

**Topology Guideline**

|  |  |
| --- | --- |
| Published by | Talking Traffic |
| Content | subWG NL profile |
| Editorial | MAPtm |
| Date | 16-11-2017 |
| Status | Final |
| Version number | 2.0 |

Contents

[1 Introduction 4](#_Toc497129025)

[1.1 Purpose of this document 4](#_Toc497129026)

[1.2 MapData (MAP) 4](#_Toc497129027)

[1.3 ITF Intersection Topology Format (ITF) 4](#_Toc497129028)

[1.4 Reading guide 4](#_Toc497129029)

[1.4.1 Background documents 4](#_Toc497129030)

[1.4.2 Relevant standards 5](#_Toc497129031)

[2 Identifiers 6](#_Toc497129032)

[2.1 StationID and TlcIdentifier 6](#_Toc497129033)

[3 Reference example 7](#_Toc497129034)

[3.1 Properties of lanes 9](#_Toc497129035)

[3.1.1 Tracked vehicles 10](#_Toc497129036)

[3.2 Nodes 10](#_Toc497129037)

[3.2.1 Absolute (ITF) versus relative (MAP) coordinates 10](#_Toc497129038)

[3.2.2 NodeList 10](#_Toc497129039)

[3.2.3 Connections 12](#_Toc497129040)

[3.2.4 Connection trajectory 14](#_Toc497129041)

[3.3 Restrictions 14](#_Toc497129042)

[4 Specific intersection or lane configurations 15](#_Toc497129043)

[4.1 Bicycle box (bike box) 15](#_Toc497129044)

[4.2 Bicycle lanes 16](#_Toc497129045)

[4.2.1 Bicycle lane with continuous lane marking 17](#_Toc497129046)

[4.2.2 Bicycle lane with broken lane marking 18](#_Toc497129047)

[4.2.3 Bidirectional separated bicycle lanes 19](#_Toc497129048)

[4.2.4 Cyclist movement in two stages 20](#_Toc497129049)

[4.2.5 Bicycle street 20](#_Toc497129050)

[4.3 Intersection lanes 20](#_Toc497129051)

[4.3.1 Fan out 20](#_Toc497129052)

[4.3.2 Road Geometry 21](#_Toc497129053)

[4.3.3 Merging lanes 21](#_Toc497129054)

[4.3.4 Joining approach roads 22](#_Toc497129055)

[4.4 Public transport lane 23](#_Toc497129056)

[4.5 Dynamic lane configuration 24](#_Toc497129057)

[4.6 Multiple intersections for 1 TLC (and ITF or MAP file) 26](#_Toc497129058)

[4.7 Remote intersection 29](#_Toc497129059)

[4.7.1 Use of remoteIntersection, no egress lanes 29](#_Toc497129060)

[4.8 Double stop lines 32](#_Toc497129061)

[4.9 Connections 33](#_Toc497129062)

[4.9.1 Connection 1:2 33](#_Toc497129063)

[4.9.2 Connection 2:2 35](#_Toc497129064)

[4.9.3 Connection 2:3 36](#_Toc497129065)

[4.10 Crosswalk 37](#_Toc497129066)

[4.10.1 Safe island 37](#_Toc497129067)

[4.10.2 Multiple signal groups 38](#_Toc497129068)

[5 Control-data: 41](#_Toc497129069)

[5.1 Sensors 41](#_Toc497129070)

[5.1.1 Sensor allocation 42](#_Toc497129071)

[5.1.2 Sensor relation 43](#_Toc497129072)

[5.2 Signal group relations 43](#_Toc497129073)

[Annex A: Bit string example 44](#_Toc497129074)

[Annex C: Conversion code absolute – relative positions 45](#_Toc497129075)

[Annex D: Members subWG NL profile 48](#_Toc497129076)

# Introduction

## Purpose of this document

This document provides recommended practices for the use and application of the data structures of MAP and ITF to convey intersection topology information. It offers examples of intersection and lane configurations and how to describe these using the available data elements.

## MapData (MAP)

The MapData (MAP) message (SAE J2735, TS19091) is used to convey many types of geographic road information. At the current time its primary use is to convey one or more intersection lane geometry maps within a single message. The map message content includes such items as complex intersection descriptions, road segment descriptions, high speed curve outlines (used in curve safety messages), and segments of roadway (used in some safety applications). A given single MapData message may convey descriptions of one or more geographic areas or intersections. The contents of this message involves defining the details of indexing systems that are in turn used by other messages to relate additional information (for example, the signal phase and timing via the Signal Phase and Timing (SPAT) message) to events at specific geographic locations on the roadway. The SPAT message is used to convey the current status of one or more signalized intersections. Along with the MapData message (which describes a full geometric layout of an intersection) the receiver of this message can determine the state of the signal phasing and when the next expected phase will occur, subject to its geographical position on the intersection.

## ITF Intersection Topology Format (ITF)

The Intersection Topology Format is largely based on the internationally standardised MAP message (SAE J2735, ISO TS 19091) and adds elements which are derived from common approaches in the Netherlands such as SPOC and V-Log. This document offers a guideline to the Intersection Topology Format as requested by the Ministry of Infrastructure and the Environment, in support of the Program Beter Benutten ITS and the Call for Innovation Partnerships Talking Traffic.

To convert from ITF to MAP, the following transformations must be made to comply to the international MAP standards:

* A layerID must be added.
* Node coordinates must be converted from absolute positions to off-sets (see Annex C).
* The SpeedLimitType ‘nominalSpeed’ must be removed.
* The NodeAttributeXY ‘yield’ must be removed.
* The regional extensions in REGION.Reg-LaneDataAttribute must be removed.
* The regional extensions in REGION.Reg-GenericLane must be removed.
* EmissionType in the regional extension REGION.Reg-RestrictionUserType must be removed.
* The entire structure for ControlData must be removed.

## Reading guide

### Background documents

This document does not stand on its own. Beside the international standards mentioned hereafter, the reader should take not of the following documents which this guidelines builds upon:

* 171102 MAP profile v1.8 [subWG NL profiel].docx
* 171102 ITF profile v1.8 [subWG NL profiel].docx

What is stated and explained in these documents is not repeated in this guideline. The reader is expected to be aware of these documents and their content.

### Relevant standards

The following standards have been used to prepare aforementioned profiles and this guideline.

* SAE J2735, Dedicated Short Range Communications (DSRC) Message Set Dictionary, March 2016
* ISO TS19091, Intelligent transport systems — Cooperative ITS — Using V2I and I2V communications for applications related to signalized intersections, 2016(E)
* ETSI 103 301, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services, V1.1.1 (2016-11)
* ETSI TS102 894-2, Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary, V1.2.1 (2014-09)

# Identifiers

## StationID and TlcIdentifier

There are multiple identifiers in use to recognize a roadside ITS station and intersection. For uniformity it is important that there is a clear relation between the different identifiers.

Consider:

* StationID ::= INTEGER (0..4294967295)
* RoadRegulatorID ::= INTEGER (0..65535)
* IntersectionID ::= INTEGER (0..65535)
* TlcIdentifier ::= string (IA5/ASCII) - 8 chars

Example: one controller with two intersections (91 & 92) requires the following identifiers:

* RoadRegulatorID: 31396
* IntersectionID: 90 (for this purpose rounded to ten)
* TlcIdentifier: 7AA4005A (the combination of the hexadecimal representation of the

RoadRegulatorID [7AA4] and IntersectionID [005A]

* StationID: 2057568346 (31396\*65536 + 90)

Consequently, the hexadecimal representation of the StationID is equal to the TlcIdentifier.

This approach does not support the case when 1 TLC serves 2 ITS applications. In that case, TLEX expects two SPAT-streams each with their own unique TlcIdentifier. It was accepted by the subWG NL profile that this is an exceptional circumstance, therefore left out of consideration.

# Reference example

This chapter describes the use of all data frames and data elements of the MapData (MAP) data structure on the basis of a simply example. The intersection layout as shown in Figure 1 is used as a reference example to detail the configurations.

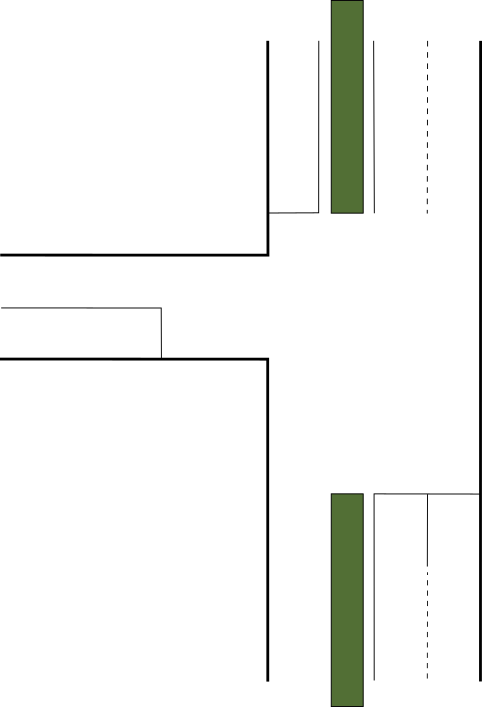


Figure 1 Intersection layout

MapData can describe the geometry of one or more intersections. In this example, there is only one intersection. Each intersection contains a reference point: the centre of an intersection (conflict area), see Figure 2.

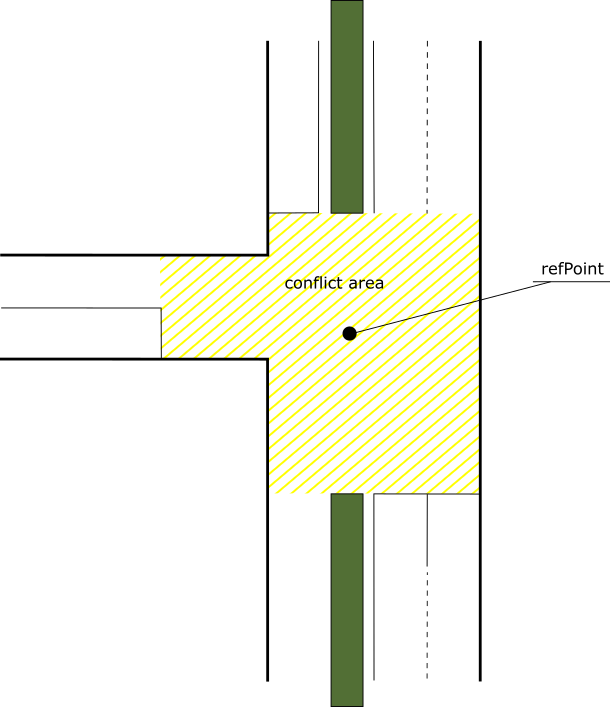


Figure 2 Reference point of the intersection

The general configuration of the intersection are detailed in Table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| name [DescriptiveName] |  | Intersection 456 Foo-Bar |  |
| id [IntersectionReferenceID] | region [RoadRegulatorID] | 101 |  |
| id [IntersectionID] | 456 |  |
| revision [MsgCount] |  | 1 |  |
| refPoint [Position3D] | lat [Latitude] | 520679333 | Integer Multiply by 10000000 to obtain integer  Divide by 10000000 to obtain coordinate |
| long [Longitude] | 50787649 | Integer Multiply by 10000000 to obtain integer  Divide by 10000000 to obtain coordinate |
| altitude [Altitude] | - |  |
| laneWidth  [LaneWidth] |  | 300 |  |
| speedLimits [SpeedLimitList]  regulatorySpeedLimit  [RegulatorySpeedLimit] | type [SpeedLimitType] | vehicleMaxSpeed |  |
| speed [Velocity] | 694 | units of 0.02 m/s 50 km/h = 13.89 m/s 13.89 / 0.02 = 694 |
| laneSet [LaneList]  genericLane  [GenericLane] |  |  | See paragraph 0 |

Table 1 General intersection configuration

## Properties of lanes

The laneSet [LaneList] data frame contains the properties of all the lanes of an intersection. Figure 3 shows all vehicle lanes, lane ID numbers, and allowed movements of the intersection.

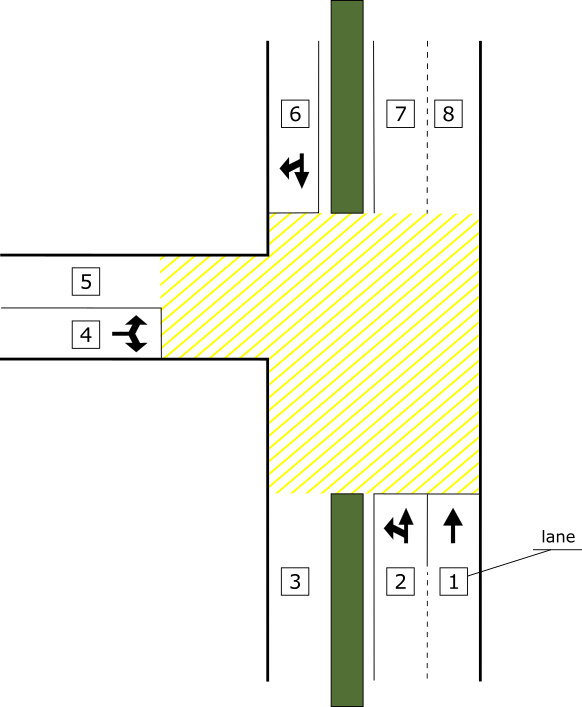


Figure 3 Intersection vehicle lanes

Each lane is part of an approach. There are two kinds of approaches, an ingress approach and an egress approach. The intersection approaches are shown in Figure 4.

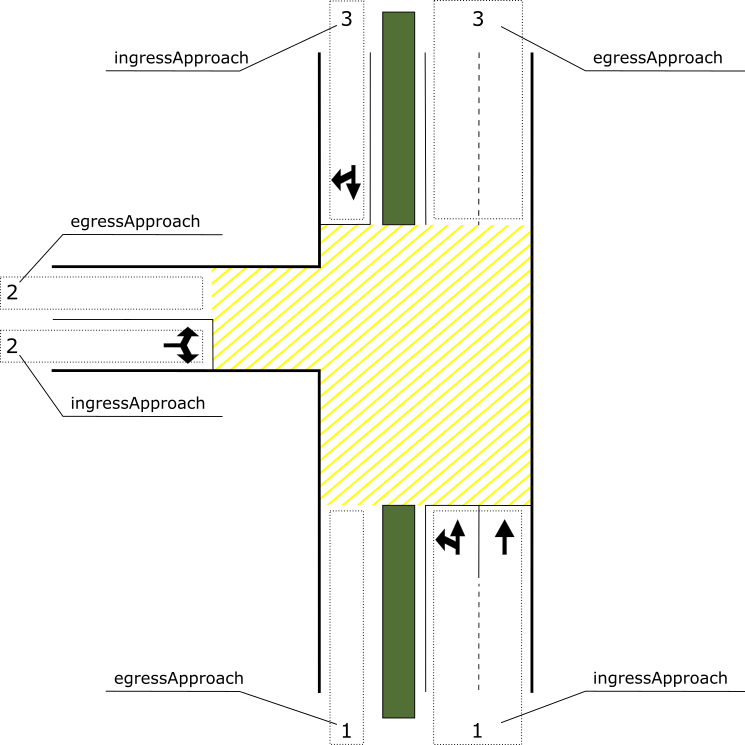


Figure 4 Intersection approaches

In more detail, the laneSet [LaneList] data frame contains a list of lane [GenericLane] data frames which include a set of attributes. As an example, the configuration of the data frames lane [GenericLane] for all vehicle lanes – as part of the ingress- and egress approach number 1 (the bottom approach, lane numbers 1, 2 and 3) – are included in Table 2. All other vehicle lanes can be configured in a similar matter.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| laneID [LaneID] |  | 2 | 5 |  |
| name [DescriptiveName] |  | fc02 | egress02 |  |
| ingressApproach [ApproachID] |  | 1 | - |  |
| egressApproach [ApproachID] |  | - | 2 |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | 01 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]   00000000 | vehicle [LaneAttributes-Vehicle]   00000000 | BIT STRING (read from left to right) |
| nodes [NodeSetXY] |  |  |  | See paragraph **Error! Reference source not found.** |
| connectsTo [ConnectsToList] |  | connection [Connection] | connection [Connection] | See paragraph 3.2.3 |

Table 2 General lane configuration

### Tracked vehicles

In case a tracked vehicle shares a lane with other traffic, this can be indicated by the sharedWith element (trackedVehicleTraffic (8)). If only part of a lane is shared with a tracked vehicle, the SegmentAttributeXYList can be used to indicate this (sharedWithTrackedVehicle(20)).

## Nodes

### Absolute (ITF) versus relative (MAP) coordinates

One difference between the data formats of the MAP message and the Intersection Topology Format is the format of node points: a node point in ITF is described by its absolute coordinates, whereas a node point in MAP is described by off-sets relative to the reference point of the intersection. When the ITF MapData is converted to MAP message, the node coordinates should be converted to off-sets (see Annex C for conversion code), for the purpose of making the MAP message as small as possible. The example below describe the off-set approach.

### NodeList

One of the properties of a lane is the nodeList: a sequence of signed offset node point values for determining the Xs and Ys to build a path for the centreline of the lane. Note that the sequence difference for an ingress- and egress lane as both lanes should always start at the conflict area. An ingress lane starts from the stop bar. An egress lane starts at the end of the conflict area. See Figure 5 for a visualisation.

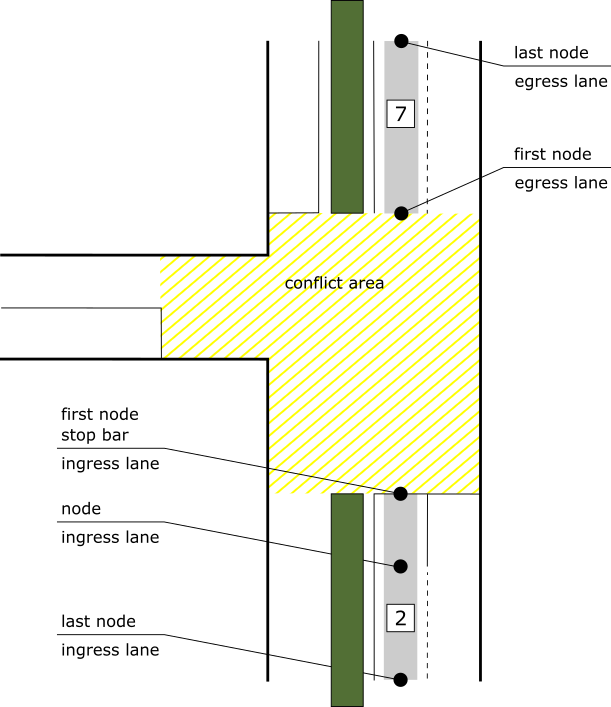


Figure 5 Node configuration

The data frame nodes [NodeSetXY] contains a list of node [NodeXY]. The first node of a lane is described as an offset from the RefPoint [Position3D] while the other nodes are described as a delta from the previous node.

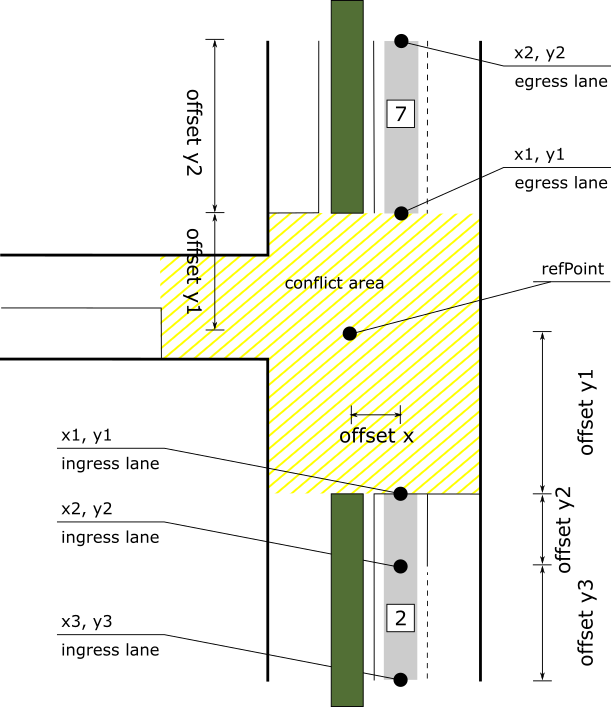


Figure 6 Node offsets from the reference point

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value (x1, y1) | Value (x2, y2) | Value (x3, y3) |
| delta [NodeOffsetPointXY] |  | node-XY1 [Node-XY-20b] | node-XY2 [Node-XY-22b] | node-XY6 [Node-XY-32b] |

Table 3 Node property delta [NodeOffsetPointXY]

#### Node attributes

Each node may contain attributes [NodeAttributeSetXY] which are valid at the node only or remain valid until disabled at another node. See the table below for an example of stop line, white line and curb on the left. NodeAttributes are considered ‘nice to have’ unless essential for the deployment of a service or the perspective of traffic safety, this is indicated in the ITF profile.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value (x1, y1) | Value (x2, y2) | Value (x3, y3) | Comments |
| localNode [NodeAttributeXYList] | nodeAttributeXY [NodeAttributeXY] | 1 (stopline) | - | - | - |
| disabled [SegmentAttributeXYList] | segmentAttributeXY [SegmentAttributeXY] | - | 2 (whiteline) | - | - |
| enabled [SegmentAttributeXYList] | segmentAttributeXY [SegmentAttributeXY] | 2 (whiteline) | - | - | - |
|  | segmentAttributeXY [SegmentAttributeXY] | 5 (curbOnLeft) | - | - | - |
| data [LaneDataAttributeList]  laneDataAttribute  [LaneDataAttribute] | speedLimits  [SpeedLimitList] | - | - | - | See Table 1 |
|  | regional [REGION.Reg-LaneDataAttribute]  addGrpC  [LaneDataAttribute- addGrpC]  maxVehicleHeight  [VehicleHeight]  maxVehicleWeight  [VehicleMass] | **-** | **-** | **-** |  |
| dWidth [Offset-B10] |  | - | 25 | 50 | - |
| dElevation [Offset-B10] |  | - | - | - | - |

Table 4 Node property attributes [NodeAttributeSetXY]

Attributes shall be enabled/disabled as seen from the order of the nodes. i.e. inside out from the intersection. The functional logic, however, should be provided as seen from the direction of driving (e.g. mergingLaneLeft indicates the presence of another lane on the left side of the current lane, as seen from the driving direction).

speedLimits provided in the LaneDataAttributeLIst persists with the provided values for all segments unless changed again. For bicycle and pedestrian lanes, no speedLimits will be provided (or corrected), therefore should be ignored.

### Connections

A vehicle manoeuvre in an intersection is conducted by the following actions. A vehicle approaches the intersection driving along the ingress lane, enters the conflict area, and leaves the intersection using the egress lane. Figure 7 shows the allowed manoeuvres for lane “2”. There are two allowed manoeuvres due to the “connectsTo” link from lane “2” to lane “7” (straight) and from lane “2” to lane “5” (left). The first node (L2-01) of the ingress lane (the stop bar) is connected to the first node (L5-01) of the egress lane “5” and to the first node (L07-1) of the egress lane “7”.

In case a connection links two ingress lanes, possibly from two different intersections (see paragraph 4.7 on the use of remote intersections), the connection connects the first node of upstream ingress lane with the last node of downstream ingress lane.

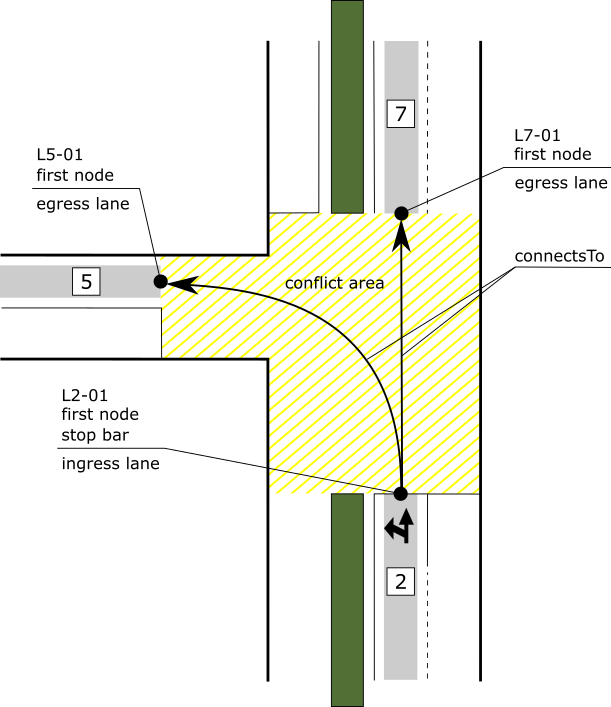


Figure 7 Vehicle manoeuvres from lane 2 to lane 5 and 7

A vehicle manoeuvre is configured using the connectsTo [ConnectsToList] data frame. This data frame contains a connection [Connection] data frame which includes a set of attributes. As an example, the configuration for lane 2 is detailed in Table 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| connectsTo  [ConnectsToList] |  |  |  |  |
| connection  [Connection] |  |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 5 | 7 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | 10000000000 | BIT STRING (read from left to right) BIT0 = maneuverStraightAllowed BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | xxxx | - |  |
|  | id [IntersectionID] | 789 | - |  |
| signalGroup  [SignalGroupID] |  | 1 | 1 |  |
| userClass  [RestrictionClassID] |  | - | - |  |
| connectionID  [LaneConnectionID] |  | 1 | 0 |  |

Table 5 connectsTo configuration

### Connection trajectory

The regional data frame “ConnectionTrajectory-addGrpC” defines the trajectory for travelling through the conflict area of an intersection. The trajectory is defined by two or more nodes. The first node of the ingress lane (see L2-01 in Figure 8) and the first node of the ingress lane (L2-01) share the same position (i.e. the node is duplicated). The ending node of the trajectory (T2-07) and the first node of the connected egress lane (L5-01) share the same position.

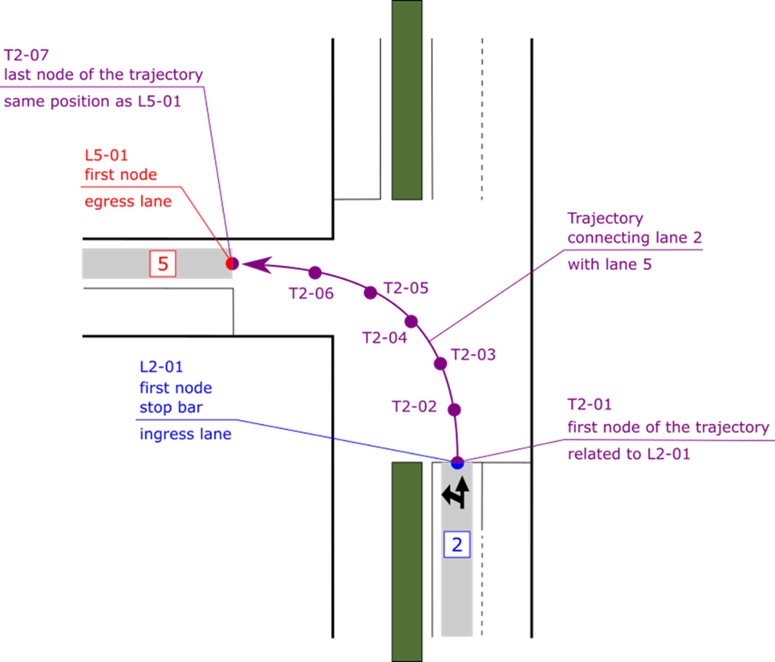


Figure 8 Connection trajectory from lane 2 to lane 5

All nodes of the trajectory can be configured as detailed in paragraph 0.

## Restrictions

The restrictionList [RestrictionClassList] is used to assign a list of typical user classes, for instance public transport vehicles. A RestrictionClassList consists of 1 or multiple RestrictionClassAssignments. A restriction [RestrictionClassAssignment] is used to assign (or bind) a single RestrictionClassID data element to a list of all user classes to which it applies. The established index is then used in the ConnectTo data frame (as part of the lane object), to qualify to whom a SignalgroupID applies when it is sent by the SPAT message about a movement. For instance, when a SignalGroup is a ‘negenoog’ a restriction can be set to assign only public transport vehicles to the connection (with a particular SignalGroup). As an example, the configuration for a restriction is detailed in Table 5. The restriction id then can be filled in the userClass as shown in Table 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| id [RestrictionClassId] |  | 1 | the unique value (within an intersection or local region) that is assigned to this group of users |
| users [RestrictionUserTypeList] |  |  |  |
| user  [RestrictionuserType] | basicType  [RestrictionAppliesTo] | equippedTransit | Public transport vehicles |
|  | Regional  [REGION.Reg-RestrictionUserType] | - | Used to define emission type and fuel type restrictions. |

Table 6 restriction configuration

# Specific intersection or lane configurations

## Bicycle box (bike box)

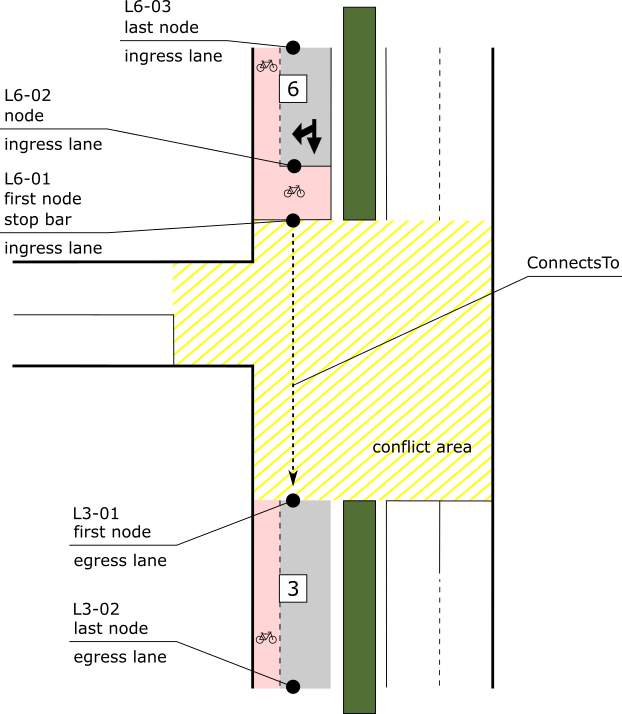


Figure 9 Bicycle box

A bike box must be modelled using segmentAttribute of a vehicle lane. In Figure 9, lane 6 must have the attribute [adjacentBikeLaneOnRight] set. The laneSharing bits for vehicles and bicycles must be set to 1. The relevant lane attributes are described as shown in Table 7. The corresponding nodeSetXY is shown in Table 8. The ConnectsToList is shown in Table 9.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| laneID [LaneID] |  | 6 |  |
| name [DescriptiveName] |  | ingressVehicle |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000100 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic  BIT7 = cyclistVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |
| nodeList [NodeListXY] |  | nodes [NodeSetXY] | See Table 8. |
| connectsTo [ConnectsToList] |  | [Connection] | See Table 9. |

Table 7 Lane configuration for bikeBox.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value (L6-01) | Value (L6-02) | Value (L6-03) | Comments |
| localNode [NodeAttributeXYList] | nodeAttributeXY | - | 1 (stopline) | - | - |
| disabled [SegmentAttributeXYList] | segmentAttributeXY | - | 16 (bikeBoxInFront) | - | - |
| enabled [SegmentAttributeXYList] | segmentAttributeXY | - | 14 (adjacentBikeLaneOnRight) | - | - |
|  | segmentAttributeXY | 16 (bikeBoxInFront) | - | - | - |

Table 8 nodeSetXY for lane 6

|  |  |  |  |
| --- | --- | --- | --- |
| Data element  For LaneID | Sub-data element | Value  6 | Comments |
| connectsTo  [ConnectsToList] |  |  |  |
| connection  [Connection] |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 3 |  |
|  | maneuver  [AllowedManeuvers] | 10000000000 | BIT STRING (read from left to right) BIT0 = maneuverStraightAllowed |
| signalGroup  [SignalGroupID] |  | 1 | SignalGorupIDs for both connections are same since they are part of the same signalGroup. |
| connectionID  [LaneConnectionID] |  | 1 |  |

Table 9 connectsToList for lane 6

## Bicycle lanes

In the Netherlands there exist many different configurations for bike lanes, including different types of lane markings, lane sharing rules and longitudinal configuration changes. To define a common practice and for the sake of simplicity, it was decided to break down all these situations into two variants shown the in next two paragraphs. The configuration of the bike lane at the stop line is considered leading and representative for the entire bike lane. In other words, if a bike lane is protected with continuous lane marking at the stop bar, it is assumed that the entire bike lane has a continuous lane marking, even if this is not the case in reality.

Future requirements may change this approach.

### Bicycle lane with continuous lane marking

A bicycle lane with continuous lane marking, where there’s no lane-sharing with other vehicles (other than allowed by law), should be modelled with a separate lane and therefore a separate connection. The sharedWith should not be set. The laneType should be set to bikeLane.

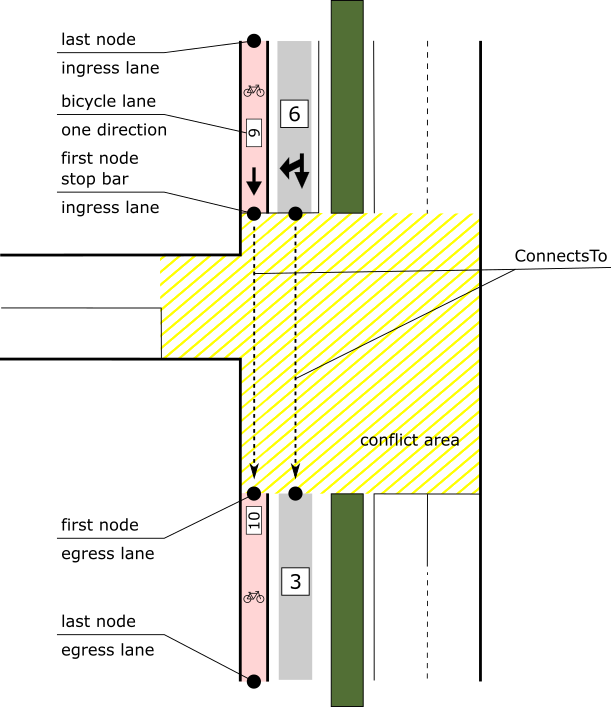


Figure 10: Bicycle lane with no lane-sharing.

### Bicycle lane with broken lane marking

Bicycle lanes with broken lane marking, where lane-sharing is present, should be modelled by the lane that is also used for other vehicles. The element sharedWith should contain cyclistVehicleTraffic (7) and the laneType should be set to vehicle.

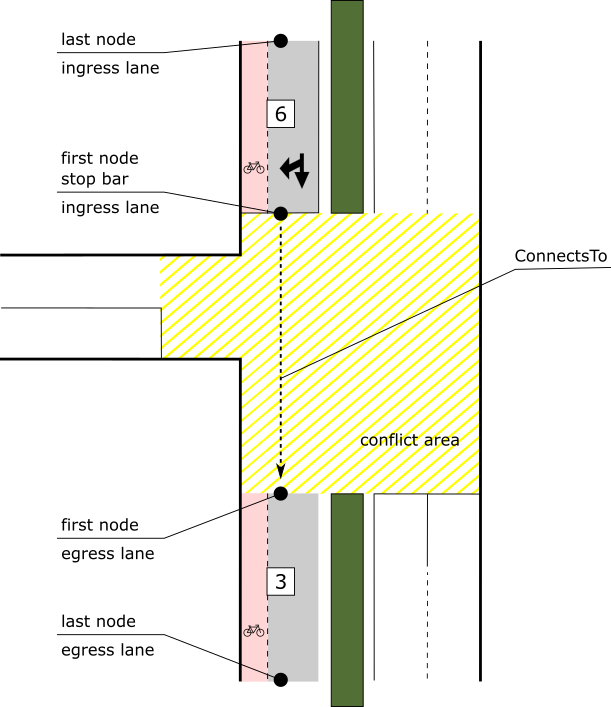


Figure 11: Bicycle lane with lane-sharing.

### Bidirectional separated bicycle lanes

Bidirectional bicycle lanes separated from vehicle lanes shall be defined as shown in the figure below. All bicycle lanes are defined as bidirectional lanes and where they intersect, the overlapping nodes of both lanes have the mergePoint and divergePoint attribute set. In addition, all bicycle lanes in one quadrant of an intersection (e.g. lanes 10 and 11) have the same ingressApproachID which is unique within the intersection. This allows easy identification of all bicycle lanes which are related.

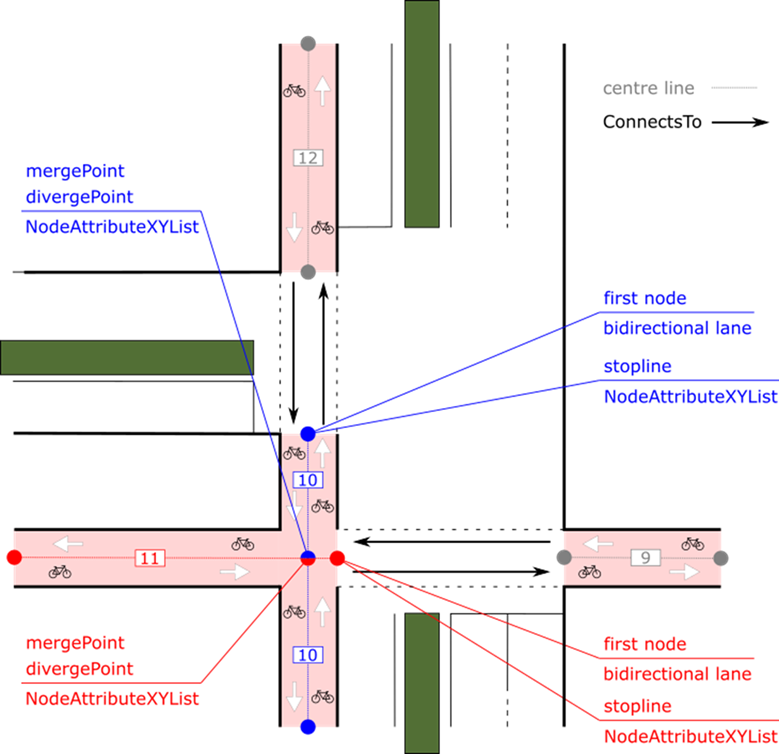


Figure 12: bidirectional separated bicycle lane

### Cyclist movement in two stages

For turns of cyclists in two stages a separate lane is used for the second stage of the turn. This lane is assigned to arm A in the image. The maneuver should be set to maneuverStraightAllowed (0).

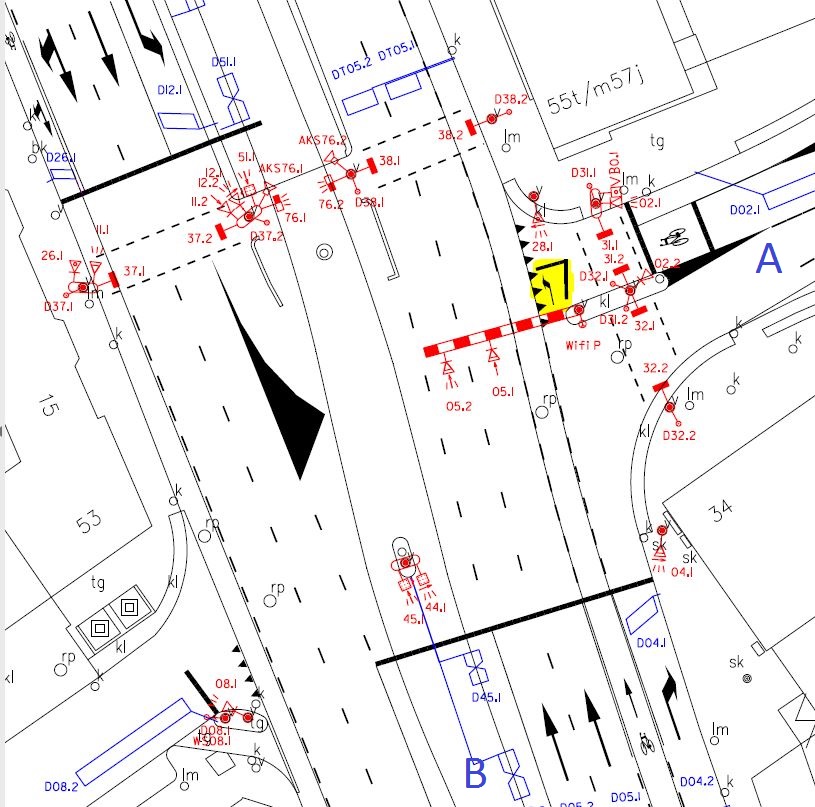


Figure 13: cyclist movement in two stages

### Bicycle street

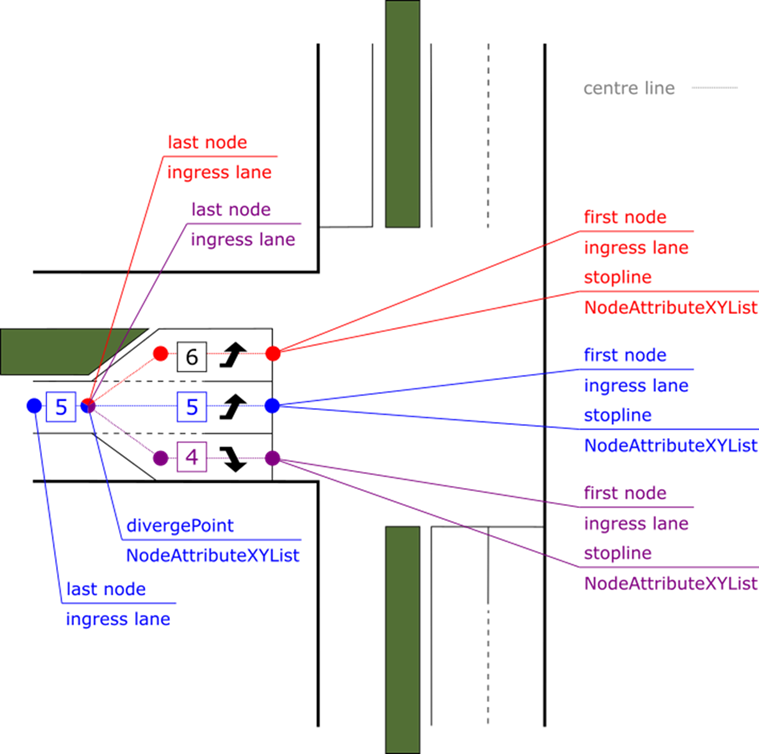
A bicycle street is a street designed as a bike route, but on which cars are also allowed. However, this car use is limited by the character and the layout of the bicycle street. This is common practise in the Netherlands, often visible by red pavement. In this case the laneType should be set to vehicle and the attribute sharedBikeLane should be enabled in the segmentAttributeXY.

The attribute sharedBikeLane in the segmentAttributeXY can also be used when bicycles on a bicycle lane on the right have to cross the vehicle lane to reach the shared vehicle lane on the left. In that case the area where bicycle traffic can cross the vehicle lane has to be marked by enabling and disabling the attribute sharedBikeLane.

## Intersection lanes

### Fan out

In many cases the road fans out at an intersection to allow separate lanes for the left and/or right turns. In this case new lanes arise. The lane(s) before the fan out must be the one(s) for through traffic; this is lane 5 in *Figure 14*. In general, this will be the straight direction, but exceptions are possible where the through traffic takes a turn. For a T junction, the major road must be selected as the through direction. If the left and right directions are equal roads, one of them can be chosen. All lanes that fan out must have the same ingressApproachID.



*Figure 14 Fan out of lanes at the intersection*

### Road Geometry

Lanes must smoothly follow the road geometry, and care must be taken that the heading of the road segments is in line with the heading of the road. A too large deviation in the heading of a lane could lead to failing map-matches.

wrong

better

good

Figure 15 Wrong and good ways to describe a lane

### Merging lanes

To indicate that lane merging is possible/allowed the segment attribute ‘mergingLaneLeft’ or ‘mergingLaneRight’ shall be set. Typically, the use of the attribute like ‘whiteLine’ is limited to segments longer than 15 meters, unless it concerns a physical separation of lanes.

Attributes are enabled/disabled as seen from the order of the nodes. i.e. inside out from the intersection. The functional logic, however, should be provide as seen from the direction of driving.

When lanes merge, this must be indicated on both affected lanes with a mergePoint, while the mergingLaneRight/mergingLaneLeft attributes shall be enabled from this node point onwards. The tapering of the merging road is indicated with ta taperToLeft or taperToRight, as shown in Figure 16.

mergingLaneRight enabled

taperToLeft disabled

mergePoint

mergePoint

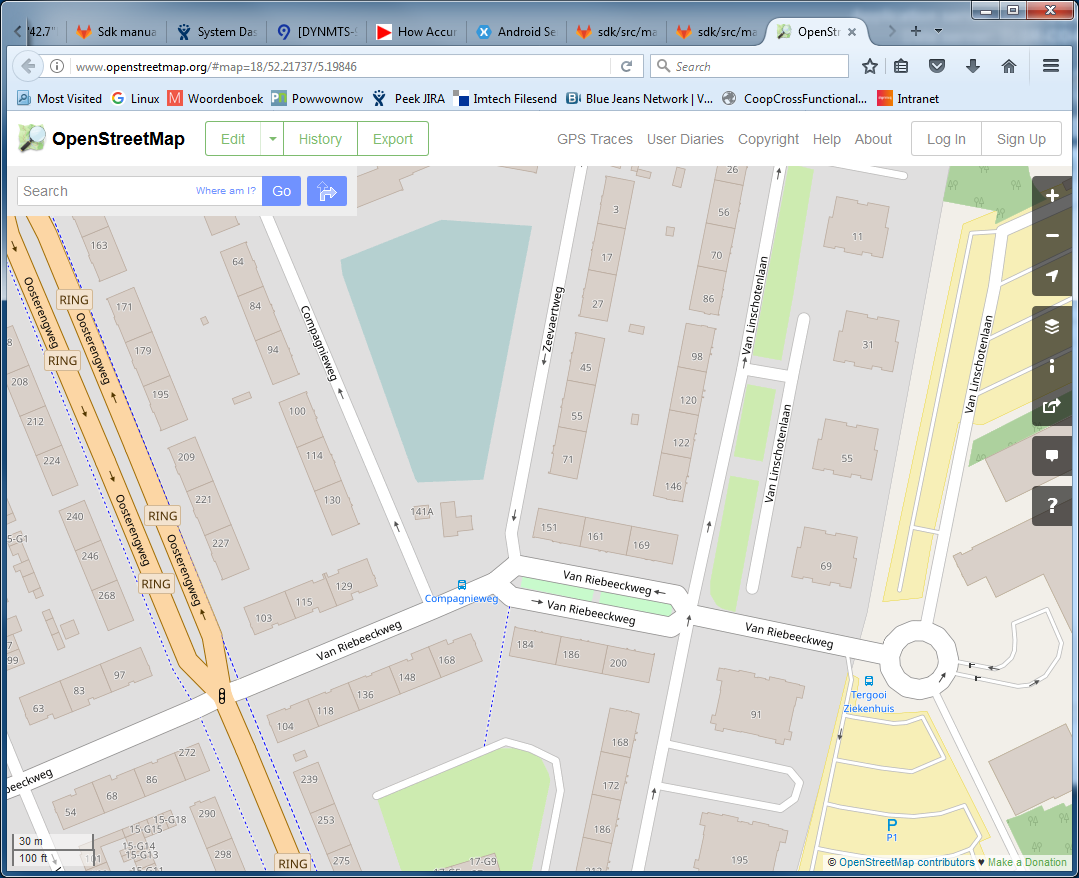
mergingLaneLeft + taperToLeft enabled

stopline

Figure 16 Merging lanes (driving direction left to right)

### Joining approach roads

Sometimes the approach road merges just before the intersection. In the example below the merge is defined as an additional lane which merges with the main approach (blue).



mergePoint

mergePoint

20

2

Figure 17 A lane merge on the approach lane

If the side road is relevant to the traffic light controller (e.g. to estimate traffic demand and estimate time of arrival of priority vehicles), it should be defined as shown. In other cases, e.g. drives to houses, parking areas and petrol stations, the NodeAttribute turnOutPointRight (or Left) shall be used. This NodeAttribute shall only be defined for the lane adjacent to the side street or drive. For turnOutPointRight this means the most right lane, for turnOutPointLeft this means the most left lane.

## Public transport lane

The following figure displays all nodes of lanes 4, 5 and 9 of ingress approach 2, with coloured dots for each node. Lane 4 (blue) and lane 5 (red) start at the stop line of the intersection. The bus lane 9 (purple) starts at the mergePoint of lanes 4 and 9.

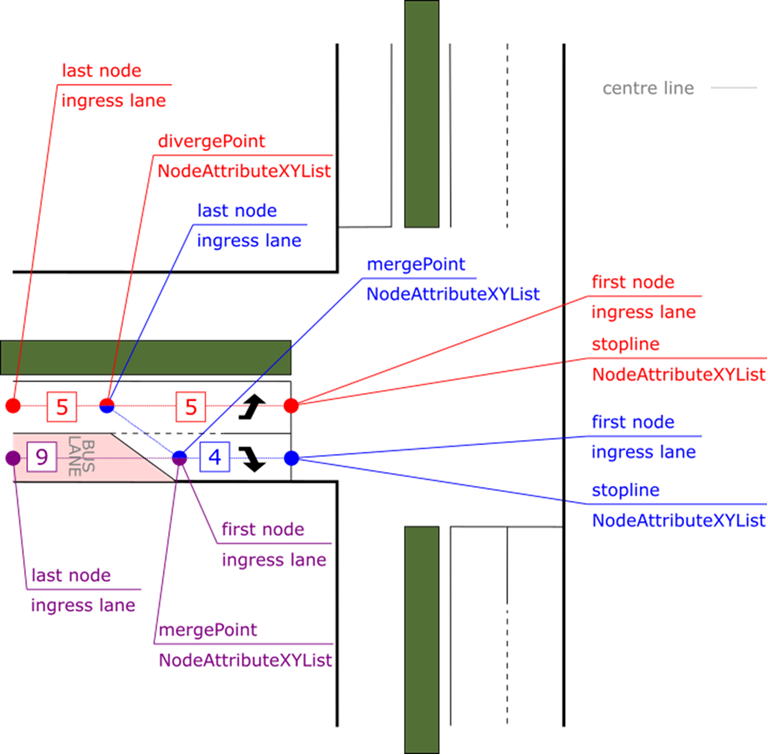


Figure 18: An example of a setback bus lane that transfers into a right turning lane

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| laneID [LaneID] |  | 4 | 9 |  |
| name [DescriptiveName] |  | fc07 | bus lane 47 |  |
| ingressApproach [ApproachID] |  | 2 | 2 |  |
| egressApproach [ApproachID] |  | - | - |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | 0000110000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic  BIT4 = busVehicleTraffic  BIT5 = taxiVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle] | vehicle [LaneAttributes-Vehicle] | BIT STRING (read from left to right) |
|  | Vehicle  [LaneAttributes-  Vehicle] | 00000000 | 00000000 |  |
| nodeList [NodeListXY] |  | nodes [NodeSetXY] | nodes [NodeSetXY] | See paragraph 3.2.2 |
| connectsTo [ConnectsToList] |  | [Connection] | [Connection] | See paragraph 3.2.3 |

Table 10 Lane configuration with set back bus lane

## Dynamic lane configuration

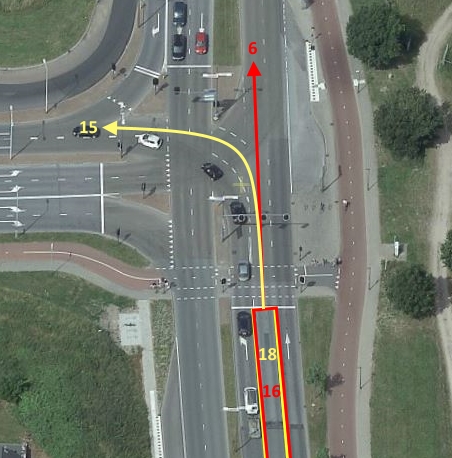


Figure 19 Dynamic lane in Deventer A060

Dynamic lanes in an intersection are configured using ‘Variants’. As an example, in Deventer A060 (as shown in Figure 19), lanes 16 and 18 are overlaying dynamic lanes of the same physical lane. In the figure, lane 16 and its connection is shown in red and lane 18 an its connection is shown in yellow. During the morning peak on weekdays, i.e., from 06:30 to 10:30, the left turn is allowed and the straight turn is not allowed. During all other times, the left turn is not allowed and the straight turn is allowed. This can be configured in the ITF in the following way.

In mapData 🡪 … 🡪 genericLane,

* Two lanes - 16 and 18 are described with the same NodeSet since they share the same position.
* In the [connectsToList] for lane 16, the connectingLane is set to “6”. In the [connectsToList] for lane 18, the connectingLane is set to “8”. All other attributes in the connection are configured as explained previously.
* In Lane-Attributes-Vehicle, bit 0, i.e., [isVehicleRevocableLane] bit is set to 1 for both these lanes.

In controlData->…Variants,

* Two variants are configured and each of them have either lane 16 or lane 18 disabled in the ‘disabledLanes’ list.
* The variantType is configured accordingly.
* The vlogIndicator and its value is specified if available.
* If vlogIndicator is not available, the activePeriods list is specified as shown in Table 11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| variants  [VariantList] |  |  |  |  |
| variant  [Variant] |  |  |  |  |
| variantID  [VariantID] |  | 1 | 2 |  |
| name  [DescriptiveName] |  | variant –  normalOperation | variant - congestion |  |
| variantCategory  [VariantCategory] |  | normalOperation | congestion |  |
| disabledLanes  [DisabledLaneList] | laneID  [LaneID] | 18 | 16 |  |
| vlogIndicator  [VlogIndicator] | vlogCat  [VlogCat] | US | US |  |
|  | vlogIdx  [VlogIdx] | 198 | 198 | This refers to the outputsignal “u\_dyn\_rystr” in the VlogStream |
|  | matchValue  [MatchValue] | 0 | 1 |  |

Table 11 variants configuration

If vlogIndicator is not available, the activePeriods attribute is used to specify when a specific variant is active. In the example considered, Variant 2 is active during the specified activePeriods.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data element | activePeriod | activePeriod | activePeriod | activePeriod | activePeriod |
| Days  [Days] | 1 | 2 | 3 | 4 | 5 |
| beginTime  [BeginTime] | 06:30:00 | 06:30:00 | 06:30:00 | 06:30:00 | 06:30:00 |
| endTime  [EndTime] | 10:30:00 | 10:30:00 | 10:30:00 | 10:30:00 | 10:30:00 |

Table 12 activePeriods for variant 2

## Multiple intersections for 1 TLC (and ITF or MAP file)

The IntersectionGeometry shall be created for each independent conflict area, this being:

* A conflict area having own stop lines and signal heads for all conflicting directions.
* A conflict area that – theoretically – can be controlled independently from other conflict areas safely.
* A conflict area does not share the conflict matrix with another conflict area.

See the examples below.

Note that MapData aims to objectively describe the IntersectionGeometry as it can be observed. How the conflict areas are controlled functionally and how they are grouped as a consequence is a different perspective. The motivation for creating one IntersectionGeometry for each independent conflict area is the limited array size of several data elements (e.g. lanes) as defined by the standards.

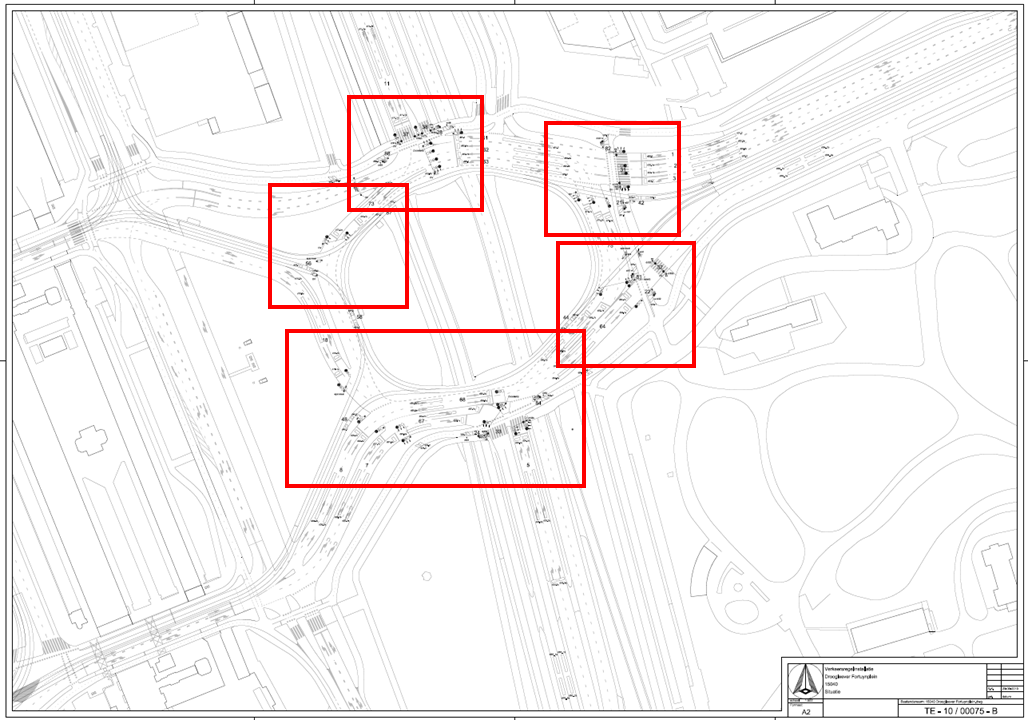


Figure 20: multiple intersections example 1

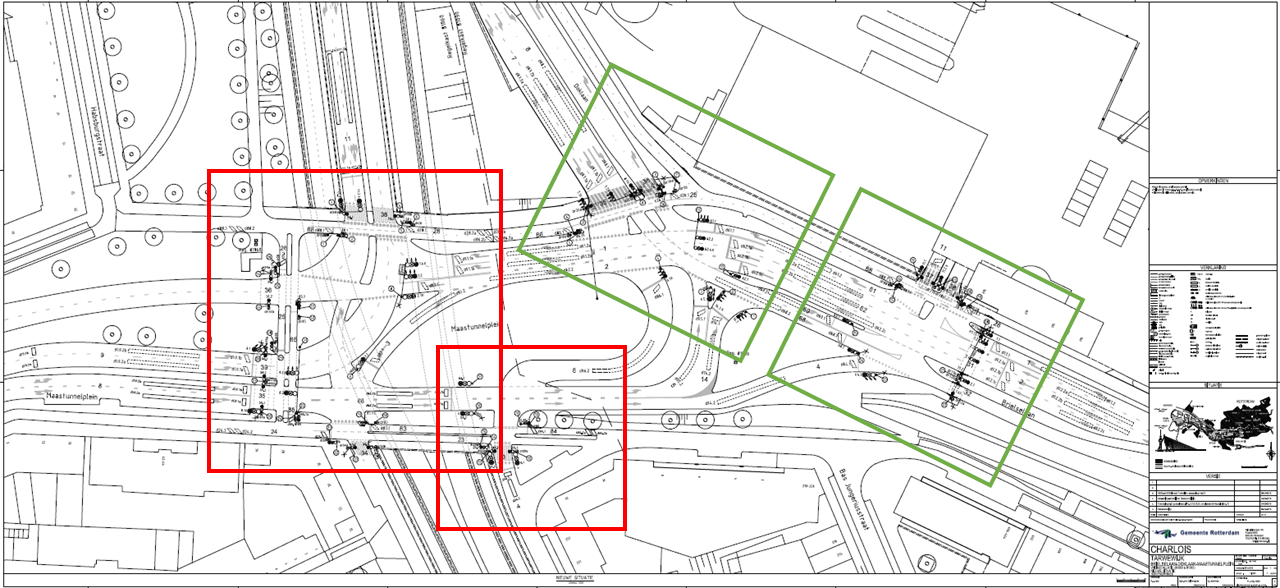


Figure 21: multiple intersections example 2

Note: this example contains two Traffic Light Controllers, each controlling two intersections (those in red and in green).

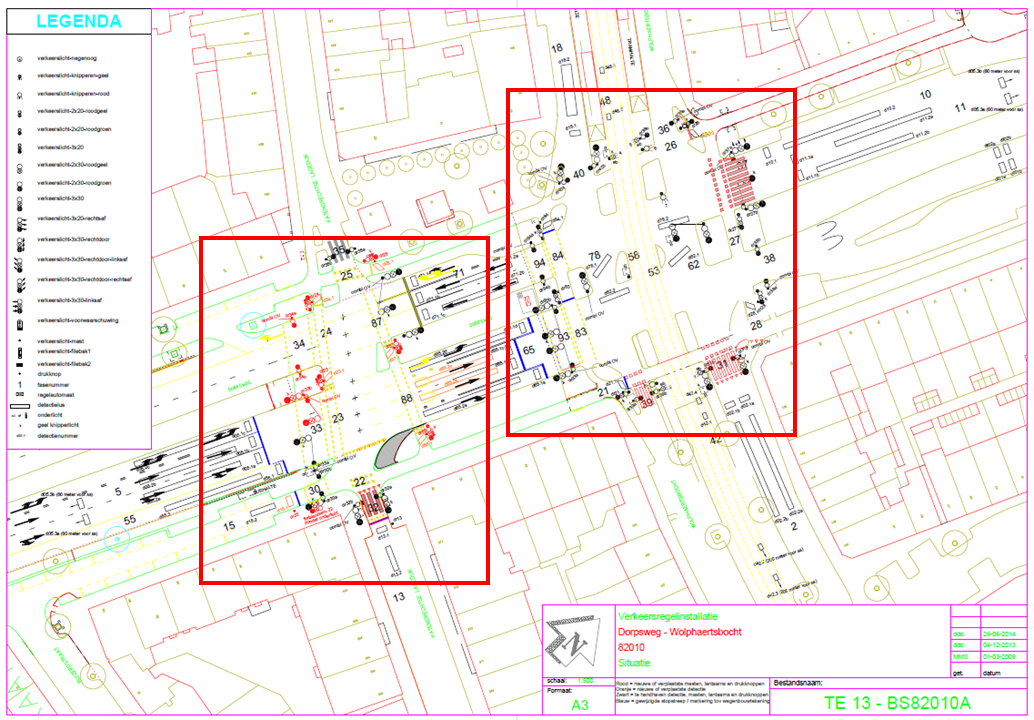


Figure 22: multiple intersections example 3

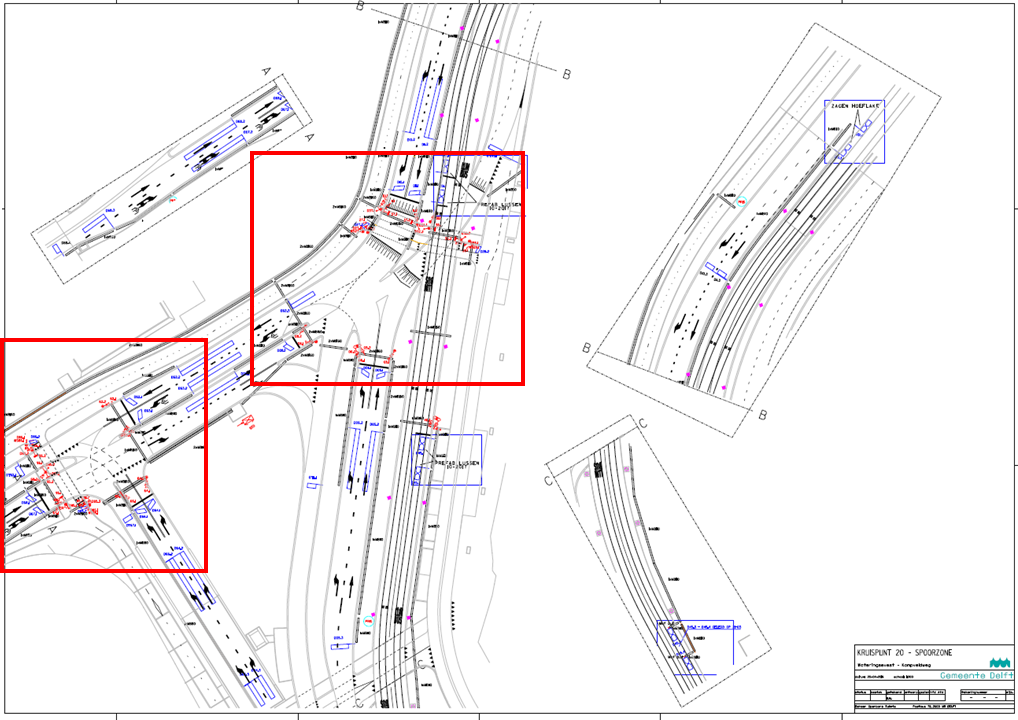


Figure 23: multiple intersections example 4

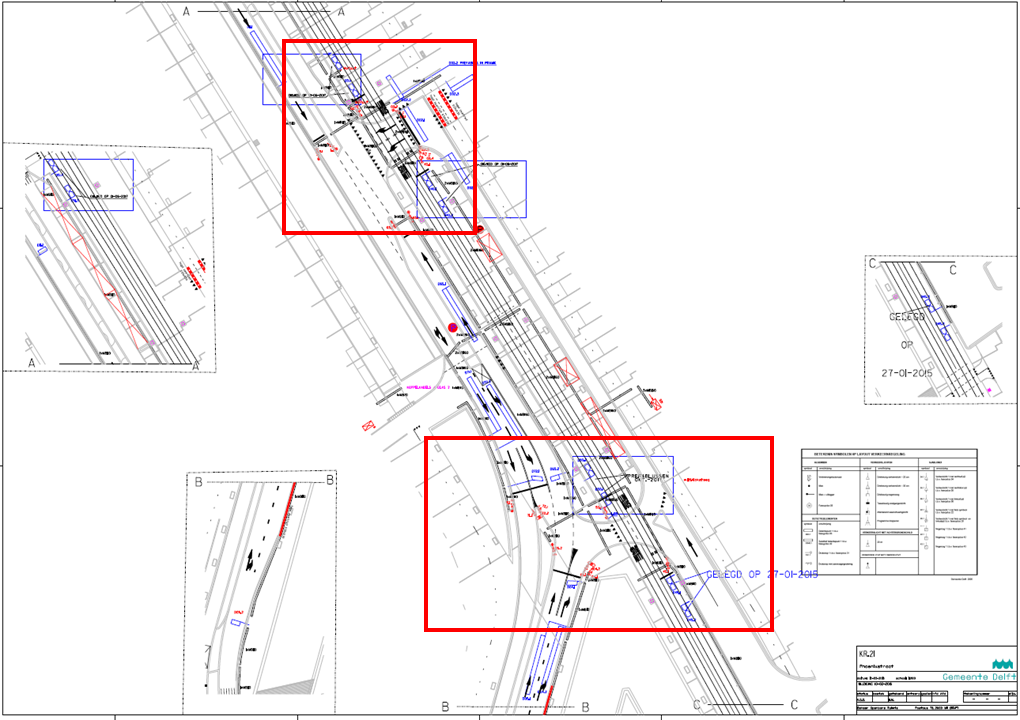


Figure 24: multiple intersections example 5

## Remote intersection

Intersections that are within a short distance of each other can be linked using the ‘remoteIntersection’ value in the connectsTo. With this option, ingress lanes of one intersection are directly linked to ingress lanes of another intersection, without providing egress lanes.

### Use of remoteIntersection, no egress lanes

This example, as shown in Figure 25, shows the configuration of one topology file (MAP A) with two intersections (A and B) within a short distance of each other.

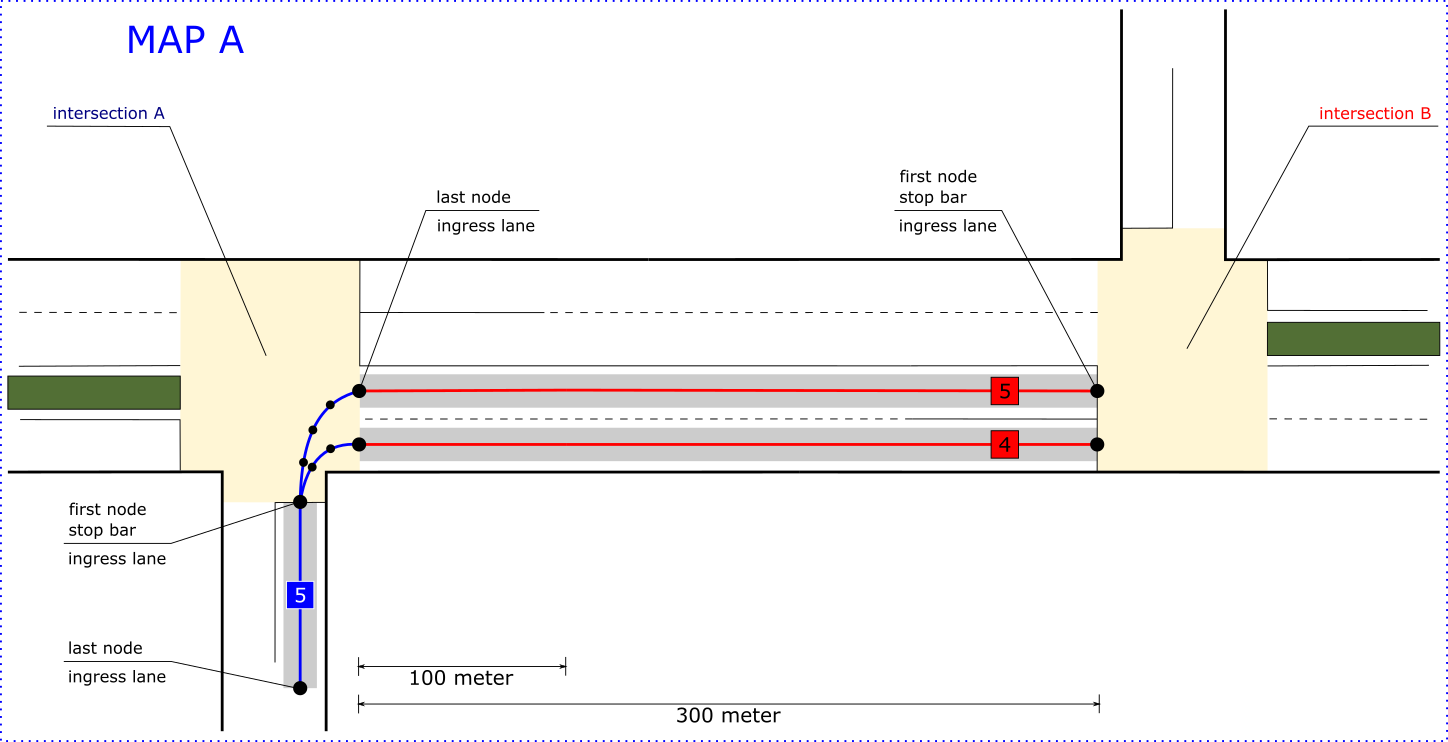


Figure 25 use of remoteIntersection, no egress lanes

The first node (stop bar) of lane number 5 of the ingress approach of intersection A will be connected to the last node of lane number 4 and the last node of lane number 5 of the ingress approach of intersection B. The following two tables will detail the required configurations of the connection.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| laneID [LaneID] |  | 5 |  |
| name [DescriptiveName] |  | ingress03 |  |
| ingressApproach [ApproachID] |  | 2 |  |
| egressApproach [ApproachID] |  | - |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |

Table 13 Lane configuration intersection A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| laneID [LaneID] |  | 4 | 5 |  |
| name [DescriptiveName] |  | ingress02 | ingress03 |  |
| ingressApproach [ApproachID] |  | 2 | 2 |  |
| egressApproach [ApproachID] |  | - | - |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicle-Traffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |

Table 14 Lane configuration intersection B

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |  |
| connection  [Connection] |  |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 4 | 5 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | 100 | 100 |  |
|  | id [IntersectionID] | 2 | 2 |  |
| signalGroup  [SignalGroupID] |  | 3 | 3 |  |
| userClass  [RestrictionClassID] |  | - | - |  |
| connectionID  [LaneConnectionID] |  | 1 | 2 |  |

Table 15 Configuration connectsTo for laneID 5

Use of overlapping egress lanes Figure 26 shows the same example only now with egress lanes for intersection A. When applying the MAP or ITF profile, minimum length requirements for lanes imply that egress lanes of intersection A shall overlap with ingress lane of intersection B.

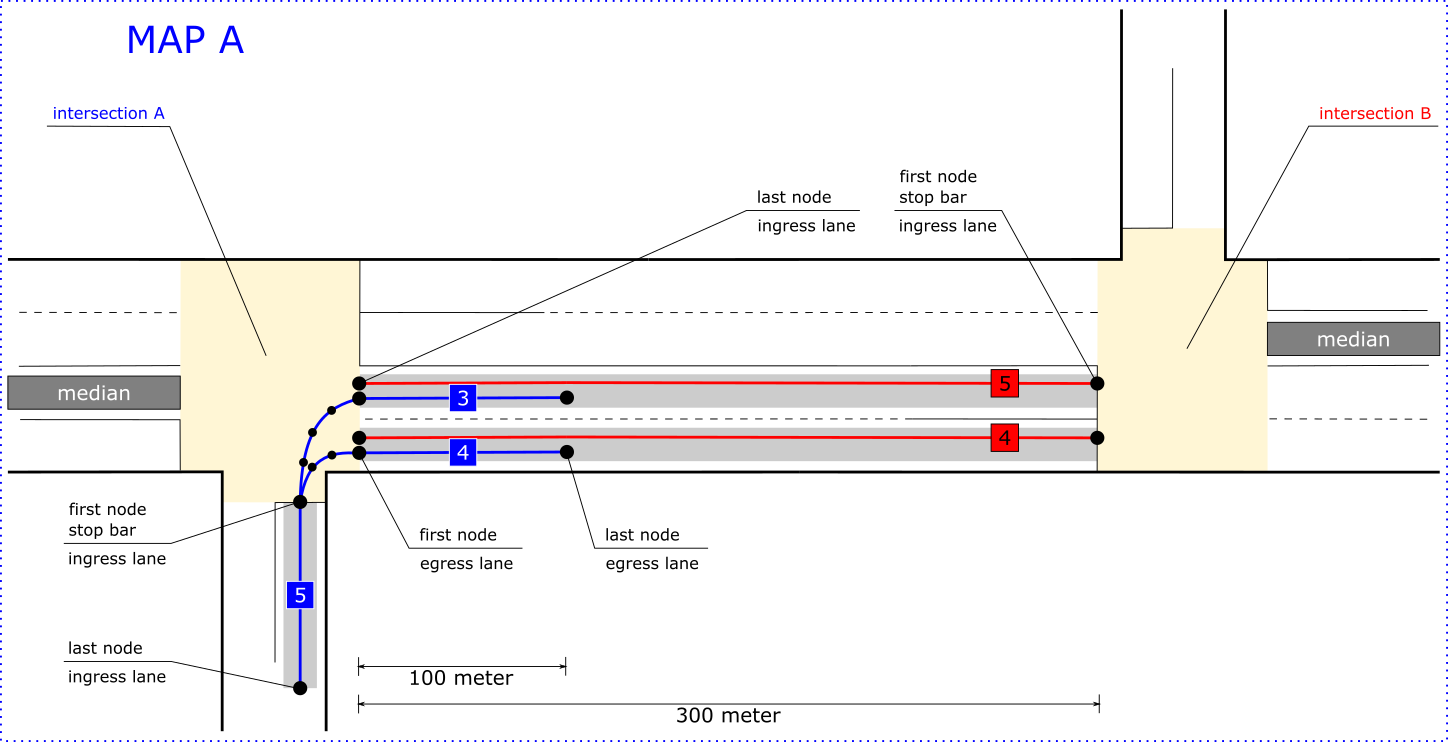


Figure 26 use of overlapping egress lanes

The following two tables will detail the required configurations of the connection.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Value | Comments |
| laneID [LaneID] |  | 5 | 3 | 4 |  |
| name [DescriptiveName] |  | ingress03 | egress01 | egress02 |  |
| ingressApproach [ApproachID] |  | 2 | - | - |  |
| egressApproach [ApproachID] |  | - | 1 | 1 |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | 01 | 01 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | 0001000000 | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicle-Traffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | vehicle [LaneAttributes-Vehicle]  00000000 | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |

Table 16 Lane configuration intersection A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| laneID [LaneID] |  | 4 | 5 |  |
| name [DescriptiveName] |  | ingress02 | ingress03 |  |
| ingressApproach [ApproachID] |  | 2 | 2 |  |
| egressApproach [ApproachID] |  | - | - |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |

Table 17 Lane configuration intersection B

## Double stop lines

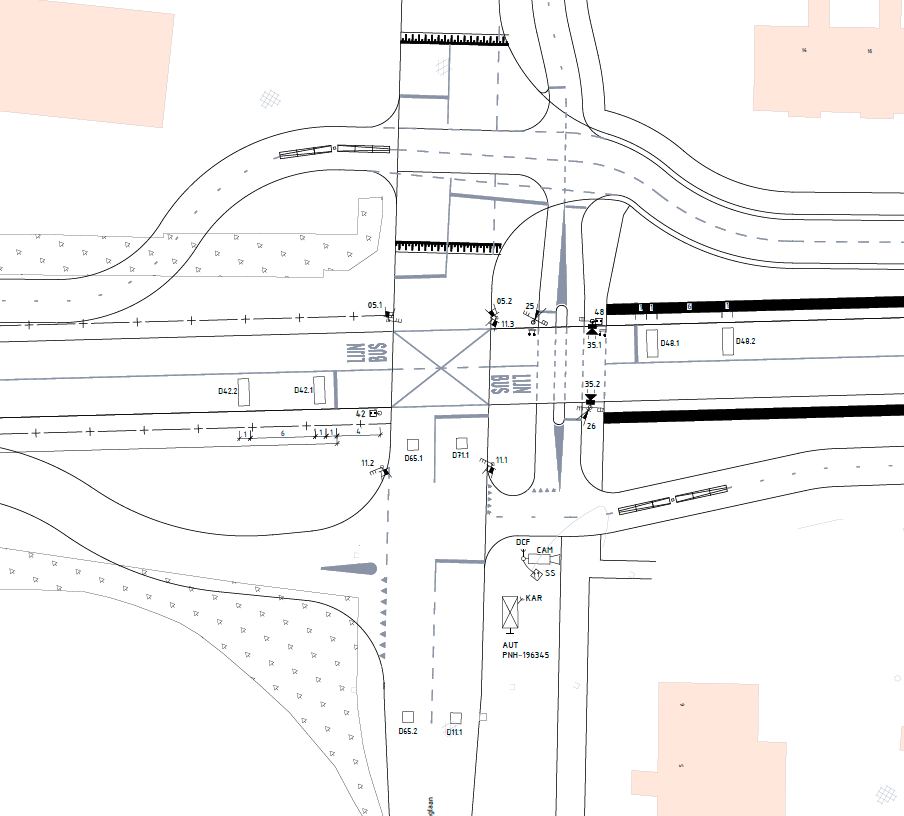


Figure 27: example double stop lines

Some lanes have double stop lines. For example, in the figure above starts one lane at the stop line near detection loop D71.1 and ends below D11.1. This lane has 2 stop lines. This lane requires a node to be placed at each stop line which sets the nodeattribute stop line. In the next table is this example detailed:

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| laneID [LaneID] |  | 1 |  |
| name [DescriptiveName] |  | 1 |  |
| ingressApproach [ApproachID] |  | 2 |  |
| egressApproach [ApproachID] |  | - |  |
| laneAttributes [LaneAttributes] | directionalUse [LaneDirection] | 10 | BIT STRING (read from left to right) BIT0 = Ingresspath BIT1 = Egresspath |
|  | sharedWith [LaneSharing] | 0001000000 | BIT STRING (read from left to right) BIT3 = individualMotorizedVehicleTraffic |
|  | laneType [LaneTypeAttributes] | vehicle [LaneAttributes-Vehicle]  00000000 | BIT STRING (read from left to right) |
|  |  |  |  |
|  | Node | Localnode |  |
| nodes [NodeSetXY] | 1 | StopLine |  |
|  | 2 | mergePoint | (node available for merging from right road) |
|  | 3 | StopLine |  |
|  | 4 |  |  |
|  |  |  |  |
| connectsTo [ConnectsToList] |  | connection [Connection] |  |
| regional [REGION.Reg-GenericLane] |  | addGrpC [ConnectionTrajectory-addGrpC] |  |

## Connections

Connections between lanes are configured using the ‘connectsTo’ data frame. This paragraph describes three cases on how to configure the ‘connectTo’ data frames of an intersection.

### Connection 1:2

The first example shows how to connect a single lane of an ingress approach to two lanes of an egress approach.

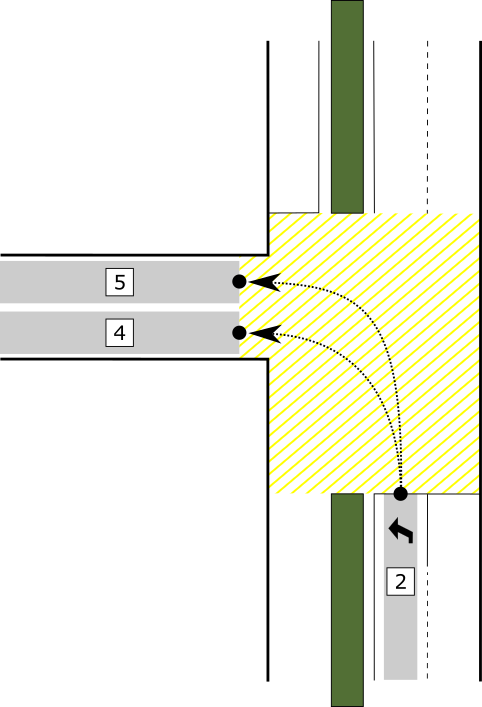


Figure 28 Connection from a single lane of an ingress approach to two lanes of an egress approach

A single lane of an ingress approach has to be connected to all possible lanes of its egress approach. In this case lane number 2 of ingress approach 1 has to be connected to both lane number 4 and lane number 5 of egress approach 2. The following two table details the required configurations of the connection.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |  |
| connection  [Connection] |  |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 4 | 5 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | - | - |  |
|  | id [IntersectionID] | - | - |  |
| signalGroup  [SignalGroupID] |  | 2 | 2 |  |
| userClass  [RestrictionClassID] |  | - | - |  |
| connectionID  [LaneConnectionID] |  | 1 | 2 |  |

Table 18 Configuration of the connectsTo data frame of laneID 2

### Connection 2:2

The second example shows how to connect two lanes of an ingress approach to two lanes of an egress approach.



Figure 29 Connection from two lanes of an ingress approach to two lanes of an egress approach

When the number of lanes of an ingress approach are connected to the equal number of lanes of its egress approach, only one connections should be made. In this case lane number 1 of ingress approach 1 has to be connected to lane number 5 of egress approach 2. And lane number 2 of ingress approach 1 has to be connected to lane number 4 of egress approach 2. The following tables will detail the required configurations of the connection.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |
| connection  [Connection] |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 5 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | - |  |
|  | id [IntersectionID] | - |  |
| signalGroup  [SignalGroupID] |  | 2 |  |
| userClass  [RestrictionClassID] |  | - |  |
| connectionID  [LaneConnectionID] |  | 1 |  |

Table 19 Configuration of the connectsTo data frame of laneID 1

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |
| connection  [Connection] |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 4 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | - |  |
|  | id [IntersectionID] | - |  |
| signalGroup  [SignalGroupID] |  | 2 |  |
| userClass  [RestrictionClassID] |  | - |  |
| connectionID  [LaneConnectionID] |  | 2 |  |

Table 20 Configuration of the connectsTo data frame of laneID 2

### Connection 2:3

The third and final example shows how to connect two lanes of an ingress approach to three lanes of an egress approach. Typically, extra lanes add only linked to the most left lane (for right hand driving). However, this strongly depends on road markings and turning lanes.

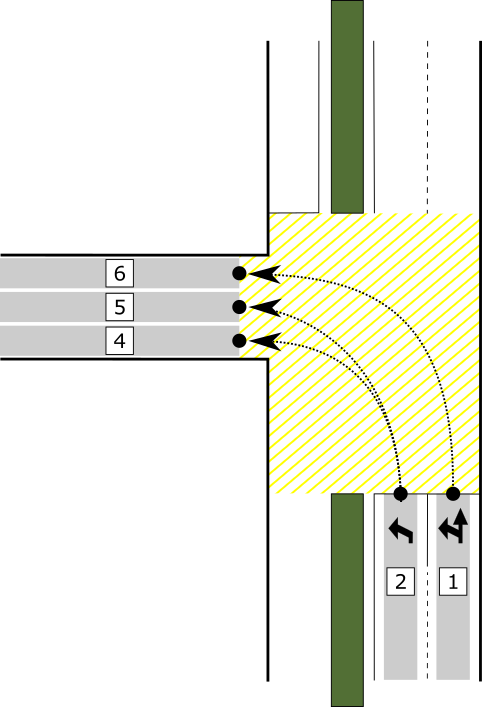


Figure 30 Connection from two lanes of an ingress approach to three lanes of an egress approach

This case is a combination of the previous two. Lane number 1 of ingress approach 1 has to be connected to lane number 6 of egress approach 2. And lane number 2 of ingress approach 1 has to be connected to both lane number 4 and lane number 5 of egress approach 2. The following three tables will detail the required configurations of the connection.

|  |  |  |  |
| --- | --- | --- | --- |
| Data element | Sub-data element | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |
| connection  [Connection] |  |  |  |
| connectingLane  [ConnectingLane] | Lane  [LaneID] | 6 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | - |  |
|  | id [IntersectionID] | - |  |
| signalGroup  [SignalGroupID] |  | 2 |  |
| userClass  [RestrictionClassID] |  | - |  |
| connectionID  [LaneConnectionID] |  | 1 |  |

Table 21 Configuration of the connectsTo data frame of laneID 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | Value | Value | Comments |
| connectsTo [ConnectsToList] |  |  |  |  |
| connection  [Connection] |  |  |  |  |
| connectingLane  [ConnectingLane] | lane  [LaneID] | 4 | 5 |  |
|  | maneuver  [AllowedManeuvers] | 01000000000 | 01000000000 | BIT STRING (read from left to right) BIT1 = maneuverLeftAllowed |
| remoteIntersection  [Intersection- ReferenceID] | region [RoadRegulatorID] | - | - |  |
|  | id [IntersectionID] | - | - |  |
| signalGroup  [SignalGroupID] |  | 2 | 2 |  |
| userClass  [RestrictionClassID] |  | - | - |  |
| connectionID  [LaneConnectionID] |  | 2 | 3 |  |

Table 22 Configuration of the connectsTo data frame of laneID 2

## Crosswalk

### Safe island

A crosswalk can be divided in separate crosswalks, for instance one crosswalk over the ingressApproach and one crosswalk over the egressApproach. Both crosswalks may be controlled by different signal groups and even multiple signal groups (see next section). The figure below shows how to define the crosswalk-lanes at the safe island. They are defined as two bidirectional lanes with one overlapping node, which has the mergePoint attribute set.

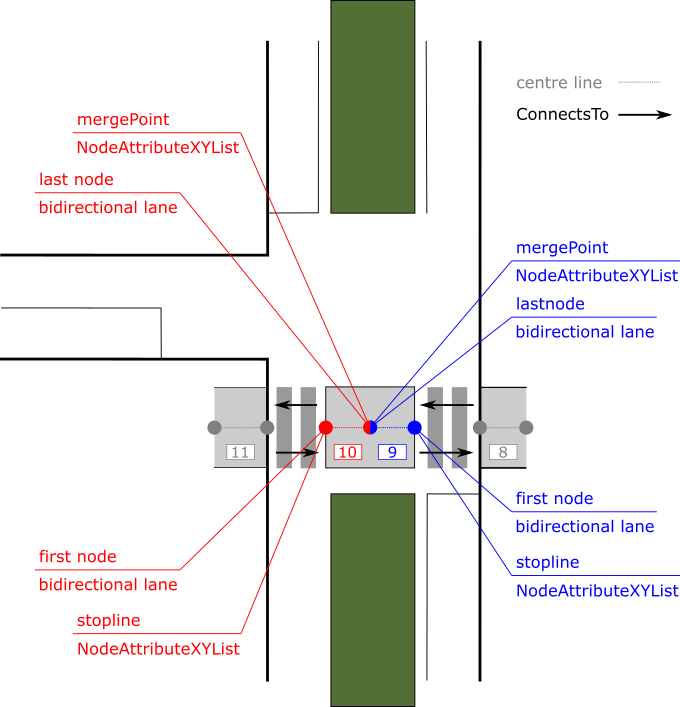


Figure 31: crosswalk with safe island

### Multiple signal groups

In general there are three different situations that are common practice in the Netherlands:

1. Each crosswalk is controlled by a different SignalGroup for pedestrians crossing in both directions (i.e. SignalGroup 31 and SignalGroup 32);
2. Each crosswalk is controlled by different SignalGroups for each direction separately (i.e. SignalGroups 31 and 91 for the ingressApproach en SignalGroups 32 and 92 for the egressApproach);
3. The outer “waiting” pedestrian area on both crosswalks are controlled by one SignalGroup and the inner “waiting” pedestrian area (between the crosswalks) are controlled by another SignalGroup for both directions.

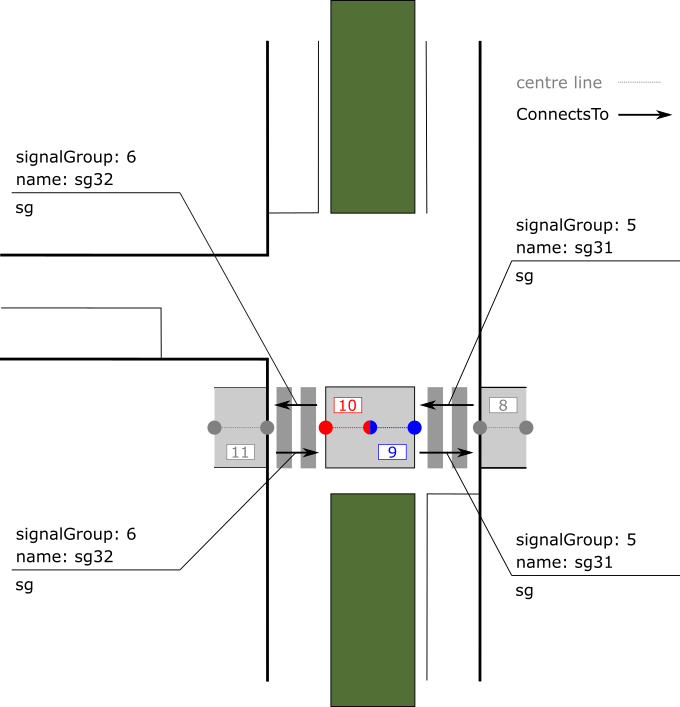


Figure 32 Situation A: Standard split Crosswalk with 2 signal groups

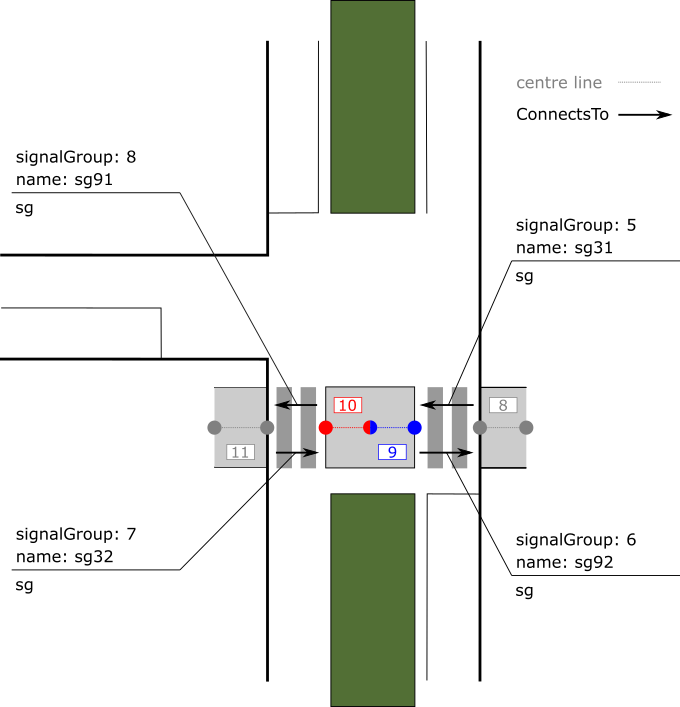


Figure 33 Situation B: Split crosswalk with 4 exclusive signal groups

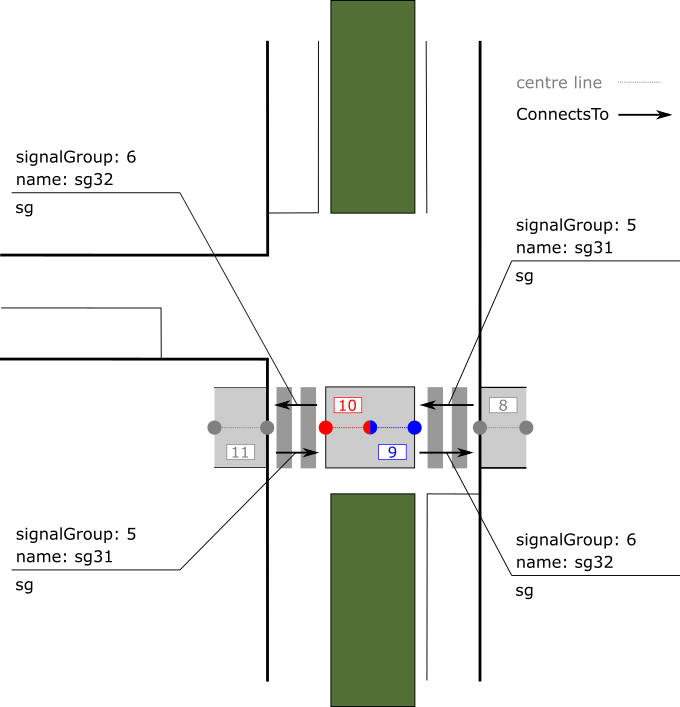


Figure 34 Situation C: Split crosswalk with ‘inner’-‘outer’ signal groups

The way to describe the lane data element is the same as described in Table 9. Note however that lanes 11 and 12 cross the ingressApproach and lanes 12 and 13 cross the egressApproach.

The referring to the signalGroup in the connectTo data element however is different for each situation. In the next table the three situations are described. In the first column points the applied situation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Situation | Data element | Sub-data element | Value | Value | Value | Value |
|  | **laneID** |  | 11 | 12 | 13 | 14 |
|  | **connectsTo**  **[ConnectsToList]** |  |  |  |  |  |
|  | **connection**  **[Connection]** |  |  |  |  |  |
|  | **connectingLane**  **[ConnectingLane]** | lane  [LaneID] | 12 | 11 | 14 | 13 |
|  |  | maneuver  [AllowedManeuvers] | 10000000000 | 10000000000 | 10000000000 | 10000000000 |
|  | **remoteIntersection**  **[Intersection- ReferenceID]** | region [RoadRegulatorID] | - | - | - | - |
|  |  | id [IntersectionID] | - | - | - | - |
| A | **signalGroup**  **[SignalGroupID]** |  | 6 [sg31] | 6 [sg31] | 7 [sg32] | 7 [sg32] |
| B | **signalGroup**  **[SignalGroupID]** |  | 6 [sg31] | 18 [sg91] | 7 [sg32] | 19 [sg92] |
| C | **signalGroup**  **[SignalGroupID]** |  | 6 [sg31] | 7 [sg32] | 7 [sg32] | 6 [sg31] |
|  | **userClass**  **[RestrictionClassID]** |  | - | - | - | - |
|  | **connectionID**  **[LaneConnectionID]** |  | 8 | 9 | 10 | 11 |

Table 23 General ConnectsTo configuration in case of crosswalk

# Control-data:

## Sensors

The figure below shows a variety of sensors, such as induction loops (white) and push buttons (green). Configuring sensors involves three steps, namely:

1. Entering the properties of the sensor itself;
2. Linking the sensor to the lane on which it is physically located (if it is physically located on a lane, skip otherwise);
3. Linking the sensor to one or more lanes. For example, a sensor on an ingress lane can also be linked to egress lanes. Another example, detectors “D02.3” and “D03.3” can function as verification of detectors “D02.1”, “D02.2”, “D03.1” and “D03.2”. This is because traffic that passes the latter four detectors must also have passed the first two.

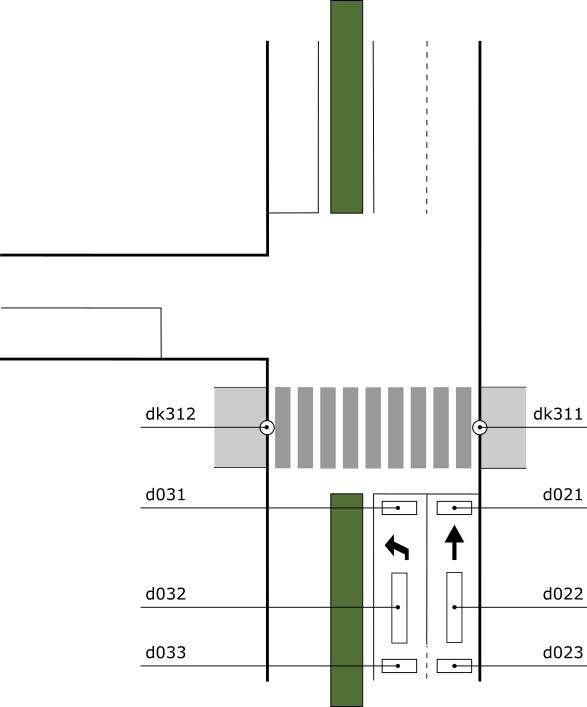


Figure 35 example sensors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data element | Sub-data element | D2.2 | DK31.1 | Comments |
| sensors  [SensorList] |  |  |  |  |
| sensor  [Sensor] |  |  |  |  |
| sensorID  [SensorID] |  | 1 | 2 |  |
| name  [DescriptiveName] |  | 2.2 | 31.1 |  |
| alias  [Alias] |  | D2.2 | DK31.1 | Optional in ITF 1.2 |
| sensorDeviceType  [SensorDeviceType] |  | inductionLoop | pushButton | (Enum = only one option available) |
| sensorOutput  [SensorOutput] |  | occupation | occupation | (Bitstring = more options available for the same detector) |
| vlogidx  [VlogIdx] |  | 4 | 30 | Optional in ITF 1.2 |
| sensorPosition  [Position] |  |  |  |  |
|  | lat [Latitude] | x | x |  |
|  | long [Longitude] | y | y |  |
| length [Length] |  | 10 | - | Optional in ITF 1.2 |
| width [Width] |  | 1 | - | Optional in ITF 1.2 |
| goShape  [GeoShape] |  | - | - | Optional in ITF 1.2 |
|  | indexpoint  [IndexedPosition] | - | - |  |
|  | index[Index] |  |  |  |
|  | lat [Latitude] | - | - |  |
|  | long [Longitude] | - | - |  |
| sensorAllocations  [SensorAllocationList] |  | - | - |  |
|  | sensorAllocation  [SensorAllocation] | - | - |  |
|  | laneID  [LaneID] | 1 | 3 |  |
|  | Ddstance  [LaneDistance] | 10 | - | Distance from stopline Optional in ITF 1.2 |
| sensorRelations  [SensorRelationList] |  |  |  |  |
|  | sensorRelation  [SensorRelation] | 1 |  |  |
|  | laneID  [LaneID] | 1 | 3 |  |
|  | purpose  [Purpose] | occupation | occupation | Optional in ITF 1.2 |
| gapTime  [GapTime] |  | 0 | - | Optional in ITF 1.2 |
| occupationTime  [OccupationTime] |  | 0 | 0 | Optional in ITF 1.2 |

Table 24 sensor examples

### Sensor allocation

Sensor allocation sets the lane(s) on which the sensor is located. It is possible to select more than one lane in case a sensor is physically located on multiple lanes. It is also possible to not select any lanes at all, in case a sensor is not physically located on any lane, for example sensors which are located on the conflict area. For these sensors the position data along with the sensor relation data can be used.

### Sensor relation

Sensor relation indicates the lanes a vehicle could use after passing this detector, but before crossing the intersection. Induction loops that are located farther away from the intersection (‘verweglussen’) do not provide information about the direction a vehicle will take farther downstream. Therefore, SensorRelations should contain a list of lanes that a vehicle could reach on the arm after passing the sensor. At the same time, vehicles located on induction loops located just before the stopline (‘koplussen’) often do not have the option to switch lanes anymore, which means SensorRelations would contain only one lane. Sensors which are located on the conflict area are not ‘allocated’ but only ‘related’, which indicates all manoeuvres which pass the sensor.

## Signal group relations

Signal group relations is marked as optional in ITF 1.2 but highly recommended. It contains a list of all conflicting signal groups and its clearance times that are protected and guarded in the TLC, thus not the clearance times used in the ITS Application. There are two different types of clearance times: protectedByClearance (green-yellow conflicts) and protectedByIntergreen (green-green conflicts). The clearance time types can be used together.

The signal group relations and clearance times can be used to – automatically – configure the guard of the TLC.

# Annex A: Bit string example

A bit string is an arbitrarily long array of bits. Specific bits can be identified by parenthesized integers and assigned names. As an example, the bit string for the data element LaneSharing is shown in Figure 36.

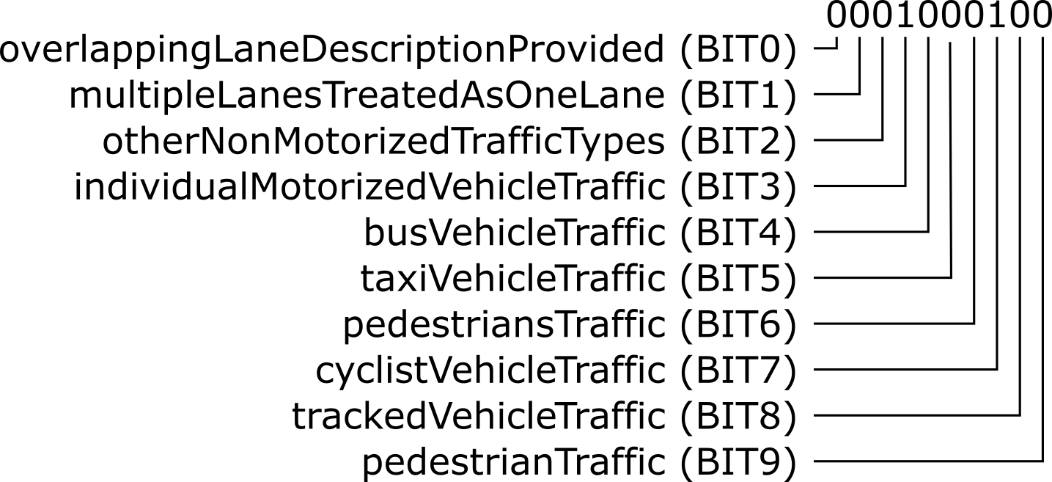


Figure 36 Bit string example

The example shows the 10 bit sting ‘0001000100’, where BIT3and BIT7 are set from left to right. This indicates that user types individualMotorizedVehicleTraffic and cyclistVehicleTraffic can access and use the respective lane.

# Annex C: Conversion code absolute – relative positions

//Copyright (c) 2017, Dynniq (www.dynniq.com)

//All rights reserved.

//

//Redistribution and use in source and binary forms, with or without

//modification, are permitted provided that the following conditions are met:

//

//1. Redistributions of source code must retain the above copyright notice, this

// list of conditions and the following disclaimer.

//2. Redistributions in binary form must reproduce the above copyright notice,

// this list of conditions and the following disclaimer in the documentation

// and/or other materials provided with the distribution.

//

//THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

//ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

//WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

//DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR

//ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

//(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

//LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND

//ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

//(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

//SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

package com.dynniq.geotools;

import java.text.DecimalFormat;

/\*\*

\* Class dealing with WGS84 locations

\*

\* @author eckoende (Eric Koenders, Dynniq)

\*

\*/

public class GeoPoint {

private double lon;

private double lat;

private static DecimalFormat df = new DecimalFormat("#.######");

public GeoPoint(double lon, double lat) {

this.lon = lon;

this.lat = lat;

}

public double getLon() {

return lon;

}

public void setLon(double lon) {

this.lon = lon;

}

public double getLat() {

return lat;

}

public void setLat(double lat) {

this.lat = lat;

}

public GeoPoint clone() {

return new GeoPoint(lon, lat);

}

public final static double EarthRadius = 6367000.0; // in meters

/\*\*

\* Calculate the distance between two points in meters.

\*

\* @param other the other GeoPoint

\* @return the geographic distance between this point and the other in meters

\*/

public double geodistance(GeoPoint other)

{

// convert to radians

double lon1 = Math.toRadians(this.lon);

double lat1 = Math.toRadians(this.lat);

double lon2 = Math.toRadians(other.lon);

double lat2 = Math.toRadians(other.lat);

// Haversine formula

double dlon = lon2 - lon1;

double dlat = lat2 - lat1;

double a = haversin(dlat) + Math.cos(lat1) \* Math.cos(lat2) \* haversin(dlon);

return EarthRadius \* haverasin(a);

}

/\*\*

\* Calculate the longitude difference between two point in meters.

\* A negative value is returned if the other point is to the west.

\* @param other the other GeoPoint

\* @return the geographic distance between this point and the other in meters

\*/

public double geodistance\_lon(GeoPoint other)

{

// convert to radians

double lon1 = Math.toRadians(this.lon);

double lat1 = Math.toRadians(this.lat);

double lon2 = Math.toRadians(other.lon);

// Haversine formula

double dlon = lon1 - lon2;

double a = Math.cos(lat1) \* Math.cos(lat1) \* haversin(dlon);

return EarthRadius \* haverasin(a) \* (dlon < 0 ? -1 : 1);

}

/\*\*

\* Calculate the latitude difference between two point in meters.

\* A negative value is returned if the other point is to the south.

\* @param other the other GeoPoint

\* @return the geographic distance between this point and the other in meters

\*/

public double geodistance\_lat(GeoPoint other)

{

// convert to radians

double lat1 = Math.toRadians(this.lat);

double lat2 = Math.toRadians(other.lat);

// Haversine formula

double dlat = lat1 - lat2;

double a = haversin(dlat);

return EarthRadius \* haverasin(a) \* (dlat < 0 ? -1 : 1);

}

/\*\*

\* @brief Move the longitude by the given distance

\* A negative value must be used when moving to the west.

\* @param distance The distance to offset the longitude in meters

\*/

public void geodisplace\_lon(double distance) {

double reflat = Math.toRadians(this.lat);

double reflon = Math.toRadians(this.lon);

double cosreflat = Math.cos(reflat);

double dlon = haverasin(haversin(distance / EarthRadius) / cosreflat / cosreflat);

if (distance < 0)

this.lon = Math.toDegrees(reflon - dlon);

else

this.lon = Math.toDegrees(reflon + dlon);

}

/\*\*

\* @brief Move the latitude by the given distance

\* A negative value must be used when moving to the south.

\* @param distance The distance to offset the latitude in meters

\*/

public void geodisplace\_lat(double distance) {

double reflat = Math.toRadians(this.lat);

double dlat = distance / EarthRadius;

this.lat = Math.toDegrees(reflat + dlat);

}

/\*\*

\* Haversine formula, see https://en.wikipedia.org/wiki/Haversine\_formula

\*

\* @param a

\* @return the haversine of a

\*/

public static double haversin(double a)

{

return Math.pow(Math.sin(a / 2), 2);

}

/\*\*

\* Inverse Haversine formula, see https://en.wikipedia.org/wiki/Haversine\_formula

\*

\* @param a

\* @return the haverasine of a

\*/

public static double haverasin(double a)

{

return 2 \* Math.asin(Math.min(1, Math.sqrt(a)));

}

public String toString() {

return "[" + df.format(lon) + "," + df.format(lat) +"] ";

}

public static void main(String args[]) {

/\* take a reference location \*/

GeoPoint refloc = new GeoPoint(5.420362, 52.173284);

/\* take a point \*/

GeoPoint pnt = new GeoPoint(5.420022, 52.173569);

/\* calculate the delta differences \*/

double deltax = pnt.geodistance\_lon(refloc);

double deltay = pnt.geodistance\_lat(refloc);

System.out.println("Refloc = " + refloc);

System.out.println("Point = " + pnt + " delta [x,y] = [" + deltax + ", " + deltay + "]");

/\* take a point at the reference location \*/

GeoPoint node = refloc.clone();

/\* move the point by a delta \*/

node.geodisplace\_lon(deltax);

node.geodisplace\_lat(deltay);

System.out.println("Node = " + node);

}

}

# Annex D: Members subWG NL profile

Jaap Vreeswijk – MAPtm

Martijn Harmenzon – MAPtm

Martin Barto – Vialis

Eric Koenders – Dynniq

Peter Luns – Siemens

Eddy Verhoeven – Siemens

Peter Smit – Swarco

Jaap Zee – Swarco

Kartik Mundaragi Shivakumar – DHDHV

Klaas-Jan op den Kelder – RHDHV

Wannes de Smet – BeMobile

Arie Schreuders – Sweco

Bram Schiltmans – RWS