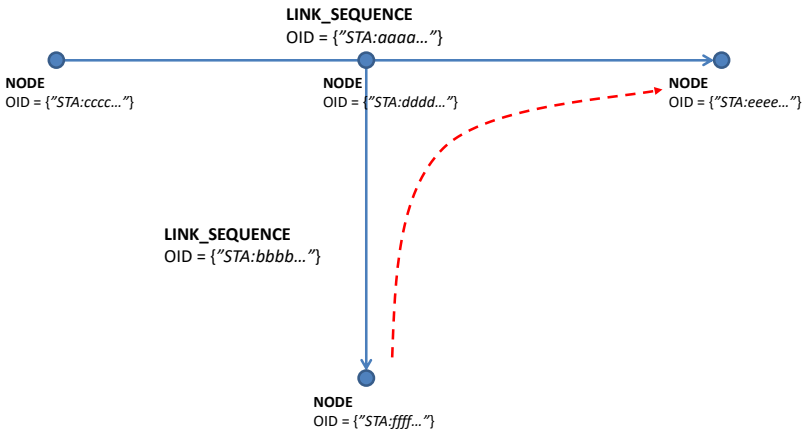


Figure 9 - Definition of a turn, example

To define the above turn according to the definitions in TNF_NETWORK_REFERENCE, the following attributes shall be set:

Attribute	Value	Comment
NETWORK_REFERENCE_TYPE	64	Turn
APPLICABLE_DIRECTION	1	Agrees with the defined direction of the turn
NETWORK_ELEMENT_REF.NODE_OID	{"STA:dddd…"}	Turn through node {"STA:dddd…"}
TURN_OID_LINEAR_ELEMENT_FROM LINK_SEQUENCE_OID	{"STA:bbbb…"}	Start at link sequence {"STA:bbbb…"}

TURN_FROM_DIRECTION	2	The turn's direction is opposite to the direction of the link sequence. Note: In this particular case a TURN_FROM_DIRECTION would not be needed to specify the direction of the turn since the LINEAR_ELEMENT_FROM starts at the NODE. However, since the direction is needed in some cases, the specification requires that all turns specify this value.
TURN_OID_LINEAR_ELEMENT_TO LINK_SEQUENCE_OID	{"STA:aaaa…"}	End at link sequence {"STA:aaaa…"}
TURN_TO_DIRECTION	1	The turn's direction is same as the direction of the link sequence Note: In this case, the direction is necessary since we otherwise would not know if the turn is to the left or to the right.

3.3.4.4 Example on NETWORK_REFERENCE_TYPE = Turn with several turns (Maneuvre)

The example below is often referred to as a maneuver since it is represented by a sequence of turns. There is no explicit way to handle maneuvers in OpenTNF. However, it is possible to specify that a TNF_PROPERTY_OBJECT_TYPE allows for more than one network reference for each instance and that the order of network references shall be maintained. This way a property object may have one or more network references of type turn which are ordered.

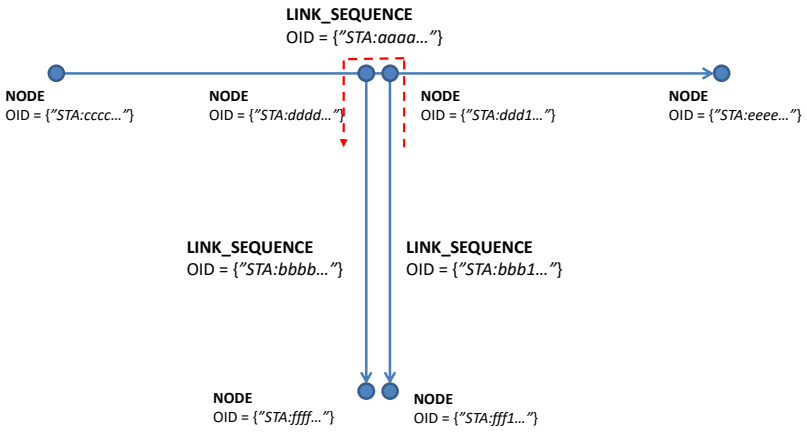


Figure 10 - Definition of a maneuver, example

First turn (SEQ_NO = 1):

Attribute	Value	Comment
NETWORK_REFERENCE_TYPE	64	Turn
APPLICABLE_DIRECTION	1	Agrees with the defined direction of the turn
NETWORK_ELEMENT_REF.NODE_OID	{"STA:ddd1…"}	Turn through node {"STA:ddd1…"}
TURN_OID_LINEAR_ELEMENT_FROM LINK_SEQUENCE_OID	{"STA:bbb1…"}	Start at link sequence {"STA:bbb1…"}
TURN_FROM_DIRECTION	2	The turn's direction is opposite to the direction of the link sequence
TURN_OID_LINEAR_ELEMENT_TO LINK_SEQUENCE_OID	{"STA:aaaa…"}	End at link sequence {"STA:aaaa…"}
TURN_TO_DIRECTION	2	The turn's direction is opposite to the direction of the link sequence

Second turn (SEQ_NO = 2):

Attribute	Value	Comment
NETWORK_REFERENCE_TYPE	64	Turn
APPLICABLE_DIRECTION	1	Agrees with the defined direction of the turn
NETWORK_ELEMENT_REF.NODE_OID	{“STA:dddd…”}	Turn through node {“STA:dddd…”}
TURN_OID_LINEAR_ELEMENT_FROM LINK_SEQUENCE_OID	{“STA:aaaa…”}	Start at link sequence {“STA:aaaa…”}
TURN_FROM_DIRECTION	2	The turn’s direction is opposite to the direction of the link sequence
TURN_OID_LINEAR_ELEMENT_TO LINK_SEQUENCE_TO	{“STA:bbbb…”}	End at link sequence {“STA:bbbb…”}
TURN_TO_DIRECTION	1	The turn’s direction is same as the direction of the link sequence

3.3.5 TNF_DIRECT_LOCATION_REFERENCE (OPTIONAL)

A direct location reference specifies the location of a transport property, using a direct location reference method. This type of location reference may be used together with the network references above for the same property object. The combined set of locations/network references shall be used as the location of the property object. The dimensionality of the location reference shall be correct with regards to what has been specified for the corresponding property object type.

Name	Type	Comment
PROPERTY_OID Foreign Key (TNF_PROPERTY_OBJECT)	CharacterString	Reference to the property to which this network reference belongs.

LOCATION_REFERENCE_TYPE	CharacterString	Specifies the type of location reference (i.e. the type for the content in the LOCATION_REFERENCE field). Currently, the following types of location references are supported: - AGORA_BINARY (binary encoding according to ISO 17572-3) - AGORA_XML (xml encoding according to ISO 17572-3) - OPENLR_BINARY (binary encoding according to OpenLR) - OPENLR_XML (xml encoding according to OpenLR) Specifications regarding physical (binary and xml) formats for OpenLR may be found at http://www.openlr.org.
LOCATION_REFERENCE	CharacterString	The location reference. The content of this field shall be interpreted according to what has been specified in the LOCATION_REFERENCE_TYPE field. A binary location reference is encoded as a base64 string and xml is encoded as a string.
SEQ_NO	Integer (CONDITIONAL)	A sequence number to record the order of the network references within one TNF_PROPERTY. The SEQ_NO attribute must be set for TNF_PROPERTY_OBJECT:s whose TNF_PROPERTY_OBJECT_TYPE specifies ORDERED_NETWORK_REFERENCES = True

3.4 Generic model for transport property object catalogues

3.4.1 Overview

OpenTNF defines a generic model for attribution and features related to the transport network. Attribution in this context relates to the representation of characteristics of the individual elements in the network (or parts of the individual elements). Features relates to abstractions of real world phenomena that exists on their own right but has a physical or logical relationship with locations in the network. To be able to interpret the data in this generic model, the various types of characteristics and features shall be described in a catalogue. The structure for the catalogue is described in this chapter.

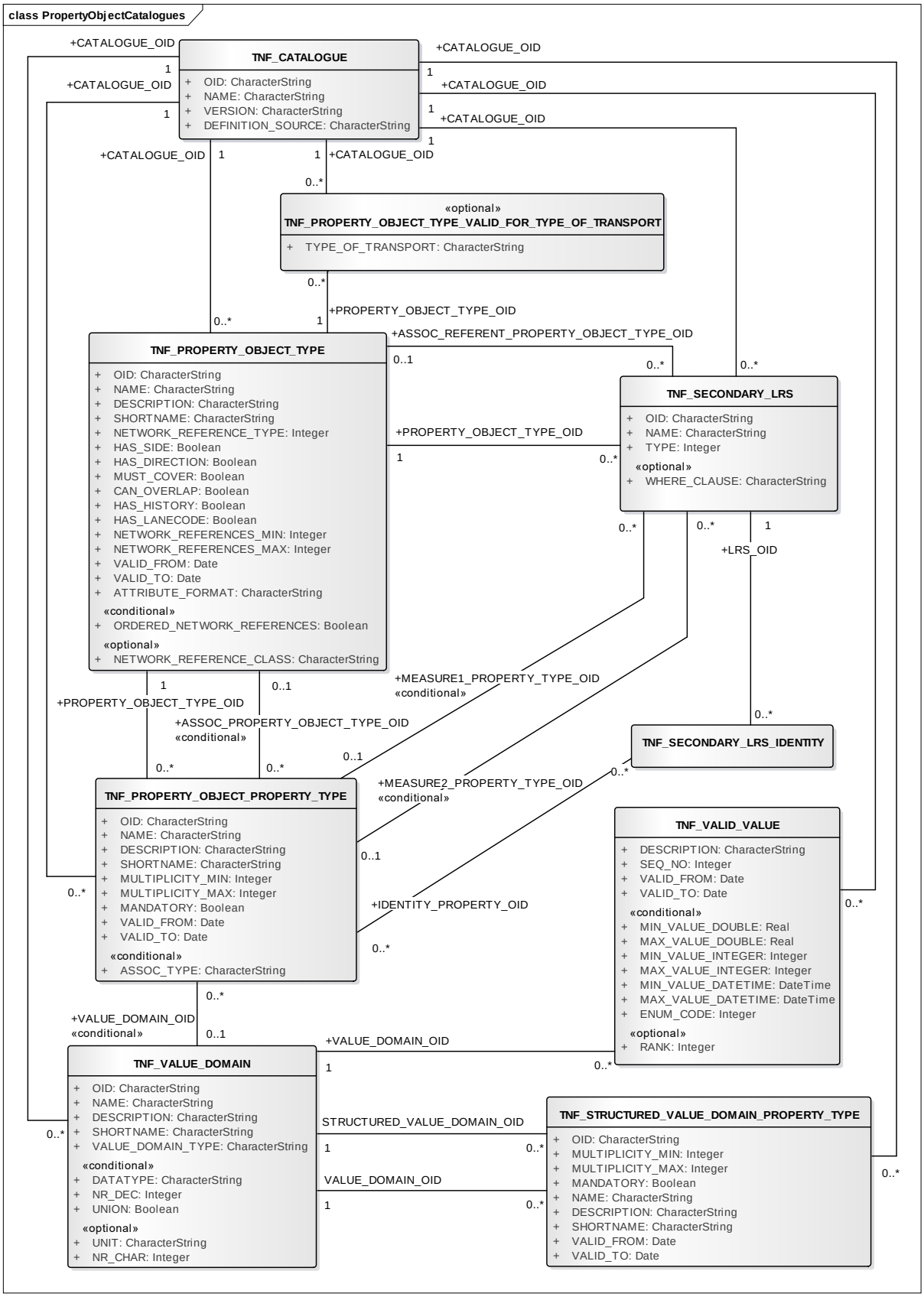


Figure 11 - Generic model for transport property object catalogues overview

3.4.2 TNF_CATALOGUE

A catalogue specifies a set of transport property object types. Typically, a single catalogue has one origin of responsibility and a coherent content meant to be used within a certain domain of applications.

Name	Type	Comment
OID Primary Key	CharacterString	Catalogue id. Should be agreed and unique within a certain community.
NAME	CharacterString	The name of the catalogue.
VERSION	CharacterString	The catalogue version
DEFINITION_SOURCE	CharacterString (OPTIONAL)	URI or a bibliographic reference, including author, title, edition, publisher, place of publication, and date of publication, to a published external source of definitions for information included in feature catalogue
DESCRIPTION	CharacterString (OPTIONAL)	A description of the catalogue, e.g. a description of the content and intended use.

3.4.3 TNF_PROPERTY_OBJECT_TYPE

A property object type is a definition of a transport property object. As described earlier in this document a property object type may define both characteristics of the elements in the transport network and features with some relationship with the network. Examples of property object types may be *Speed Limit, Functional Road Class, Accident, Traffic Sign.*

Name	Type	Comment
OID Primary Key	CharacterString	Identification of the property object type which is unique within the catalogue.
CATALOGUE_OID Primary Key Foreign Key (TNF_CATALOGUE)	CharacterString	Identifies the feature catalogue to which this property object type belongs
NAME	CharacterString	A descriptive name for the property object type

DESCRIPTION	CharacterString	A description/definition of the property object type
NETWORK_REFERENCE_TYPE	Integer	Specifies the type of network references that are allowed for property objects of this type. If more than one NETWORK_REFERENCE_TYPE is allowed, the values are added: 1 : Node 4 : PointOnLinearElement 8 : SegmentOnLinearElement 64 : Turn 512 : RailwayTurn
HAS_SIDE	Boolean	True if the transport property object type is side dependent
HAS_DIRECTION	Boolean	True if the transport property object type is direction dependent
MUST_COVER	Boolean	True if this type shall cover the entire network
CAN_OVERLAP	Boolean	True if several instances (transport property object) of this type may overlap in time and space
HAS_HISTORY	Boolean	True if transport property objects of this type are allowed to record history.
HAS_LANECODE	Boolean	True if the transport property object type is lane dependent.
NETWORK_REFERENCES_MIN	Integer	Specifies the minimum allowed number of network references per transport property object of this type
NETWORK_REFERENCES_MAX	Integer	The maximum allowed number of network references per transport property object of this type -1 = Unlimited

VALID_FROM	Date	The date from which the transport property object type is valid and in use
VALID_TO	Date	The date from which the transport property object type is no longer valid and in use
SHORTNAME	CharacterString	A short mnemonic name or code for the transport property object type. *Note: Only alphanumeric characters (i.e. A-Z, a-z and 0-9) are recommended for compatibility reasons.*
ATTRIBUTE_FORMAT	CharacterString	”text” – Xml encoded as text ”binary” – Xml text zipped to binary with GZIP and base64-encoded
ORDERED_NETWORK_REFERENCES	Boolean (CONDITIONAL)	True if the ordering of instances of network references for objects of this type has to be maintained. This attribute has to be set for property object types with NETWORK_REFERENCES_MAX = -1 or NETWORK_REFERENCES_MAX > 1
NETWORK_REFERENCE_CLASS	CharacterString (OPTIONAL)	Name of the class (table) containing network references for transport property objects of this type. The referenced class shall be a true extension of the mandatory parts of the TNF_NETWORK_REFERENCE class for the specified NETWORK_REFERENCE_TYPE. If null, then the used class shall be TNF_NETWORK_REFERENCE. *Note: This option may be usable if someone wants to extend the TNF_NETWORK_REFERENCE class to include geometry attributes and/or the property type instances as separate and explicit attributes (columns) in addition to the xml.*

BASE_CATALOGUE_OID Foreign Key (TNF_CATALOGUE)	Integer (CONDITIONAL)	Identifies the catalogue containing the type referred to by BASE_PROPERTY_OBJECT_TYPE_OID below Shall be specified if BASE_PROPERTY_OBJECT_TYPE_OID below has been specified.
BASE_PROPERTY_OBJECT_TYPE_OID Foreign key (TNF_PROPERTY_OBJECT_TYPE)	Integer (OPTIONAL)	If present, identifies that this type is a sub type (or a specialized type) of the type referred to by the value of BASE_PROPERTY_OBJECT_TYPE_OID.
IS_DERIVED	Boolean (OPTIONAL)	The value True specifies that the instances of this property object type are derived by a system (i.e. not intended to be edited by an end user). Default is False.

3.4.4 TNF_PROPERTY_OBJECT_TYPE_VALID_FOR_TYPE_OF_TRANSPORT (OPTIONAL)

This class specifies which type/s of transport a property object type can be used for.

Name	Type	Comment
PROPERTY_OBJECT_TYPE_OID Primary Key Foreign Key (TNF_PROPERTY_OBJECT_TYPE)	CharacterString	Identifies the property object type to define valid transport type(s) for
CATALOGUE_OID Primary Key Foreign Key (TNF_CATALOGUE)	CharacterString	Identifies the catalogue to which the property object type belongs.
TYPE_OF_TRANSPORT Primary Key	CharacterString	Type of transport network (TNF_Network) the property object type applies to. *Note: Can be set to one of the following "air", "cable","rail","road", "water"*

3.4.5 TNF_PROPERTY_OBJECT_PROPERTY_TYPE

This class specifies the properties (characterizing attributes) for a property object type. Examples may be *Speed Limit.Maximum speed, Functional Road Class.Class, Accident.Type, Traffic Sign.Type.*

Name	Type	Comment
OID Primary Key	CharacterString	Id for the property type. Shall be unique within the property object type.
CATALOGUE_OID Primary Key Foreign Key (TNF_CATALOGUE)	CharacterString	Identifies the catalogue to which the property type belongs
PROPERTY_OBJECT_TYPE_OID Primary Key Foreign Key (TNF_PROPERTY_OBJECT_TYPE)	CharacterString	Identifies the property object type to which this property type belongs
MULTIPLICITY_MIN	Integer	The minimum allowed number of values for this property type
MULTIPLICITY_MAX	Integer	The maximum allowed number of values for this property type
MANDATORY	Boolean	True if the property type is mandatory (i.e. an instance must contain a value for this property type)
NAME	CharacterString	A descriptive name for this property type
DESCRIPTION	CharacterString	A description/definition of this property type
SHORTNAME	CharacterString	A short/mnemonic name or code for this property type. *Note: Only alphanumeric characters (i.e. A-Z, a-z and 0-9) are recommended for compatibility reasons.*

VALID_FROM	Date	The date from which the property type is valid and in use
VALID_TO	Date	The date from which the property type is no longer valid and in use
ASSOC_PROPERTY_OBJECT_TYPE_OID Foreign Key (TNF_PROPERTY_OBJECT_TYPE)	CharacterString (CONDITIONAL)	If the property type represents an association, this field specifies the identity of the associated transport property object type
ASSOC_TYPE	CharacterString (CONDITIONAL)	If the property type represents an association, this field specifies the type of association. Valid values are: - BYREF - BYVAL BYREF indicates that the object and the associated object do not share lifespan. BYVAL indicates that the object and the associated object share lifespan (i.e. if the object is deleted, the associated object shall also be deleted)
VALUE_DOMAIN_OID Foreign Key (TNF_VALUE_DOMAIN)	CharacterString (CONDITIONAL)	If the property type represents an attribute, this field specifies the identity of the associated value domain

3.4.6 TNF_VALUE_DOMAIN

This class specifies a value domain. A value domain is reusable and specifies a basic data type, set of valid values, unit etc. An example may be *Speed* (DATATYPE=Integer, UNIT=km/h).

Namn	**Typ**	**Anm.**
OID Primary Key	CharacterString	Id for the value domain. Shall be unique within the catalogue.
CATALOGUE_OID	CharacterString	Identifies the catalogue to which the ValueDomain belongs

Primary Key Foreign Key (TNF_CATALOGUE)		
VALUE_DOMAIN_TYPE	CharacterString	”SIMPLE” or ”STRUCTURED”. Instances of ”SIMPLE” attributes shall be encoded as <SimpleAttribute>. Instances of ”STRUCTURED” attributes shall be encoded as <StructuredAttribute> and shall have properties (see TNF_STRUCTURED_VALUE_DOMAIN_PROPERTY_TYPE).
NAME	CharacterString	A descriptive name for the value domain
SHORTNAME	CharacterString	A short/mnemonic name or code for the value domain. *Note: Only alphanumeric characters (i.e. A-Z, a-z and 0-9) are recommended for compatibility reasons.*
DESCRIPTION	CharacterString	A description/definition for the value domain
DATATYPE	CharacterString (CONDITIONAL)	Specifies basic data type for the (SIMPLE) value domain. These names are specified in ISO/TS 19103 – conceptual schema language : • Boolean – Logical value (true/false) • Date – Date • DateTime – Date and time • Time – Time • Enum – Enumeration. Valid enumeration values shall be specified in TNF_VALID_VALUE • Real – Real • Integer – Integer • CharacterString – character string • Point – Point (spatial attribute)

		• LineString – LineString (spatial attribute) • Polygon – Polygon (spatial attribute) • BLOB – Binary data • ShortDate – Date without year *Note: Spatial attribute values shall be encoded according to the TNF_SPATIAL_ATTRIBUTE_ENCODING metadata key.*
NR_DEC	Integer (CONDITIONAL)	Number of significant decimals for value domains of type real
IS_UNION	Boolean (OPTIONAL)	For a structured value domain, this value specifies whether the domain shall be regarded and used as a union, i.e. for each value instance for the domain only one of the structured value domain property types may be used.
UNIT	CharacterString (OPTIONAL)	A unit of measure for the value domain if applicable (e.g. ”km/h”, ”mph” or ”m”) If not specified (=NULL) a MEASURE_QUANTITY may be specified and in such case is the SI unit assumed.
MEASURE_QUANTITY	CharacterString (OPTIONAL)	Specifies the quantity used for the value domain if applicable. Valid values follow the international system of units (SI). For each quantity there is a defined unit. Examples are: - length (m) - mass (kg) - time (s) - speed (m/s) This attribute specifies the quantity and assumes the standard unit to be used (if not specified otherwise in the UNIT attribute)
NR_CHAR	Integer (OPTIONAL)	Number of characters for presentation of numbers or maximum number of characters for character strings

3.4.7 TNF_STRUCTURED_VALUE_DOMAIN_PROPERTY_TYPE

This class specifies the properties (characterizing attributes) for a structured value domain.

Name	Type	Comment
OID Primary Key	CharacterString	Id for the property type. Shall be unique within the value domain.
CATALOGUE_OID Primary Key Foreign Key (TNF_CATALOGUE)	CharacterString	Identifies the catalogue to which the property type belongs
STRUCTURED_VALUE_DOMAIN_OID Primary Key Foreign Key (TNF_VALUE_DOMAIN)	CharacterString	Identifies the structured value domain to which this property type belongs
VALUE_DOMAIN_OID Foreign Key (TNF_VALUE_DOMAIN)	CharacterString	Specifies the identity of the associated value domain (which may be simple or structured)
MULTIPLICITY_MIN	Integer	The minimum allowed number of values for this property type
MULTIPLICITY_MAX	Integer	The maximum allowed number of values for this property type
MANDATORY	Boolean	True if the property type is mandatory (i.e. an instance must contain a value for this property type)
NAME	CharacterString	A descriptive name for this property type
DESCRIPTION	CharacterString	A description/definition of this property type
SHORTNAME	CharacterString	A short/mnemonic name or code for this property type. *Note: Only alphanumeric characters (i.e. A-Z, a-z and 0-9) are recommended for compatibility reasons.*
VALID_FROM	Date	The date from which the property type is valid and in use

VALID_TO	Date	The date from which the property type is no longer valid and in use

3.4.8 TNF_VALID_VALUE

This class specifies sets of valid values for value domains. All value domains where "DATATYPE = Enum" shall have at least one instance of this class. Other value domains may have valid values.

Namn	**Typ**	**Anm.**
VALUE_DOMAIN_OID Primary Key Foreign Key (TNF_VALUE_DOMAIN)	CharacterString	Identifies the value domain for which this is a valid value
CATALOGUE_OID Primary Key Foreign Key (TNF_CATALOGUE)	CharacterString	Identifies the catalogue
DESCRIPTION	CharacterString	Description for the valid value
SEQ_NO Primary Key	Integer	A sequence number for keeping a certain sequence within the set of valid values for a value domain.
VALID_FROM	Date	Date from which this valid value is valid and in use.
VALID_TO	Date	Date from which this valid value is no longer valid and in use.
MIN_VALUE_DOUBLE	Real (Double precision) (CONDITIONAL)	Minimum value for real value interval
MAX_VALUE_DOUBLE	Real (Double precision) (CONDITIONAL)	Maximum value for real value interval
MIN_VALUE_INTEGER	Integer (CONDITIONAL)	Minimum value for integer interval
MAX_VALUE_INTEGER	Integer	Maximum value for integer interval

	(CONDITIONAL)	
MIN_VALUE_DATETIME	DateTime (CONDITIONAL)	Minimum value for date-/time interval
MAX_VALUE_DATETIME	DateTime (CONDITIONAL)	Maximum value for date-/time interval
ENUM_CODE	Integer (CONDITIONAL)	Code for enumerated value (this is the value that shall be stored for attribute instances). A textual definition is specified in the DESCRIPTION field
RANK	Integer (OPTIONAL)	Used in generalisation to specify which valid value is the least (low rank) and the most limiting (high rank).
VALUE_STRING	Text (CONDITIONAL)	Specifies a valid string value for a TNF_VALUE_DOMAIN whose DATATYPE is a CharacterString. *Note: The absence of any VALUE_STRING values for such a TNF_VALUE_DOMAIN indicates that any text value is valid. If VALUE_STRING values exist, only these values are valid.*

3.4.9 TNF_ SECONDARY_LRS

This class specifies any secondary linear referencing systems that may exist. In different applications, several and different linear referencing systems may exist. Examples may be road numbers and mileposts. TNF_SECONDARY_LRS contains metadata that serves as basis for a formal definition of an LRS which enables transformations between secondary and primary LRS and also between different LRS:s defined on top of the primary LRS. The basic idea is that property objects are defined which describe the individual linear elements and optionally also referents.

The following options exist for the definition of a secondary LRS:

1. Define a transport property object type which has a number of identifying attributes where each transport property object has network references in an identifiable order and direction. Each position in the secondary LRS is identified by using identification and geometric length.

2. Define a transport property object type which has a number of identifying attributes where each transport property object has network references in an identifiable order and direction and two attributes which define the measure (to be used as LRS measure) at the beginning and end of the accumulated extent. Each position in the

LRS is identified by specifying the identity and a measure which is linearly interpolated between the measure values.

3. Define two transport property object types, one with ordered and directed extents which define the linear (and measurable) feature. This transport property object shall have a number of attributes which identifies the linear feature. Also define an associated transport property object type where each transport property object instance has only one point network reference which specifies a reference marker (see ISO 19148). One attribute shall keep the measure value. The set of reference markers associated with the linear transport property object defines the ”length coordinate system” for the linear element.

Name	Type	Comment
OID Primary Key	CharacterString	A globally unique id for the LRS
NAME	CharacterString	A descriptive name for the LRS
TYPE	Integer	The type of LRS (according to listing above): 1=LRS with geometric measurement 2=LRS with start- and end measure 3=LRS with point referents
CATALOGUE_OID Foreign Key (TNF_CATALOGUE)	CharacterString	Catalogue for the linear property object type
PROPERTY_OBJECT_TYPE_OID Foreign Key (TNF_PROPERTY_OBJECT_TYPE)	CharacterString	Identification of the linear property object type
ASSOC_REFERENT_PROPERTY_OBJECT_TYPE_OID Foreign Key (TNF_PROPERTY_OBJECT_TYPE)	CharacterString (CONDITIONAL)	Identification of the point property object type whose instances are used as referents if LRS_TYPE=3
MEASURE1_PROPERTY_TYPE_OID Foreign Key (TNF_PROPERTY_OBJECT_PROPERTY_TYPE)	CharacterString (CONDITIONAL)	Name for the property type which contains the start position (LRS_TYPE=2) or the referent position (LRS_TYPE=3). This field is required if LRS_TYPE=2 or 3.

		Note: The datatype of the attribute (PROPERTY_OBJECT_PROPERTY_TYPE) type must be a numeric.
MEASURE2_PROPERTY_TYPE_OID Foreign Key (TNF_PROPERTY_OBJECT_PROPERTY_TYPE)	CharacterString (CONDITIONAL)	Name for the property type which contains the end position. This field is required if LRS_TYPE=2 *Note: The datatype of the attribute (PROPERTY_OBJECT_PROPERTY_TYPE)type must be a numeric.*
WHERE_CLAUSE	CharacterString (OPTIONAL)	May contain a limiting expression (limits the set of linear elements available)
SEQUENCE_PROPERTY_TYPE_OID Foreign key: TNF_PROPERTY_OBJECT_PROPERTY_TYPE	Text (OPTIONAL)	Name of a property that contains a sortable value, which shall act as a sequencer when sequencing many LRS instances.
ORDER_DESCENDING	Boolean (CONDITIONAL)	True if the sequencer describes a descending order. False if the sequencer describes an ascending order. Value must be provided if SEQUENCE_PROPERTY_TYPE_OID is set

3.4.10 TNF_SECONDARY_LRS_IDENTITY

This class defines the attributes that constitute the identity of each individual linear element for the given secondary LRS.

Name	Type	Comment
LRS_OID Primary Key Foreign Key (TNF_SECONDARY_LRS)	CharacterString	A globally unique id for the LRS
IDENTITY_PROPERTY_OID Primary Key Foreign Key (TNF_PROPERTY_OBJECT_PROPERTY_TYPE)	CharacterString	OID of a property type in the linear property object type that is a part of the identification of a single linear property object.

3.4.11 Examples

3.4.11.1 LRS type 1

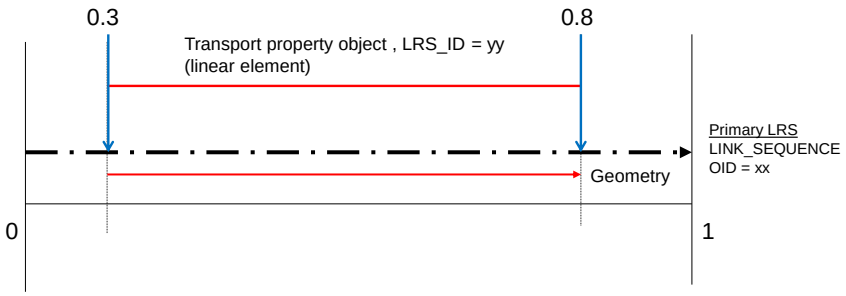


Figure 12 - Example for LRS type 1

A property object specifies its position relative to the primary LRS by using a network reference. The network reference specifies the interval 0.3-0.8 on link sequence = “xx”. In the example above, the primary LRS is defined within the interval 0-1. The geometry of the position of the property object may be derived from the geometry of the primary LRS in the interval 0.3-0.8. Also, the property object type has a property object property type “LRS_ID” which may be used as a unique identification. In this instance, the value of that property type is “yy”. A position referring to the secondary LRS has to specify:

- The identity of the LRS (TNF_SECONDARY_LRS.OID)
- The identity of the linear element (“yy”)
- A position relative to length of the derived geometry for the property object in the interval 0-L. The absolute position is calculated from a linear interpolation of the geometric length of the extent of the linear element. The rules for calculating geometric length (for example how z values are used) are defined within the actual implementation.

3.4.11.2 LRS type 2

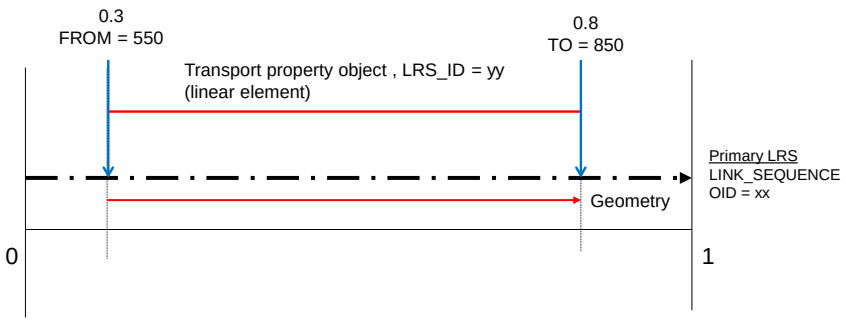


Figure 13 - Example of LRS type 2

A property object specifies its position relative to the primary LRS by using a network reference. The network reference specifies the interval 0.3-0.8 on link sequence = “xx”. In the example above, the primary LRS starts is defined within the interval 0-1. The geometry of the position of the property object may be derived from the geometry of the primary LRS in the interval 0.3-0.8. Also, the property object type has a property object property type “LRS_ID” which may be used as a unique identification. In this instance, the value of that property type is “yy”. The difference between type 2 and type 1 above is that with type 2, the property object type has two property types defining from measure and to measure. In the example above the property types are named “FROM” and “TO” and the values are 550 and 850. The relationship between the interval “TO”-“FROM” (850-550) and the underlying derived geometry is linear. This means that the geometry has a length of 300 units and the interval starts at 550.

A position referring to this secondary LRS has to specify:

- The identity of the LRS (TNF_SECONDARY_LRS.OID)
- The identity of the linear element (“yy”)
- A position in the interval “FROM”-“TO” (550-850). The absolute position is calculated from a linear interpolation of the interval 550-850 on the derived geometry of the property object.

3.4.11.3 LRS type 3

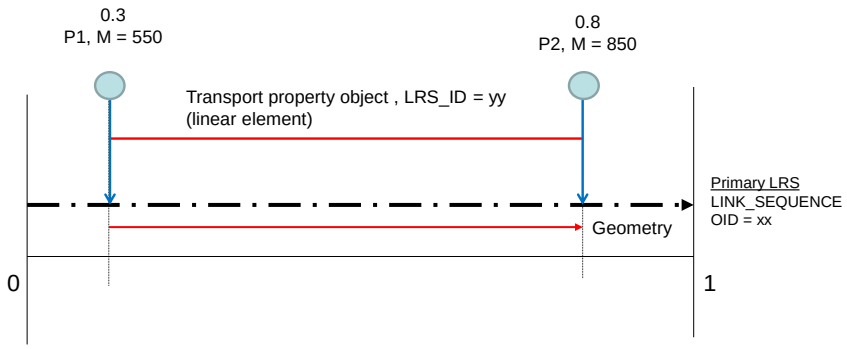


Figure 14 – Example of LRS type 3

A property object specifies its position relative to the primary LRS by using a network reference. The network reference specifies the interval 0.3-0.8 on link sequence = “xx”. In the example above, the primary LRS starts is defined within the interval 0-1. The geometry of the position of the property object may be derived from the geometry of the primary LRS in the interval 0.3-0.8. Also, the property object type has a property object property type “LRS_ID” which may be used as a unique identification. In this instance, the value of that property type is “yy”. The linear property object shall also have a list of property objects with point locations on the same underlying network elements. In the example above, the property objects “P1” and “P2” are associated with the linear property object “yy”. The property object type that defines “P1” and “P2” has a property type specifying the linear measure at the point location. In the example above the property type is “M” and the values for “P1” and “P2” is 550 and 850.

A position referring to this secondary LRS has to specify:

- The identity of the LRS (TNF_SECONDARY_LRS.OID)
- The identity of the linear element (“yy”)
- A position in the interval of measures specified by the list of associated point referents. In the example above this is 550-850. The absolute position is calculated from a linear interpolation of between the adjacent referents

3.5 Model for metadata

3.5.1 TNF_METADATA

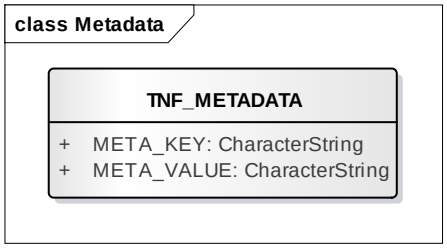


Figure 15 - Metadata overview

This class contains the dataset level metadata as a series of tags and values.

Name	Type	Comment
META_KEY Primary Key	CharacterString	Metadata key identifier
META_VALUE	CharacterString	Metadata value

A number of predefined keys exist:

META_KEY	META_VALUE
TNF_VERSION	The version of the specification to which the dataset complies
TNF_DATASET_IDENTIFIER	identifies the dataset
TNF_DATASET_TIMESTAMP	specifies when the dataset was created
TNF_HISTORY_DATA	"YES" indicates that the dataset contains historic data
TNF_CRS_NAME	Identification of the coordinate reference system used within the dataset (e.g. EPSG:xxxx)
TNF_DATASET_TYPE	"SNAPSHOT" or "UPDATES" If "UPDATES", the class (table) TNF_CHANGE_TRANSACTION shall have at least one instance (contain at least one row).
TNF_METADATA_XML (OPTIONAL)	a zipped (GZIP) and base64-encoded xml metadata document, preferably according to ISO 19139 (http://www.isotc211.org/schemas/2005/gmd/gmd.xsd).

TNF_AGORA_VERSION	Specifies the version of AGORA-C that is used for dynamic location references (if AGORA-C is used)
TNF_OPENLR_VERSION	Specifies the version of OPENLR that is used for dynamic location references (if OPENLR is used)
TNF_SPATIAL_ATTRIBUTE_ENCODING	Specifies the encoding format to use when storing spatial attribute values. Possible values: • **WKT**: Well Known Text format • **WKB**: Well Known Binary format • **GML**: Geometry Markup Language (version 3.2 or later) This key is required only if any spatial attribute values are included in the dataset. *Note: WKT and WKB definitions available at:* http://www.opengeospatial.org/standards/sfa *GML definition available at:* http://www.opengeospatial.org/standards/gml
TNF_VIEW_DATE	If present, specifies the view date to which the OpenTNF dataset corresponds. Shall be specified as a DateTime value compliant with ISO 8601, e.g. according to https://www.w3.org/TR/NOTE-datetime *Note: The presence of this value indicates that data valid for other points in time may be absent.*

3.6 Model for update transactions

This data is present only if the DATASET_TYPE is “UPDATES”.

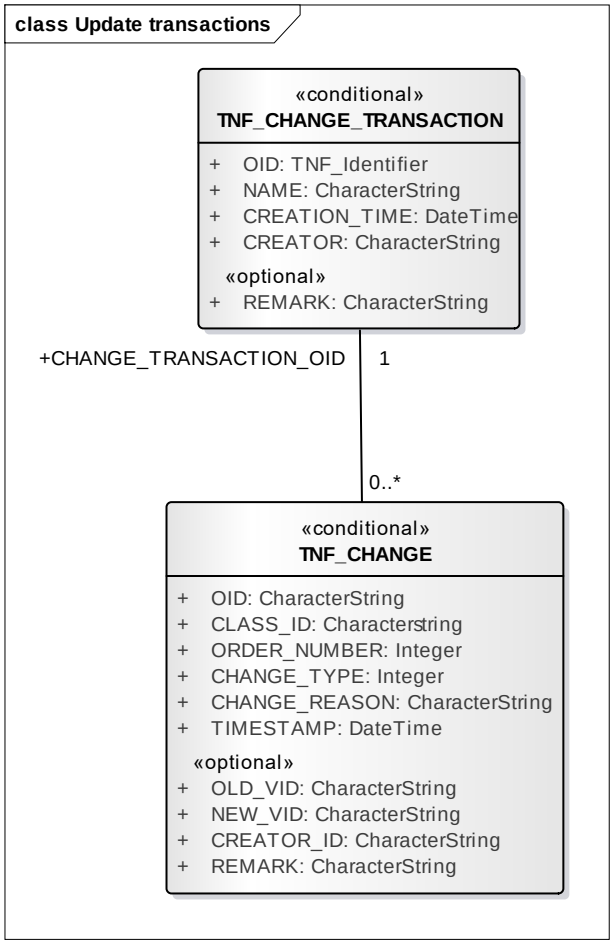


Figure 16 - Update transactions overview

3.6.1 TNF_CHANGE_TRANSACTION (CONDITIONAL)

A change transaction represents a set of changes (or updates) that are to be treated as an atomic operation ("all or nothing").

Name	Type	Comment
OID Primary Key	CharacterString	Globally unique transaction id.
NAME	CharacterString	Descriptive name for the transaction
CREATION_TIME	DateTime	Specifies when the transaction was created in the source dataset
CREATOR	CharacterString	Username for the originator of the transaction

REMARK	CharacterString (OPTIONAL)	An optional remark for the transaction

3.6.2 TNF_CHANGE (CONDITIONAL)

A change represents an update event that occurred to an object in the source dataset. A change always occurs in the context of a change transaction.

Name	Type	Comment
OID Foreign Key (TNF_LINK_SEQUENCE,TNF_LINK, TNF_NODE, TNF_PROPERTY_OBJECT, TNF_NETWORK, TNF_NETWORK_CONNECTION)	CharacterString	Refers to the OID of the changed object
CLASS_ID	CharacterString	Identifies the class of object that has been changed: - LINK_SEQUENCE - LINK - NODE - PROPERTY_OBJECT CATALOGUE_OID PROPERTY_OBJECT_TYPE_OID Example: "PROPERTY_OBJECT/ABC123/12345", where "ABC123" is the CATALOGUE_OID and "12345" is the OID of the transport property object type - NETWORK - NETWORK_CONNECTION - TOPOLOGY_LEVEL
CHANGE_TRANSACTION_OID Foreign Key (TNF_CHANGE_TRANSACTION)	CharacterString	Identifies the transaction of which this Change is a part

ORDER_NUMBER	Integer	The sequence number for the change in the context of the transaction.
CHANGE_TYPE	Integer	Update type: 0=A comment. This type may be used to insert free text (in the REMARK) instances in the update transaction. 1=Create/Insert 2=Modify/Update 3=Delete
CHANGE_REASON	CharacterString	The reason for the change. Allowed values are: - Correction Used when data has been updated to represent the real world more correctly - Real world Used when data has been updated to reflect a real world change - Unknown Used when the reason for update is unknown
TIMESTAMP	DateTime	A timestamp indicating the point in time when the change occurred in the source dataset
OLD_VID	CharacterString (OPTIONAL)	Refers to the previous version id of the changed object. The object in the receiving dataset should have this version. This value may be defined if the change concerns a modification or deletion (CHANGE_TYPE = {2,3}) and that the data exchange scheme uses versioning of objects.
NEW_VID	CharacterString (OPTIONAL)	The next version id of the object. The receiving dataset is expected to update its version id for the object to this value. This value may be defined if the change concerns a creation or modification (CHANGE_TYPE = {1,2}) and that the data exchange scheme uses versioning of objects.
CREATOR_ID	CharacterString (OPTIONAL)	Identification of the originator of the change. *Note: The content has to be agreed between the parties involved in the exchange.*

REMARK	CharacterString (OPTIONAL)	Remark

4 SUPPLEMENT 1 – Specializations for Swedish/Norwegian national road databases

4.1 Model for transport network

In the model for transport networks support for the port concept is added. The concept is defined and described in SS637004, class *NW_Port* and subclasses.

Furthermore, the concept of topology levels is introduced.

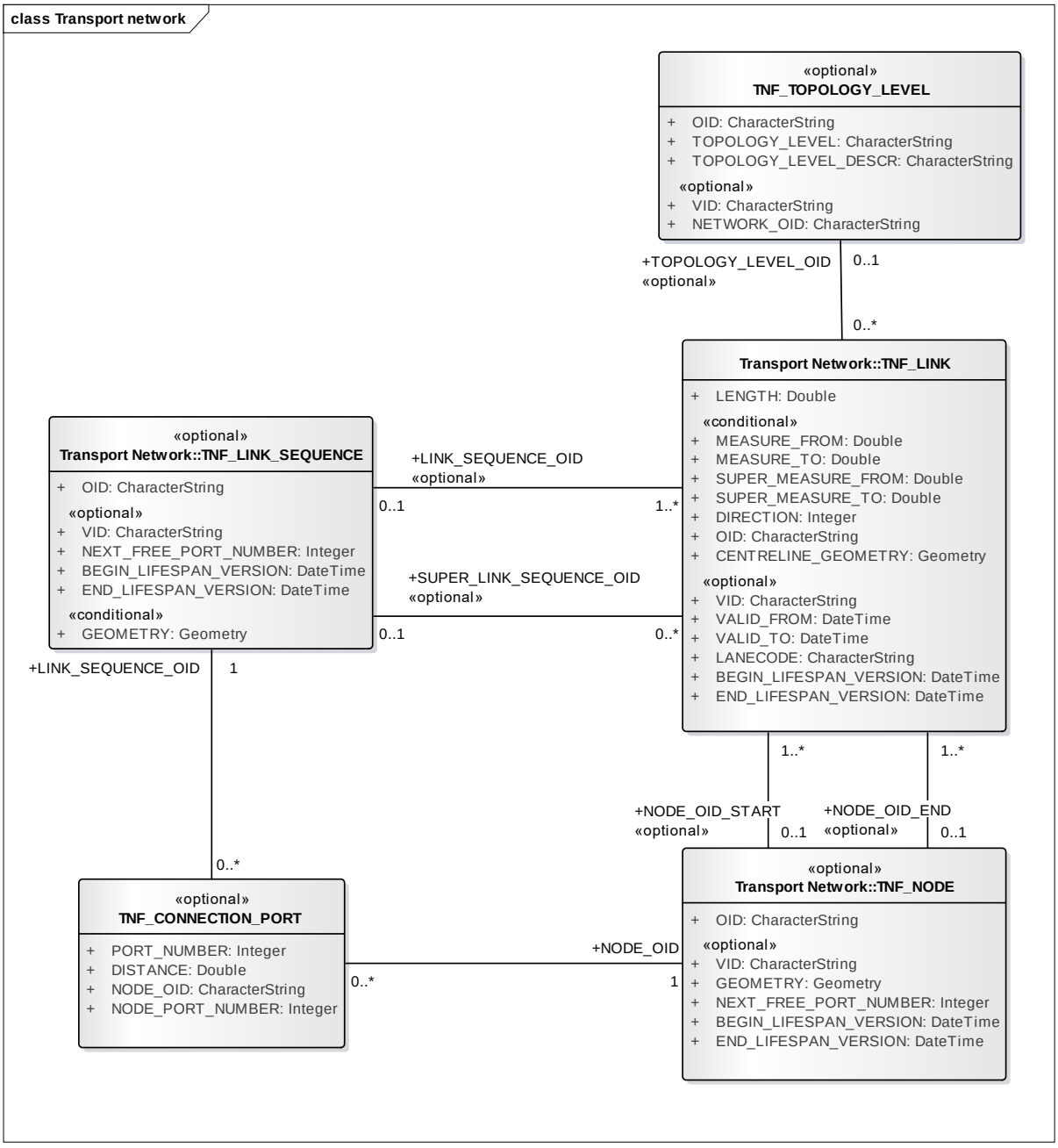


Figure 17 - Transport network specializations overview

4.1.1 TNF_LINK_SEQUENCE (OPTIONAL)

An attribute NEXT_FREE_PORT_NUMBER is added. The purpose is to be able to generate new and unique port numbers per link sequence instance since port numbers may not be reused.

Name	**Type**	Comment
NEXT_FREE_PORT_NUMBER	Integer (OPTIONAL)	Next free port number (see SS637004). *Note: A port number may never be reused within the same link sequence.*

4.1.2 TNF_LINK

An attribute LANECODE is added according to the table below. Links also supports representations of different topological levels in the same or different datasets.

Each link may be related to a topological level (TNF_TOPOLOGY_LEVEL). Each link may also specify a mapping to a corresponding part of a link sequence on another topological level.

Name	**Type**	**Comment**
LANECODE	CharacterString (OPTIONAL)	Indicates (using user defined notation) what lanes that exist on this transport link.
SUPER_LINK_SEQUENCE_OID Foreign Key (TNF_LINK)	CharacterString (CONDITIONAL)	Specifies if applicable a corresponding link sequence on another topology level to which this transport link corresponds.
SUPER_ MEASURE_FROM	Real (Double precision) (CONDITIONAL)	Specifies the start point on the corresponding transport link sequence to which the start point of this transport link is mapped. The value shall be set if SUPER_OID is not null.
SUPER_ MEASURE_TO	Real (Double precision) (CONDITIONAL)	Specifies the end point on the corresponding transport link sequence to which the end point of this transport link is mapped.

		The value shall be set if SUPER_OID is not null.
DIRECTION	Integer (CONDITIONAL)	Specifies if the direction of this transport link is the same (=1) or the opposite (=2) of the direction of the transport link sequence referenced by SUPER_OID. The value shall be set if SUPER_OID is not null.
TOPOLOGY_LEVEL_OID Foreign Key (TNF_TOPOLOGY_LEVEL)	CharacterString (OPTIONAL)	Specifies the topology level for this transport link.

EXAMPLE A dataset may represent both the road and the carriageway levels in a case where the road consists of multiple carriageways (for example a motorway).

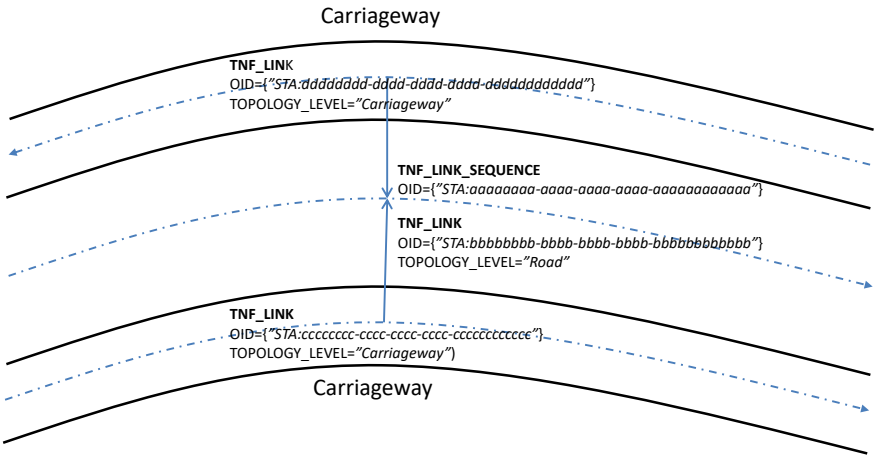


Figure 18 - Topological levels example

In the example above, the two carriageways are each represented by a link (OID {*"STA:cccccccc-cccc-cccc-cccc-cccccccccccc"*} and {*"STA:dddddddd-dddd-dddd-dddd-dddddddddddd"*}). The road level is represented by a link sequence (OID {*"STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"*}) consisting

of one link (OID *{"STA:bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb"}*). The relationship between the carriageway level and the road level is specified using the SUPER_LINK_SEQUENCE_OID, SUPER_MEASURE_FROM and SUPER_MEASURE_TO attributes. These attributes maps the lower level links to the higher level link sequence and provides also the parameters needed to map linear positions and directions between the levels. SUPER_MEASURE_FROM and SUPER_MEASURE_TO shall be linearly interpolated in relation to MEASURE_FROM and MEASURE_TO on the super (=Road) level. If link sequence at road level has MEASURE_FROM=0 and MEASURE_TO=1, this means that the carriageway links maps to the whole road link. The link with OID={*"STA:dddddddd-dddd-dddd-dddd-dddddddddddd"*} is directed opposite to the road link sequence.

The tables below shows how this example could be represented:

TNF_LINK_SEQUENCE:

Attribute	**Value**	**Comment**
OID	{*"STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"*}	

TNF_LINK:

Attribute	**Value**	**Comment**
OID	*{"STA:bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb"}*	
LINK_SEQUENCE_OID	{*"STA: aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"*}	
TOPOLOGY_LEVEL	"Road"	
OID	{*"STA:cccccccc-cccc-cccc-cccc-cccccccccccc"*}	
SUPER_LINK_SEQUENCE_OID	{*"STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"*}	
DIRECTION	1	
TOPOLOGY_LEVEL	"Carriageway"	

OID	{*"STA:dddddddd-dddd-dddd-dddd-dddddddddddd"*}	
SUPER_LINK_SEQUENCE_OID	{*"STA:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa"*}	
DIRECTION	2	
TOPOLOGY_LEVEL	"Carriageway"	

4.1.3 TNF_NODE (OPTIONAL)

An attribute NEXT_FREE_PORT_NUMBER is added. The purpose is to be able to generate new and unique port numbers per node instance since port numbers may not be reused.

Name	Type	Comment
NEXT_FREE_PORT_NUM BER	Integer (OPTIONAL)	Next free port number (see SS637004). *Note: A port number may never be reused within the same node.*

4.1.4 TNF_CONNECTION_PORT (OPTIONAL)

A connection port is used for compatibility with SS 637004 since topological connectivity and the specification of LRS (relative lengths) for the linear reference system is handled by transport link sequence, transport links and transport nodes.

Name	Type	Comment
LINK_SEQUENCE_OID Primary Key Foreign Key (TNF_LINK_SEQUENCE)	CharacterString	OID for the transport link sequence (link sequence) to which the port belongs
PORT_NUMBER Primary Key	Integer	The official number (unique within the link sequence) of the port.
DISTANCE	Real (Double precision)	A relative position (measure) on the transport link sequence within the interval ([0,1]).
NODE_OID	CharacterString	OID for the connected transport node.

Primary Key Foreign Key (TNF_NODE)		
NODE_PORT_NUMBER Primary Key	Integer	The official number (unique within the node) for the connected port.

4.1.5 TNF_TOPOLOGY_LEVEL (OPTIONAL)

A topology level specifies classification of topology levels for a transport link.

Name	Type	Comment
OID Primary Key	CharacterString	An id
TOPOLOGY_LEVEL	CharacterString	A descriptive and short name for the topology level.
TOPOLOGY_LEVEL_DESCR	CharacterString	A description of the topology level.
VID	CharacterString (OPTIONAL)	Globally unique version id *Note: Whenever the data has changed for the object it shall receive a new version id regardless if the data is changed due to real world changes or corrections of the data.*
NETWORK_OID Foreign Key (TNF_NETWORK)	CharacterString (OPTIONAL)	OID for the transport network for which this topology level is valid. May be missing if table TNF_NETWORK is not used and therefore only one network occurs in the dataset.

4.2 Generic model for attribution and features related to the road network

4.2.1 TNF_NETWORK_REFERENCE

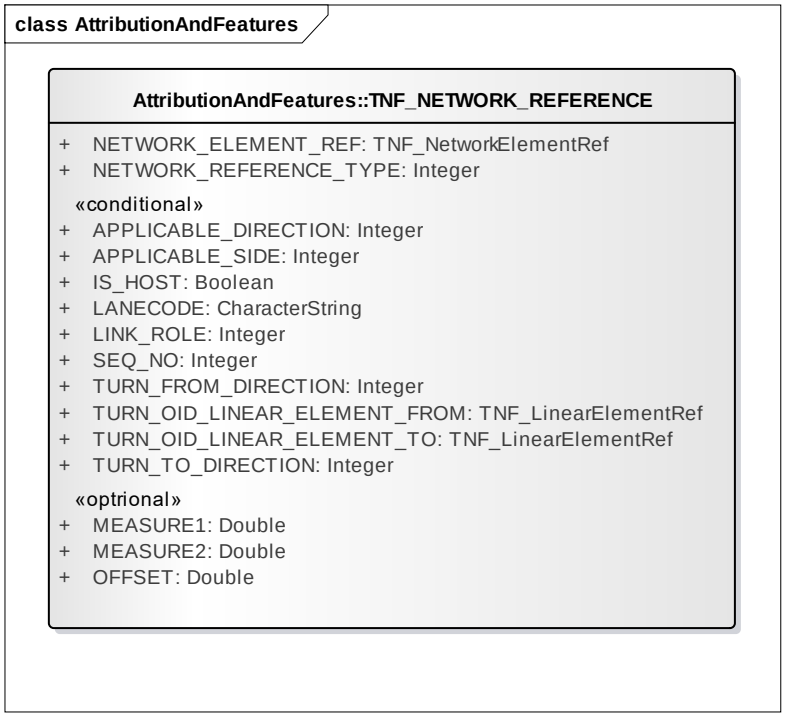


Figure 19 - TNF_NETWORK_REFERENCE specialization

Support for network references of type road and road with host are added. This corresponds to the class *NW_RoadExtent* in SS637004.

Name	Type	Comment
NETWORK_REFERENCE_TYPE	Integer	The following types of network references are added: 16 : Road 256 : RoadWithHost *Note: These network reference types are explained in the specification for the Swedish National Road database, se http://www.trafikverket.se/.*

LANECODE	CharacterString (CONDITIONAL)	Specifies a lane for which this network reference is valid. Shall be specified if the corresponding transport property object type specifies that the type is lane dependent. Note: The notation for specifying lane codes is outside the scope for this specification and has to be defined, agreed and described separately (e.g. a separate agreement between parties involved in the data exchange)
APPLICABLE_DIRECTION	Integer (CONDITIONAL)	This attribute is already defined in chapter 3.3.4. The list of NETWORK_REFERENCE_TYPE when this attribute is applicable is extended with {16 and 256}
LINK_ROLE	Integer (CONDITIONAL)	Link role for NETWORK_REFERENCE _TYPE={16,256}. The possible values are: 1 = Normal 2 = Sibling forward 3 = Sibling backwards 4 = Branch
IS_HOST	Boolean (CONDITIONAL)	True if property object (e.g. a road) is the host for the underlying linear element. A value shall be specified for NETWORK_REFERENCE _TYPE={256}
MEASURE1	Real (double precision) (OPTIONAL)	This attribute is already defined in chapter 3.3.4. The list of NETWORK_REFERENCE_TYPE when this attribute is applicable is extended with {16 and 256}
MEASURE2	Real (double precision) (OPTIONAL)	This attribute is already defined in chapter 3.3.4. The list of NETWORK_REFERENCE_TYPE when this attribute is applicable is extended with {16 and 256}

4.3 Generic model for transport property object catalogues

Support for network references of type road and road with host are added. This corresponds to the class *NW_RoadExtent* in SS637004.

4.3.1 TNF_PROPERTY_OBJECT_TYPE

A property object type is a definition of a transport property object.

Name	Type	Comment
NETWORK_REFERENCE_TYPE	Integer	Specifies the type of network references that are allowed for property objects of this type. If more than one NETWORK_REFERENCE_TYPE is allowed, the values are added: 16 : Road 256 : RoadWithHost