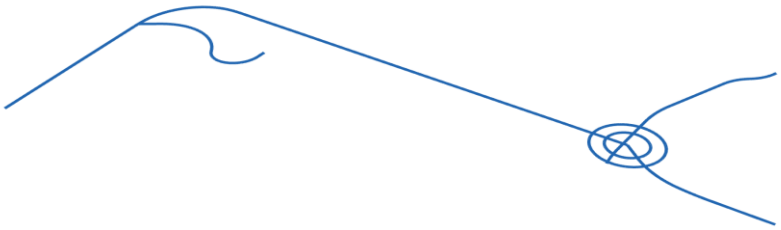
OKSTRA Internationalization



Julian Amann, André Borrmann
Chair of Computational Modeling and Simulation
Leonhard Obermeyer Center
Technical University of Munich

on behalf of the

Federal Highway Research Institute (Bundesanstalt für Straßenwesen)
Bergisch Gladbach

Contents

What is OKSTRA?

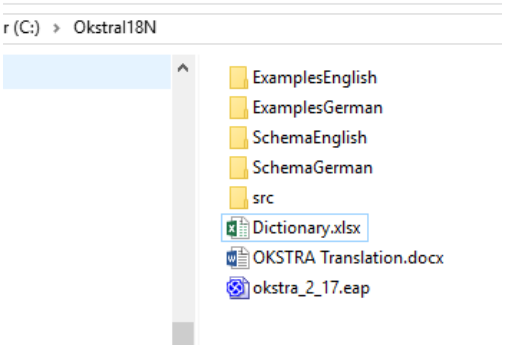
The German OKSTRA standard (Objektkatalog für das Strassen- und Verkehrswesen = Object catalog for road and traffic related data) unifies the data description of objects from traffic engineering and contains a data model for alignment data. It is owned by BASt (German Federal Highway Research Institute) and distributed under an open license that permits its use for commercial projects. The main design goal of OKSTRA is the simple data exchange between different applications that implement the standard. The current OKSTRA standard (version 2.017) was published in April 2016. In the past, the OKSTRA standard was provided as an EXPRESS schema along with an XSD schema, but the current version provides the data model solely as an XSD schema. The CTE data format (STEP-based data format to store EXPRESS-based instance files) was also retired. The OKSTRA developers moved from EXPRESS to XSD, from NAIM Diagrams (like EXPRESS-G Diagrams) to UML and from STEP for storing instance files (also called CTE files in the OKSTRA standard) to XML. The official website of the OKSTRA standard is http://www.okstra.de/.

Purpose of this document

The OKSTRA standard was developed for the German market. As such, the official XSD schema specifies all identifiers in German and the documentation is also only available in German. This limits its potential to be considered as a good candidate for an international standard. This document represents a first translation of a subset of the OKSTRA standard to make it internationally accessible to experts from other countries. Since road design is being targeted in current developments by organizations such as buildingSMART or the Open Geospatial Consortium (OGC), this translation focuses primarily on road design.

Accompanying files

This document is accompanied by several files:

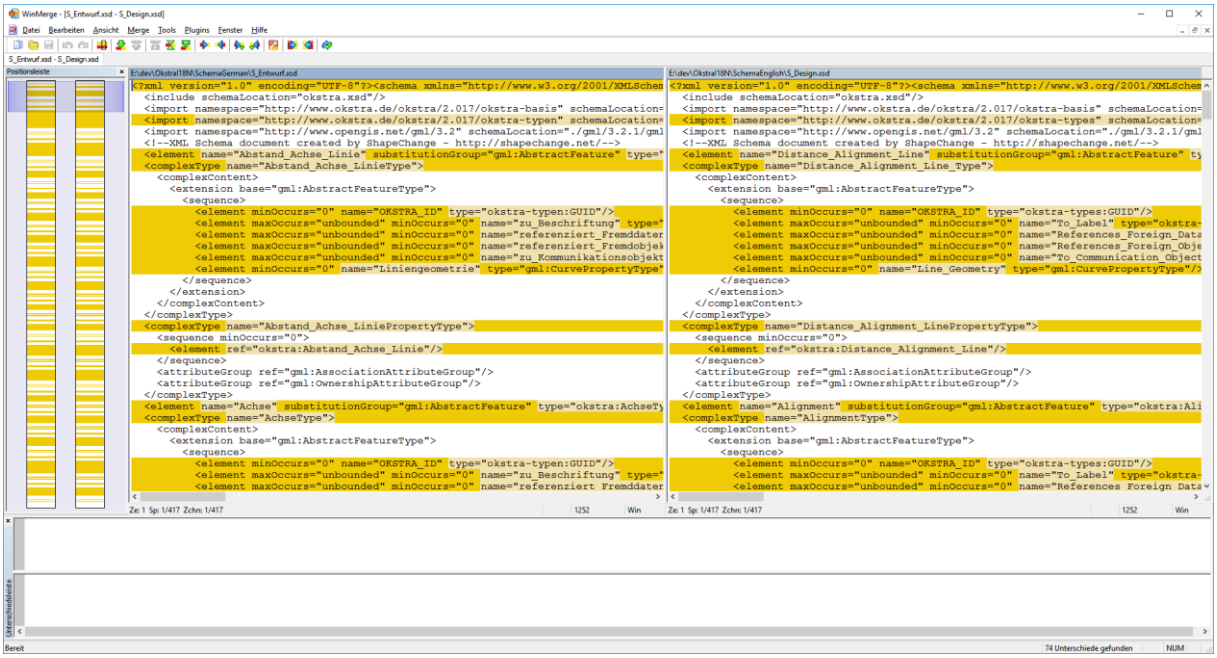


- The folder “SchemaGerman” contains the original non-modified OKSTRA XML schema (version 2.017)
- The folder “SchemaEnglish” contains a modified OKSTRA XML schema (version 2.017) that has been partially translated to the English language
- The folder “ExamplesGerman” contains some example files for basic road design using the original non-modified German OKSTRA Standard/XML Schema
- The folder “ExamplesEnglish” contains some example files for basic road design using the modified OKSTRA XML schema that has been partially translated to the English language
- The folder “src” contains tools as C# Code that have been developed for the translation of the schema and instance files
- “Dictionary.xlsx” provides an excel sheet overview of the most important translated items

- “okstra_2_17.eap” is an Enterprise Architect project that contains all the UML schema shown in this document

Direct schema comparison

To compare the German schema against the English version, simply use a diff tool such as WinMerge or diff on a Unix-based system to reveal the differences.



The screenshot shows that the German term “Abstand_Achse_Linie” has been translated to “Distance_Alignment_Line”.

Naming conventions for translations

Underscores are used to separate different words in identifiers. As the German language has long compound words, these have where necessary been translated as multiple words. For example, *S_Allgemeine_Geometrieobjekte* has been translated as *S_General_Geometry_Objects*. Different words are always separated by an underscore.

German and English also handle upper and lower case differently. Here we have elected to always capitalize the first letter of each word. For example, *fachliche_Bedeutung* becomes *Technical_Meaning*, and *allgemeines_Linienobjekt* is translated as *General_Line_Object*.

> *Note:* With a more GML-like naming convention, *fachliche_Bedeutung* would be translated to *TechnicalMeaning* without an underscore. We have, however, elected to continue using underscores to remain as close as possible to the original OKSTRA naming convention.

Schema file translation

The resources that come with this document include the German (folder “SchemaGerman”) schema along with a partial English translation of the schema (folder “SchemaEnglish”). The following table provides an overview of how the schema names were translated. As mentioned above, this translation focuses on individual subschemas of the entire schema. Large sections of the schema have therefore not yet been translated – the blue highlighting of the subschemas/rows indicate the areas we have concentrated on in this translation:

German	English
Datentypen.xsd	Data_Types.xsd
okstra.xsd	okstra.xsd
okstra-basis.xsd	okstra-basis.xsd
okstra-typen.xsd	okstra-types.xsd
S_Administration.xsd	S_Administration.xsd
S_Allgemeine_Geometrieobjekte.xsd	S_General_Geometry_Objects.xsd
S_Allgemeine_Mengenberechnung.xsd	S_General_Quantity_Calculation.xsd
S_Allgemeine_Objekte.xsd	S_General_Objects.xsd
S_Arbeitsstelle_an_Strassen.xsd	S_Work_Place_On_Road.xsd
S_Bauliche_Strasseneigenschaften.xsd	S_Structural_Road_Properties.xsd
S_Bauwerke.xsd	S_Buildings.xsd
S_Dynamische_Beschilderung.xsd	S_Dynamic_Signage.xsd
S_Dynamische_Verkehrsdaten.xsd	S_Dynamic_Traffic_Data.xsd
S_Entwurf.xsd	S_Design.xsd
S_Flaechenmodell.xsd	S_Area_Model.xsd
S_Grunderwerb.xsd	S_Land_Acquisition.xsd
S_Hausnummern.xsd	S_House_Numbers.xsd
S_Historisierung.xsd	S_Historicisation.xsd
S_Kataster.xsd	S_Land_Register.xsd
S_Kostenmanagement.xsd	S_Cost_Management.xsd
S_Kreuzungen.xsd	S_Junctions.xsd
S_Landschaftsplanung.xsd	S_Landscape_Planning.xsd
S_Lichtsignalanlage.xsd	S_Traffic_Light_System.xsd
S_Liegenschaftsverwaltung.xsd	S_Property_Management.xsd
S_Netzaenderungsprotokoll.xsd	S_Net_Activity_Log.xsd
S_Oekologie.xsd	S_Ecology.xsd
S_Organisation.xsd	S_Organisation.xsd
S_Projektressourcen.xsd	S_Project_Resources.xsd
S_REB_22013.xsd	S_REB_22013.xsd
S_Schwertransport.xsd	S_Heavy_Transport.xsd
S_Statische_Beschilderung.xsd	S_Static_Signage.xsd
S_Strassenausstattungen.xsd	S_Road_Equipment.xsd
S_Strassennetz.xsd	S_Road_Network.xsd
S_Strassenverzeichnis.xsd	S_Road_Directory.xsd
S_Strassenzustandsdaten.xsd	S_Road_Condition_Data.xsd
S_Topografie.xsd	S_Topography.xsd
S_Umfeldmessstelle.xsd	S_Environment_Monitoring_Station.xsd
S_Unfall.xsd	S_Accident.xsd
S_Verkehr.xsd	S_Traffic.xsd
S_Vermessungspunkt.xsd	S_Triangulation_Station.xsd
Schluesseltabellen.xsd	Key_Tables.xsd

Encoding

XML instance files of the German OKSTRA Standard are encoded as ISO-8859-1.

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!-- generiert von OKLABI, (c) 2009-2016 Bundesanstalt für Straßenwesen -->
```

For the translation, we have switched to UTF-8.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- generated by TUM Open Infra Platform, based on OKLABI (c) 2009-2016 Bundesanstalt für Straßenwesen -->
```

More details on UTF-8 and other encodings can be found here: http://utf8everywhere.org.

UML Diagrams

The following section shows the most important parts of our OKSTRA translation in the form of UML diagrams.

Package S_General_Geometry_Objects (S_Allgemeine_Geometrieobjekte)

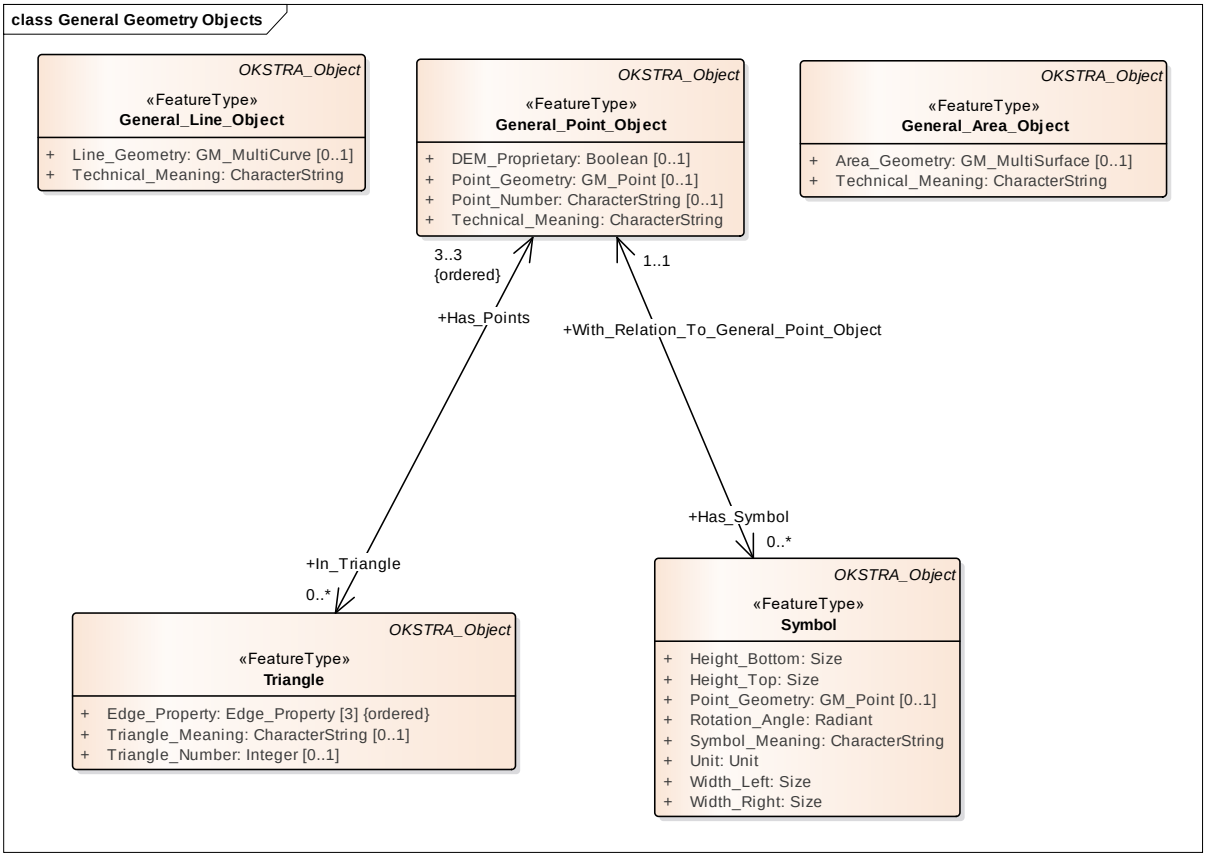
This package contains general geometry objects, the object type DGM (DEM = Digital Elevation Model) as well as different object types that are associated with plan representations.

The general geometry objects such as *General_Point_Object*, *General_Line_Object* and *General_Area_Object* extend the ability of OKSTRA to exchange geometric information.

Each general geometry object has, in addition to a geometry attribute based on the type of geometry chosen, an attribute entitled “Technical_Meaning” that can be used to store the technical meaning of the object. The possible values of the attributes are defined in so-called OKSTRA-Fachbedeutungslisten (lists of technical meanings). These lists are published by the OKSTRA maintainers.

General geometry objects can be seen as an interim solution for objects that are not defined by the OKSTRA standard and should only be used if OKSTRA does not provide a specific object type.

General Geometry Objects (Allgemeine Geometrieobjekte)



General_Line_Object

An object type for the representation of a line with an annotated technical meaning.

General_Point_Object

An object type for the representation of a point with an annotated technical meaning.

In contrast to the *General_Line_Object* and *General_Area_Object*, the *General_Point_Object* has the following additional properties:

1. A *General_Point_Object* can have a point number (via the attribute "Point_Number").
2. A *General_Point_Object* can be a vertex of a triangle within a DEM.
3. The *General_Point_Object* has an additional attribute "DEM_Proprietary" that defines if a *General_Point_Object* is only valid within a DEM or globally valid (i.e. in a record).If the attribute *"*DEM_Proprietary" is not defined, then the *General_Point_Object* is globally valid. A defined point number must only be unique in the case of a propriety DEM. Otherwise it must be unique in the global view.
4. A *General_Point_Object* can have a *Symbol* in the case where the location of the representation is translated or the RAS-Verm (Richtlinien fuer die Anlage von Strassen: RAS - Teil: Vermessung = Road construction directive – part survey) requires a different symbol.

General_Area_Object

An object type for the representation of an area object with an annotated technical meaning.

Triangle

An object type for the representation of a triangle in the *DEM*.

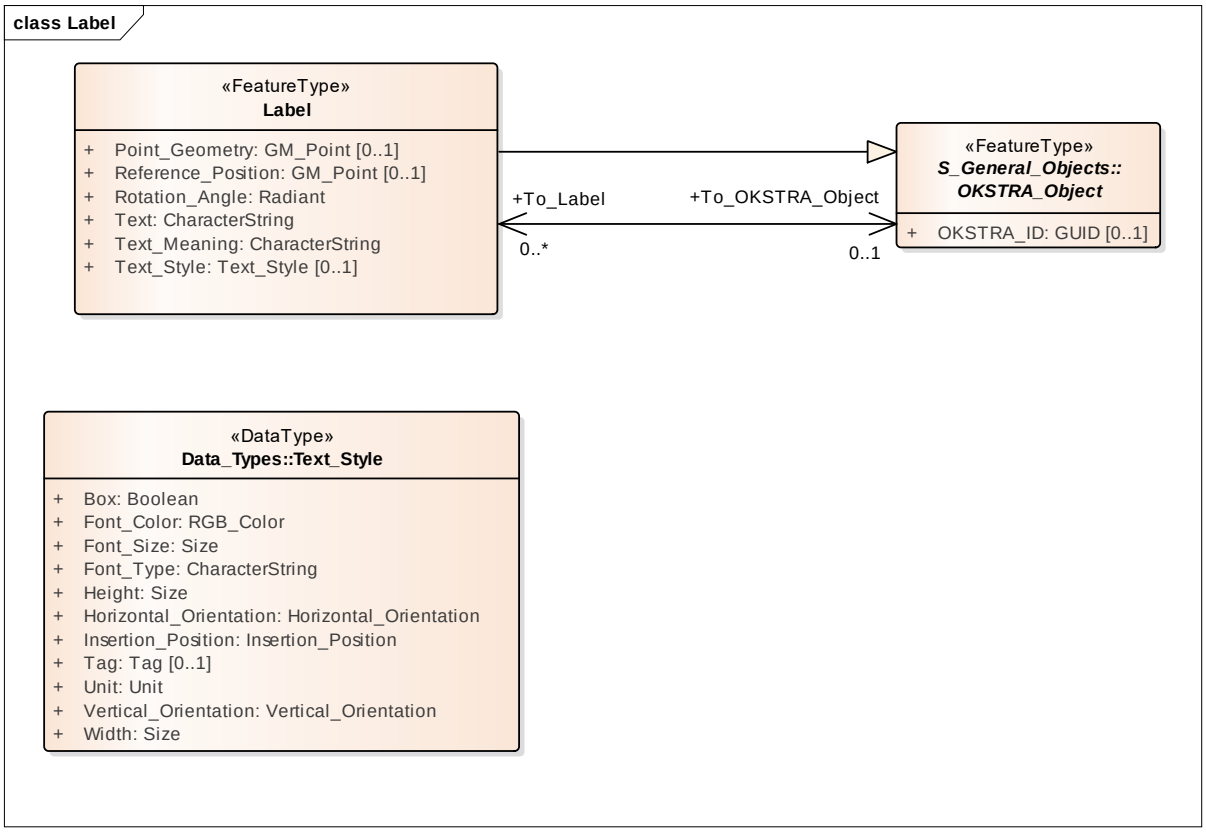
The vertices of a *Triangle* must be specified as *General_Point_Object* objects. For every of the three edges of the *Triangle* an *Edge_Property* must be defined that determines whether an edge is a normal edge, a break line or a form line.

The indication of the *General_Point_Object's* and *Edge_Property's* is done in an ordered set.

Using the positions in both lists, a *General_Point_Object* correlates to an *Edge_Property.* The following convention applies: An *Edge_Property* relates to the edge that is opposite the corresponding *General_Point_Object* located at the corresponding list position.

Example: The second edge property relates to the triangle edge that is opposite the second *General_Point_Object.*

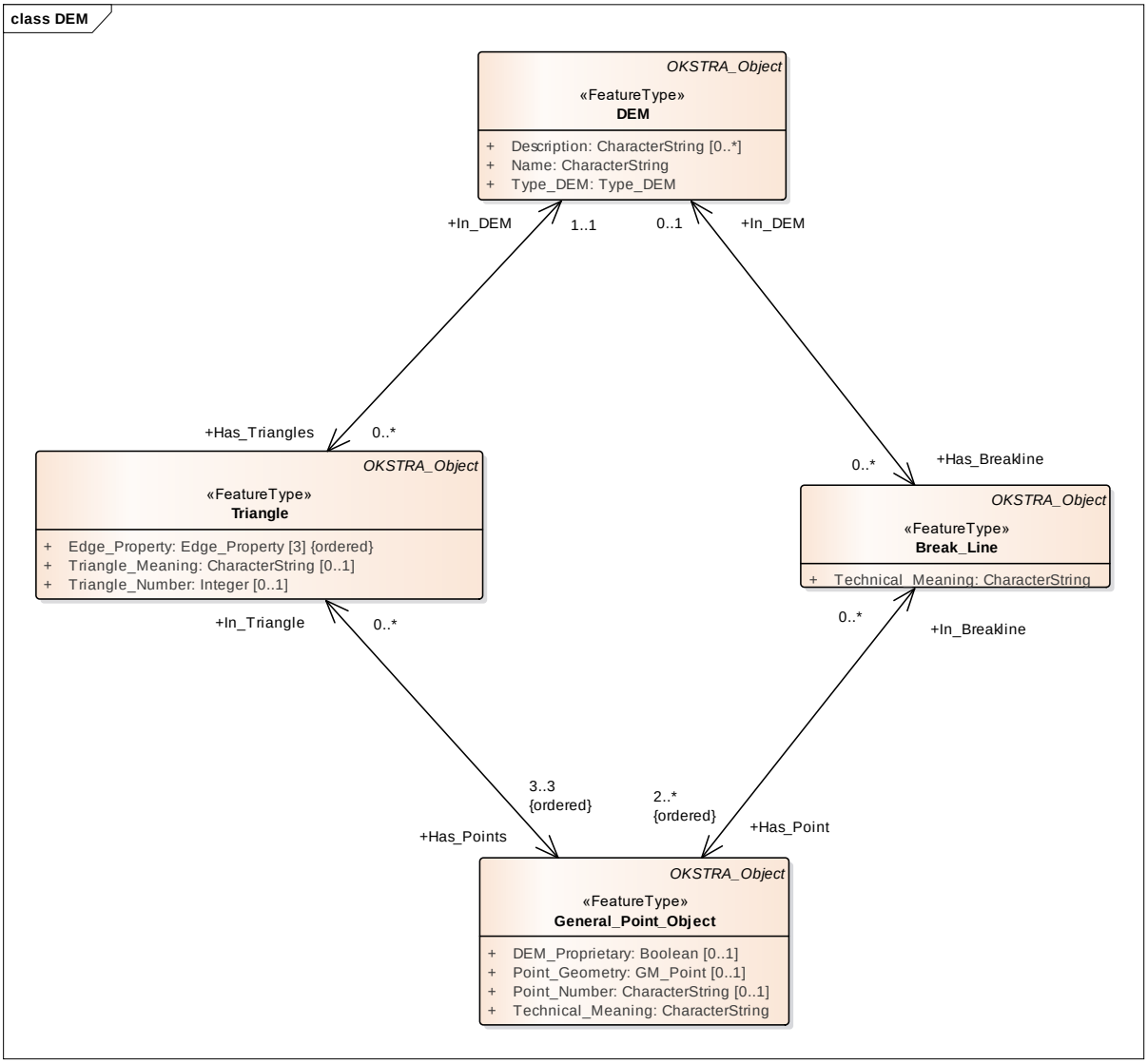
Label (Beschriftung)



Label

An object type for specifying an annotation text in a map presentation.

DEM (DGM)



Breakline

An object type for representing a break line. A break line is geometrically defined over an ordered set of *General_Point_Object* objects. If the break line belongs to a DEM, the general point objects must at the same time be components of the *DEM* (or its triangles) and must be within the triangle edge bounds.

DEM

An object type for the representation of a digital elevation model (*DEM*).

A *DEM* consists of triangles and, if necessary, also break lines, with each triangle belonging to exactly one *DEM*. Using the key table *Type_DEM* the technical meaning of a *DEM* can be described in more detail.

A triangle can have a positive integer triangle number. If triangle numbers are specified, they must be unique across the entire DEM. In addition, a “triangle meaning” can optionally be specified for a triangle (as a text string).

A triangle is defined by a list of three *General_Point_Objects*. In addition, a triangle has a list of three edge properties (key table), whereby the convention is that the specified edge property refers to the triangle side opposite to the corner of the point located at the same position in the point list (the

second edge property refers to the triangular edge which is opposite to the second *General_Point_Object*). The possible values of the key table edge property are “normal side”, “break line”, and “shape line”.

General_Point_Object

An object type for the representation of a point with an attached technical meaning.

Compared to the *General_Line_Object* object and the *General_Area_Object* object, the *General_Point_Object* object has the following additional properties:

1. A point number can be specified for a *General_Point_Object* (attribute “point number”).
2. A *General_Point_Object* can be a vertex of a triangle within a *DEM*.
3. The General_Point_Object has the additional attribute “DEM_proprietary”. This attribute can be used to specify whether a *General_Point_Object* object is valid only within the scope of a specified DEM or globally (i.e. within a complete dataset). If the attribute "DEM_proprietary" is not specified, the *General_Point_Object* object is valid globally. A specified point number must be unique in the DEM-proprietary case only within one DEM, but in the other case globally unambiguous.
4. A *General_Point_Object* can have a symbol if the representation is translated, or a symbol graphic when the RAS-Verm is to be selected.

Triangle

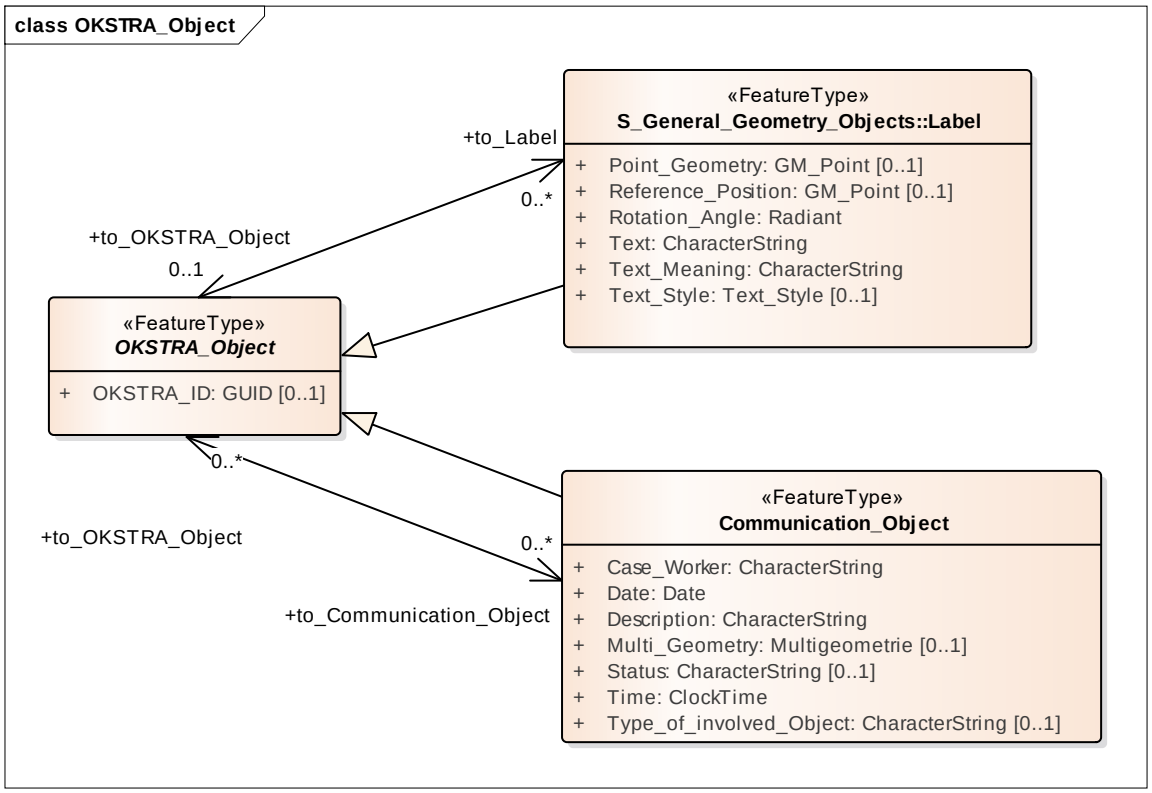
An object type for representing a triangle of a DEM.

The vertices of a triangle must be specified as *General_Point_Object* objects. For each of the three sides of a triangle, a lateral feature must be used to indicate whether the side is a normal side, a break line, or a shape line.

The specification of the *General_Point_Object* and the side properties are arranged in each case in order. Via the positions in the two lists, a *General_Point_Object* object and a side property are assigned to each other. In this case, the convention applies that a side property refers to the side opposite to the general point object assigned via the corresponding list position.

Package S_General_Objects (S_Allgemeine_Objekte)

OKSTRA_Object (OKSTRA_Objekt)



Communication_Object

An object type to support communications in data exchange and the transmission of requests, indications, etc. from the sender to the recipient of the data. A *Communication_Object* can be located by an arbitrarily definable point, line or plane geometry and can have explicit references to any other objects (a relation to an *OKSTRA_Object*).

OKSTRA_Object

An abstract Supertype from which all OKSTRA object types inherit directly or indirectly. The *OKSTRA_Object* has the optional attribute “OKSTRA_ID” which is a Global Unique Identifier (GUID) with a length of 128 bits (corresponding to 32 characters).

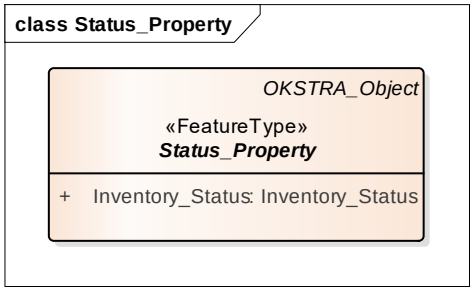
Within a GUID, the following characters are allowed: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The advantage of GUIDs is that they can be generated decentrally, and are at the same time also unambiguous.

In the case of versionable object types, all versions of an object, that is all instances describing the state of the same object in different time periods, are assigned the same GUID (if given). In a system that supports OKSTRA historization, the “OKSTRA_ID” alone is not sufficient to uniquely identify the object, but the combination of “OKSTRA_ID” and time. This convention is necessary in order to be able to use the “OKSTRA_ID” in the communication between historically and non-historically viable systems.

An *OKSTRA_Object* that appears in a plan presentation can be assigned a label. In addition, it is possible to specify additional information (requests, notes, etc.) for an *OKSTRA_Object* during data exchange via the *Communication_Object*.

Status_Property (Status_Eigenschaft)

Status_Property



An abstract supertype for those object types that contain information on the inventory.

Package S_Design (S_Entwurf)

This package includes object types for the representation of road designs.

Basic agreements

At all points in which the same values are used, the same units and signs should be applied. These agreements therefore apply to all object types.

Angle

Angle system according to computer internal representation: ± Pi (radians). Angles are recorded in the mathematical system. A specified rotation angle of 0 degrees means horizontal alignment. The sign corresponds to the mathematical system.

Distances

Distances to the right have a positive sign, distances to the left a negative sign.

Inclinations

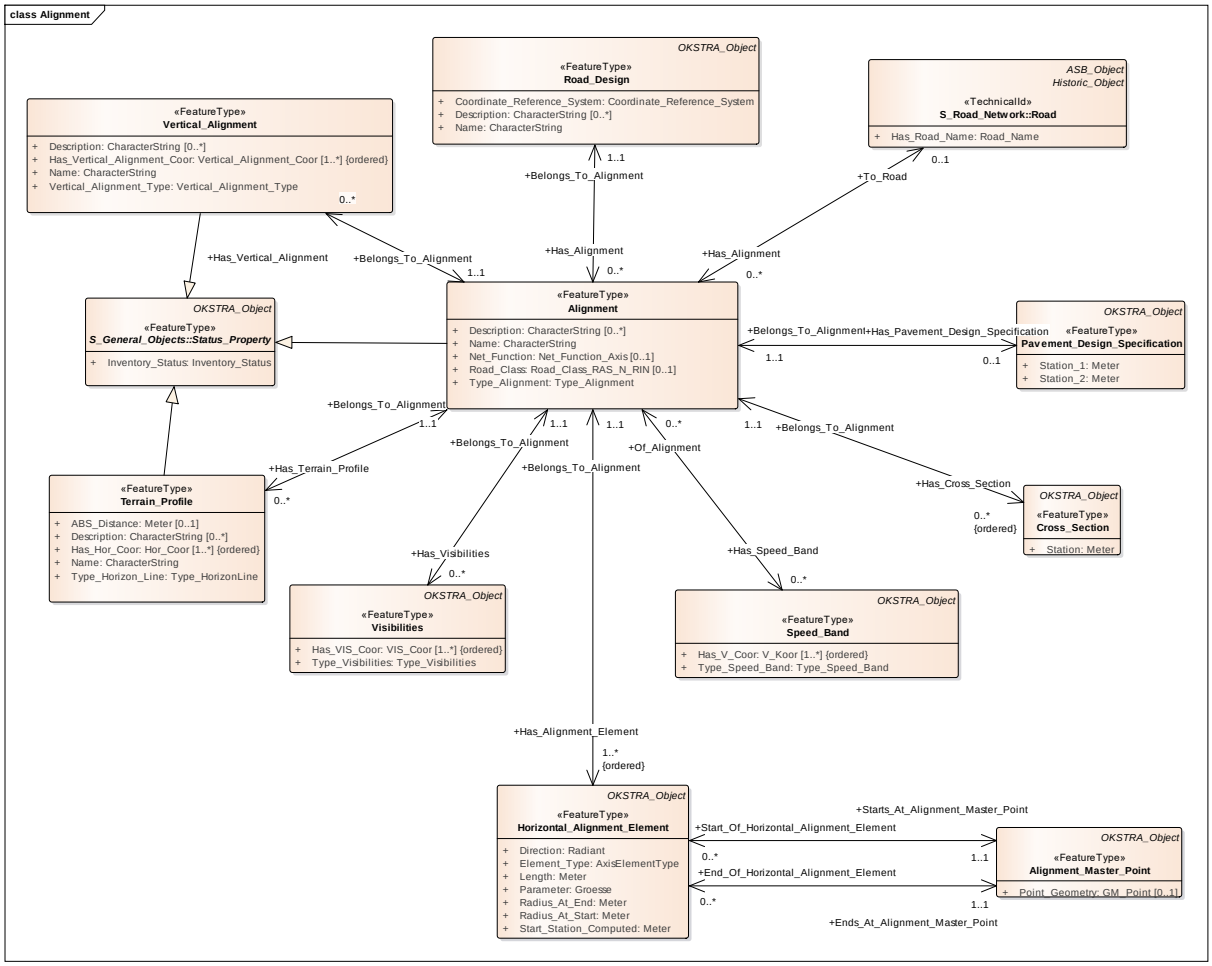
All inclinations are positively defined from left to right.

Reference systems

Points and positions are described at many places in the object types. These are not all in the same coordinate system. The following reference systems are used:

Location (survey): right, high, altitude
Longitudinal section: Station (in the ground plan projection), height
Cross section: center distance, height
Widths: station, width
Lateral inclinations: station, transverse inclination
Visibility: station, visibility
Speeds: station, speed
Other axis-dependent data: station, value

Alignment (Achse)



Alignment

An object type to represent a road alignment from the geometrical viewpoint.

The “Name” attribute is mandatory and holds the alignment identifier, which must be defined exclusively. It can be a number (“alignment number”) or an alphanumeric short name and must be unique within a path in the range of the first 8 characters. The customer and the contractor need to agree on whether only numerical or also alphanumeric axis identifiers may be used. In the case of numeric axis identifiers, these must be given without leading or trailing spaces. Compliance with this condition must be ensured by a design system for OKSTRA data export. In addition, a design system must ensure the uniqueness of the axis identifiers in the range of the first 8 characters.

The attribute “Type_Alignment” must be used to specify the technical meaning of the axis.

The attribute “Description” is used to specify other explanations or descriptions.

If alignments from an OKSTRA design data record are to be transferred to the stock data management, they must be designated as “ASB stock alignment” or “ASB auxiliary alignment” in the “Type_ Alignment” attribute. In addition, the “Net_Function” attribute must be specified in this case. This information can be used to decide whether the axis is transferred to the stock data as a section or as a branch.

If the axis describes part of a classified road, it should be indicated as a symbolic link (i.e. the street name is indicated) by means of the relation “To_Road”.

Road_Design (Trasse)

An object type that forms the central access point to the objects of a road design. It bundles all alignments (and thus all objects attached to it). Through the connection of the key table coordinate reference system, the structural geometries of the road design related to the alignment of the route are necessarily assigned to a uniform coordinate reference system. In a design project, exactly one instance of the object type Alignment must exist.

Horizontal_Alignment_Element

An object type for describing part of an axis with a particular axis element type (straight, clothoid or arc).

Radii to the axis element (attributes “Radius_To_Start” and “Radius_At_End”) are transmitted with a sign: In a left turn (in the direction of stations) the radius is negative, in a right curve it is positive, in a straight line it is 0.

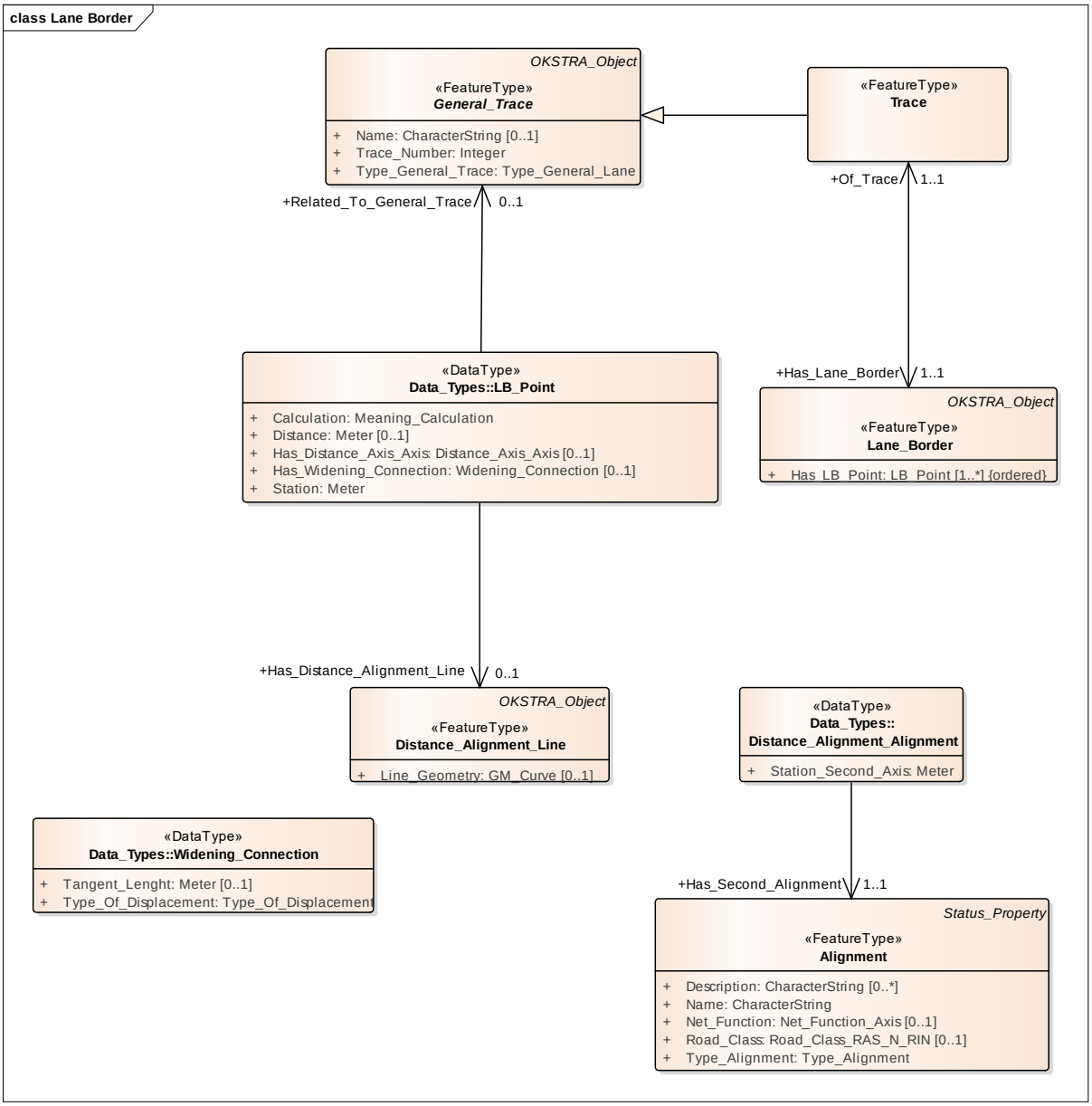
Clothoid parameters (attribute “parameter”) are passed without a sign.

Horizontal alignment elements are limited by alignment master points. Successive axis elements must hang on the same instance of alignment master points.

Alignment_Master_Point

An object type for marking the ends of axis elements. Axis principal points have coordinate geometries and thus allow a conversion of construction geometries specified in the axis elements as well as in related objects into coordinate geometries.

Lane_Border (Breitenband)



General_Trace

An abstract super-type (“generalized trace”) for pooling the common characteristics of the different types of pavement design specification (i.e., the lane and the curb step trace). In the *Type_General_Trace* object type contains the attribute “trace number”.

Trace

An object type for the representation of a “normal trace” (in contrast to a curb step trace) in the pavement design specification.

Each trace is assigned a lane border, which describes the position of the outer edge of the track. Since the trace 0 runs along the alignment, it always has the fixed distance 0 (cf. the object type LB_points).

In addition, a lane can be assigned to a slope band for indicating transverse inclinations, as well as a height profile for indicating the height of the lane. The elevation of the height profile is always valid for the outer edge of the lane described by the slope band.

All the lanes of a pavement design specification must be fixed in a certain manner, whereby a fixation can also be implicitly given via a known neighboring lane and a height profile. Over determination of the elevation and inclination is not allowed.

LB_Point

A complex data type to represent a point of a lane border.

An *LB_point* determines the position of a lane border at a particular station and contains information on how the lane border passes to the next *LB_point*.

There are three possibilities for defining the position of a lane border at the station of an *LB_point*:

1. In the "distance" attribute, a fixed distance (without a sign) can be specified in relation to a *General_Lane*. In the case of lane 0 (the axis of pavement design book), the distance 0 is entered; In this case, there is also no reference to a *General_Trace*.
2. The location can be defined by a line geometry (relation to object type *Distance_Alignment_Line*).
3. The position can be defined by a second axis (relation to object type *Distance_Alignment_Alignment*).

If the position of the border band in an LB point is determined by a line geometry or an axis, the border band follows the next LB point of the indicated line or axis. If the specified line or axis ends before the next LB point, the width band continues to run constantly.

Lane_Border

An object type for describing the course of the outer edge of a lane in the pavement design specification. A lane border is defined by an ordered set of LB points.

Note on the term "lane border": The LB points are very often defined by the specification of a distance from another trace (or curb step trace). If they refer to the next inner lane in a pavement design specification, the distances actually indicate the width of the track at the respective station. However, the model also allows references to any other lanes or even the definition of the lane border over other axes or lines; consequently the case that the lane border actually indicates the width of a lane is only true in special cases.

Distance_Alignment_Line_Border

An object type for the representation of a line with coordinate geometry, which can be used to define the position of a lane border.

Distance_Alignment _ Alignment

A complex data type for specifying a second axis that can be used to determine the position of a lane border. The "Station_Second_Alignment" attribute specifies the station on the second axis, from which the width of this axis follows.

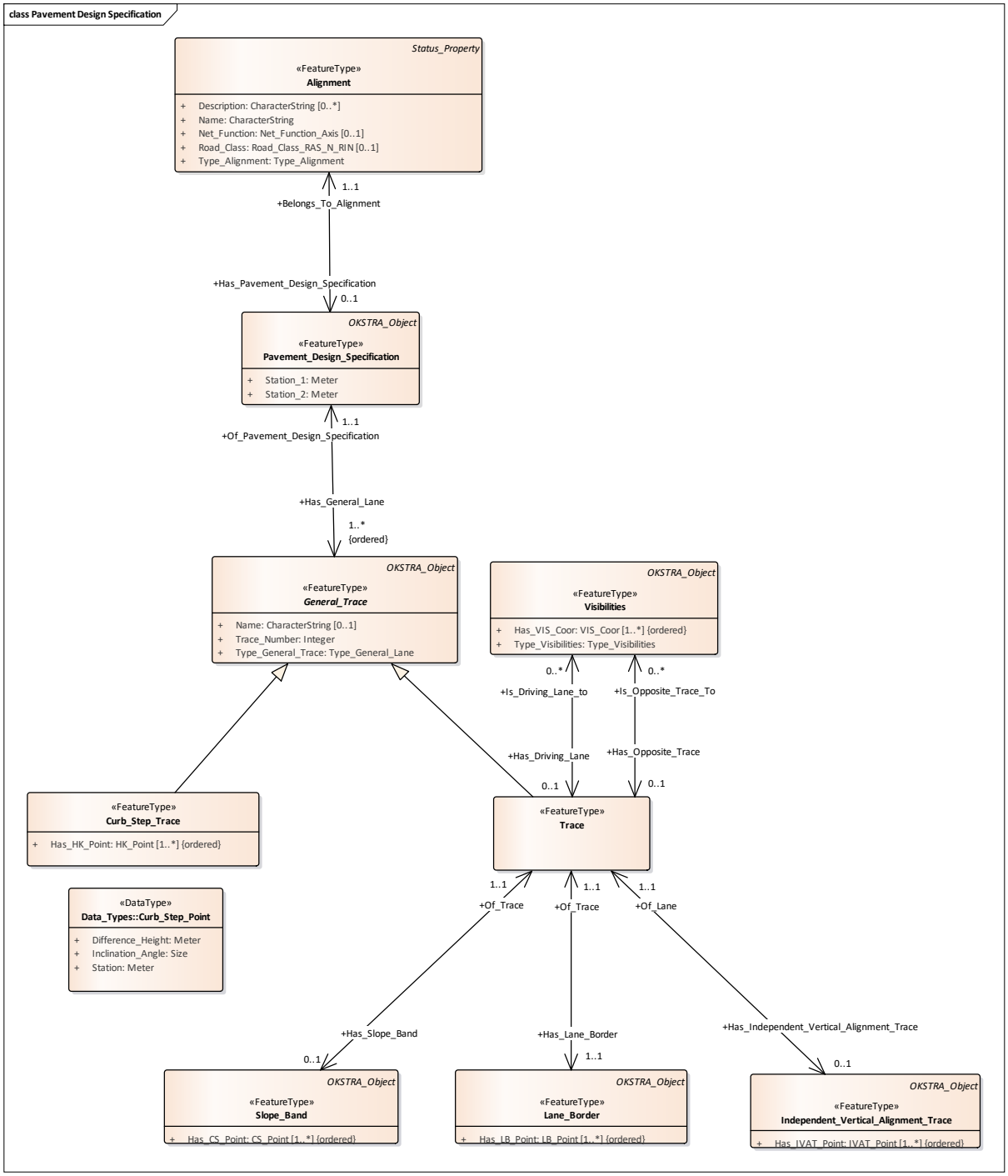
Distance_Alignment _Line

An object type for representing a line with coordinate geometry, which can be used to define the position of a lane border.

Widening_Connection

"Widening, broadening, connection"; Complex data type for specifying a relationship between two LB points of a lane border.

Pavement_Design_Specification (Deckenbuch)



Pavement_Design_Specification

An object type for representing a pavement design specification. A pavement design specification describes the structure of the road surface along an axis. The road surface is subdivided into individual general tracks (i.e., tracks or uprights), which are negative to the left (-1, -2, ...) of the axis, and positive (1, 2, ...) to the right of the axis. Track 0 is on the axis itself.

Curb_Step_Trace

An object type for representing a curb step trace within a pavement design specification. An elevated trace is considered a special trace in the pavement design specification. The outer edge of the elevated trace can be constructively attached to its inner edge (which is defined by the outer edge of the inner

adjacent lane of the pavement design specification) by way of the *Curb_Step_Points* assigned to the elevated lane.

A *Curb_Step_Trace* is always inclined towards the outside (in the direction from the inside to the outside in the pavement design specification); Overhangs inside cannot occur.

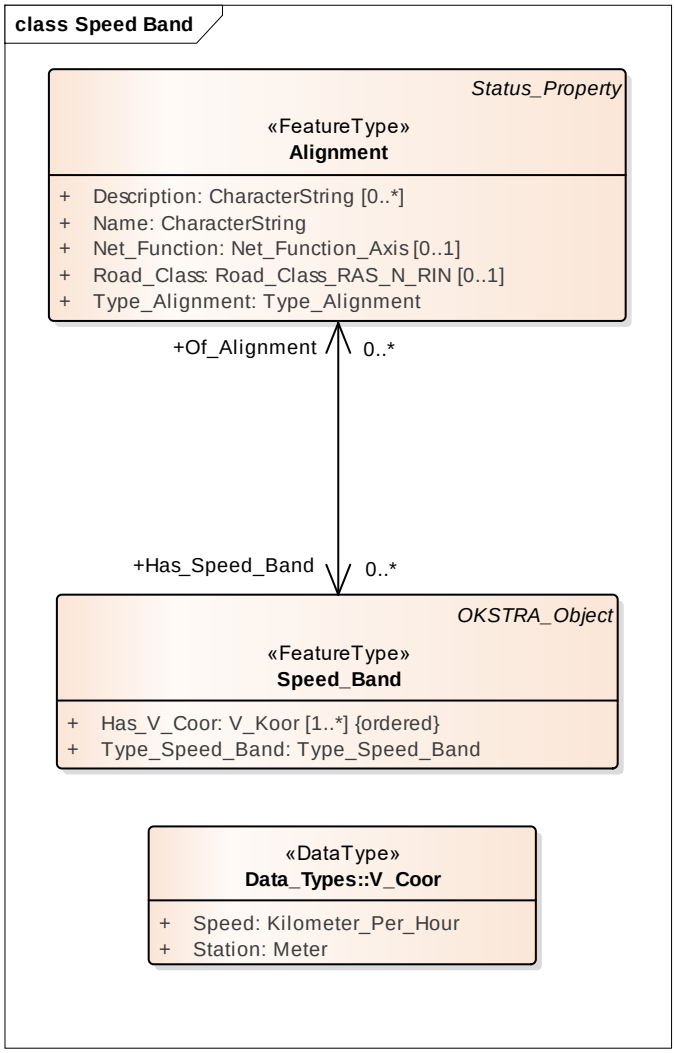
Curb_Step_Point

A complex data type for describing a point of a curb step trace. A *Curb_Step_Point* in the *Difference_Height* attribute contains the height difference of the high kerb lane with respect to the road surface of the inner adjacent lane as well as the *Inclination_Angle* attribute to the slope of the lorry lane at a particular station.

The sign of the *Difference_Height* is positive when the high kerb lane goes up from the inside adjacent lane, and negative when the high kerb lane goes down.

In the *Inclination_Angle* attribute, the denominator n of the inclination specification 1: n used in practice is entered in the form of a non-negative real number. If the sign is needed, it must be derived from the attribute *Difference_Height*. A vertical slope is conventionally given as n = 0.

Speed_Band (Geschwindigkeitsband)



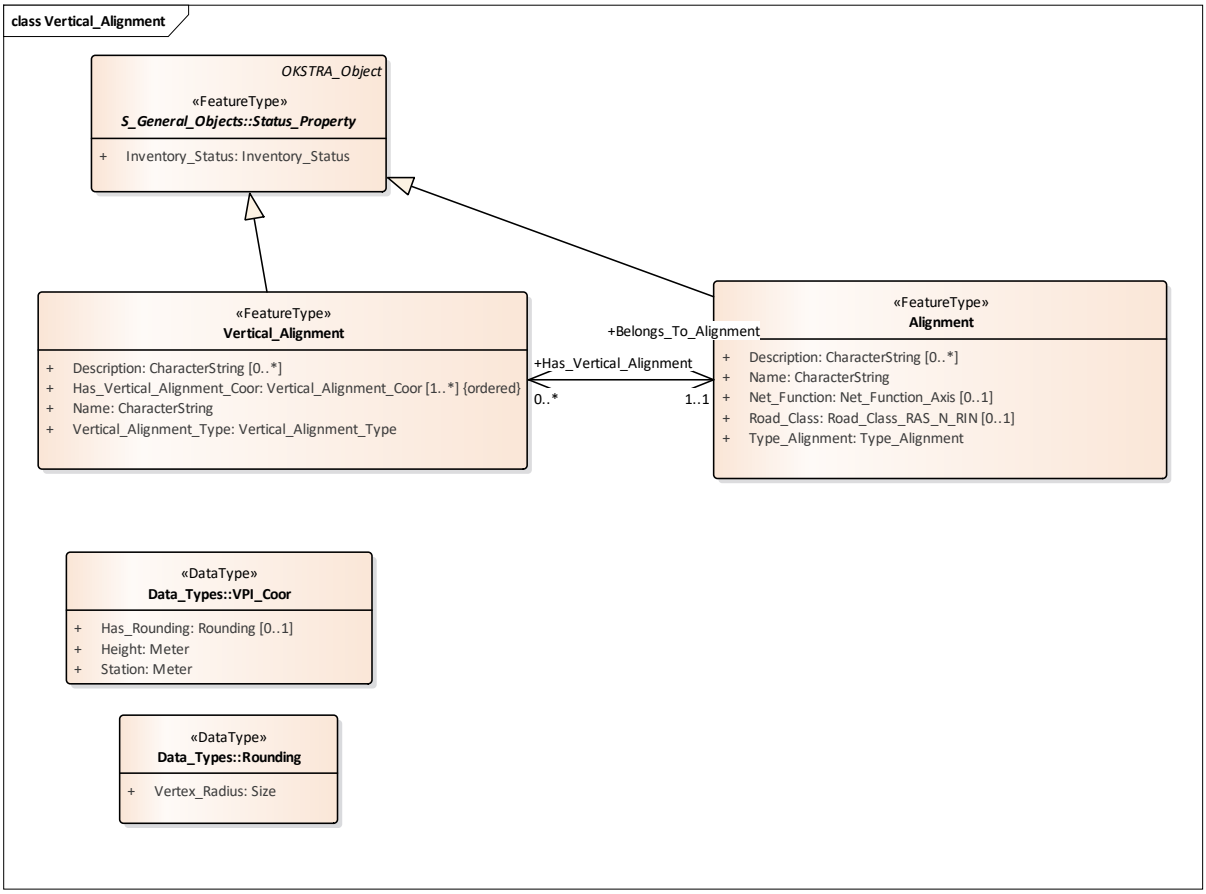
Speed_Band

An object type for specifying a speed band. It specifies the velocities (V85, Ve, Vk) of a road design at the various stations of an axis. The object type *V_Coor* is used to specify a single speed at a specific station.

V_Coor

A complex data type for describing a point of the speed band (i.e., indicating a speed at a particular station)

Vertical_Alignment (Gradiente)



Vertical_Alignment

An object type for the representation of a gradient, i.e. the description of the height profile of an alignment or a track of a pavement design specification.

The first case occurs when there is no pavement design specification for the axis. In this case, only one gradient can be assigned to the axis to avoid ambiguities.

In the second case, the gradient is referenced by the *IVAT_Points* of a vertical altitude, which describes the height profile of a track of the pavement design specification. In this case, several gradients can be assigned to the axis to which the pavement design specification refers (in order to be able to specify different gradients for the different tracks). Since track 0 of a blanket book is always on the associated axis, the height profile of the axis is then determined via track 0 and not from the (possibly ambiguous) relation between the axis and the gradients.

The height profile of a gradient is described by a polyline that is rounded at its vertices (see object type *VPI_Coor*).

VPI_Coor

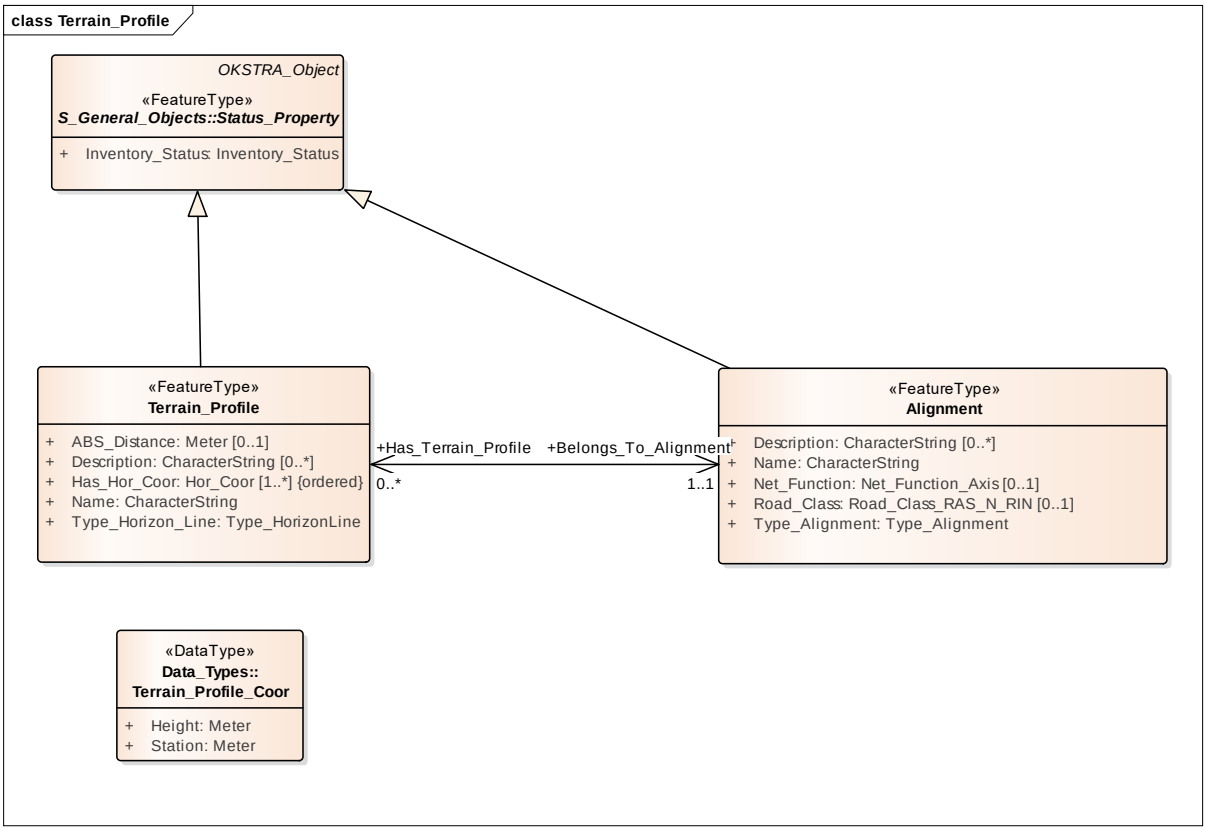
A complex data type for representing the kinking point of a polyline, which describes the gradient of the profile. The station specification of a *VPI_Coor* always refers to the axis to which the profile is

assigned. Since gradients usually contain roundings, a *VPI_Coor* can be used to add rounding information to the profile in the relevant *VPI_Coor*.

Rounding

A complex data type to describe the rounding of a profile in a particular *VPI_Coor*. The rounding always described as a square parabola; The corresponding “vertex radius” is to be specified as a compulsory attribute of the rounding (without sign).

Terrain_Profile (Horizontlinie)



Terrain_Profile

An object type for describing the height profile of a terrain horizon in the longitudinal section.

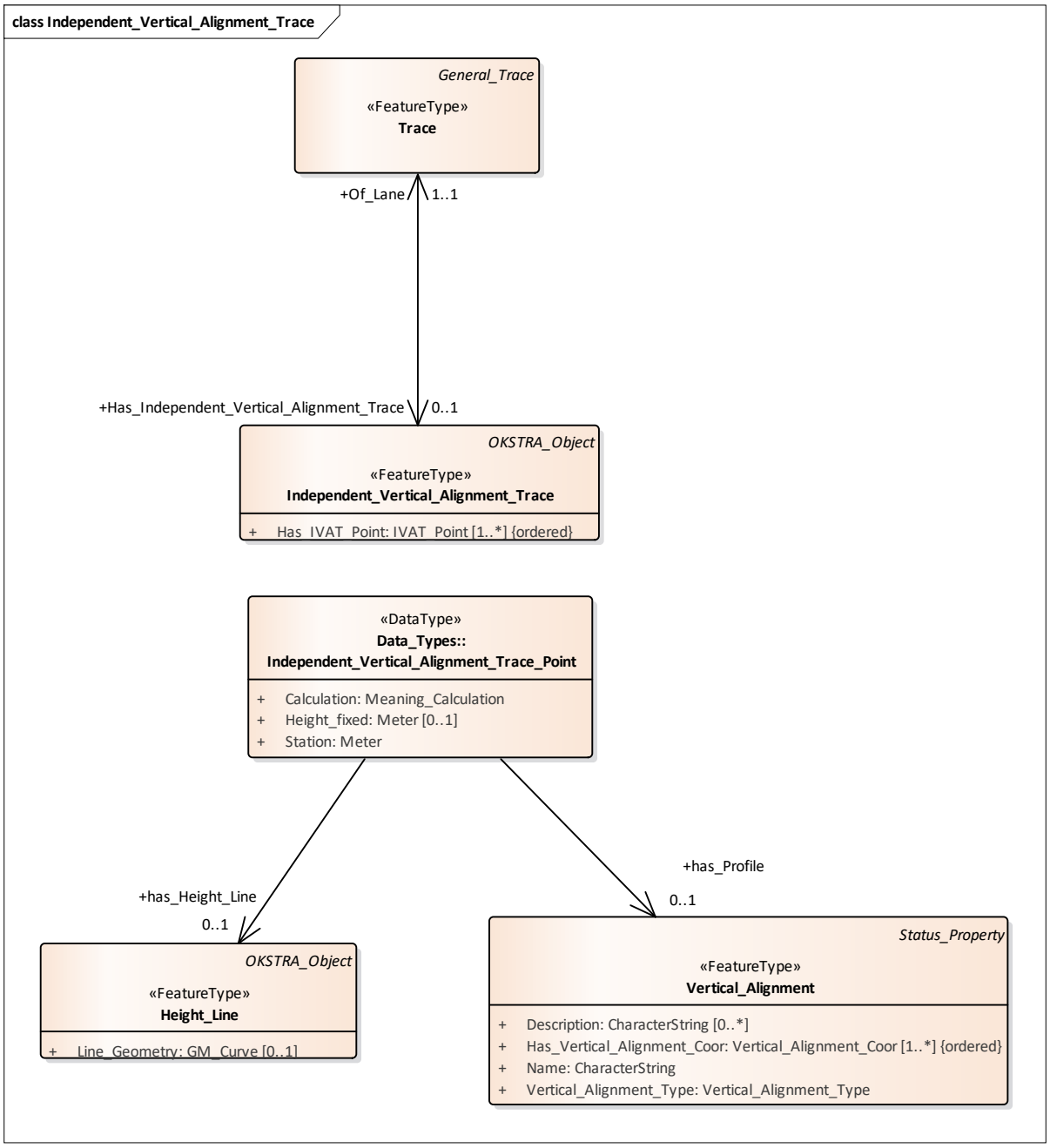
A horizon line runs either on the associated axis or parallel to it. In the second case, the distance from the axis in the attribute “*ABS_Distance*” must be specified (distances to the left are indicated as negative, distances to the right as positive).

The height profile of a horizontal line is described by a polyline.

Hor_Coor

A complex data type for the representation of a kinking point of a polyline, which describes the height profile of a horizon line. The station specification of a *Hor_Coor* always refers to the axis to which the horizon line is assigned.

Independent_Vertical_Alignment_Trace (Höhenzug)



Independent_Vertical_Alignment_Trace

An object type for specifying the height of a lane of the pavement design specification. An elevation is defined by an ordered set of *IVAT_points*.

A gradient or a Height_Line can be used by several vertical alignment traces.

IVAT_Point

A complex data type to indicate the height of an Altitude at a particular station. There are three ways to assign a height indication to an *IVAT_Point*:

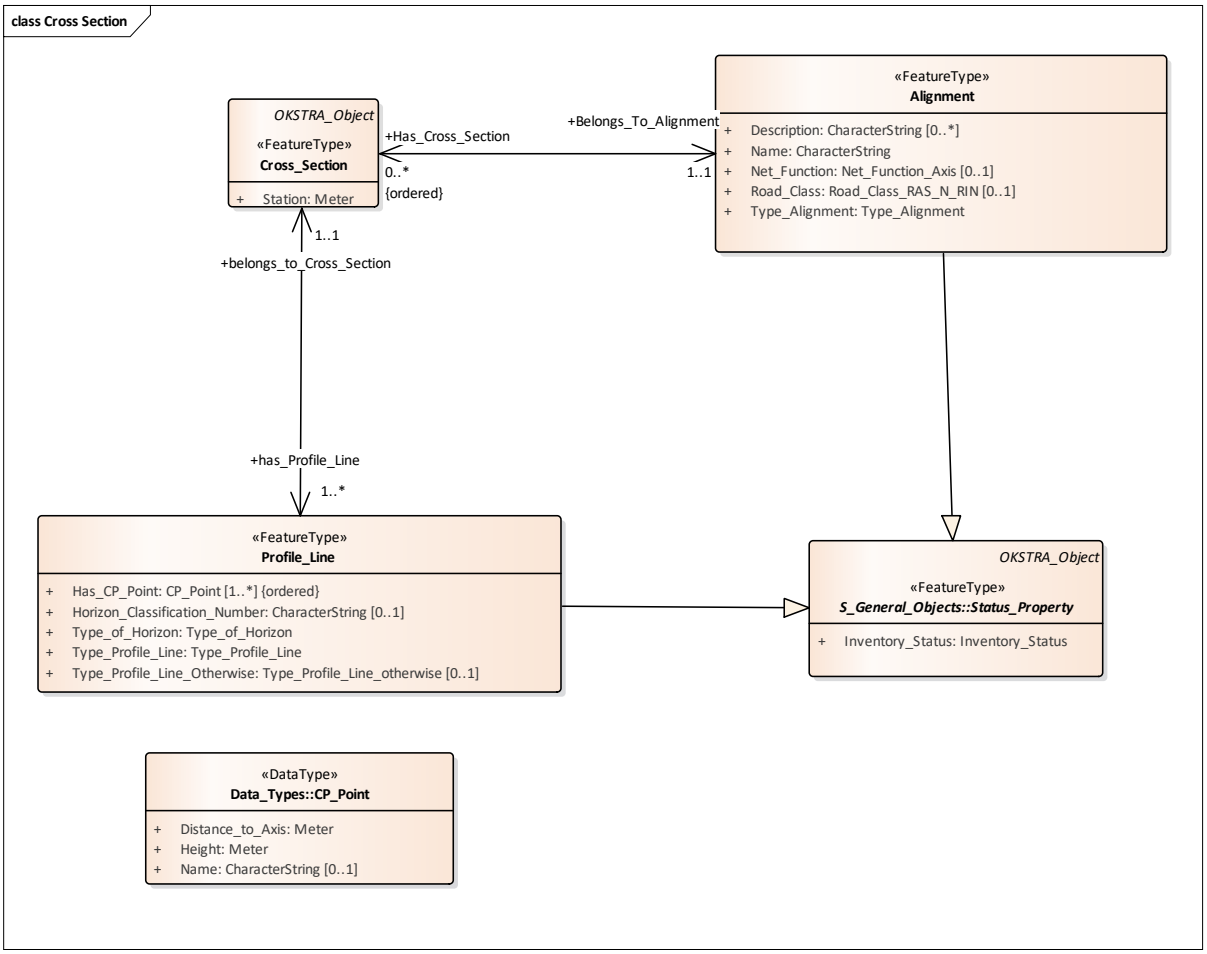
1. A fixed height value can be specified (attribute “Height_Fixed”).
2. The height can be taken over from a profile.
3. The height can be taken from a 3D line geometry (object type Height_Line).

If the elevation of an *IVAT_Point* is derived from one of the object types Profile or *Height_Line*, the altitudes of the Altitude will follow the next *IVAT_Point* of the profile or the line. If a fixed height is specified for an *IVAT_Point*, a linear elevation is obtained between it and the following *IVAT_Point*.

Height_Line

An object type for displaying a line with 3D coordinate geometry, which can be used to determine the height of a vertical line.

Cross Section (Querprofil)



Cross_Section

An object type for describing a cross-section profile. A cross-section describes all the horizons of a road in a cross-section at a specific station of the assigned axis. The individual horizons of the cross-section profile are specified by the object type *Profile_Line*.

Profile_Line

An object type for describing a horizon in a cross-section. A *Profile_Line* has a constructive line geometry that can be a simple (not closed) line as well as a (closed) face ring. Whether a *Profile_Line* is a simple line or an area border is specified using the key table *Type_of_Horizon*.

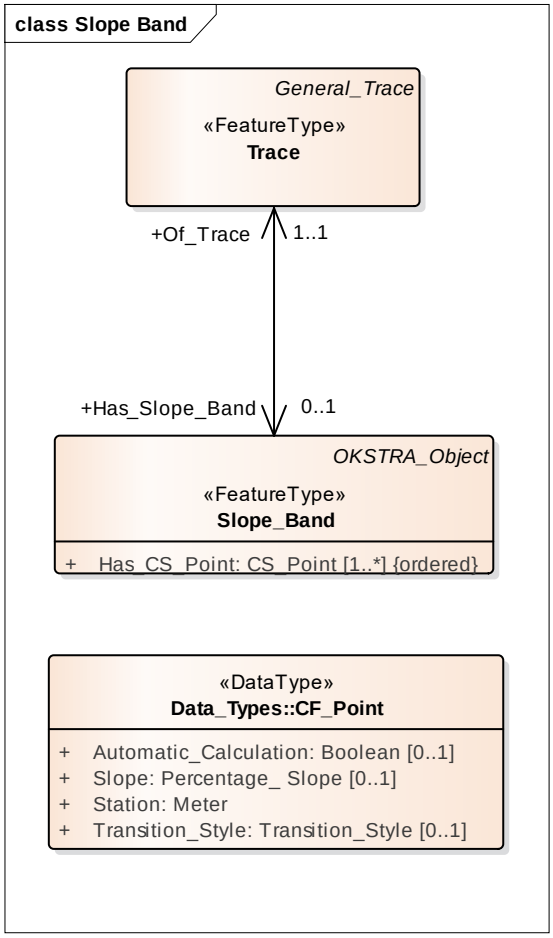
The points of the profile line are represented by the object type *CP_Point*. If a profile line is an area border, its start point and its end point must be given by the same instance of the object type *CP_Point*.

The REB name of the profile line can be entered in the “Horizon number” attribute (for a simple line, a double-digit number between 10 and 99, a seven-digit position specification as per the REB data type DA53 for a face circumference). If an REB name is not available, nothing is specified.

CP_Point

A complex data type *CP_Point* for describing a point in a profile of a cross section. A CP_Point can be assigned a name in the “Name” attribute.

Slope_Band (Querneigungsband)



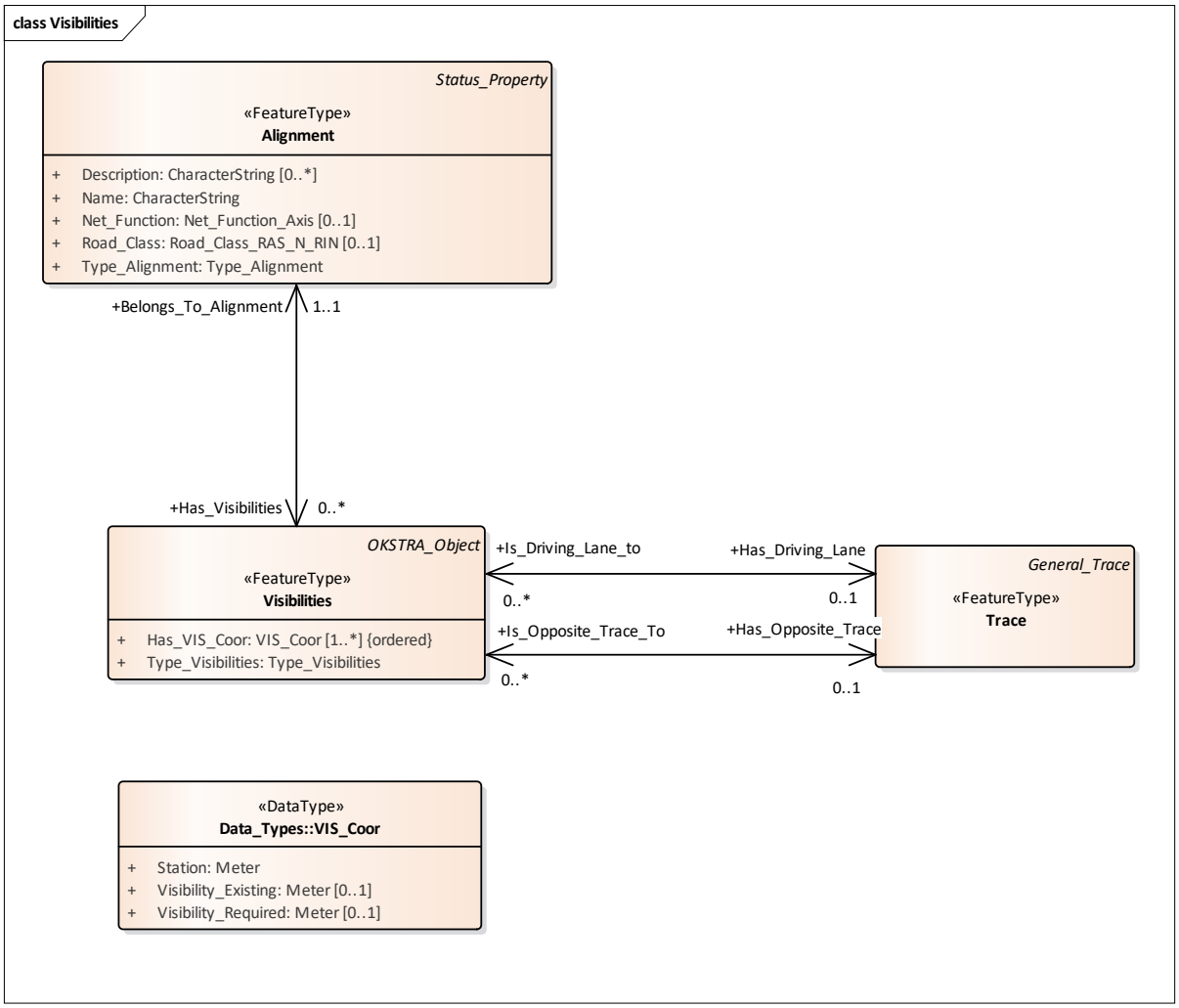
Slope_Band

An object type for describing the transverse inclinations of a lane of the pavement design specification.

CF_Point

A complex data type for describing the transverse inclination at a specific station of the slope band. For a *CF_Point*, you can either specify an explicit slope (“*Slope*” attribute) or specify that the cross-tilt at the relevant station is to be calculated automatically (“*Automatic_Calculation*” attribute). In the case of automatic calculation, the transverse inclination is derived from the heights of the adjacent lane.

Visibilities (Sichtweiten)



Visibilities

An object type for specifying a visibility strip (sight distance from a specific station). Both the existing and the required overtaking and stop sightings can be stored. The visual scope is defined by an ordered set of *VIS_Coors*.

Trace

An object type for the representation of a “normal trace” (in contrast to the curb step trace) in the cover pavement design specification.

Each track is assigned to a lane border, which describes the position of the outer edge of the track. Since track 0 is on the axis, it always has the fixed distance 0 (cf. the object type *LB_points*).

In addition, a lane can be assigned to a slope band for indicating its transverse inclinations, as well as an altitude to indicate the height of the track. The elevation of the tread is always valid for the outer edge of the lane described by the slope band.

All the lanes of pavement design specification must be fixed in terms of the height profile, although this can also be defined implicitly from the height profile and a slope band of a neighboring lane. Excesses for the elevation and inclination are not allowed, and no overdetermination is allowed.

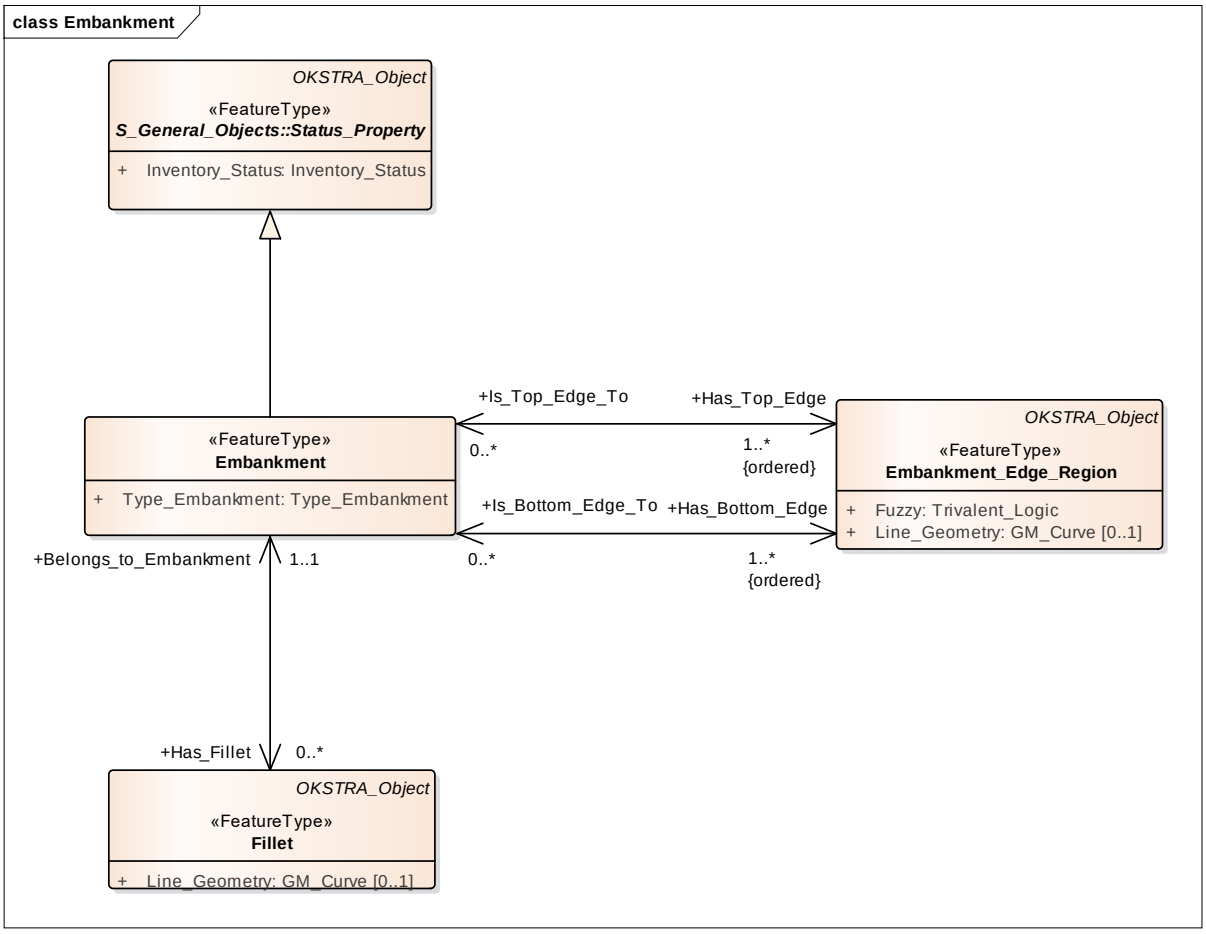
VIS_Coor

A complex data type for describing a point in the sight distance band. A *VIS_Coor* can be used to specify the required or available visibility at a specific station.

Package S_Topography (S_Topografie)

This package contains object types for the representation of topographical conditions.

Embankment (Böschung)



Embankment

An object type for representing an embankment.

The upper edge and the lower edge of an embankment can be composed of several embankment edge sections, each with its own line geometry. For this reason, the two relations from the embankment to the embankment edge region are multiple. The embankment edge regions of an edge must be connected linearly (the end point of one segment is the starting point of the next). They are also specified in the relation from the beginning to the end of the respective edge. The slopes can meet at the lateral extremities of the embankment, but this is not mandatory. A possible special case is represented by an annular embankment: here, the upper edge and the lower edge respectively meet themselves and form two polygons, one of which runs completely into the other.

From the line geometries of the upper and lower edges of an embankment, a surface geometry for the slope can be constructed if necessary. In the case of an annular embankment, a surface with a hole is produced. In all other embankments, the surface circumference is formed by joining the top and bottom edges at the beginning and end of the embankment. If the ends do not meet at one point (naturally escaping embankment), a straight line connection should be inserted.

If an embankment should be shown with hatching in a system, this must be interpolated by the system (it is not transported via OKSTRA). The interpolation of a hatched section starts at the beginning of the

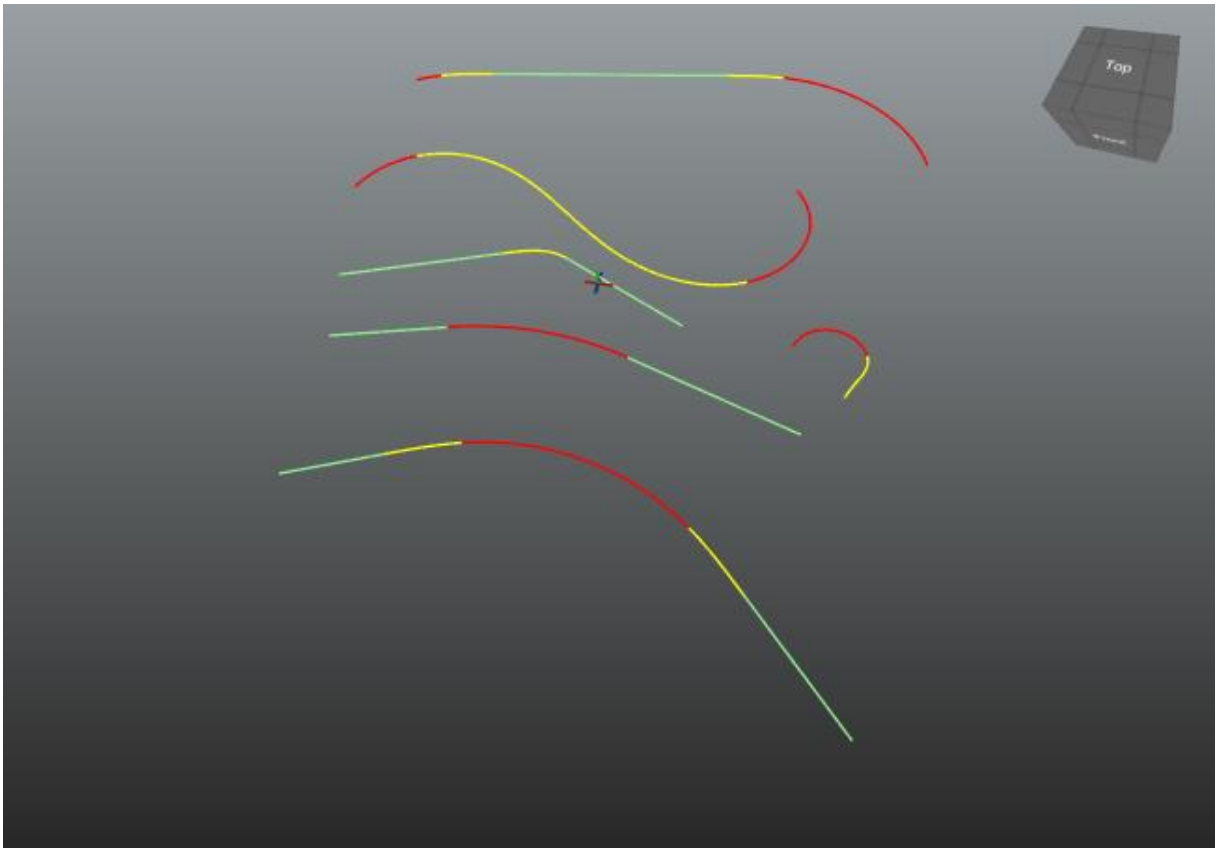
line of the upper edge. This must be taken into account when generating the line geometry of the top edge during data export (if necessary, the order of the points in the line geometry must be reversed so that the hatching interpolation of the target system starts on the desired side).

An embankment can also contain fillets.

Examples

Example 1 (ex-horizontal-alignment.xml)

This example shows six alignment curves, all of which have only a horizontal alignment and no vertical alignment. The horizontal alignment elements consist of different horizontal alignment elements such as straights (green color), arcs (red color) and clothoids (yellow color).



The alignment without an arc element (there is only one in this example) is named “Alignment - (1)” and is described in OKSTRA by an *okstra:Alignment* element:

```
<okstra:Alignment gml:id="Alignment_8012">
    <okstra:Inventory_Status>6</okstra:Inventory_Status>
    <okstra:Name>Alignment - (1)</okstra:Name>
    <okstra:Type_Alignment>1</okstra:Type_Alignment>
    <okstra:Has_Horizontal_Alignment_Element Object_Class="Horizontal_Alignment_Element" xlink:href="#Horizontal_Alignment_Element_8106"/>
    <okstra:Has_Horizontal_Alignment_Element Object_Class="Horizontal_Alignment_Element" xlink:href="#Horizontal_Alignment_Element_8188"/>
    <okstra:Has_Horizontal_Alignment_Element Object_Class="Horizontal_Alignment_Element" xlink:href="#Horizontal_Alignment_Element_8256"/>
    <okstra:Belongs_To_Alignment Object_Class="Road_Design" xlink:href="#Road_Design_7863"/>
</okstra:Alignment>
```

This axis element has three horizontal alignment elements. The first horizontal alignment element looks like this:

```
<okstra:Horizontal_Alignment_Element gml:id="Horizontal_Alignment_Element_8106">
    <okstra:Element_Type>1</okstra:Element_Type>
    <okstra:Start_Station_Computed uom="m">0.0</okstra:Start_Station_Computed>
    <okstra:Length uom="m">509.3707922553564</okstra:Length>
    <okstra:Direction uom="rad">0.3564193707695</okstra:Direction>
    <!-- Property Parameter missing -->
    <okstra:Parameter>-0</okstra:Parameter>
    <!-- Property Radius_At_Start missing -->
    <okstra:Radius_At_Start>-0</okstra:Radius_At_Start>
    <!-- Property Radius_At_End missing -->
    <okstra:Radius_At_End>-0</okstra:Radius_At_End>
    <okstra:Belongs_To_Alignment Object_Class="Alignment" xlink:href="#Alignment_8012"/>
    <okstra:Ends_At_Alignment_Master_Point Object_Class="Alignment_Master_Point" xlink:href="#Alignment_Master_Point_8146"/>
    <okstra:Starts_At_Alignment_Master_Point Object_Class="Alignment_Master_Point" xlink:href="#Alignment_Master_Point_8145"/>
</okstra:Horizontal_Alignment_Element>
```

The axis element type is 1 which means that is a straight element. The axis element also has a length and a direction value. The start position is described by the following alignment master point:

```
449     <gml:featureMember>
450         <okstra:Alignment_Master_Point gml:id="Alignment_Master_Point_8145">
451             <okstra:Point_Geometry>
452                 <gml:Point gml:id="Point.8163">
453                     <gml:pos srsDimension="3">6236.632 4031.979 0.0</gml:pos>
454                 </gml:Point>
455             </okstra:Point_Geometry>
456             <okstra:Start_Of_Alignment_Element Object_Class="Horizontal_Alignment_Element" xlink:href="#Horizontal_Alignment_Element_8106"/>
457         </okstra:Alignment_Master_Point>
458     </gml:featureMember>
459     <gml:featureMember>
```

The second axis element of the axis looks like this:

```
117         <okstra:Horizontal_Alignment_Element gml:id="Horizontal_Alignment_Element_8188">
118             <okstra:Element_Type>12</okstra:Element_Type>
119             <okstra:Start_Station_Computed uom="m">509.3707922553564</okstra:Start_Station_Computed>
120             <okstra:Length uom="m">200.0</okstra:Length>
121             <okstra:Direction uom="rad">0.3564194738865</okstra:Direction>
122             <okstra:Parameter>155.2742763525292</okstra:Parameter>
123             <okstra:Radius_At_Start uom="m">0.0</okstra:Radius_At_Start>
124             <okstra:Radius_At_End uom="m">120.550504484008</okstra:Radius_At_End>
125             <okstra:Belongs_To_Alignment Object_Class="Alignment" xlink:href="#Alignment_8012"/>
126             <okstra:Ends_At_Alignment_Master_Point Object_Class="Alignment_Master_Point" xlink:href="#Alignment_Master_Point_8195"/>
127             <okstra:Starts_At_Alignment_Master_Point Object_Class="Alignment_Master_Point" xlink:href="#Alignment_Master_Point_8194"/>
128         </okstra:Horizontal_Alignment_Element>
```

The element type here is “12” which corresponds to a clothoid element. The different parameters are described accordingly.

This concept applies similarly to the other axis and axis elements.

Example 2 (vertical-alignment.xml)

The second example focuses on vertical alignment. This file also contains a horizontal alignment, as already discussed in Example 1.

The OKSTRA standard describes vertical alignment using a vertical point of intersection approach (PVI approach) instead of the segment-based approach used, for example, by the IFC Alignment standard. The segment-based approach stores data about the individual segments, e.g. the start and end point is stored for each line and every parabola. The PVI approach stores only the points of vertical intersection and the radius of curves. With this approach, the start and end points of the individual segments have to be computed explicitly. Figure 1 shows the PVI approach compared with the segment-based approach. Each of the two approaches can be converted to the other.

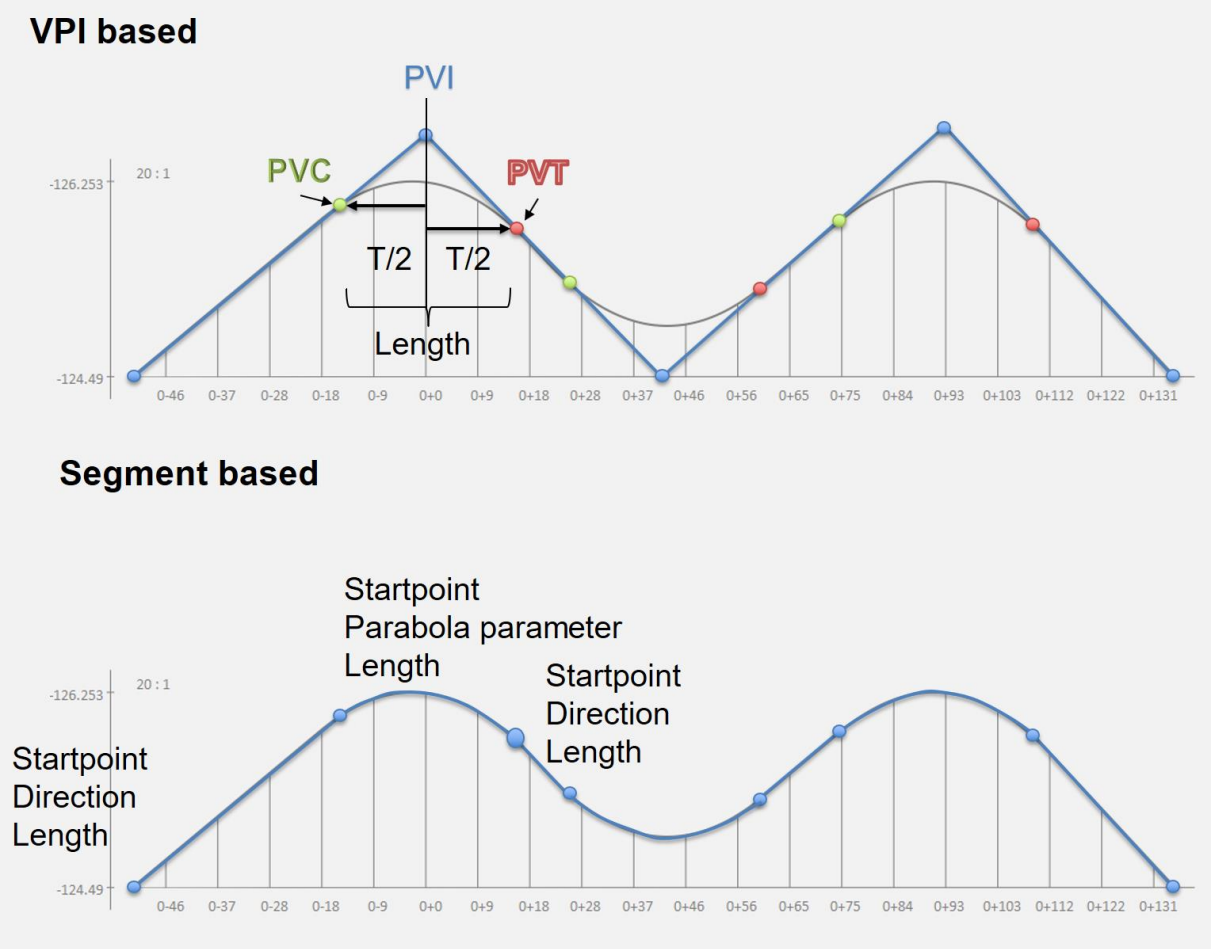
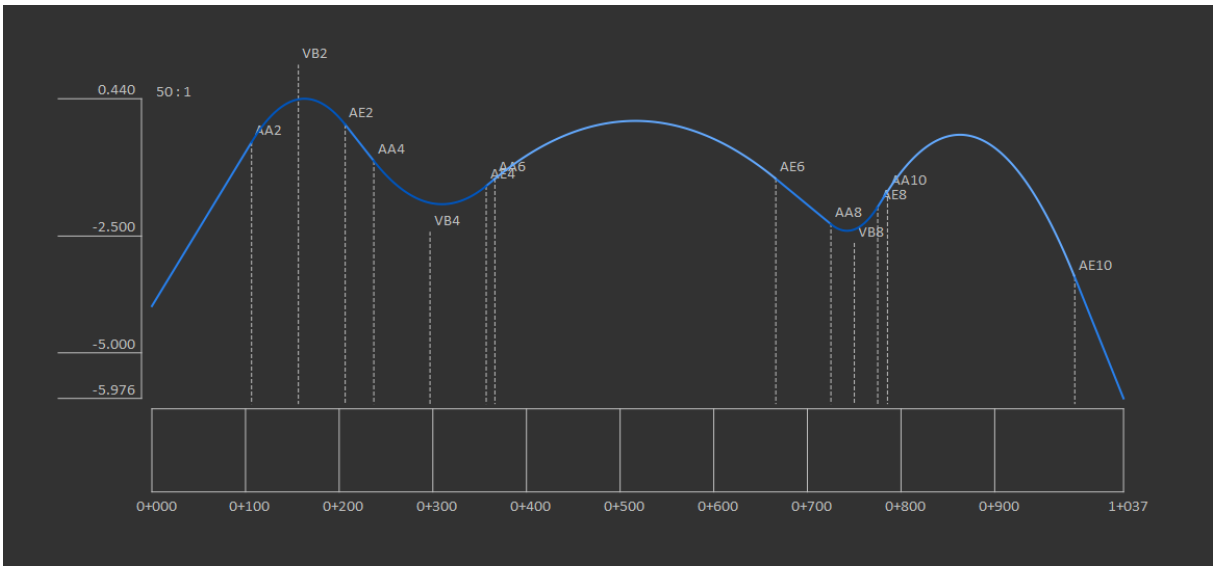


Figure 1: PVI approach compared to the segment-based approach

The vertical alignment is stored in the *Profile* class that is referenced by the *Axis* class.

The following images show an overview of the example vertical alignment (profile):



The profile is described by *VPI_Coor* elements.

```
76          <okstra:Vertical_Alignment gml:id="Vertical_Alignment_8219">
77              <okstra:Inventory_Status>6</okstra:Inventory_Status>
78              <okstra:Name>unbenannt</okstra:Name>
79              <okstra:Type_Vertical_Alignment>1</okstra:Type_Vertical_Alignment>
80              <okstra:Has_VPI_Coor>
81                  <okstra:VPI_Coor>
82                      <okstra:Station uom="m">0.0</okstra:Station>
83                      <okstra:Height uom="m">-4.0</okstra:Height>
84                  </okstra:VPI_Coor>
85              </okstra:Has_VPI_Coor>
86              <okstra:Has_VPI_Coor>
87                  <okstra:VPI_Coor>
88                      <okstra:Station uom="m">156.438683478683</okstra:Station>
89                      <okstra:Height uom="m">1.157814831387</okstra:Height>
90                      <okstra:Has_Rounding>
91                          <okstra:Rounding>
92                              <okstra:Vertex_Radius>1711.8272359552093</okstra:Vertex_Radius>
93                          </okstra:Rounding>
94                      </okstra:Has_Rounding>
95                  </okstra:VPI_Coor>
96              </okstra:Has_VPI_Coor>
97              <okstra:Has_VPI_Coor>
```

Everything needed to reconstruct the vertical alignment is stored in *VPI_Coor (Grad_Koor* in the German version*)*:

Grad_Koor:Station	-1.568700	Station
Grad_Koor:Hoehe	125.980000	Height
Grad_Koor:hat_Ausrundung	1 Fachobjekt: Ausrundung	Rounding
Ausrundung:Scheitelradius	499.911587	Parabola parameter

The parabola can be found in the *Rounding* object and the pvi-type determined by the following rules:

- Start:
  - First pvi
- Line:
  - *VPI_Coor* doesn't have a *Rounding* object
- Parabola
  - *VPI_Coor* does have a *Rounding* object

The following shows how this OKSTRA PVI approach can be converted to a segment-based representation. We assume the following pseudo-code definitions for the described algorithm:

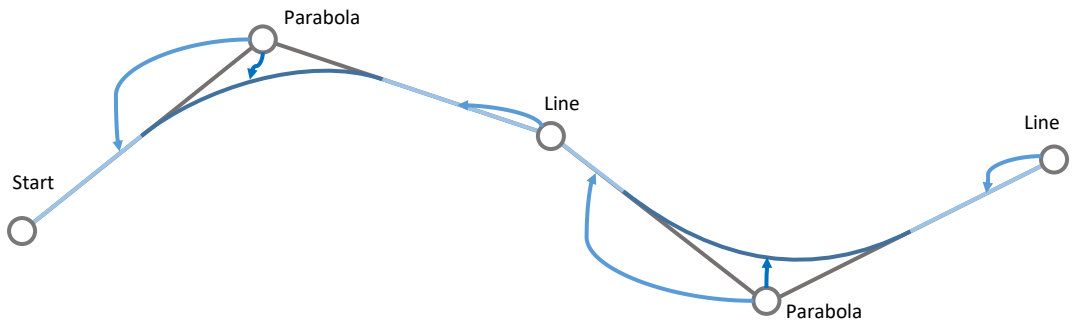
```
enum OKSTRA_PVI_Type
{
	Start,
	Line,
	Parabola
};

struct OKSTRA_PVI
{
	OKSTRA_PVI_Type pvi_type;
	double Station;
	double Height;
	double Rounding;
	double StartSlope;
	double EndSlope;
};
```

Only used if pvi_type is Parabola

Generating Alignment Elements

The vector of PVIs can be used to generate alignment elements (segments).
This is achieved by iterating through the OKSTRA_PVIs:

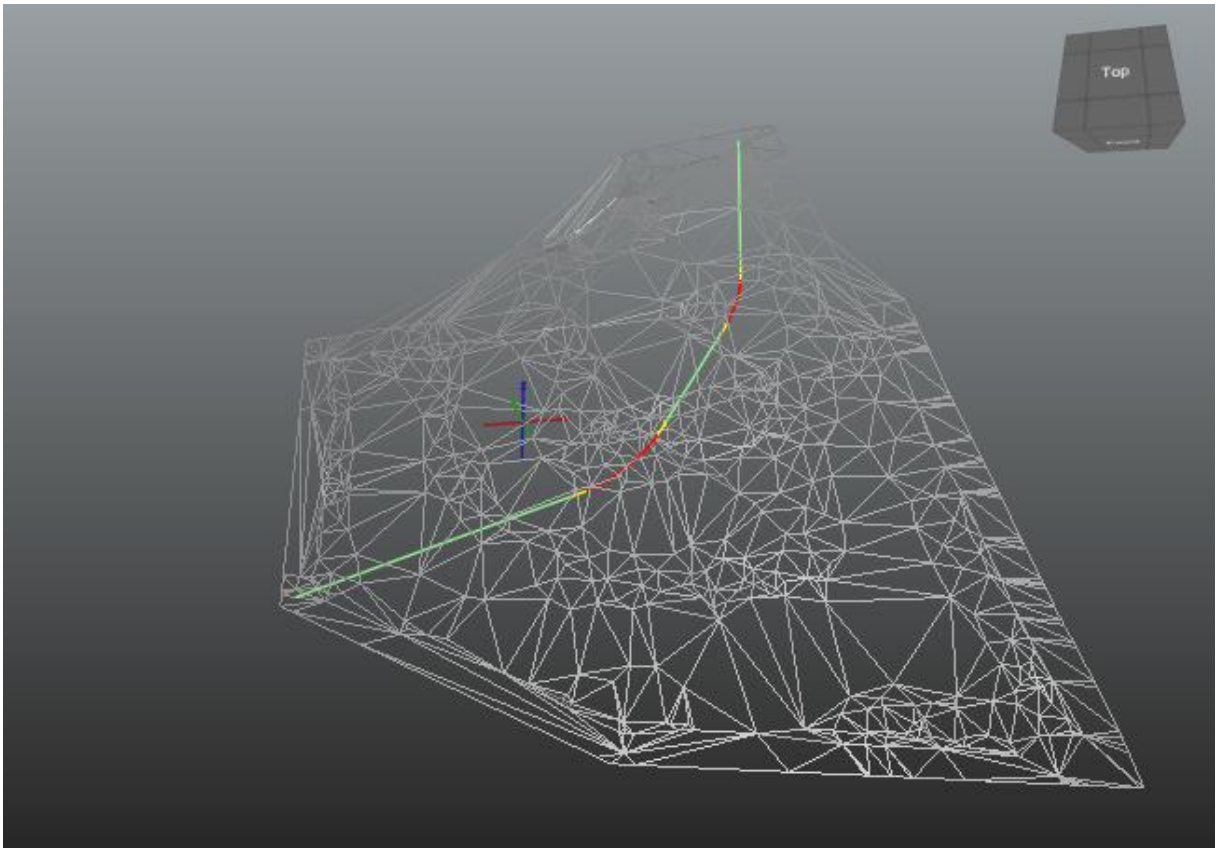
- Start:
  - Temporarily save position
- Parabola:
  - Calculate start and end position of the parabola
  - Generate line element from the last position to the start position
  - Generate parabola element between start and end position
  - Temporarily save end position
- Line:
  - Generate line element from the last position to the current position
  - Temporarily save position



Example 3 (ex-terrain-surface.xml)

This example describes the alignment of a digital elevation model. A digital elevation model (DEM) can be stored in the form of a triangle description. The class *DGM* (short for Digitales-Gelände-Modell = digital elevation model) serves this purpose. It manages a list of triangles of type *Dreieck* (triangle).

The following shows a screenshot of the example file:



The DEM (digital elevation model consist of triangles):

```
<gml:featureMember>
    <okstra:DEM gml:id="DEM_359522">
        <okstra:Name>Geländemodell</okstra:Name>
        <!-- Property Type_DEM missing -->
        <okstra:Type_DEM Object_Class="Type_DEM"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_376877"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_376905"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_376931"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_376957"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_376983"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377009"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377035"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377061"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377087"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377113"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377139"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377165"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377191"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377217"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377243"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377269"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377295"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377321"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377347"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377373"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377399"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377425"/>
        <okstra:Has_Triangles Object_Class="Triangle" xlink:href="#Triangle_377451"/>
```

The triangle object references individual points:

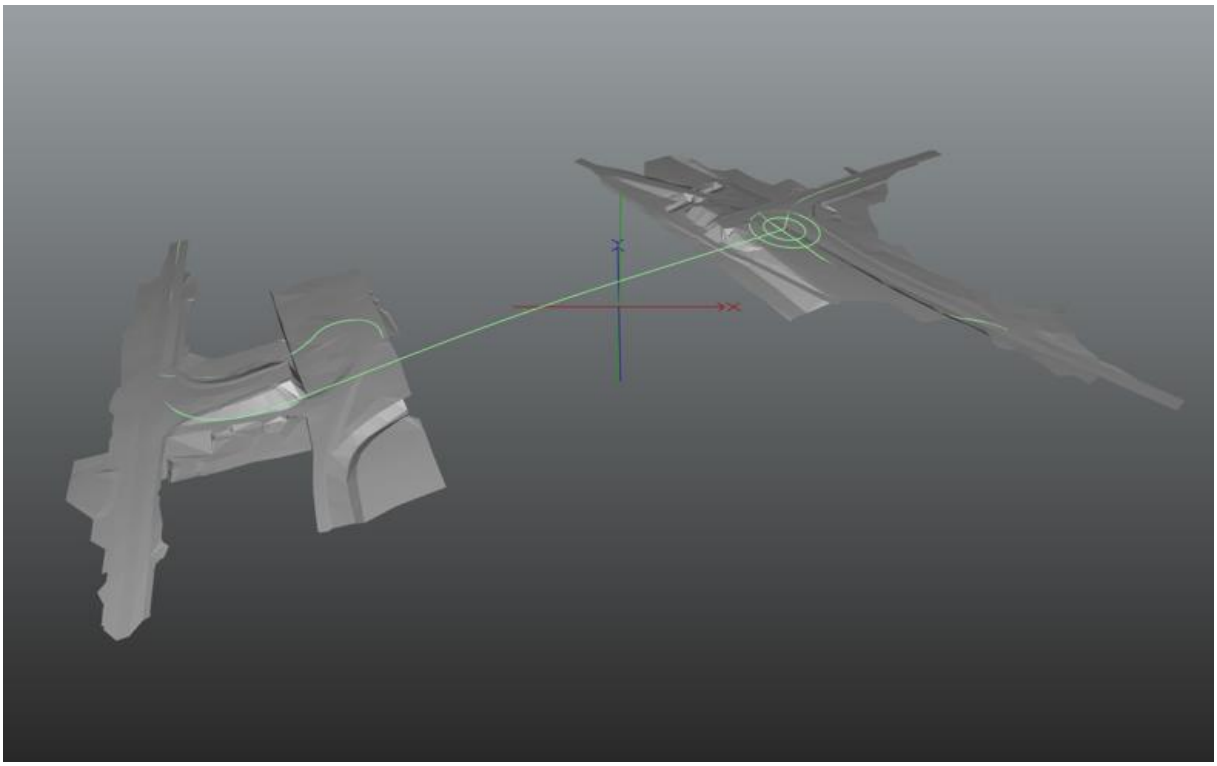
```
53872     <gml:featureMember>
53873         <okstra:Triangle gml:id="Triangle_376877">
53874             <!-- Eigenschaft Edge_Property fehlt -->
53875             <okstra:Edge_Property Object_Class="Edge_Property"/>
53876             <okstra:Edge_Property Object_Class="Edge_Property"/>
53877             <okstra:Edge_Property Object_Class="Edge_Property"/>
53878             <okstra:In_DEM Object_Class="DEM" xlink:href="#DEM_359522"/>
53879             <okstra:Has_Points Object_Class="General_PointObject" xlink:href="#General_PointObject_359525"/>
53880             <okstra:Has_Points Object_Class="General_PointObject" xlink:href="#General_PointObject_359529"/>
53881             <okstra:Has_Points Object_Class="General_PointObject" xlink:href="#General_PointObject_359533"/>
53882         </okstra:Triangle>
53883     </gml:featureMember>
```

The point object stores the corresponding coordinates:

```
363     <gml:featureMember>
364         <okstra:General_PointObject gml:id="General_PointObject_359525">
365             <okstra:Point_Geometry>
366                 <gml:Point gml:id="Point.359526">
367                     <gml:pos srsDimension="3">1240.55 1297.057 187.211</gml:pos>
368                 </gml:Point>
369             </okstra:Point_Geometry>
370             <!-- Property technical meaning missing -->
371             <okstra:Technical_Meaning/>
372             <okstra:In_Triangle Object_Class="Triangle" xlink:href="#Triangle_376877"/>
373         </okstra:General_PointObject>
374     </gml:featureMember>
375     <gml:featureMember>
```

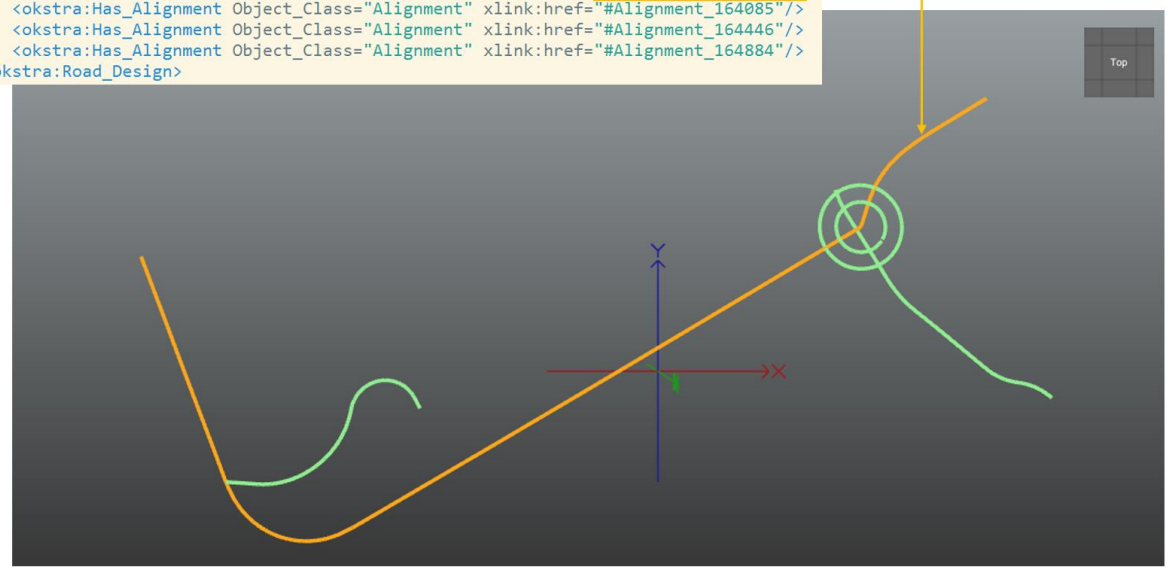
Example 4 (ex-terrain-alignment.xml)

The following shows an overview of the example scenario:



The example scenario contains a digital elevation model and an alignment model.

```
<okstra:Road_Design gml:id="Road_Design_162502">
    <okstra:Name>Road_Design</okstra:Name>
    <okstra:Has_Alignment Object_Class="Alignment" xlink:href="#Alignment_162651"/>
    <okstra:Has_Alignment Object_Class="Alignment" xlink:href="#Alignment_163182"/>
    <okstra:Has_Alignment Object_Class="Alignment" xlink:href="#Alignment_164085"/>
    <okstra:Has_Alignment Object_Class="Alignment" xlink:href="#Alignment_164446"/>
    <okstra:Has_Alignment Object_Class="Alignment" xlink:href="#Alignment_164884"/>
</okstra:Road_Design>
```



Consider Examples 1, 2 and 3 for more detail on the horizontal alignment, vertical alignment and digital elevation model.

Translation table

The document “Dictionary.xlsx” located in the same folder as this document contains an overview of the used Translations.