OpenDRIVE®
managing the road ahead

Project:	OpenDRIVE®	Document No.	Issue:
Title:	Format Specification, Rev. 1.3	VI2010.002	D
Date:	August 17, 2010	no. of pages:	83
Issuing Party:	VIRES Simulationstechnologie GmbH		
Author:	Marius Dupuis e.a.		
Distribution List:	public		

issued by
VIRES Simulationstechnologie GmbH
Oberaustrasse 34
83026 Rosenheim
Germany
phone +49.8031.463640
fax +49.8031.463645
e-mail opendrive@vires.com
web www.vires.com

Note:

Modifications to the previous release are highlighted with yellow background color.
For new chapters / items, only the title of the respective chapter has been highlighted.

OpenDRIVE® Format Specification, Rev. 1.3

Printed in Germany
August 2010

PUBLIC LICENSE STATEMENT
FOR
OpenDRIVE® Format Specification

Table of Contents

1 Preface

1.1 Scope

The OpenDRIVE® format provides a common base to describe track-based road networks. The data stored in an OpenDRIVE® file describes the geometry of roads as well as features along the roads that influence the logics (e.g. lanes, signs, signals).

The format is organized in nodes which can be extended with user-defined data. By this, a high degree of specialization for individual applications is feasible while maintaining the commonality required for the exchange of data between different applications.

1.2 Developers

The initial OpenDRIVE® format was developed by VIRES Simulationstechnologie GmbH, Germany, in close co-operation with Daimler Driving Simulator, Sindelfingen, Germany.

The contents of the file format are reviewed by a core team before publication. For the current members of the core team, please visit the OpenDRIVE® website (see below).

This standard has been created for its users. So, if you feel anything is missing, should be clarified or modified, please don't hesitate to contact us (see below).

1.3 Point of Contact

Further assistance on OpenDRIVE® is provided

via the OpenDRIVE® website

www.opendrive.org

via email

opendrive@vires.com

and via the "classic style" of direct contact with human beings:

VIRES Simulationstechnologie GmbH
Oberaustrasse 34
83026 Rosenheim
Germany
phone +49.8031.463640
fax +49.8031.463645

2 Conventions

2.1 Naming Conventions

In this document, the following conventions apply:

`data types`	are given according to IEEE standard
`track`	signifies the reference line of a road

2.2 Units

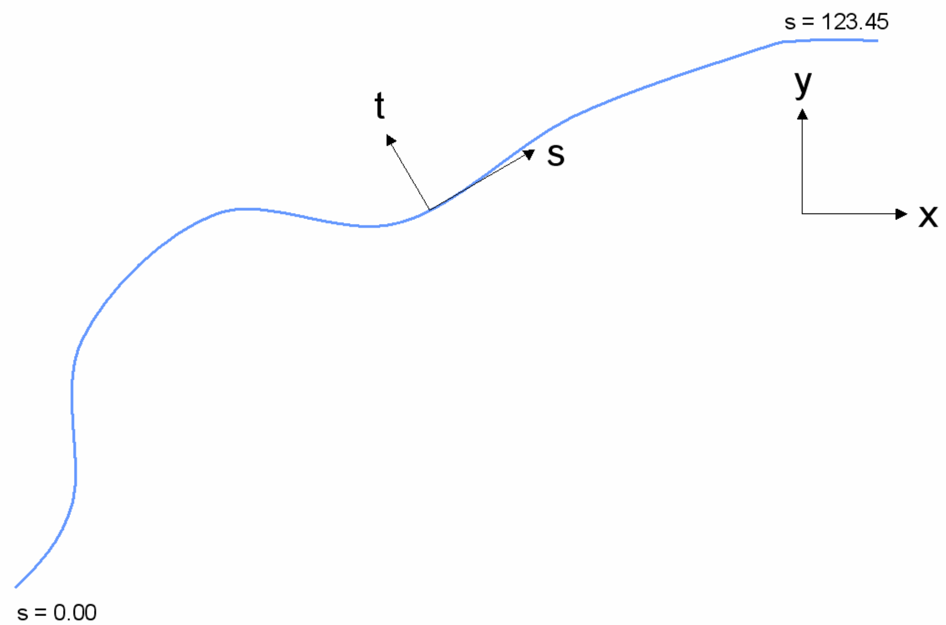
All numeric values within this specification are in SI units, e.g.:

position/distance	in [m]
angles	in [rad]
time	in [s]
speed	in [m/s]

2.3 Co-ordinate Systems

2.3.1 Overview

The following figure gives an overview of the two co-ordinate systems most frequently used in this specification – track co-ordinates and inertial co-ordinates (for details, see the following chapters).



2.3.2 Inertial System

The inertial system is a right-handed co-ordinate system according to ISO 8855 with the axes pointing to the following directions:

x	forward
y	left
z	up

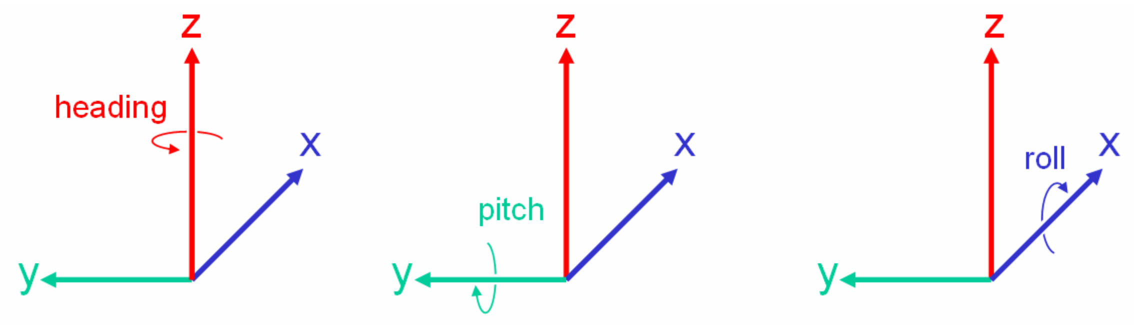
For geographic reference, the following convention applies:

x	east
y	north
z	up

Within the inertial system, the following angles are defined:

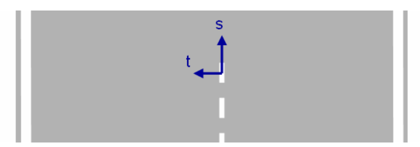
heading	around z-axis, 0.0 = east
pitch	around y-axis, 0.0 = level
roll	around x-axis, 0.0 = level

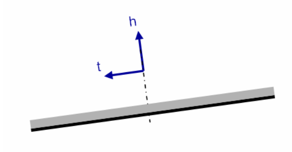
The following image shows the positive axes and positive directions of the corresponding angles.



2.3.3 Track System

The track co-ordinate system applies along the reference line of a road. It is also a right-handed co-ordinate system. The following degrees of freedom are defined:

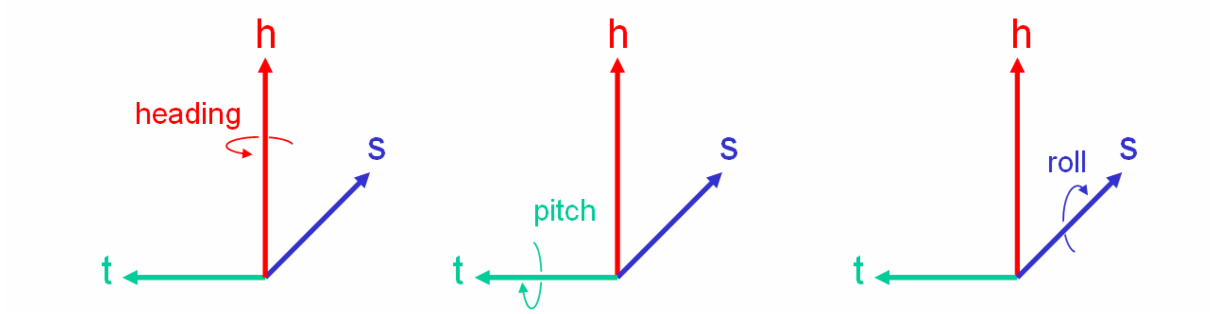




s	position along reference line, measured in [m] from the beginning of the track, calculated in the xy-plane (i.e. not taking into account the elevation profile of the track)
t	lateral position, positive to the left
h	up

heading	around h-axis, 0 = forward
pitch	around t-axis, 0 = level
roll	around s-axis, 0 = level

The following image shows the positive axes and positive directions of the corresponding angles.



2.3.4 Local System

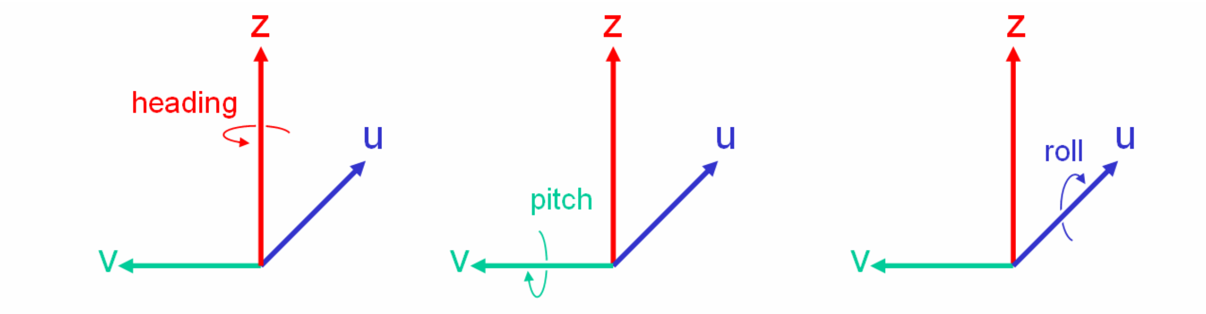
The local system is a right-handed co-ordinate system according to ISO 8855 with the axes pointing to the following directions:

u	forward
v	left
z	up

Within the local system, the following angles are defined:

heading	around z-axis, 0.0 = east
pitch	around v-axis, 0.0 = level
roll	around u-axis, 0.0 = level

The following image shows the positive axes and positive directions of the corresponding angles.



The local system can only be positioned in track space by providing the full track co-ordinates and orientation of its origin.

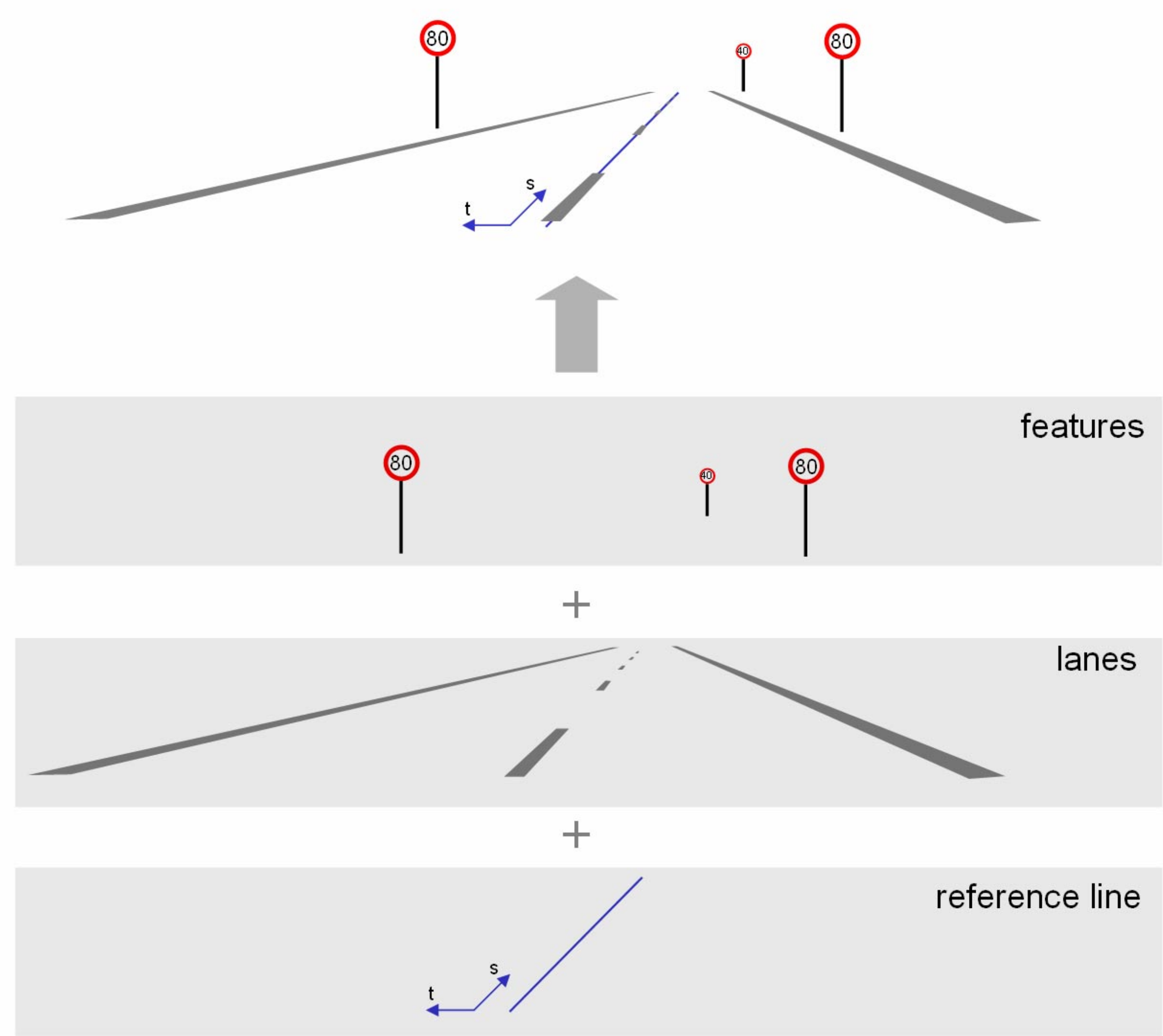
2.3.5 Curvature

Positive curvature indicates a left curve
Negative curvature indicates a right curve

3 Road Layout

3.1 General

The following figure depicts the principles of road layout covered by this specification:



All roads consist of a reference line which defines the basic geometry. Along the reference line, various properties of the road can be defined. These are, e.g. elevation profile, lanes, traffic signs etc. Roads can be linked to each other either directly (when there is only one connection possible between two given roads) or via junctions (when more than one connection is possible from a given road to other roads).

All properties may be parameterized according to the standards laid out in this specification and, optionally, by user-defined data.

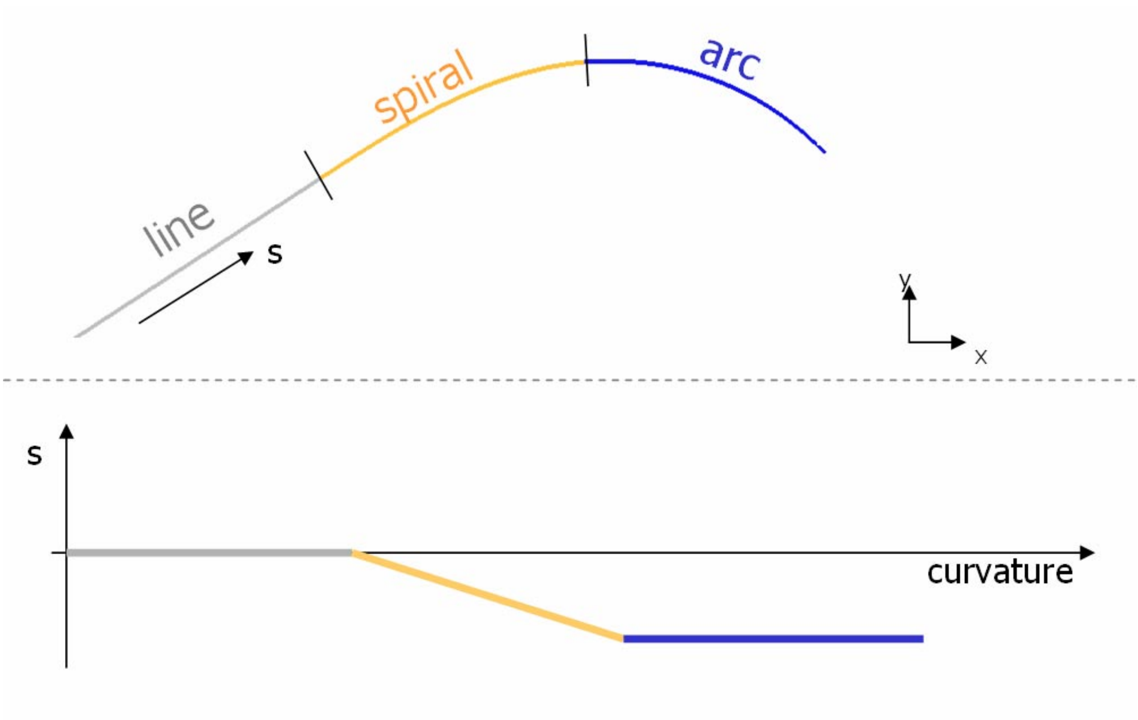
The convention applies that properties of the same type defined along a single reference line must be defined in ascending order. This means that the start co-ordinate (parameter s, see above) of a property must either be the same or greater than the start co-ordinate of the preceding property of same type on the same track.

3.2 Reference Line (Track)

The geometry of the reference line is described as a sequence of sections of various types. The available types are:

- straight line (constant zero curvature)
- spiral (linear change of curvature)
- curve (constant non-zero curvature)
- polynom (of 3^{rd} order)

The following figure illustrates this principle.



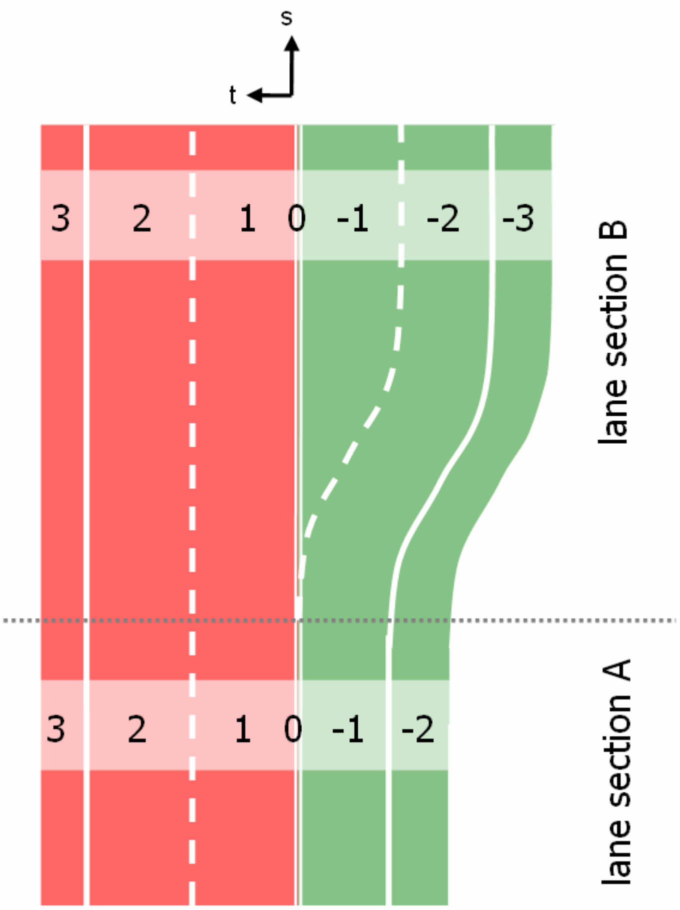
3.3 Lanes

3.3.1 General

Lanes are identified by numbers which are

- unique (per lane section, see below)
- in sequence (without gaps),
- starting from the reference line (lane no. 0)
- ascending to the left
- descending to the right

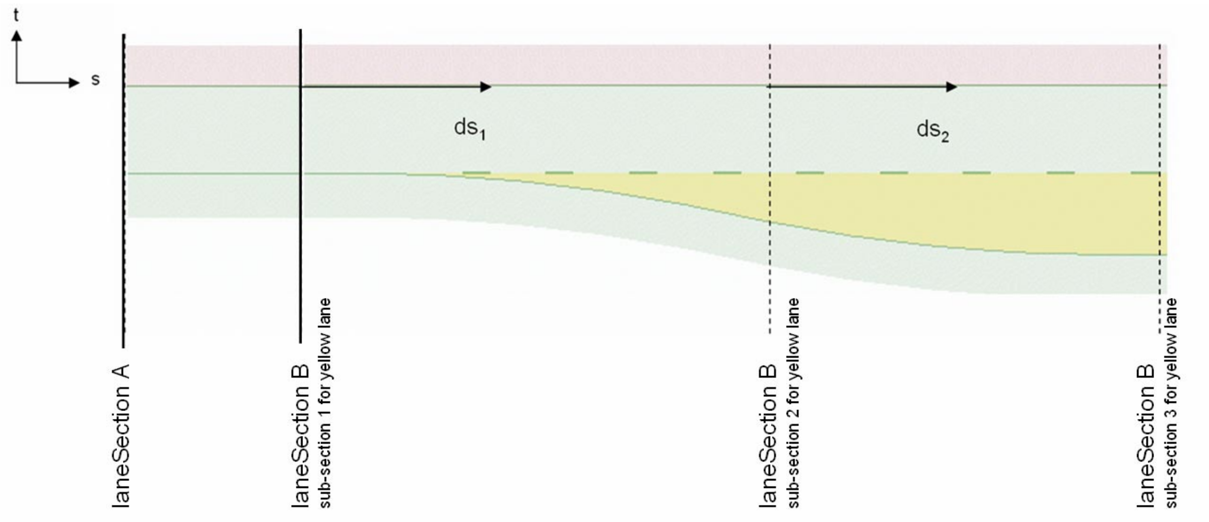
The total number of lanes is not limited. The reference line itself is defined as lane zero and must not have a width entry (i.e. the width must always be 0.0).



3.3.2 Lane Sections

The lanes appearing in a given cross-section along the road are defined in so-called lane sections. Multiple lane sections may be defined in ascending order along a reference line. Each lane section is valid until the next lane section is defined. Therefore, in order to be usable, each road must at least be equipped with one lane section starting at s= 0.0m.

The following figure depicts the principles of lane sections:



Per lane section, the number of lanes is constant. However, the properties of each lane (e.g. width, road marks, friction etc.) may change.

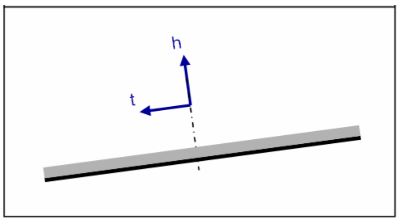
3.3.3 Lane Properties

Lane properties are defined relative to the start of the corresponding lane section. Offsets start at 0.0 for the beginning of the lane section and increase corresponding to the road co-ordinate s. Lane properties are valid until a new property of the same type is defined or the lane section ends. Lane properties of identical types must be defined in ascending order. Lane properties may be point or range properties. They are valid at a given point only or over a given range, respectively.

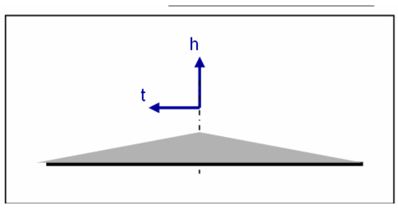
If a property is not defined within a given lane section or not covering the entire section, the application may assume default values.

3.4 Superelevation and Crossfall

In most cases, a road cross section will not be parallel to the surrounding terrain. Instead, it will be elevated to one side (e.g. in curves) or to the center (for drainage). Both properties are covered by the OpenDRIVE format with the former being called “superelevation” and the latter “crossfall”. The following figure illustrates both properties:



superelevation

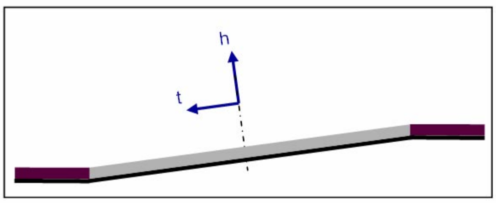


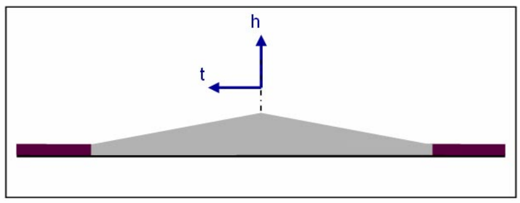
crossfall

Superelevation and crossfall can be superimposed in order to provide smooth transitions between e.g. straight sections with crossfall and curves with superelevation.

As can be seen from the above figures, the superelevation is defined per entire road cross section whereas the crossfall is defined per side of the road.

Single lanes can be excluded from the application of the superelevation and crossfall properties. Pedestrian walkways, for example, will always run on level planes (see following figure)





3.5 Road Linkage

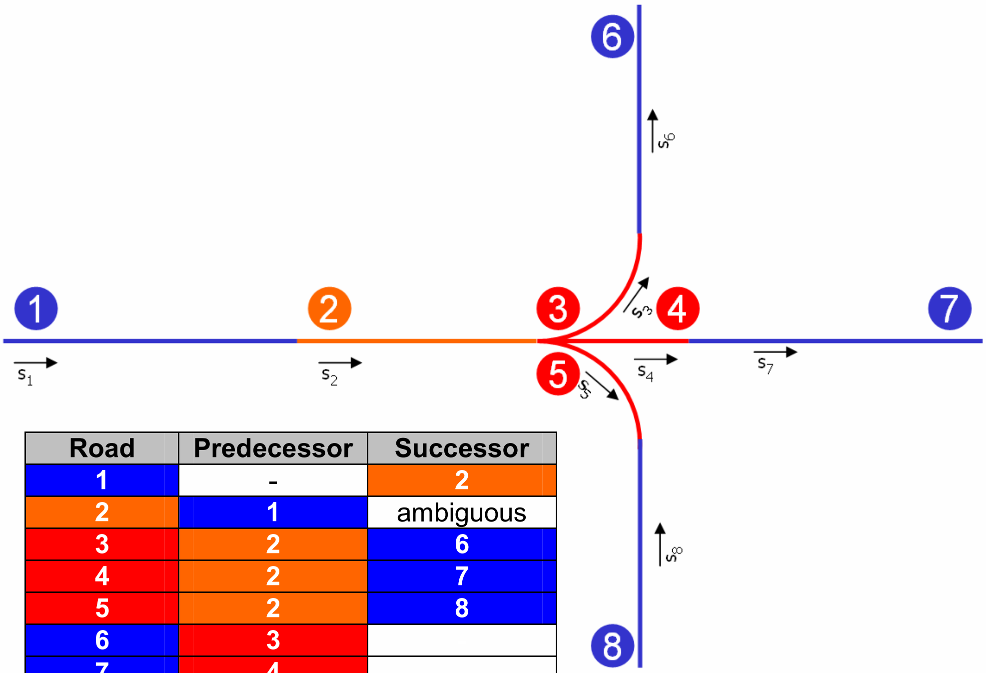
3.5.1 Overview

In order to navigate through a road network, the application must know about the linkage between different roads. Two types of linkage are possible:

- standard successor/predecessor linkage
- junctions

Whenever the linkage between two roads is clear, a standard linkage information is sufficient. A junction is required when the successor or predecessor relationship of a road is ambiguous. Here, the application needs to select one of several possibilities.

The following figure and table illustrate the different cases:



Road	Predecessor	Successor
1	-	2
2	1	ambiguous
3	2	6
4	2	7
5	2	8
6	3	
7	4	-
8	-	5

Usually, a junction would imply more ambiguous connections than listed in the table above.

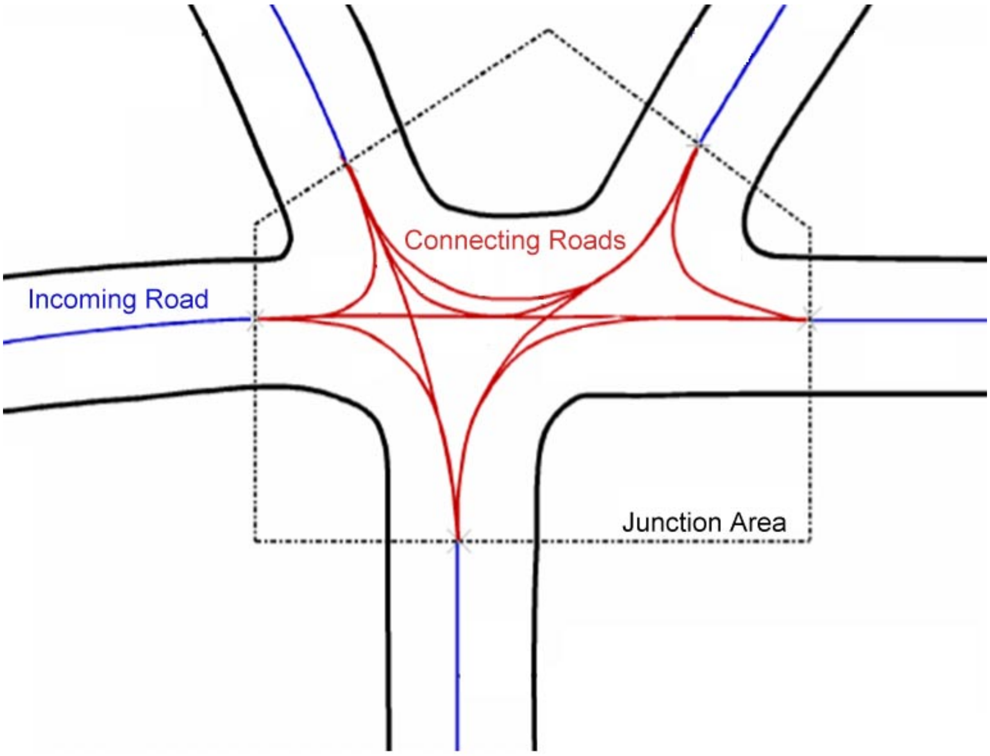
In order to facilitate navigation through a road network on a per-lane basis, additional linkage information can be provided on the lane level.

3.5.2 Junctions

The basic principle is very simple:

> Junctions link in-coming roads via paths (connecting roads) to out-going roads.

The following figure shows a complex junction scenario:



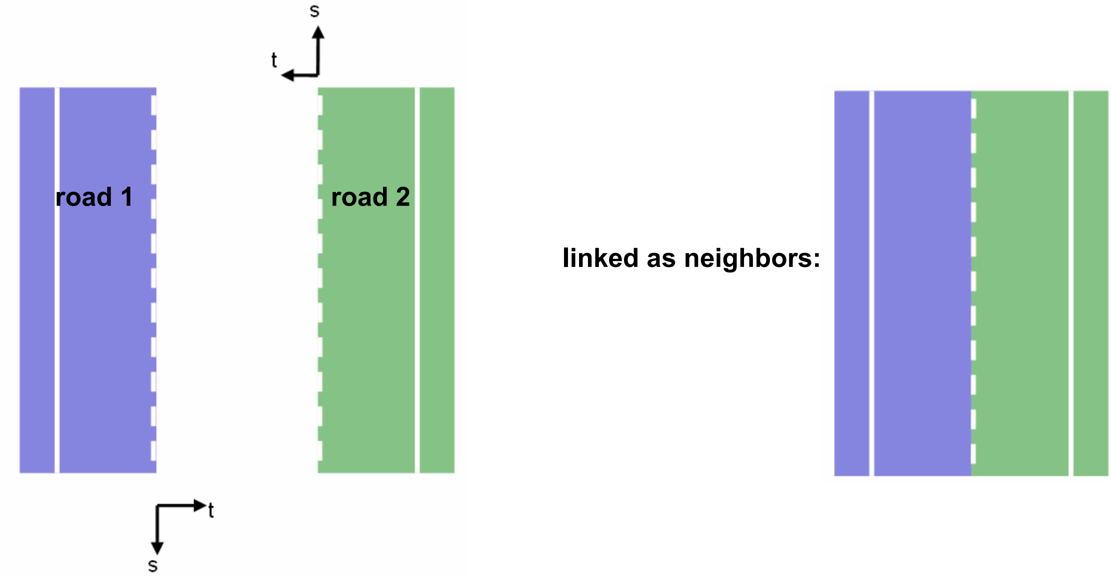
Connecting Roads are also modeled as roads according to the rules laid out here for all other roads. They consist of reference lines with lane sections etc. Usually, in-coming roads also serve as out-going roads, with the actual usage of a road being determined by its lanes.

Junctions consist of a connection matrix which indicates all possibilities to enter a connecting road from a given in-coming road. These connections are listed on a per-lane basis in order to facilitate navigation. Once a connecting road is entered, the following connection to the corresponding out-going road can be retrieved from the general successor/predecessor information that is stored with each road. Within the junctions, priorities of roads relative to each other may be stored but they may also be retrieved by evaluating the signs/signals and geometry.

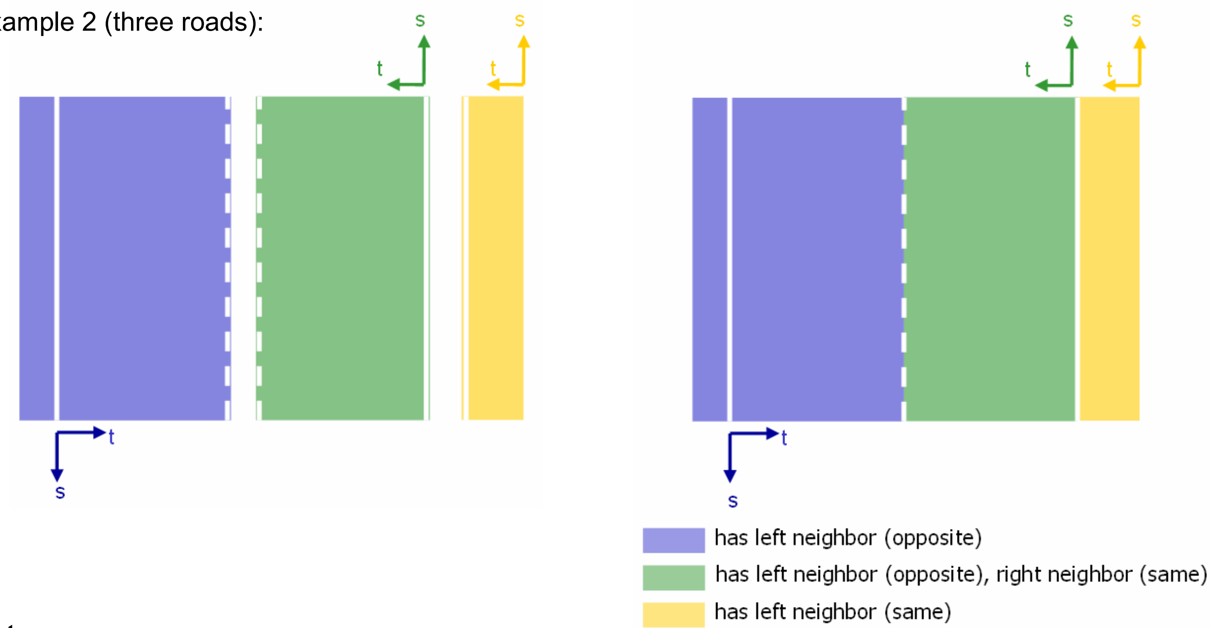
3.5.3 Neighbors

Roads may not only be linked to predecessors and successors but also to neighbors. This type of link information may be required when only one driving direction is defined per reference line (i.e. only left or only right lanes), or when only a fraction of the total number of lanes is defined per reference line. Each road may have up to two neighbors.

Example 1 (two roads being each other's neighbor):



Example 2 (three roads):



Note:
The neighbor entry has been introduced mainly for legacy purposes. For the design of new road networks, it is recommended to define both driving directions of a road along a single reference line and to avoid using the neighbor entry (this recommendation applies outside junctions only; within junctions, single reference lines may and should be used for each driving direction, however).

3.6 Surface

OpenDRIVE provides two approaches for describing surface properties:

- Standard description:

  In the standard case, a `<material>` record may be defined per lane of the road, providing parameters for
  - surface material code
  - roughness
  - friction

- Extended description (new in OpenDRIVE 1.2):

  A more detailed description of road surface data (e.g. from measurements), which is not limited to the definition of material properties within lane boundaries, may be provided within the newly introduced `<surface>` record. This data may be applied to an entire cross section or parts thereof.

  Due to the potentially very large amount of data that is contained in detailed surface information and due to the fact that publicly available data formats already exist for surface data, OpenDRIVE does not define its own XML implementation of surface descriptions but will provide references to the respective data files instead.

  Formats officially supported as surface import formats to OpenDRIVE are listed below. This list may be extended in future revisions of OpenDRIVE depending on the purposes served by additional formats.

  In order to guarantee the portability of an OpenDRIVE description, this standard recommends using the OpenCRG format as preferred extended road surface description.

  List of supported surface data formats:

no.	name	revision	description
1	Open Curved Regular Grid (OpenCRG)	June 2010	data format by TÜV Süd in co-operation with Daimler AG. Tools available by Daimler and VIRES Detailed information: http://www.opencrg.org

For the application of this data to the OpenDRIVE road see chapter 5.3.10.

3.7 Alternative Layouts

Depending on the individual application, it might be necessary to describe various possible setups of a road network's properties, so that the OpenDRIVE file does not only contain one description of these properties but provides access to all pre-defined setups.

For this purpose, the `<set>` record is being introduced (see 5.8). It allows the user to define alternative instances of a property within its level.

The `<set>` record may be used at any level without restrictions, however users should take into account that the portability of an OpenDRIVE road description may suffer with sets being defined at levels where another user might not expect them. In the future, this document shall contain hints where sets have been used successfully in applications, in order to give an indication where sets are encouraged and where they should be avoided.

4 File

4.1 Format

OpenDRIVE® data is stored in an XML file.

4.2 Extension

OpenDRIVE® files have the extension ".xodr".
Compressed OpenDRIVE® files have the extension ".xodrz".

4.3 Structure

The OpenDRIVE® file structure is laid out according to XML rules and with reference to the respective schema file.

Beads are organized in levels. Beads with a level greater than zero (0) are children of the preceding level. Beads with a level of one (1) are called primary beads.

Each bead can be extended with user-defined data. This data is stored in so-called ancillary beads (see 5.6)

4.4 Notation

All floating point numbers are "double" precision per default. It is highly recommended to use a 16 digit scientific representation for floating point numbers.

4.5 Schema

The schema file for the OpenDRIVE® format can be retrieved from www.opendrive.org

4.6 Combining Files

Multiple files can be combined by means of an `include` tag at the appropriate locations. Upon parsing this tag, OpenDRIVE readers shall immediately start reading the file specified as argument to the tag. It is the user's responsibility to make sure that contents read from an include file are consistent with the context from which the inclusion starts.

4.7 Overview of Beads

The following table provides a simplified overview of all beads that may occur within an OpenDRIVE™ file. It also indicates whether beads are optional or may occur multiple times. Children of optional beads must not be present if their respective parent is omitted. The levels of the beads are indicated by indentation and by the appropriate numbers.

bead name	level	optional	max. instances per parent	parent
OpenDRIVE	0	-	1	-
│-header	1	-	1	OpenDRIVE
│-road	1	-	unlimited	OpenDRIVE
│ │-link	2	+	1	road
│ │ │-predecessor	3	+	1	link
│ │ │-successor	3	+	1	link
│ │ │-neighbor	3	+	2	link
│ │-type	2	+	unlimited	road
│ │-planview	2	-	1	road
│ │ │-geometry	3	-	unlimited	planview
│ │ │ │-line	4	-	1	geometry
│ │ │ │-spiral	4	-	1	geometry
│ │ │ │-arc	4	-	1	geometry
│ │ │ │-poly3	4	-	1	geometry
│ │-elevationProfile	2	+	1	road
│ │ │-elevation	3	+	unlimited	elevationProfile
│ │-lateralProfile	2	+	1	road
│ │ │-superelevation	3	+	unlimited	lateralProfile
│ │ │-crossfall	3	+	unlimited	lateralProfile
│ │-lanes	2	-	1	road
│ │ │-laneOffset	3	+	unlimited	lanes
│ │ │-laneSection	3	-	unlimited	lanes
│ │ │ │-left	4	+	1	lane section
│ │ │ │ │-lane	5	-	unlimited	left
│ │ │ │ │ │-link	6	+	1	lane
│ │ │ │ │ │ │-predecessor	7	+	1	link
│ │ │ │ │ │ │-successor	7	+	1	link
│ │ │ │ │ │-width	6	-	unlimited	lane
│ │ │ │ │ │-roadMark	6	+	unlimited	lane
│ │ │ │ │ │ │ -type	7	+	1	roadMark
│ │ │ │ │ │ │ │ -line	8	-	unlimited	type
│ │ │ │ │ │-material	6	+	unlimited	lane
│ │ │ │ │ │-visibility	6	+	unlimited	lane
│ │ │ │ │ │-speed	6	+	unlimited	lane
│ │ │ │ │ │-access	6	+	unlimited	lane
│ │ │ │ │ │-height	6	+	unlimited	lane
│ │ │ │-center	4	+	1	lane section
│ │ │ │ │-lane	5	-	1	center
│ │ │ │ │ │-link	6	+	1	lane
│ │ │ │ │ │ │-predecessor	7	+	1	link
│ │ │ │ │ │ │-successor	7	+	1	link
│ │ │ │ │ │-roadMark	6	+	unlimited	lane
│ │ │ │ │ │ │ -type	7	+	1	roadMark
│ │ │ │ │ │ │ │ -line	8	-	unlimited	type
│ │ │ │-right	4	+	1	lane section
│ │ │ │ │-lane	5	-	unlimited	right
│ │ │ │ │ │-link	6	+	1	lane
│ │ │ │ │ │ │-predecessor	7	+	1	link
│ │ │ │ │ │ │-successor	7	+	1	link
│ │ │ │ │ │-width	6	-	unlimited	lane
│ │ │ │ │ │-roadMark	6	+	unlimited	lane
│ │ │ │ │ │ │ -type	7	+	1	roadMark
│ │ │ │ │ │ │ │ -line	8	-	unlimited	type
│ │ │ │ │ │-material	6	+	unlimited	lane
│ │ │ │ │ │-visibility	6	+	unlimited	lane
│ │ │ │ │ │-speed	6	+	unlimited	lane

<table>
<tr><th>bead name</th><th>level</th><th>optional</th><th>max. instances per parent</th><th>parent</th></tr>
<tr><td>│ │ │ │ │ │-access</td><td>6</td><td>+</td><td>unlimited</td><td>lane</td></tr>
<tr><td>│ │ │ │ │ │-height</td><td>6</td><td>+</td><td>unlimited</td><td>lane</td></tr>
<tr><td>│ │-objects</td><td>2</td><td>+</td><td>1</td><td>road</td></tr>
<tr><td>│ │ │-object</td><td>3</td><td>+</td><td>unlimited</td><td>objects</td></tr>
<tr><td>│ │ │ │-repeat</td><td>4</td><td>+</td><td>1</td><td>object</td></tr>
<tr><td>│ │ │ │-outline</td><td>4</td><td>+</td><td>1</td><td>object</td></tr>
<tr><td>~~│ │ │ │ │-cornerInertial~~</td><td>~~5~~</td><td>~~+~~</td><td>~~unlimited~~</td><td>~~outline~~</td></tr>
<tr><td>│ │ │ │ │-cornerRoad</td><td>5</td><td>+</td><td>unlimited</td><td>outline</td></tr>
<tr><td>~~│ │ │ │ │-cornerRelative~~</td><td>~~5~~</td><td>~~+~~</td><td>~~unlimited~~</td><td>~~outline~~</td></tr>
<tr><td>│ │ │ │ │-cornerLocal</td><td>5</td><td>+</td><td>unlimited</td><td>outline</td></tr>
<tr><td>│ │ │ │-material</td><td>4</td><td>+</td><td>1</td><td>object</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>number of lanes</td><td>object</td></tr>
<tr><td>│ │ │-objectReference</td><td>3</td><td>+</td><td>unlimited</td><td>objects</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>lanes in road</td><td>objectReference</td></tr>
<tr><td>│ │ │-tunnel</td><td>3</td><td>+</td><td>unlimited</td><td>objects</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>number of lanes</td><td>tunnel</td></tr>
<tr><td>│ │ │-bridge</td><td>3</td><td>+</td><td>unlimited</td><td>objects</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>number of lanes</td><td>bridge</td></tr>
<tr><td>│ │-signals</td><td>2</td><td>+</td><td>1</td><td>road</td></tr>
<tr><td>│ │ │-signal</td><td>3</td><td>+</td><td>unlimited</td><td>signals</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>lanes in road</td><td>signal</td></tr>
<tr><td>│ │ │ │-dependency</td><td>4</td><td>+</td><td>unlimited</td><td>signal</td></tr>
<tr><td>│ │ │-signalReference</td><td>3</td><td>+</td><td>unlimited</td><td>signals</td></tr>
<tr><td>│ │ │ │-validity</td><td>4</td><td>+</td><td>lanes in road</td><td>signalReference</td></tr>
<tr><td>│ │-surface</td><td>2</td><td>+</td><td>1</td><td>road</td></tr>
<tr><td>│ │ │-CRG</td><td>3</td><td>+</td><td>unlimited</td><td>surface</td></tr>
<tr><td>│-controller</td><td>1</td><td>+</td><td>unlimited</td><td>OpenDRIVE</td></tr>
<tr><td>│ │-control</td><td>2</td><td>-</td><td>number of signals</td><td>controller</td></tr>
<tr><td>│-junction</td><td>1</td><td>+</td><td>unlimited</td><td>OpenDRIVE</td></tr>
<tr><td>│ │-connection</td><td>2</td><td>-</td><td>unlimited</td><td>junction</td></tr>
<tr><td>│ │ │-laneLink</td><td>3</td><td>+</td><td>unlimited</td><td>connection</td></tr>
<tr><td>│ │-priority</td><td>2</td><td>+</td><td>unlimited</td><td>junction</td></tr>
<tr><td>│ │-controller</td><td>2</td><td>+</td><td>unlimited</td><td>junction</td></tr>
</table>

At any level, there may be the following beads:

<table>
<tr><th>bead name</th><th>level</th><th>optional</th><th>max. instances per parent</th><th>parent</th></tr>
<tr><td>userData</td><td>any</td><td>+</td><td>unlimited</td><td>any</td></tr>
<tr><td>include</td><td>any</td><td>+</td><td>unlimited</td><td>any</td></tr>
<tr><td>set</td><td>any</td><td>+</td><td>unlimited</td><td>any</td></tr>
<tr><td>│-instance</td><td>set+1</td><td>-</td><td>unlimited</td><td>set</td></tr>
</table>

5 File Entries

5.1 Enclosing Tag

The overall enclosing tag of the file is:

Delimiters: `<OpenDRIVE>...</OpenDRIVE>`

Maximum Instances: `1`

Optional: `no`

Arguments: `none`

5.2 Header

The header record is the very first record within the `OpenDRIVE` tag.

Delimiters: `<header>...</header>`

Parent: `<OpenDRIVE>`

Maximum Instances: `1`

Optional: `no`

Arguments:

name	***type***	***description***
`revMajor`	ushort	major revision number of OpenDRIVE® format (currently 1)
`revMinor`	ushort	minor revision number of OpenDRIVE® format (currently 3)
`name`	string	database name
`version`	float	version number of this database (format: a.bb)
`date`	string	time/date of database creation
`north`	double	maximum inertial y value in [m]
`south`	double	minimum inertial y value in [m]
`east`	double	maximum inertial x value in [m]
`west`	double	minimum inertial x value in [m]
`vendor`	string	vendor name

5.3 Road Records

Roads are the principal containers of information within a database. For an overview of records which can be stored within a road, see chapter 4.7.

5.3.1 Road Header Record

The road header record defines the basic parameters of an individual road. It is followed immediately by other records defining geometry and logical properties of the road.

Delimiters: `<road>...</road>`

Parent: `<OpenDRIVE>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	type	description
`name`	string	name of the road
`length`	double	total length of the reference line in the xy-plane
`ID`	string	unique ID within database
`junction`	string	ID of the junction to which the road belongs as a path (= -1 for none)

5.3.2 Road Link Record

This record follows the Road Header Record if the road is (as usual) linked to a successor, a predecessor, or a neighbor (see children of the link record). Isolated roads may omit this record.

Delimiters: `<link>...</link>`

Parent: `<road>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.2.1 Road Predecessor

This record provides detailed information about the predecessor of a road. The predecessor may be of type road or junction.

Delimiters: `<predecessor.../>`

Parent: `<link>`

Maximum Instances: `1`

Optional: `yes`

Arguments:

name	***type***	***description***
`elementType`	string	*`road`* or *`junction`*
`elementId`	string	ID of the linked element
`contactPoint`	string	contact point of link on the linked element, may be *`start`* or *`end`*

5.3.2.2 Road Successor

This record provides detailed information about the successor of a road. The successor may be of type road or junction.

Delimiters: `<successor.../>`

Parent: `<link>`

Maximum Instances: `1`

Optional: `yes`

Arguments:

name	***type***	***description***
`elementType`	string	*`road`* or *`junction`*
`elementId`	string	ID of the linked element
`contactPoint`	string	contact point of link on the linked element, may be *`start`* or *`end`*

5.3.2.3 Road Neighbor

This record provides detailed information about the neighbor of a road. The neighbor must be of type road.

Note: The neighbor entry has been introduced mainly for legacy purposes. For the design of new road networks, it is recommended to define both driving directions of a road along a single reference line and to avoid using the neighbor entry (this recommendation applies outside junctions only; within junctions, single reference lines may and should be used for each driving direction, however).

Delimiters: `<neighbor.../>`

Parent: `<link>`

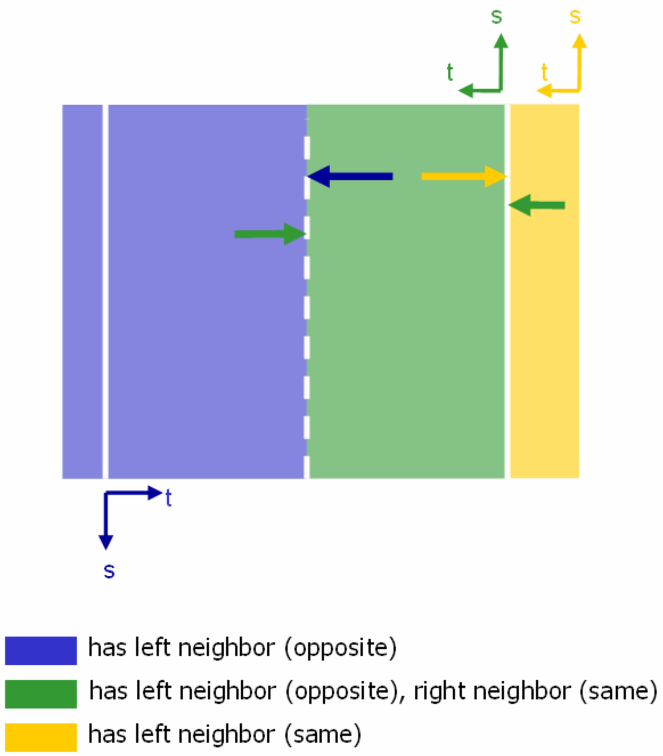
Maximum Instances: `2`

Optional: `yes`

Arguments:

name	type	description
`side`	string	*left* or *right*
`elementId`	string	ID of the linked road
`direction`	string	*`same`* or *`opposite`*

Illustration:



5.3.3 Road Type Record

The type of a road may change over its entire length. Therefore, a separate record is provided for the definition of the road type with respect to a certain section of the road. A road type entry is valid for the entire cross section of the road. It is also valid until a new road type record is provided or the road ends.

Delimiters: `<type.../>`

Parent: `<road>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	start position (s-coordinate)
`type`	string	for supported types, see chapter 6.1

5.3.4 Road Plan View Record

The plan view record contains a series of geometry records which define the layout of the road's reference line in the x/y-plane (plan view).

Delimiters:	`<planView>...</planView>`
Parent:	`<road>`
Maximum Instances:	`1`
Optional:	`no`
Arguments:	`none`

5.3.4.1 Road Geometry Header Record

A sequence of road geometry records defines the layout of the road's reference line in the in the x/y-plane (plan view). The geometry records must occur in ascending order (i.e. increasing s-position). The geometry information is split into a header which is common to all geometric elements and a subsequent bead containing the actual geometric element's data (depending on the type of geometric element).

Currently, three types of geometric elements are supported:

- straight lines
- spirals
- arcs

Delimiters:	`<geometry>...</geometry>`
Parent:	`<planView>`
Maximum Instances:	`unlimited`
Optional:	`no`

Arguments:

name	***type***	***description***
`s`	double	start position (s-coordinate)
`x`	double	start position (x inertial)
`y`	double	start position (y inertial)
`hdg`	double	start orientation (inertial heading)
`length`	double	length of the element's reference line

This record is followed immediately by a record with more information about the actual geometry element.

5.3.4.1.1 Geometry, Line Record

This record describes a straight line as part of the road's reference line.

Delimiters: `<line.../>`

Parent: `<geometry>`

Maximum Instances: 1

Optional: no

Arguments: none

5.3.4.1.2 Geometry, Spiral Record (Clothoids)

This record describes a spiral as part of the road's reference line. For this type of spiral, the curvature change between start and end of the element is linear.

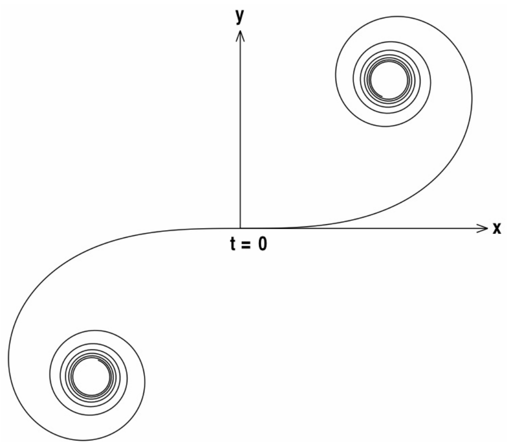
In order to provide consistency between spiral evaluations of various implementations, users are highly recommended to use identical algorithms, or, at least, algorithms with reasonable accuracy within the applicable range (i.e. curvature, length of spiral etc.).

The theory of clothoids is described (for example) at

http://en.wikipedia.org/wiki/Euler_spiral

A library for computing clothoids can be downloaded at

http://www.opendrive.org/downloads/spiral.zip



Delimiters: `<spiral.../>`

Parent: `<geometry>`

Maximum Instances: 1

Optional: no

Arguments:

name	*type*	*description*
`curvStart`	double	curvature at the start of the element
`curvEnd`	double	curvature at the end of the element

5.3.4.1.3 Geometry, Arc Record

This record describes an arc as part of the road's reference line.

Delimiters: `<arc.../>`

Parent: `<geometry>`

Maximum Instances: `1`

Optional: `no`

Arguments:

name	***type***	***description***
`curvature`	double	constant curvature throughout the element

5.3.4.1.4 Geometry, Cubic Polynom Record

This record describes a cubic polynom as part of the road's reference line. The polynom is described in the local u/v co-ordinate system of the starting point (with u pointing in the local s direction and v pointing in the local t direction). Each local co-ordinate is calculated by:

$$v_{local} = a + b*du + c*du^2 + d*du^3$$

The conversion of u/v co-ordinates into x/y co-ordinates can be performed easily by simple geometric transformations (i.e. one translation and one rotation) relative to the starting point

Delimiters: `<poly3.../>`

Parent: `<geometry>`

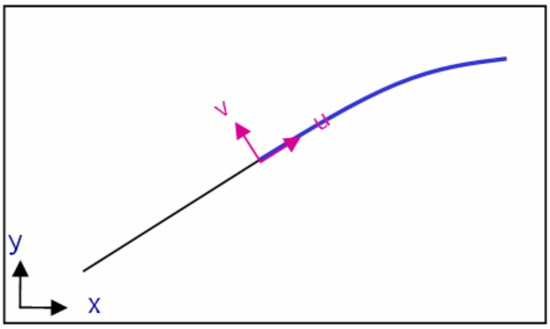
Maximum Instances: 1

Optional: no

Arguments:

name	type	description
a	double	parameter A
b	double	parameter B
c	double	parameter C
d	double	parameter D

Illustration:



5.3.5 Road Elevation Profile Record

The elevation profile record contains a series of elevation records which define the characteristics of the road's elevation along the reference line.

Delimiters:	`<elevationProfile>...</elevationProfile>`
Parent:	`<road>`
Maximum Instances:	`1`
Optional:	`yes`
Arguments:	`none`

5.3.5.1 Road Elevation Record

The elevation record defines an elevation entry at a given reference line position. Entries must be defined in increasing order along the reference line. The parameters of an entry are valid until the next entry starts or the road's reference line ends. Per default, the elevation of a road is zero.

The elevation is stored in a polynomial function of third order. It looks like:

$$elev = a + b*ds + c*ds^2 + d*ds^3$$

with

`elev`	being the elevation (inertial z) at a given position
`a, b, c, d`	being the coefficients and
`ds`	being the distance along the reference line between the start of the entry and the actual position.

Therefore, ds starts at zero for each entry. The absolute position of an elevation value is calculated by

$$s = s_{start} + ds$$

with

s	being the absolute position (track co-ordinate system)
s_{start}	being the start position of the entry in the track co-ordinate system
ds	being the delta between the start position and the requested position

Delimiters: `<elevation.../>`

Parent: `<elevationProfile>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	type	description
s	double	start position (s-coordinate)
a	double	parameter A (elevation in [m])
b	double	parameter B
c	double	parameter C
d	double	parameter D

5.3.6 Road Lateral Profile Record

The lateral profile record contains a series of superelevation and crossfall records which define the characteristics of the road surface's banking along the reference line.

Delimiters: `<lateralProfile>...</lateralProfile>`

Parent: `<road>`

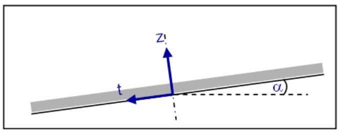
Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.6.1 Road Superelevation Record

The superelevation of the road is defined as the road section's roll angle around the s-axis (superelevation is positive for road surfaces "falling" to the right side, i.e. the following figure shows a negative superelevation).



Each superelevation record defines an entry at a given reference line position. Entries must be defined in increasing order along the reference line. The parameters of an entry are valid until the next entry starts or the road's reference line ends. Per default, the superelevation of a road is zero.

The superelevation is stored in a polynomial function of third order. It looks like:

$$sElev = a + b*ds + c*ds^2 + d*ds^3$$

with

sElev	being the superelevation at a given position, default: sElev = 0
a, b, c, d	being the coefficients and
ds	being the distance along the reference line between the start of the entry and the actual position.

Therefore, ds starts at zero for each entry. The absolute position of a superelevation value is calculated by

$$s = s_{start} + ds$$

with

s	being the absolute position (track co-ordinate system)
s_{start}	being the start position of the entry in the track co-ordinate system
ds	being the delta between the start position and the requested position

Delimiters: <superelevation.../>

Parent: <lateralProfile>

Maximum Instances: unlimited

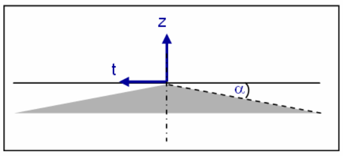
Optional: yes

Arguments:

name	type	description
s	double	start position (s-coordinate)
a	double	parameter A (superelevation in [rad])
b	double	parameter B
c	double	parameter C
d	double	parameter D

5.3.6.2 Crossfall Record

The crossfall of the road is defined as the road surface´s angle relative to the t-axis. Positive crossfall results in a road surface “falling” from the reference line to the outer boundary.



The crossfall can be defined per side of the road. It will be applied to each lane of the respective side which is not being explicitly excluded from its application.

The crossfall is stored in a polynomial function of third order. It looks like:

$$crfall = a + b*ds + c*ds^2 + d*ds^3$$

with

`crfall`	being the crossfall at a given position, default: `crfall = 0`
`a, b, c, d`	being the coefficients and
`ds`	being the distance along the reference line between the start of the entry and the actual position.

Therefore, ds starts at zero for each entry. The absolute position of a crossfall value is calculated by

$$s = s_{start} + ds$$

with

s	being the absolute position (track co-ordinate system)
s_{start}	being the start position of the entry in the track co-ordinate system
ds	being the delta between the start position and the requested position

Delimiters: `<crossfall.../>`

Parent: `<lateralProfile>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`side`	string	applicable side of the road (left / right / both)
`s`	double	start position (s-coordinate)
`a`	double	parameter A (crossfall in [rad])
`b`	double	parameter B
`c`	double	parameter C
`d`	double	parameter D

5.3.7 Road Lanes Record

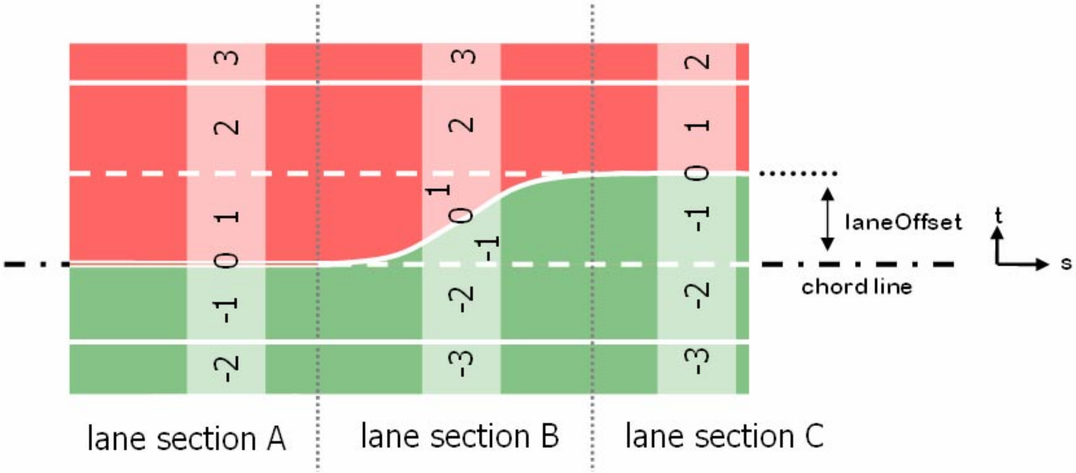
The lanes record contains a series of lane section records which define the characteristics of the road cross sections with respect to the lanes along the reference line.

Delimiters:	`<lanes>...</lanes>`
Parent:	`<road>`
Maximum Instances:	`1`
Optional:	`no`
Arguments:	`none`

5.3.7.1 Road Lane Offset Record

The lane offset record defines a lateral shift of the lane reference line (which is usually identical to the road reference line). This may be used for an easy implementation of a (local) lateral shift of the lanes relative to the road's reference line. Especially the modeling of inner-city layouts or "2+1" cross-country road layouts can be facilitated considerably by this feature.

The following figure illustrates the modeling of additional "inner" lanes along a straight reference line:



The actual offset at a given point is computed with a polynomial function of third order. It looks like:

$$\texttt{offset} = a + b*ds + c*ds^2 + d*ds^3$$

with

`offset`	being the lateral offset at a given position
`a, b, c, d`	being the coefficients and
`ds`	being the distance along the reference line between the start of the entry and the actual position.

Therefore, ds starts at zero for each entry. The absolute position of a width value is calculated by

$$s = s_{start} + ds$$

with

s	being the absolute position (track co-ordinate system)
s_{start}	being the start position of the lateral offset entry in the track co-ordinate system
ds	being the delta between the start position (s_{start}) and the requested position

A new lane offset entry is required each time the polynomial function changes.

Delimiters: `<laneOffset.../>`

Parent: `<lanes>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
s	double	start position (s-coordinate)
a	double	parameter A (offset in [m])
b	double	parameter B
c	double	parameter C
d	double	parameter D

5.3.7.2 Road Lane Section Record

The lane section record defines the characteristics of a road cross-section. It specifies the numbers and types of lanes and further features of the lanes. At least one record must be defined in order to use a road. A lane section record is valid until a new lane section record is defined. If multiple lane section records are defined, they must be listed in ascending order.

The actual lanes and their properties are children of the lane section record and the lane records, respectively.

Delimiters: `<laneSection>...</laneSection>`

Parent: `<lanes>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	***type***	***description***
s	double	start position (s-coordinate)

For the naming convention of lanes, see chapter 3.2.

5.3.7.2.1 Left / Center / Right Records

For easier navigation through a road description, the lanes under a lane section are grouped into left, center and right lanes. At least one entry (left, center or right) must be present.

Delimiters: `<left>...</left>` `<center>...</center>` `<right>...</right>`

Parent: `<laneSection>`

Maximum Instances: `1`

Optional: `no`

Arguments: `none`

5.3.7.2.1.1 Lane Record

Lane records are found within the left/center/right records. They define the IDs of the actual lanes (and, therefore, their position on the road, see conventions in 3). In order to prevent confusion, lane records should represent the lanes from left to right (i.e. with descending ID). All properties of the lanes are defined as children of the lane records.

Delimiters: `<lane>...</lane>`

Parent: `<left> / <center> / <right>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	***type***	***description***
`id`	int	id of the lane (according to convention)
`type`	string	type of the lane, see chapter 6.5
`level`	string	"true" = keep lane on level, .i.e. do not apply superelevation or crossfall "false" = apply superelevation and crossfall to this lane (default, also used if argument `level` is missing) lanes are also kept on level if the argument `level` is present but no superelevation or crossfall have been defined.

5.3.7.2.1.1.1 Lane Link Record

In order to facilitate navigation through a road network on a per-lane basis, lanes should be provided with predecessor/successor information. Only when lanes end at a junction or have no physical link, this record should be omitted.

For links between lanes on the same physical road (i.e. identical reference line), the lane predecessor/ successor information provides the ids of lanes on the preceding or following lane section. For links between lanes on different roads, i.e. roads directly connecting to each other without a junction, the complete link information must be composed from the corresponding road link record and the lane link record.

Delimiters: `<link>...</link>`

Parent: `<lane>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.7.2.1.1.1.1 Lane Predecessor

This record provides detailed information about the predecessor of a lane.

Delimiters: `<predecessor.../>`

Parent: `<link>`

Maximum Instances: `1`

Optional: `yes`

Arguments:

name	***type***	***description***
`id`	int	ID of the linked lane

5.3.7.2.1.1.1.2 Lane Successor

This record provides detailed information about the successor of a lane.

Delimiters: `<successor.../>`

Parent: `<link>`

Maximum Instances: `1`

Optional: `yes`

Arguments:

name	***type***	***description***
`id`	int	ID of the linked lane

5.3.7.2.1.1.2 Lane Width Record

Each lane within a road cross section can be provided with several width entries. At least one entry must be defined for each lane, except for the center lane which is, per convention, of zero width (see 3.2). Each entry is valid until a new entry is defined. If multiple entries are defined for a lane, they must be listed in ascending order.

The actual width at a given point is computed with a polynomial function of third order. It looks like:

$$width = a + b*ds + c*ds^2 + d*ds^3$$

with

`width`	being the width at a given position
`a, b, c, d`	being the coefficients and
`ds`	being the distance along the reference line between the start of the entry and the actual position.

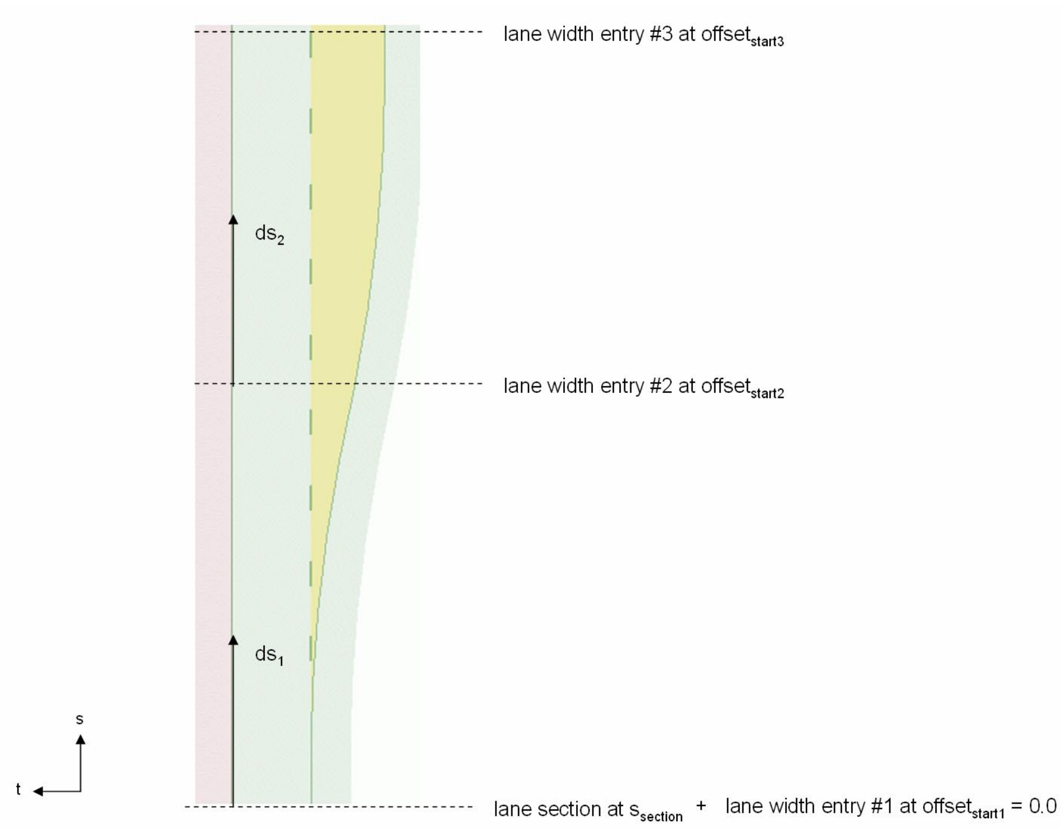
Therefore, ds starts at zero for each entry. The absolute position of a width value is calculated by

$$s = s_{section} + offset_{start} + ds$$

with

s	being the absolute position (track co-ordinate system)
$s_{section}$	being the start position of the preceding lane section record (see 5.3.7) in the track co-ordinate system
$offset_{start}$	being the offset of the entry relative to the preceding lane section record
ds	being the delta between the offset (s_{start}) and the requested position

The following figure illustrates this convention for a lane with varying width over a given range:



A new width entry is required each time the polynomial function changes.

Delimiters: `<width.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `no`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	***type***	***description***
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *`laneSection`* record
`a`	double	parameter A (width in [m])
`b`	double	parameter B
`c`	double	parameter C
`d`	double	parameter D

5.3.7.2.1.1.3 Road Mark Record

Each lane within a road cross section can be provided with several road mark entries. The road mark information defines the style of the line at the lane's outer border. For left lanes, this is the left border, for right lanes the right one. The style of the line separating left and right lanes is determined by the road mark entry for lane zero (i.e. the center lane)

Delimiters: `<roadMark>...</roadMark>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `yes`

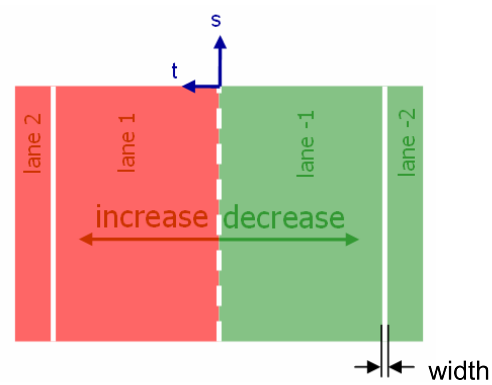
Arguments:

name	***type***	***description***
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *`laneSection`* record
`type`	string	type of the road mark, see chapter 6.2
`weight`	string	weight of the road mark, see chapter 6.3
`color`	string	color of the road mark, see chapter 6.4
`width`	double	width of the road mark in [m] – optional
`laneChange`	string	`[increase	decrease	both	none]` allow a lane change in the indicated direction taking into account that lanes are numbered in ascending order from right to left. If the argument is missing, "both" is assumed to be valid.

The parameter `weight` may be used for a categorized definition of the width of a road mark (e.g. "standard" and "bold" according to the corresponding country's design rules) whereas the optional parameter `width` may be used for an exact definition of an individual road mark's width. This may be required e.g. in cases which deviate from a common design rule.

For an exact evaluation or a road mark's borders including the width of the mark, the convention shall apply that a road mark's centerline is always positioned on the respective lane's outer border line (so that the outer half of the road mark is physically placed on the next lane).

Illustration:



5.3.7.2.1.1.3.1 Road Mark Type

Road marks may be further described in terms of their type. Instead of providing additional parameters to the `roadMark` tag, type definitions shall be defined as children to the `roadMark` tag. Each `type` definition must contain one or more `line` definitions with additional information about the lines the road mark is composed of.

Delimiters: `<type>...</type>`

Parent: `<roadMark>`

Maximum Instances: `1`

Optional: `yes`

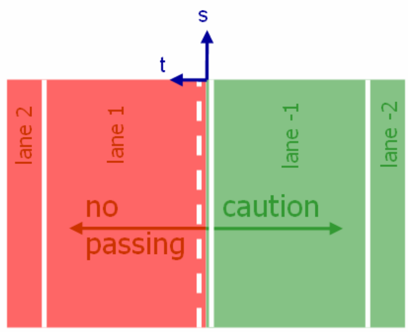
Arguments:

name	***type***	***description***
`name`	string	name of the road mark type
`width`	double	line width [m]

5.3.7.2.1.1.3.1.1 Road Mark Type – Line Definition

A road mark may consist of one or more elements. Multiple elements will usually be positioned side by side.

Currently, the elements can only be lines. Each line can be defined by its line/space setup and by a lateral offset to the road mark's reference position (i.e. the border between two lanes). A line definition is valid for a given length (i.e. the total of visible line and invisible space) and will be repeated automatically.



Delimiters: `<line.../>`

Parent: `<type>`

Maximum Instances: `unlimited`

Optional: `no`

Restrictions: `none`

Arguments:

name	***type***	***description***
`length`	double	length of the visible part [m]
`space`	double	length of the space following the visible part [m]
`tOffset`	double	lateral offset from the lane border [m]
`sOffset`	double	initial longitudinal offset from of the line definition from the start of the road mark definition [m]
`rule`	string	`[ no passing | caution | none ]` rule which is to be observed when passing the line from inside

5.3.7.2.1.1.4 Lane Material Record

Each lane within a road cross section may be provided with several entries defining its material. Each entry is valid until a new entry is defined. If multiple entries are defined, they must be listed in increasing order.

Delimiters: `<material.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `yes`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	type	description
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *laneSection* record
`surface`	string	surface material code [-], depending on application
`friction`	double	friction value [-]
`roughness`	double	roughness [-] (e.g. for sound and motion systems)

5.3.7.2.1.1.5 Lane Visibility Record

Each lane within a road cross section may be provided with several entries defining the visibility in four directions relative to the lane's direction. Each entry is valid until a new entry is defined. If multiple entries are defined, they must be listed in increasing order.

For left lanes (positive ID), the forward direction is oriented opposite to the track's direction, for right lanes, the forward direction and the track's direction are identical.

Delimiters: `<visibility.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `yes`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	type	description
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *laneSection* record
`forward`	double	visibility in forward direction [m]
`back`	double	visibility in reverse direction [m]
`left`	double	visibility to the left [m]
`right`	double	visibility to the right [m]

5.3.7.2.1.1.6 Lane Speed Record

This record defines the maximum allowed speed on a given lane. Each entry is valid in direction of the increasing s co-ordinate until a new entry is defined. If multiple entries are defined, they must be listed in increasing order.

Delimiters: `<speed.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `yes`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	type	description
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *`laneSection`* record
`max`	double	maximum allowed speed in [m/s]

5.3.7.2.1.1.7 Lane Access Record

This record defines access restrictions for certain types of road users. The record can be used to complement restrictions resulting from signs or signals in order to control the traffic flow in a scenario. Each entry is valid in direction of the increasing s co-ordinate until a new entry is defined. If multiple entries are defined, they must be listed in increasing order.

Delimiters: `<access.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

Optional: `yes`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	type	description
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *`laneSection`* record
`restriction`	string	identifier of the participant to which the restriction applies, see also 6.9

5.3.7.2.1.1.8 Lane Height Record

The surface of a lane may be offset from the plane defined by the reference line and the corresponding elevation and crossfall entries (e.g. pedestrian walkways are typically a few centimeters above road level). The height record provides a simplified method to describe this offset by setting an inner and outer offset from road level at discrete positions along the lane profile.

Delimiters: `<height.../>`

Parent: `<lane>`

Maximum Instances: `unlimited`

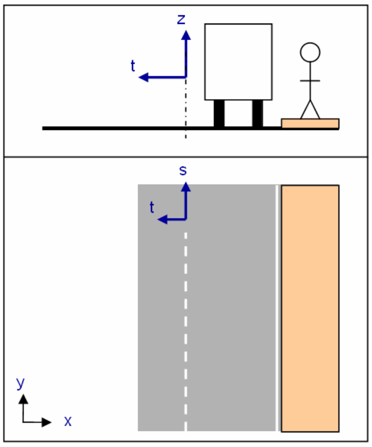
Optional: `yes`

Restrictions: `not allowed for center lane (laneId=0)`

Arguments:

name	***type***	***description***
`sOffset`	double	start position (s-coordinate) relative to the position of the preceding *`laneSection`* record
`inner`	double	inner offset from road level in [m]
`outer`	double	outer offset from road level in [m]

Illustration:



5.3.8 Road Objects Record

The objects record is the container for all objects along a road.

Delimiters: `<objects>...</objects>`

Parent: `<road>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.8.1 Object Record

The object record has been introduced for application-dependent elements. It is very flexible due to several parameters that can be used to provide the application with additional information. The most frequently used types of objects may become part of the OpenDRIVE® specification in future releases.

Delimiters: `<object>...</object>`

Parent: `<objects>`

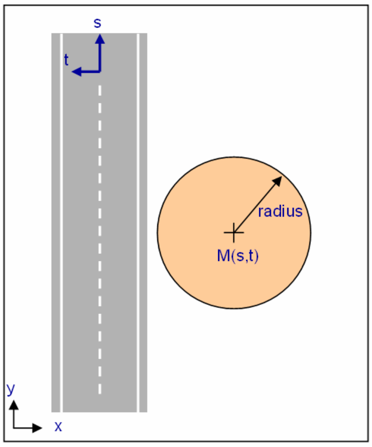
Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	***type***	***description***
`type`	string	type of the object
`name`	string	name of the object
`id`	string	unique ID within database
`s`	double	track position of object's origin [m] (s-position)
`t`	double	track position of object's origin [m] (t-position)
`zOffset`	double	z offset of object's origin from track level [m]
`validLength`	double	extent of object's validity along s-axis in [m] (0.0 for point object)
`orientation`	string	"+" = valid in positive track direction "-" = valid in negative track direction "none" = valid in both directions
`length`	double	length of the object [m] (local dx)
`width`	double	width of the object [m] (local dy)
`radius`	double	radius of the object [m]; alternative to width and length
`height`	double	height of the object [m] (local dz)
`hdg`	double	heading angle of the object relative to road direction [rad]
`pitch`	double	pitch angle of the object relative to road pitch [rad]
`roll`	double	roll angle of the object relative to road roll [rad]

Illustration for circular (or cylindric) objects:



5.3.8.1.1 Object Repeat Record

In order to avoid multiple definitions for multiple instances of the same object, repeat parameters may be defined for the original object.

Delimiters: `<repeat.../>`

Parent: `<object>`

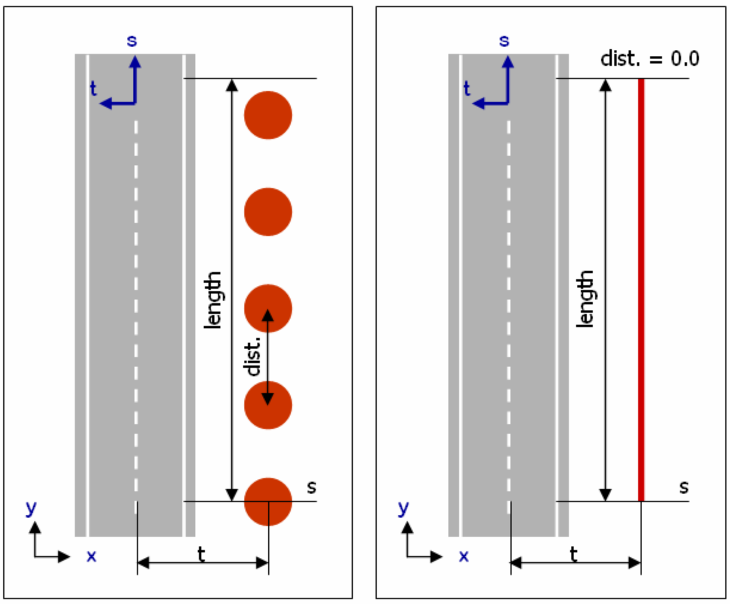
Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	start position (s-coordinate) [m], overrides the corresponding argument in the original `<object>` record
`length`	double	length of the repeat area [m]
`distance`	double	distance between two instances of the object [m] If this value is zero, then the object is considered to be a continuous feature like a railing, a wall etc.

Illustration:



5.3.8.1.2 Object Outline Record

The default parameters of the object record allow for objects with rectangular and circular footprint to be placed within the database. However, users may need to describe linear features as well as polygonal areas of non-rectangular shape along roads. For this purpose, the outline record may be used. It defines a sequence of corners including height information on the object's extent either in object co-ordinates or relative to the road's reference line (~~mixed definitions may also be used~~). For areas, the points should – preferably – be listed in CCW order.

The outline record must be followed by at least one corner record.

Delimiters: `<outline>...</outline>`

Parent: `<object>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.8.1.2.1 CornerRoad

Defines a corner point on the object's outline in road co-ordinates..

Delimiters: `<cornerRoad.../>`

Parent: `<outline>`

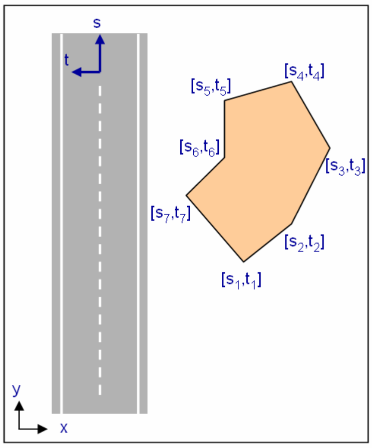
Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	type	description
s	double	s co-ordinate of the corner
t	double	t co-ordinate of the corner
dz	double	delta z of the corner relative to road's reference line
height	double	height of the object at this corner in [m]

Illustration:



5.3.8.1.2.2 CornerLocal

Defines a corner point on the object’s outline relative to the object's pivot point in local u/v co-ordinates. The pivot point and the orientation of the object are given by the `s/t/heading` arguments of the `<object>` entry.

Delimiters: `<cornerLocal.../>`

Parent: `<outline>`

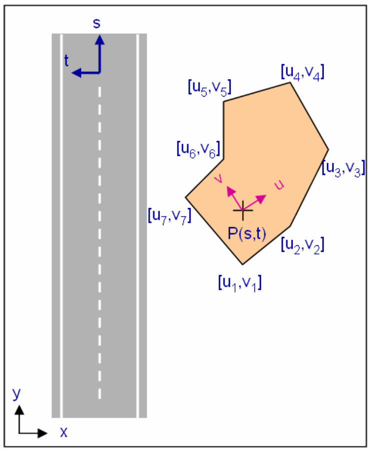
Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	type	description
`u`	double	local u co-ordinate of the corner
`v`	double	local v co-ordinate of the corner
`z`	double	local z co-ordinate of the corner
`height`	double	height of the object at this corner in [m]

Illustration:



5.3.8.1.3 Object Material Record

For objects like patches which are within the road surface (and, typically, coplanar to the surface) and which represent a local deviation from the standard road material, a description of the material properties is required. This description supercedes the one provided by the Road Material record and, again, is valid only within the outline of the parent road object.

Delimiters: `<material.../>`

Parent: `<object>`

Maximum Instances: `1`

Optional: `yes`

Arguments:

name	***type***	***description***
`surface`	string	surface material code [-], depending on application
`friction`	double	friction value [-]
`roughness`	double	roughness [-] (e.g. for sound and motion systems)

5.3.8.1.4 Lane Validity Record

Per default, objects are valid for all lanes pointing into the object's direction. This default validity may be replaced with explicit validity information for an object. The validity record is an optional child record of the object record. Multiple validity records may be defined per object.

Delimiters: `<validity.../>`

Parent: `<object>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`fromLane`	int	minimum ID of the lanes for which the object is valid
`toLane`	int	maximum ID of the lanes for which the object is valid

NOTE: For single-lane-validity of the object, provide identical values for `fromLane` and `toLane`.

5.3.8.2 Object Reference Record

Since the same object may be referred to from several roads, a corresponding record is being provided. This requires, however, that objects which are to be referred to be provided with a unique ID. The object reference record consists of a main record and an optional lane validity record.

Delimiters: `<objectReference.../>`

Parent: `<objects>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	track position [m] (s-position)
`t`	double	track position [m] (t-position)
`id`	string	unique ID of the referred object within the database
`zOffset`	double	z offset from track level [m]
`validLength`	double	extent of object's validity along s-axis in [m] (0.0 for point object)
`orientation`	string	"+" = valid in positive track direction "-" = valid in negative track direction "none" = valid in both directions

5.3.8.2.1 Lane Validity Record

see 5.3.8.1.4

5.3.8.3 Tunnel Record

The tunnel record is – like an object record – applied to the entire cross section of the road within the given range unless a lane validity record with further restrictions is provided as child record.

Delimiters: `<tunnel.../>`

Parent: `<objects>`

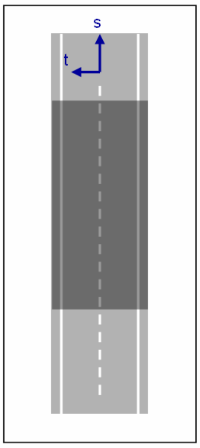
Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	track position (s co-ordinate)
`length`	double	length of the tunnel in [m] (s direction)
`name`	string	name of the tunnel
`id`	string	unique ID within database
`type`	string	type of the tunnel, see also 6.7
`lighting`	double	degree of artificial tunnel lighting [0.0..1.0]
`daylight`	double	degree of daylight intruding the tunnel [0.0..1.0]

Illustration:



5.3.8.3.1 Lane Validity Record

see 5.3.8.1.4.

Delimiters: `<validity.../>`

Parent: `<tunnel>`

5.3.8.4 Bridge Record

The bridge record is – like an object record – applied to the entire cross section of the road within the given range unless a lane validity record with further restrictions is provided as child record.

Delimiters: `<bridge.../>`

Parent: `<objects>`

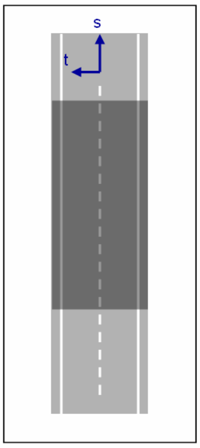
Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	track position (s co-ordinate)
`length`	double	length of the bridge in [m] (s direction)
`name`	string	name of the bridge
`id`	string	unique ID within database
`type`	string	type of the bridge, see also 6.8

Illustration:



5.3.8.4.1 Lane Validity Record

see 5.3.8.1.4.

Delimiters: `<validity.../>`

Parent: `<bridge>`

5.3.9 Road Signals Record

The signals record is the container for all signals along a road.

Delimiters: `<signals>...</signals>`

Parent: `<road>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.9.1 Signal Record

The signal record is used to provide information about signs and signals along a road. Signals are signs that can change their state dynamically (e.g. traffic lights). The signal record consists of a main record and an optional lane validity record.

Delimiters: `<signal.../>`

Parent: `<signals>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	***type***	***description***
`s`	double	track position [m] (s-position)
`t`	double	track position [m] (t-position)
`id`	string	unique ID of the signal within the database
`name`	string	name of the signal (e.g. bead name in graphics)
`dynamic`	string	`yes` or `no`
`orientation`	string	"+" = valid in positive track direction "-" = valid in negative track direction "none" = valid in both directions
`zOffset`	double	z offset from track level [m]
`country`	string	country code of the signal, see also 6.10
`type`	int	type number according to country code or -1, see also 6.11
`subtype`	int	subtype number according to country code or -1
`value`	double	value of the signal (e.g. speed for speed restricting signals, weight for weight limiting signals etc.)

5.3.9.1.1 Lane Validity Record

Per default, signals are valid for all lanes pointing into the signal's direction. This default validity may be replaced with explicit validity information for a signal. The validity record is an optional child record of the signal record. Multiple validity records may be defined per signal.

Delimiters: `<validity.../>`

Parent: `<signal>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`fromLane`	int	minimum ID of the lanes for which the object is valid
`toLane`	int	maximum ID of the lanes for which the object is valid

NOTE: For single-lane-validity of the signal, provide identical values for `fromLane` and `toLane`.

5.3.9.1.2 Signal Dependency Record

The signal dependency record provides signals with a means to control other signals. Signs can e.g. restrict other signs for various types of vehicles, warning lights can be turned on when a traffic light goes red etc. The signal dependency record is an optional child record of the signal record. A signal may have multiple dependency records.

Delimiters: `<dependency.../>`

Parent: `<signal>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`id`	string	ID of the controlled signal
`type`	string	type of the dependency, depending on the application

5.3.9.2 Signal Reference Record

Depending on the way roads (especially in junctions) are laid out for different applications, it may be necessary to refer to the same (i.e. the identical) sign from multiple roads. In order to prevent inconsistencies by multiply defining an entire signal entry, the user only needs to define the complete signal entry once and can refer to this complete record by means of the signal reference record.

This requires, however, that signals which are to be referred to be provided with a unique ID.
The signal reference record consists of a main record and an optional lane validity record.

Delimiters: `<signalReference.../>`

Parent: `<signals>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`s`	double	track position [m] (s-position)
`t`	double	track position [m] (t-position)
`id`	string	unique ID of the referenced signal within the database
`orientation`	string	"+" = valid in positive track direction "-" = valid in negative track direction "none" = valid in both directions

5.3.9.2.1 Lane Validity Record

see 5.3.9.1.1.

5.3.10 Surface

The surface record is the container for all surface descriptions which shall be applied along a road.

Delimiters: `<surface>...</surface>`

Parent: `<road>`

Maximum Instances: `1`

Optional: `yes`

Arguments: `none`

5.3.10.1 Curved Regular Grid Record

The interface to a Curved Regular Grid (CRG) surface description file is defined as arguments to the `<CRG>` tag and as an include operation between the opening and closing `<CRG>` tags.

Delimiters: `<CRG>...</CRG>`

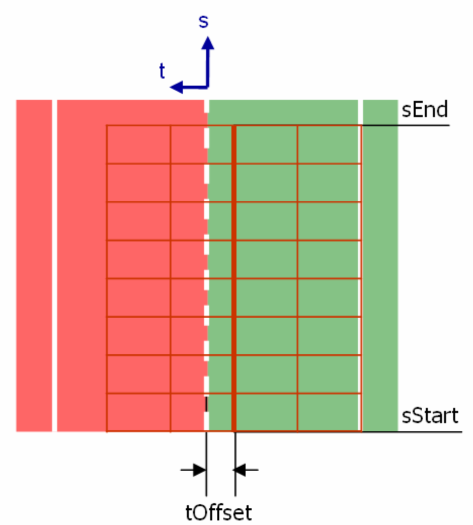
Parent: `<surface>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`file`	string	name of the file containing the CRG data
`sStart`	double	start of the application of CRG data [m] (s-position)
`sEnd`	double	end of the application of CRG [m] (s-position)
`orientation`	string	`same` or `opposite`
`mode`	string	application mode, `attached` or `genuine`
`sOffset`	double	s-offset between CRG center line and reference line of the road (optional, default = 0.0) [m]
`tOffset`	double	t-offset between CRG center line and reference line of the road (optional, default = 0.0) [m]
`zOffset`	double	z offset between CRG center line and reference line of the road (optional, default = 0.0) [m]
`zScale`	double	z scale factor for the surface description (optional, default = 1.0) [-]
`hOffset`	double	heading offset between CRG center line and reference line of the road (required for mode `genuine` only, optional, default = 0.0) [rad]



As the name indicates, CRG data is organized in a regular grid which is laid out along a reference line (comparable to an OpenDRIVE road's reference line). At each grid position, it contains the absolute elevation measured along a real road and some additional data which allows for the computation of the delta elevation relative to the reference line.

The key to combining OpenDRIVE and CRG data is to define a correlation between the two reference lines and a rule for using the elevation data of both descriptions.

CRG data may be offset from the OpenDRIVE road's reference line (see `tOffset`) and it may be oriented in the same or opposite direction as the layout direction of the road (see `orientation`).

The CRG data may be applied to a given OpenDRIVE road in two modes:

Mode `attached`:

The reference line of the CRG data set is replaced with the OpenDRIVE road's reference line, taking into account the `tOffset` and the `sOffset` parameters

The CRG local elevation values (calculated by evaluating the CRG grid and applying `zOffset` and `zScale`) will be added to the surface elevation data of the OpenDRIVE road (as derived from the combination of elevation, super-elevation and crossfall).

With this mode, the surface information relative to the original CRG data's reference line is transferred from an arbitrary CRG road to an OpenDRIVE road without having to make sure that the overall geometries of the road match. The original position, heading, curvature, elevation and superelevation of the CRG road are disregarded. The CRG grid is evaluated along the OpenDRIVE reference line instead of the CRG reference line.

$$\begin{pmatrix} u \\ v \end{pmatrix}_{CRG} = \begin{pmatrix} s - s_{Offset} \\ t - t_{Offset} \end{pmatrix}_{OpenDrive}$$

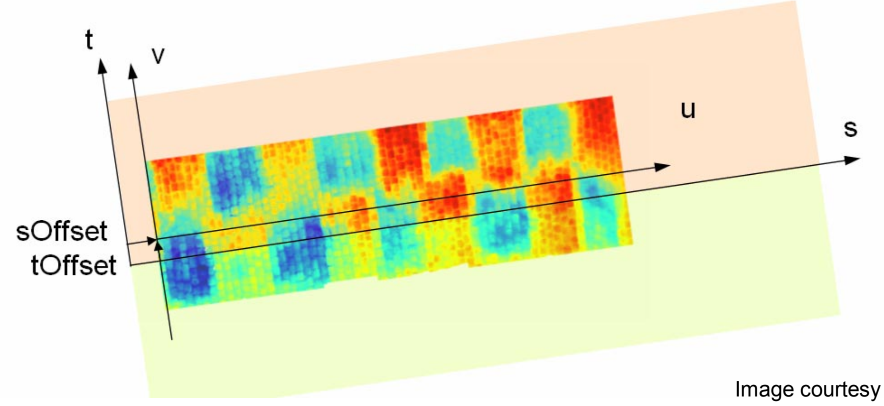


Image courtesy of Daimler AG

Mode genuine:

The start point of the CRG data set's reference line is positioned relative to the point on the OpenDRIVE road's reference line at the position defined by sStart, sOffset and tOffset.

By providing offset values for the longitudinal (sOffset) and lateral (tOffset) displacement, the heading (hOffset) and the elevation (zOffset), the correlation between the two description's reference lines is clear.

In genuine mode, the CRG data will completely replace the OpenDRIVE elevation data, i.e. the absolute elevation of a given point of the road surface is directly computed from the CRG data (think of it as combining CRG and OpenDRIVE data with the OpenDRIVE elevation, super-elevation and crossfall all being zero).

When using this method, it must of course be made sure that the geometry of the CRG data matches - within certain tolerance - the geometry of the underlying OpenDRIVE road.

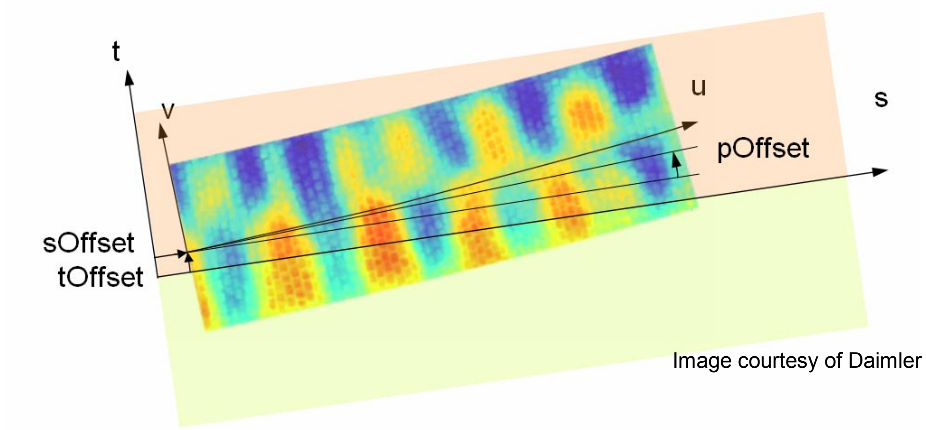


Image courtesy of Daimler AG

Since CRG data may only cover parts of a road's surface, it must be made sure that outside the valid CRG area, the elevation information derived from OpenDRIVE data can still be used.

Example:

```
<surface>
    <CRG file="fancyData.crg" sStart="0.0" sEnd="100.0"
         orientation=”same” mode=”relative”
         tOffset=”0.0”>
    </CRG>
</surface>
```

5.4 Controller Record

A controller provides consistent states for a group of signals. This may be a set of signals within a junction or a set of dynamic speed restrictions on a motorway. The entire record consists of a header followed by a number of dependency records.

Delimiters: `<controller.../>`

Parent: `<OpenDRIVE>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`id`	string	unique ID within database
`name`	string	name of the controller
`sequence`	int	sequence number (priority) of this controller with respect to other controllers of same logical level

5.4.1 Control Entry Record

The control entry record provides information about a single signal controlled by the corresponding controller. This record is a child record of the controller record.

Delimiters: `<control.../>`

Parent: `<controller>`

Maximum Instances: `unlimited`
l
Optional: `no`

Arguments:

name	***type***	***description***
`signalId`	string	ID of the controlled signal
`type`	string	type of control

5.5 Junction Record

When a road can be linked to more than one successor (or predecessor, depending on the orientation), a junction record is required. It contains the information about all possible connections between roads meeting at a physical junction.

For junctions, two types of roads must be distinguished:

- standard roads
- paths

`Standard roads` are in-coming and out-going roads of the junction. Usually, they don't require any special treatment except for the fact that they end or begin at a junction.

`Paths` are roads within a junction. For each connection leading from one standard road to another, a path is defined. Paths may link only single lanes or a set of lanes at once. Paths must only contain lanes of one side (left or right). However, within the database, paths having only left lanes and paths having only right lanes may both be contained. The geometric and logic description of paths complies with the rules defined for standard roads (i.e. they contain lanes, elevation entries etc.).

Due to the fact that paths cannot have lanes of both directions, junctions only provide information linking incoming roads to paths. This is the only ambiguous part of a connection. The link between a path and the corresponding outgoing road is clear from the ROAD LINK record of the path (see 5.3.2).

The junction record is split into a header record and a series of link records.

Delimiters: `<junction>...</junction>`

Parent: `<OpenDRIVE>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`name`	string	name of the junction
`id`	string	unique ID within database

5.5.1 Connection Record

The connection record provides information about a single connection within a junction. It is a child of the junction header record.

Delimiters: `<connection>...</connection>`

Parent: `<junction>`

Maximum Instances: `unlimited`

Optional: `no`

Arguments:

name	type	description
`id`	string	unique ID within the junction
`incomingRoad`	string	ID of the incoming road
`connectingRoad`	string	ID of the connecting path
`contactPoint`	string	contact point on the connecting road, may be *`start`* or *`end`*

5.5.1.1 Junction Lane Link Record

The junction lane link record provides information about the lanes which are linked between incoming road and connecting road. This record may be omitted if all incoming lanes are linked to lanes with identical IDs on the connecting road. However, it is strongly recommended to provide this record.

Delimiters: `<laneLink.../>`

Parent: `<connection>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	type	description
`from`	int	ID of the incoming lane
`to`	int	ID of the connecting lane

5.5.2 Junction Priority Record

The junction priority record provides information about the priority of a connecting road over another connecting road. It is only required if priorities cannot be derived from signs or signals in a junction or on tracks leading to a junction.

Delimiters: `<priority.../>`

Parent: `<junction>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`high`	string	ID of the prioritized connecting road
`low`	string	ID of the connecting road with lower priority

5.5.3 Junction Controller Record

Junction controller records list the controllers which are used for the management of a junction.

Delimiters: `<controller.../>`

Parent: `<junction>`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	***type***	***description***
`id`	string	ID of the controller
`type`	string	type of control for this junction
`sequence`	int	sequence number (priority) of this controller with respect to other controllers in the same junction

5.6 Ancillary Data

Ancillary data for a bead may be defined by generating an entry that directly follows the corresponding bead. It is designed to contain user-defined data that is required for specific applications or is not yet covered by the OpenDRIVE® standard.

Delimiters: `<userData.../>`

Parent: `any`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	*type*	*description*
code	string	code for the user data (application specific)
value	string	user data as string (e.g. hexdump)

5.7 Include Tag

The inclusion of another file can be triggered with an include tag at any location.

Delimiters: `<include.../>`

Parent: `any`

Maximum Instances: `unlimited`

Optional: `yes`

Arguments:

name	*type*	*description*
file	string	location of the file which is to be included

5.8 Alternative Layouts (Sets)

Sets indicate that more than one setup of the properties enclosed by the set's opening and closing tags are available and that the application may choose one of these setups (the activation of none or more than one setup at a given time is not supported). Sets may be used e.g. for different road mark setups, different signaling etc.

Delimiters: `<set>...</set>`

Parent: any

Maximum Instances: unlimited

Optional: yes

Arguments:

name	type	description
id	string	unique ID within the database

5.8.1 Layout Instance

Each set may contain one or more alternative setups of the enclosed property. In order to identify a specific setup, it must be enclosed with instance tags.

Delimiters: `<instance>...</instance>`

Parent: `<set>`

Maximum Instances: unlimited

Optional: no

Arguments:

name	type	description
id	string	unique ID within the database

Example:

```
<road ...>
    <set id="roadWorks">
        <instance id="regular">
            <lanes>
            :
            </lanes>
        </instance>
        <instance id="construction">
            <lanes>
            :
            </lanes>
        </instance>
    </set>
</road>
```

6 Constants

6.1 Road Type Information

The known keywords for the road type information are:

```
unknown
rural
motorway
town
lowSpeed
pedestrian
```

NOTE: In Germany, `lowSpeed` is equivalent to a 30km/h zone

6.2 Road Mark Type Information

The known keywords for the road mark type information are:

`none`
`solid`
`broken`
`solid solid` (for double solid line)
`solid broken` (from inside to outside, exception: center lane - from left to right)
`broken solid` (from inside to outside, exception: center lane - from left to right)

6.3 Road Mark Weight Information

The known keywords for the road mark weight information are:

```
standard
bold
```

6.4 Road Mark Color Information

The known keywords for the road mark color information are:

```
standard
yellow
```

6.5 Lane Type Information

The known keywords for the lane type information are:

```
none
driving
stop
shoulder
biking
sidewalk
border
restricted
parking
mwyEntry
mwyExit
special1
special2
special3
tram
```

6.6 Object Types

The known keywords for the object type information are:

```
obstacle
wind
patch
```

6.7 Tunnel Types

The known keywords for the tunnel type information are:

```
standard
underpass (i.e. sides are open for daylight)
```

6.8 Bridge Types

The known keywords for the bridge type information are:

```
concrete
steel
brick
```

6.9 Access Restriction Types

The known keywords for the restriction information are:

```
simulator
autonomous traffic
pedestrian
none
```

6.10 Signal Country Codes

The known keywords for the signal country codes are:

```
OpenDRIVE
France
Germany
USA
```

6.11 Signal Types

For the country codes “Germany” and “OpenDRIVE”, the following signal types shall be defined in addition to the numbers given by the corresponding rule books:

traffic light	1,000,001
pedestrian traffic light	1,000,002

Others may be added in future version of this specification.

7 Further Explanations and Examples

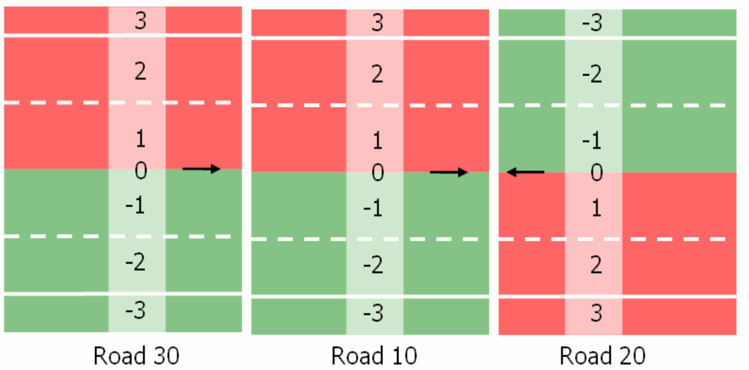
Important Note: All of the following examples do not provide complete OpenDRIVE data sets. Instead, they are reduced to the contents necessary for explaining the respective issues.

7.1 Road and Lane Linkage

7.1.1 Direct Link of Roads and Lanes

Two roads which connect directly with each other, can be handled without any junction entries. Instead, the link information will be provided within the road tags.

Let's assume the following situation:



Road 10 is directly linked to road 20 and road 30. The cross sections of all roads are identical, however the directions do not match between road 10 and 20.

The following database fragment provides the link information on two levels. First, the direct link between the roads is indicated in the <link>-tag which follows the <road>-tag.

```
<road name="" length="1.0000000000000000e+00" id="10" junction="-1">
   <link>
       <predecessor elementType="road" elementId="30" contactPoint="end" />
       <successor elementType="road" elementId="20" contactPoint="end" />
   </link>
```

Road 30 is the predecessor of road 10 and road 10 connects to the end of road 30
Road 20 is the successor of road 10 and road 10 connects to the end of road 20

Now that the basic road connections are known, one can provide additional information about the lane links. If a road has multiple lane sections then the first and last lane section are of interest for the transition from one road's lane to another road's lane. Here, let's assume, our roads have only one lane section each.

So, the entries for road 10 look like the following:

```
<lanes>
    <laneSection s="0.0000000000000000e+00">
        <left>
            <lane id="3" type="border" level= "0">
                <link>
                    <predecessor id="3"/>
                    <successor id="-3"/>
                </link>
            </lane>
            <lane id="2" type="driving" level= "0">
                <link>
                    <predecessor id="2"/>
                    <successor id="-2"/>
                </link>
            </lane>
            <lane id="1" type="driving" level= "0">
                <link>
                    <predecessor id="1"/>
                    <successor id="-1"/>
                </link>
            </lane>
        </left>
        <center>
            <lane id="0" type="border" level= "0">
                <link>
                </link>
            </lane>
        </center>
        <right>
           <lane id="-1" type="driving" level= "0">
                <link>
                    <predecessor id="-1"/>
                    <successor id="1"/>
                </link>
            </lane>
            <lane id="-2" type="driving" level= "0">
                <link>
                    <predecessor id="-2"/>
                    <successor id="2"/>
                </link>
            </lane>
            <lane id="-3" type="border" level= "0">
                <link>
                    <predecessor id="-3"/>
                    <successor id="3"/>
                </link>
            </lane>
        </right>
    </laneSection>
</lanes>
```

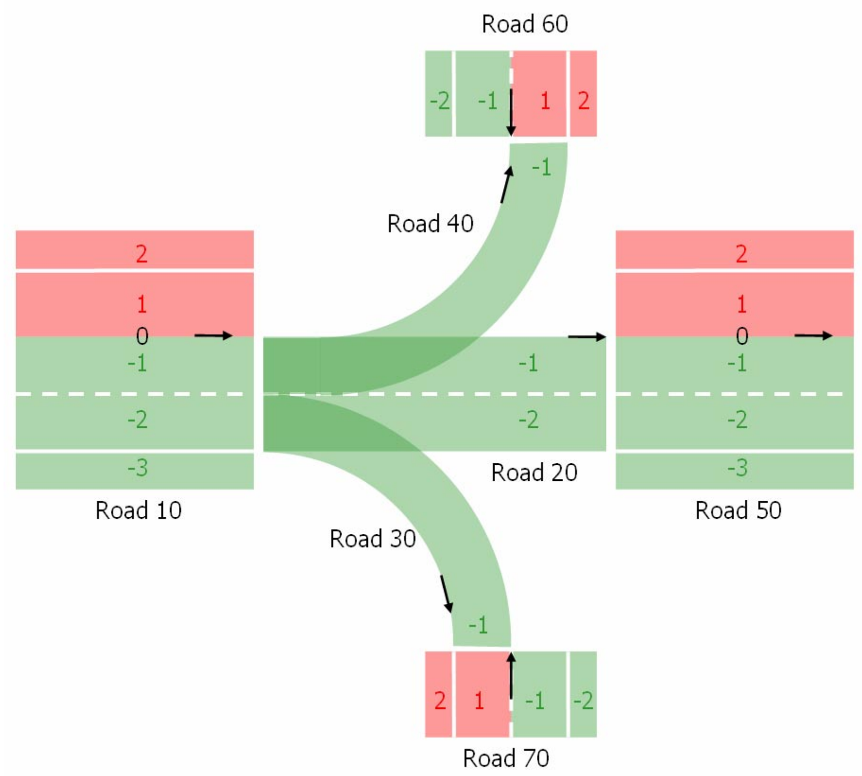
The predecessor of lane 3 on road 10 is lane 3 on road 30.

The successor of lane 3 on road 10 is lane -3 on road 20.

The same scheme applies to all other lanes of the cross section.

7.1.2 Links using Junctions

If the successor or predecessor of a road becomes ambiguous, then a junction is required. Let's take a look at the following example:



Start on the left side: Road 10 links to a junction. The depicted connections (which do not represent the entire connection matrix) allow for the following ways to navigate the junction:

Road 10, lane -1	via	road 20, lane -1	to	road 50, lane -1
	or via	road 40, lane -1	to	road 60, lane 1
Road 10, lane -2	via	road 20, lane -2	to	road 50, lane -2
	or via	road 30, lane -1	to	road 70, lane 1

So, we have two incoming lanes and four possible destinations.

The links between the connecting roads 20, 30, and 40 and their respective successors and predecessors (note that all connecting roads in the example only have right lanes) are clear and can be handled like the standard linking described in the previous chapter.

Thus, for road 30, the link information can be provided in the following way:

```
<road name="" length="1.0000000000000000e+00" id="30" junction="-1">
   <link>
       <predecessor elementType="road" elementId="10" contactPoint="end" />
       <successor elementType="road" elementId="70" contactPoint="end" />
   </link>
   <lanes>
       <laneSection s="0.0000000000000000e+00">
           <left>
           </left>
           <center>
               <lane id="0" type="border" level= "0">
                   <link>
                   </link>
               </lane>
           </center>
           <right>
              <lane id="-1" type="driving" level= "0">
                   <link>
                       <predecessor id="-2"/>
                       <successor id="1"/>
                  </link>
               </lane>
          </right>
       </laneSection>
   </lanes>
```

For road 10, the successor is now a junction instead of a road. This is indicated in the link information following the <road> tag:

```
<road name="" length="2.0000000000001661e+00" id="10" junction="-1">
   <link>
       <predecessor elementType="road" elementId="99" contactPoint="end" />
       <successor elementType="junction" elementId="25" />
   </link>
```

```
<lanes>
    <laneSection s="0.0000000000000000e+00">
        <left>
            <lane id="2" type="driving" level= "0">
                <link>
                    <predecessor id="2"/>
                </link>
            </lane>
            <lane id="1" type="driving" level= "0">
                <link>
                    <predecessor id="1"/>
                </link>
            </lane>
        </left>
        <center>
            <lane id="0" type="border" level= "0">
                <link>
                </link>
            </lane>
        </center>
        <right>
           <lane id="-1" type="driving" level= "0">
                <link>
                    <predecessor id="-1"/>
                </link>
            </lane>
            <lane id="-2" type="driving" level= "0">
                <link>
                    <predecessor id="-2"/>
                </link>
            </lane>
            <lane id="-3" type="border" level= "0">
                <link>
                    <predecessor id="-3"/>
                </link>
            </lane>
        </right>
    </laneSection>
</lanes>
```

The sample above assumes road 10 has a predecessor no. 99 (not depicted above, so let's assume it is of the same cross-section as road 10) and a successor which is junction no. 25. The junction itself is not shown since it is just a logical container for further connection information (see below)

The presence of the junction is also considered in the lane links of road 10. Only the links to the predecessor are clear. The links to the successors are omitted. This can also be seen in the code sample on the left.

Now for the corresponding junction matrix which closes the gap between the incoming road 10 and all possible destinations (roads 50, 60 and 70). Again, the information is provided on two levels, first the road level, then the lane level. Look at the following code fragment:

```
<junction name="" id="25">
    <connection id="0" incomingRoad="10" connectingRoad="20" contactPoint="start">
        <laneLink from="-1" to="-1"/>
        <laneLink from="-2" to="-2"/>
    </connection>
    <connection id="1" incomingRoad="10" connectingRoad="30" contactPoint="start">
        <laneLink from="-2" to="-1"/>
    </connection>
    <connection id="2" incomingRoad="10" connectingRoad="40" contactPoint="start">
        <laneLink from="-1" to="-1"/>
    </connection>
</junction>
```

As can be seen, the connection from road 10 to road 20 may be entered on two different lanes, whereas the other connections may be entered on one lane only.

As state above, the links between the connecting roads and their successors is, again, clear and can therefore be provided as standard <link> information below the <road> tag and the respective <lane> tags.

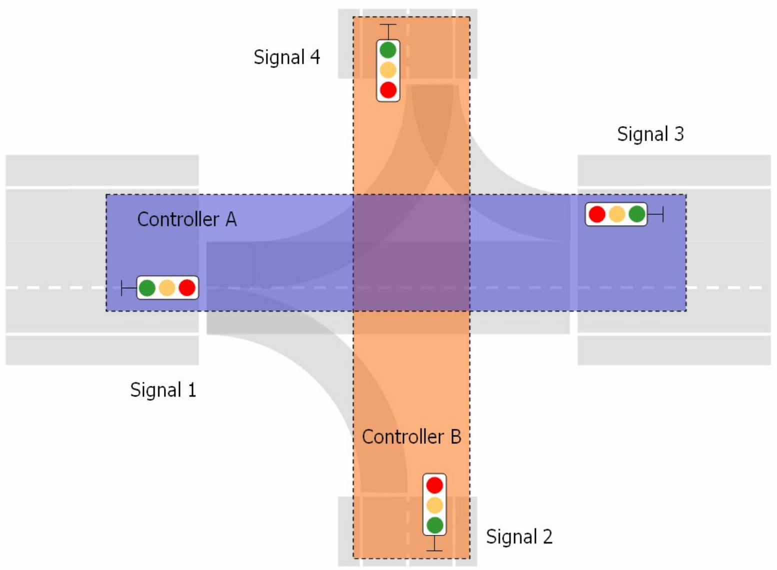
7.2 Junction Control

The designer of a road network does not only provide lane and road linkage information but also details of the signalization. It is assumed to be his (or her) task to group signals in a fashion that they can be controlled consistently from the scenario simulation which reads and interprets the OpenDRIVE file.

Signal control is split into two levels which allow for the grouping of

- various signals under a single controller and for
- various controllers within one junction

The following example shall provide a control scheme of a "standard" crossing. It represents one possible solution only which is, of course, non-binding.



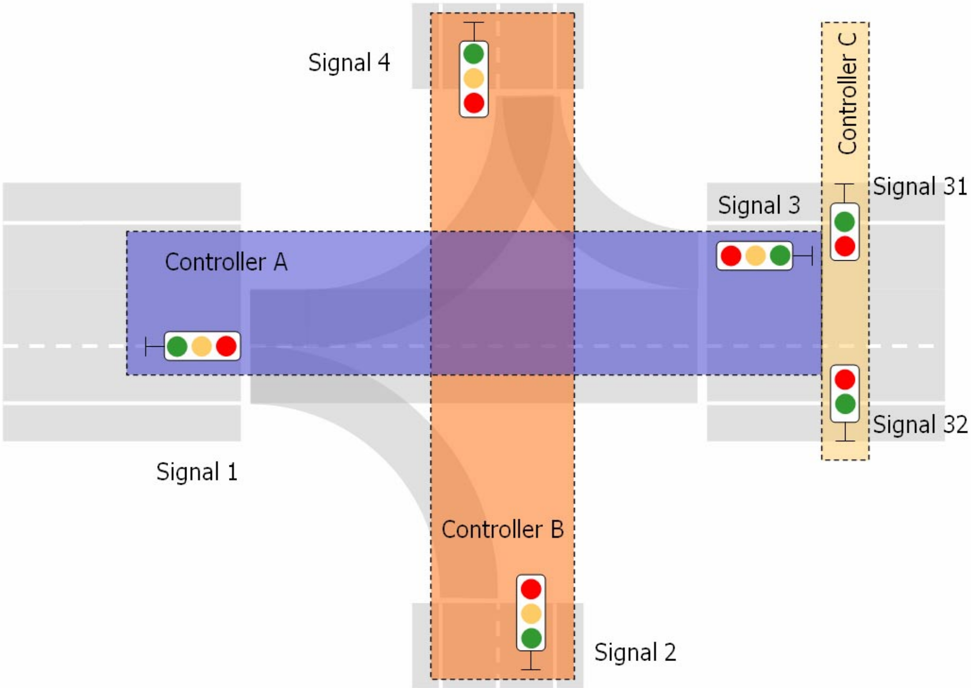
It is recommended to assign synchronous signals of same type to identical controllers instead of having one controller for each signal. So, in the example above, both through-directions will be controlled by individual controllers.

In the OpenDRIVE file, the controllers would be defined in the following way:

```
<OpenDRIVE>
    <controller id="Controller A" type="">
        <control signalId="Signal 1" type=""/>
        <control signalId="Signal 3" type=""/>
    </controller>
    <controller id="Controller B" type="">
        <control signalId="Signal 2" type=""/>
        <control signalId="Signal 4" type=""/>
    </controller>
</OpenDRIVE>
```

Please note that the type of control is not yet actually used for controller or control entry.

If, for example, pedestrian lights should be added to the above junction, it is recommended to group them under additional controllers. The following image illustrates this situation:



In this example, the signals 31 and 32 are controlled by an additional controller "Controller C". If corresponding pedestrian signals existed at the location of "Signal 1", they could also be controlled by "Controller C".

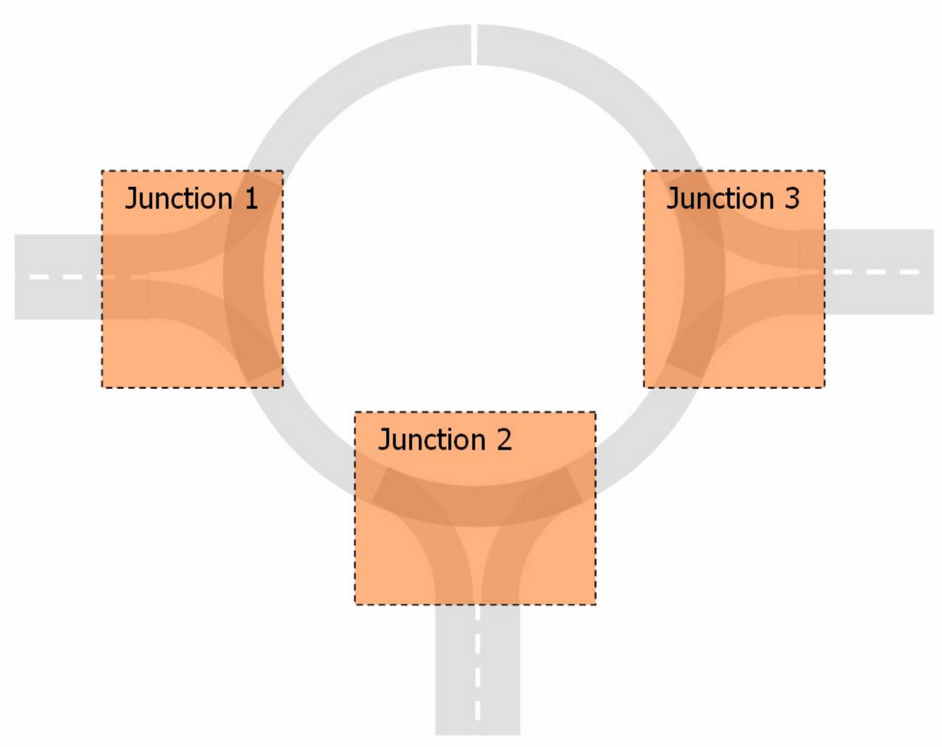
Finally, all controllers responsible for a junction may be grouped within the <junction> tag. Again, let's assume, we are talking about junction no. 25 in the example above.

The junction entry would look similar to the following one (not all connections and controls are given):

```
<OpenDRIVE>
    <junction name="" id="25">
        <connection id="0" incomingRoad="10" connectingRoad="20" contactPoint="start">
            <laneLink from="-1" to="-1"/>
            <laneLink from="-2" to="-2"/>
        </connection>
        <connection id="1" incomingRoad="10" connectingRoad="30" contactPoint="start">
            <laneLink from="-2" to="-1"/>
        </connection>
        <connection id="2" incomingRoad="10" connectingRoad="40" contactPoint="start">
            <laneLink from="-1" to="-1"/>
        </connection>
        <controller id="Controller A" type=""/>
        <controller id="Controller B" type=""/>
        <controller id="Controller C" type=""/>
    </junction>
</OpenDRIVE>
```

7.3 Roundabouts

Roundabouts are to be considered as collections of junctions with each junction being constructed according to the rules explained above. Also for the signaling, the rules laid out before apply.



8 Change Log

Version 1.3C vs. Version 1.2A

- modified / extended:
  - using the term “reference line” instead of “chord line” and “center line”
  - removed “inertial” object outlines
  - “relative” object outlines are called “local” outlines
  - track co-ordinate system is s/t/h instead of s/t/z
  - updated object repeat record with continuous road-side objects
  - debugged some typos

- introduced:
  - local co-ordinate system
  - road mark types (detailed definition)
  - vendor information in header
  - lane offset
  - object material record
  - examples / further explanations

Version 1.2A vs. Version 1.1D

- modified / extended:
  - material record (“surface” argument)
  - object record is defined as range, not as single-line definition
  - unique IDs of most elements are allowed to be strings instead of unsigned integers
  - using string parameters for type definitions
  - neighbor entry provides left and right neighbors
  - roadMark entry extended with laneChange information

- introduced:
  - road surface description
  - CRG data
  - roadMark sub-types
  - repeat entry for objects
  - object reference record
  - property sets

Version 1.1D vs. Version 1.1C

- corrected:
  - chapter 2.3.2, description of inertial co-ordinate system
  - minor bugfixes

Version 1.1 vs. Version 1.0

- introduced:
  - chapter 3.4 explaining superelevation and crossfall
  - lateralProfile (replacing crossfallProfile)
  - superelevation record (replacing old “crossfall” record)
  - crossfall record per side of the road
  - definition of object corners relative to a given pivot point
  - signal references
  - include tag

- extended:
  - country code for signal type and subtype IDs
  - lane record (application of crossfall may be restricted)

- illustration of co-ordinate systems in chapter 2

Version 1.0 vs. Version 0.7

- introduced:
    - geometry primitive poly3
    - lane speed information
    - lane access restrictions
    - lane height information
    - object radius (alternatively to width and length)
    - object outline information
    - tunnel record
    - bridge record
- extended:
    - road mark entry may be provided with exact width information
- corrected:
    - some type information of the existing records was incorrect