IPG AUTOMOTIVE

Taking you to the next level

IPG Documentation

ROAD5

File Format Description
Version 1.0

SimulationSolutions • TestSystems • EngineeringServices

Table of Contents

Chapter 1

IPGRoad File Format

The IPGRoad file format is based on the IPG InfoFile format, therefore each road file should start with the following line:

#INFOFILE1.1 - Do not remove this line!

1.1 General Parameters

FileIdent = *product version*

Specifies the version of IPGRoad

Example `FileIdent = IPGRoad 5.0`

FileCreator = *string*

Optional. Specifies who, what and when the file was created

Example `FileCreator = roadutil 5.0.2, created on 2015-09-09 16:27:07`

NoOfLinks =	***totalNumberOfLinks***
NoOfJunctions =	***totalNumberOfJunctions***
NoOfRoutes =	***totalNumberOfRoutes***
RoadNetworkLength =	***totalLengthOfAllLinks***
RouteLength.<routeId> =	***lengthOfRoute***
XYRange =	***minX maxX minY maxY***

Optional. Specifications which can be read and shown by the CarMaker GUI:

GCS = *mode x y z lat lon elev*

Optional. Used method and reference points for converting geocoded road data from WGS84 system to the IPGRoad internal x,y,z coordinate system.
*mode*FlatEarth, (GaussKruger, NOT YET SUPPORTED)
*x,y,z*reference position in the IPGRoad coordinate system (Units [meters] [meters] [meters]).
*lat,lon,elev*reference position in in WGS84 coordinates (Units [deg] [deg] [meters])

Example
```
GCS = FlatEarth 0 0 0 49.99477 8.37352 111
```

RST = *urban countryroad motorway roundabout ramp dirttrack user undefined*

Optional. Specifies speed limits in meters/s for the available road types.

Example
```
RST = 13.8889 27.7778 -1 8.33333 19.4444 2.777 -1 -1
```

1.2 Junction Parameters

Junctions are used to connect *links.* Junctions are placed at a x,y,z position and can define up to 6 junction arms

The identifiers (*id*) for all junctions given in the infofile must increase strictly monotone.

Junction.<id>.Knot = *x y z*

Junction position in road coordinate system (Units [meters] [meters] [meters]).

Example
```
Junction.0.Knot = 0 0 0
```

Junction.<id>.ArmAlpha = *alpha0 alpha1 alpha2 [alpha3 [alpha4 [alpha5]]]*

This defines the number and angles of junction arms. The angles are in deg and given with respect to the x-axis.
The junction arms must be defined sequencially in the counter clockwise direction.
Min. 2 to max. 6 angles for the junction arms can be defined.

Example `Junction.2.ArmAlpha = 0.0000 90.0000 180.0000 270.0000`

Junction.<id>.ArmRadius = *r0 r1 [r2 [r3 [r4 [r5]]]]*

Optional. Defines the transition radius in meters between two consecutive junction arms. eg. r0 defines radius between alpha0 and alpha1.
The number of radii must match with the number of junction arms.
Default: 1.5

Example `Junction.2.ArmRadius = 1 2 2 1`

Junction.<id>.ArmLength = *d0 d1 [d2 [d3 [d4 [d5]]]]*

Optional. Junction arm length defines the distance between junction knot and junction entry for each arm in meters.
Default: 1.5

Example `Junction.2.ArmLength = 10 8 8 9`

Attention: If neither the ArmRadius nor ArmLength are specified, then the first or the last segment of the connected link will be used as the distance between junction knot and junction entry.

Junction.<id>.MainArms = *arm0 arm1*

Defines the main path represented by the junction arms along the intersection.

Example `Junction.2.MainArms = 1 3`

Junction.<id>.RST = *rst*

Optional. Specifies a certain road speed type for the junction. Default: undefined

rst urban,countryroad,motorway,roundabout,ramp,dirttrack,user,undefined

Example `Junction.2.RST = roundabout`

Junction.<id>.Friction = *friction*

Optional. Specifies a friction for the junction area. Default: 1

Example `Junction.2.Friction = 1`

1.3 Link Parameters

A link is defined by at least one segment. The segments are describing the x,y-course of the link. The link endpoints are called node0 and node1.

Please note that all s-coordinate reports should have at least four decimal places!

Link.<id>.Junctions = *junc0Id arm0Id junc1Id arm1Id*

Specifies the connected junctions at link node0 and node1. The arm id is the id of the junction arm of the connetced junction. If the link node0/1 is unconnected, set junc0/1Id and arm0/1Id to -1.

Example `Link.0.Junctions = -1 -1 0 2`

Example `Link.1.Junctions = 0 1 -1 -1.`

Link.<id>.Node0 = *x y z ang*

Defines the x,y,z position of node0 and the start direction of the link. If node0 is connected to a junction, this key will be ignored (Units [meters] [meters] [meters] [deg]).

Link.<id>.Node1 = *x y z ang*

Defines x,y,z position of node1 and the end direction of the link.
This key is only needed if node1 is unconnected and the segment list contains a segment of type ST_Connect (Units [m] [m] [m] [deg]).

Link.<id>.Friction = *friction*

Optional. Specifies a friction for the link. Default: 1

Example `Junction.2.Friction = 1`

Link.<id>.Seg.<segId>.Type = *type*

Defines the type of segment added to a link

*type*Straight, TurnLeft, TurnRight, ClothLeft, ClothRight, PointList, File, Connect

Link.<id>.Seg.<segId>.Type = type

Link.<id>.Seg.<segId>.Param = *p0 p1 p2 ... p7*

The meaning of the parameters *p* is depending on the segment type (see Table 1.10:).

Link.<id>.Seg.<segId>.File = *fname*

Optional. Specifies an external file for the x,y profile of the reference line.

Supported file formats are:
- former ipg road data files, binary or ascii
- kml files as created by route export from google earth (Measured converted data might not work)
- OpenCRG data files

Altitude or lateral slope data in the file may overwrite given elevation or slope profiles. Grid data as given by OpenCRG is also considered.

Link.<id>.Seg.<segId>.FileAddParam = *tType useMesh useElev useSlope useCamber rampU0 rampU1 rampV0 rampv1 tOff uScale vScale zScale*

Optional. Additional parameters to handle external file data (OpenCRG)
*tType*transition type, see Table 1.1:
*useMesh*use available grid data
*useElev*use available eleavtion data
*useSlope*use available slope data
*useCamber*use available camber data
*rampU0*extension in front of grid data to ramp z from zero to first longitudinal grid value
*rampU1*extension at the of grid data to ramp z from last longitudinal grid data to zero
*rampV0*extension right of grid data to ramp z from zero to first lateral grid value
*rampV1*extension left of grid data to ramp z from last lateral grid data to zero
*tOff*lateral offset of grid data reference line to link reference line
*uScale*longitudinal scaling factor for grid data
*vScale*lateral scaling factor for grid data
*zScale*scaling factor for grid z values

Link.<id>.Seg.<segId>.File = *fname*

Optional. Specifies an external file for the x,y profile of the reference line.

Supported file formats are:
- former ipg road data files, binary or ascii
- kml files as created by route export from google earth (Measured converted data might not work)
- OpenCRG data files

Altitude or lateral slope data in the file may overwrite given elevation or slope profiles. Grid data as given by OpenCRG is also considered.

Link.<id>.Seg.<segId>.FileAddParam = *tType useMesh useElev useSlope useCamber rampU0 rampU1 rampV0 rampv1 tOff uScale vScale zScale*

Optional. Additional parameters to handle external file data (OpenCRG)
*tType*transition type, see Table 1.1:
useMeshuse available grid data
useElevuse available eleavtion data
useSlopeuse available slope data
useCamberuse available camber data
rampU0extension in front of grid data to ramp z from zero to first longitudinal grid value
rampU1extension at the of grid data to ramp z from last longitudinal grid data to zero
rampV0extension right of grid data to ramp z from zero to first lateral grid value
rampV1extension left of grid data to ramp z from last lateral grid data to zero
tOfflateral offset of grid data reference line to link reference line
uScalelongitudinal scaling factor for grid data
vScalelateral scaling factor for grid data
zScalescaling factor for grid z values

Link.<id>.Seg.<segId>.PointList:
x0 y0
x1 y1
. .
. .
xn yn

Indicates the points list for the segment type *PointList*
It defines the x and y coordinates in meters.

Example
```
Link.0.RoadMarking.3.PointList:
   0 1.5
   0 1.2
   1.5 -1.4
   2.5 -1.5
```

Z Profile:

Link.<id>.ElevationProfile:
s0 z0 dz0
s1 z1 dz1
. . .
. . .
sn zn dzn

Optional. The height profile is specified along the s-coordinate of the reference line. For each support point, a slope can be specified with dz. A spline defined by the support points is set to -999 dz. If no value is specified for s = 0 or s = *length of link* values are adopted either from the following intersection or possibly the first / last entry for the start / end. Z-values for s = 0 and s = *length of link* should match with z-values of intersections.

Example

```
Link.0.ElevationProfile:
    0.0000 5.0000 -999.0000
    68.0000 1.5000 -999.0000
```

Link.<id>.SlopeProfile:
s0 q0 dq0
s1 q1 dq1
. . .
. . .
sn qn dqn

Optional. The slope profile is specified along the s-coordinate of the reference line. The q-value corresponds to dz / dt and is in meters. i.e. for 10% slope, q = 0.1. A spline is used for the slope s-curve if *dq* is set to -999.

Example

```
Link.0.SlopeProfile:
    0.0000 0.0200 0.0000
    30.0000 0.0000 0.0000
```

Link.<id>.CamberProfile:
s0 c0 dc0
s1 c1 dc1
. . .
. . .
sn cn dcn

Optional. The camber profile is specified along the s-coordinate of the reference line.
A spline is used for the camber s-curve if *dc* is set to -999.

Example
```
Link.0.CamberProfile:
   0.0000 -0.0050 -999
   25.0000 -0.0150 -999
```

Lanes:
Optional: Default is one lane for each side of the road.
A lane section describes a section with a constant number of lanes on either side of the reference line.
The link lane section ids must start with 0 and increase strictly monotone.

Link.<id>.LaneSection.<lsId>.Start = *start*

Start specifies at which s-coordinate the lane section begins. A lane section ends with the start of a new lane section or at the end of a link.

Example
```
Link.1.LaneSection.0.Start = 0
```

Link.<id>.LaneSection.<lsId>.Lane<*side*>.<*laneId*> = *tType w0 w1 lType u0 u1 u2*

Defines a lane for a lane section. LaneIds must start with 0 and increase strictly monotone.
side L (Left), R (Right)
*laneId*0-9. Only Integers allowed. Note: roadside bumps need own lane
*tType*transitiontype for width (see Table 1.1:)
*w0*lane start width
*w1*lane end width
*lType*lane type (see Table 1.4:)
*u0,u1,u2*currently inused

Example
```
Link.1.LaneSection.0.LaneL.0 = 0 3 0 0 0 0 0
```

Link.<id>.LaneSection.<lsId>.Lane<*side*>.<*laneId*>.Width:
s0 lonRef0 tType0 wa0 wb0 wc0 wd0
s1 lonRef1 tType1 wa1 wb0 wc1 wd1
. . . .
. . . .
sn lonRefn tTypen wan wbn wcn wdn

Table with width specifications for lane:
*s*s coordinate for width with respect to the start/end of the lane section
*lonRef*longitudinal reference, see Table 1.2:
*tType*transition type for width, see Table 1.1:
*wa*width
*wb,wc,wd*extra polynomial coefficients (only read if tType == 3)

Example
```
Link.1.LaneSection.0.LaneL.0.Width:
   8 0 0 3 0 0 0
   23 0 0 3 0 0 0
   58 0 0 0 0 0 0
   0 1 3 0 0 0 0
```

Link.<id>.LateralOffset:
s0 lonR0 tType0 t0 dt0
s1 lonR1 tType1 t1 dt1
. . . . .
sn lonRn tTypen tn dtn

Link.<id>.LaneSection.<lsId>.LateralOffset:
s0 lonR0 tType0 t0 dt0
s1 lonR1 tType1 t1 dt1
. . . . .
sn lonRn tTypen tn dtn

Lateral offset can be defined relativ to a link or lane section.

ss-coordinate
lonRlongitudinal reference, see Table 1.2:
tType transition type, see Table 1.1:
tlateral offset
dtlateral derivation with respect to s (dt/ds)

Marker:

Link.<id>.Marker.<mId>.Type = *type*

Marker ids (mId) must start with 0 and increase strictly monotone.
*type*DrvSpeed, DrvStop, DrvPylon, EnvWind, DrivManTrigger, DrivManCmd, DrivManJump

Please Note:
IPGRoad Version 5.1 and below need the following types:
typeDrvSpeed, DrvStop, Pylon, Wind, DrivManTrigger, DrivManCmd, DrivManJump, Bridge, Tunnel

For compatibility reasons these types will be supported at least up to IPGRoad version 6.0, any further support is not guaranteed.
To define bridges and tunnels with version 5.1.1 and newer please see *Link.<id>.Bridge* or *Link.<id>.Tunnel.*

Marker-Dependent Parameter List:

Link. <Id> .Marker. <mId> .param = *s0 lonR0 s1 lonR1 t LatR invisible dir speed unit*

DrvSpeed:
*s0*start position of speed limit
*lonR0*integer value for longitudinal reference of s0 (see Table 1.2:)
*s1*end position of speed limit
*lonR1*integer value for longitudinal reference for s1 (see Table 1.2:)
*t*lateral offset
*latR*integer value for lateral reference for offset (see Table 1.3:)
*invisible*movie visibility flag: 0 visible, 1 not visible
*dir*effective direction: 1 in link direction, -1 counter link direction
*speed*speed limit
*unit*string for unit of speed: kmh or mph

Example `Link.1.Marker.0.Param = 0 2 140 2 0.5 1 1 -1 70 kmh`

Link. <Id> .Marker. <mId> .param = *s lonR t LatR invisible dir time*

DrvStop:
*s*DrvStop position
*lonR*integer value for longitudinal reference of s (see Table 1.2:)
*t*lateral offset
*latR*integer value for lateral reference for offset (see Table 1.3:)
*invisible*movie visibility flag: 0 visible, 1 not visible
*dir*effective direction: 1 in link direction, -1 counter link direction
*time*stop time in seconds

Example `Link.4.Marker.0.Param = 1 2 0.2 1 1 -1 1`

Link.<id>.Marker.<mId>.Param = *s lonR t latR invisible dir width*

DrvPylon:
*s*pylon position
*lonR*integer value for longitudinal reference of s (see Table 1.2:)
*t*lateral offset
*latR*integer value for lateral reference for offset (see Table 1.3:)
*invisible*movie visibility flag: 0 visible, 1 not visible
*dir*effective direction: 1 in link direction, -1 counter link direction
*width*pylon clearance width

Link.<id>.Marker.<mId>.Param = *s lonR dir mode id*

DrivManTrigger:
*s*trigger position
*lonR*integer value for longitudinal reference of s (see Table 1.2:)
*dir*effective direction: 1 in link direction, -1 counter link direction
*mode*mode = 1: define new reference point, Mode = 2: take all measurements, mode = 3: take all measurements and define new reference point.
*id*this is a positive integer value and should be numbered consecutively.

Link.<id>.Marker.<mId>.Param = *s lonR dir expr*

DrivManCmd:
s DrivManCmd position
*lonR*integer value for longitudinal reference of s (see Table 1.2:)
*dir*effective direction: 1 in link direction, -1 counter link direction
*expr*command string

Link.<id>.Marker.<mId>.Param = *s lonR dir label*

DrivManJump:
s DrivManJump position
*lonR*integer value for longitudinal reference of s (see Table 1.2:)
*dir*effective direction: 1 in link direction, -1 counter link direction
*label*jump label

Link.<id>.Marker.<mId>.Param = *s0 lonR0 s1 lonR1 invisible dir speed angle unit*

EnvWind:
*s0*start position of wind
*lonR0*integer value for longitudinal reference of s0 (see Table 1.2:)
*s1*end position of wind
*lonR1*integer value for longitudinal reference for s1 (see Table 1.2:)
*invisible*movie visibility flag: 0 visible, 1 not visible
*dir*effective direction is always in both direction, here the dir flag only indicates the position of the wind symbols in the movie
*speed*wind speed
*angle*absolute angle in deg with respect to the x-axis
*unit*unit string for the unit of the given speed value, allowed are *ms* or *kmh*

Link.<id>.Marker.<mId>.Param = *s0 lonR0 s1 lonR1 h bwl bwr ang0 ang1 type mat [pillar0 [pillar1]]*

Bridge/Tunnel:
*s0*bridge/tunnel start position
*lonR0*integer value for longitudinal reference of s0 (see Table 1.2:)
*s1*bridge/Tunnel end position
*lonR1*integer value for longitudinal reference for s1 (see Table 1.2:)
*h*height in meters.
*bwl*distance to the left of road in meters.
*bwr*distance to the right of road in meters.
*ang0*angle in degrees with s0 (Currently without effect).
*ang1*angle in degrees with s1 (Currently without effect).
*type*string, currently only "Standard"
*mat*texture
*pillar0*optional. Bridge only. String: "None" Or "Standard" (Default "Standard")
*pillar1*optional. Bridge only. String: "None" Or "Standard" (Default "Standard")

The optional parameters *pillar0* and *pillar1* allow to suppress the illustration of the bridge pillars in IPGMovie.

Example
```
Link.11.Marker.0.Param = 5 0 35 0 6 0 0 0 0 Standard "Concrete_1.jpg"
None None
```

Bumps:

Link. <Id> .Bump. <bId> .Type = *<type>*

Bump ids (bId) must start wit0 and increase strictly monotone
typeFriction, Beam, Wave, Cone, Mesh, LatProfile

Please note:
IPGRoad Version 5.1 and below need the following types:
*type*Friction, Beam, Wave, Cone, Mesh, LatProfileRSL, LatProfileRSR

Link. <Id> .Bump. <bId> .Material = *mat*

Optional. Specifies a texture file or a color for the bump

Example `Link.0.Bump.0.Material = 1,0,0,0.5`

Example `Link.0.Bump.0.Material = MyTexture.png`

Bump dependent Parameter list:

Link.<id>.Bump.<bId>.Param = *s0 lonR0 s1 lonR1 t latR reg fric ang width*

Friction:
s0 friction start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 friction end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
reg 1 for rectangular, 0 for following reference line
fric friction
ang only when reg = 1; defines rotation relative to the reference line
width width of friction strip.

Link.<id>.Bump.<bId>.Param = *s lonR t latR reg fric ang height lUp lTop lDown width*

Beam:
s beam start position
lonR integer value for longitudinal reference of s0 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
reg 1 for rectangular, 0 for following reference line
fric local friction
ang only used if reg = 1; defines rotation relative to the reference line
height height over street
lUp length of ramp up
lTop length of top
lDown length of ramp down
width width

Link. <Id> .Bump. <bId> .Param = *s0 lonR0 s1 lonR1 t latR fric ang amplitude width waveLen rampLT rampRT rampOnS rampOffS*

Wave:
s0 wave start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 wave end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
fric friction
ang only if reg = 1; defines rotation relative to the reference line
amplitude wave amplitude
width width
waveLen wave length
rampLT/rampRT lateral transition length on left / right
rampOnS/rampOffS longitudinale transition length

Link. <Id> .Bump. <bId> .param = *s lonR t latR reg fric ang height rS0 rT0 rS1 rT1*

Cone
s cone center position
lonR integer value for longitudinal reference of s0 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
reg 1 for rectangular, 0 for following reference line
fric local friction
ang only if reg = 1; defines rotation relative to the reference line
height height over street
rS0 base cone longitudinal radius
rT0 base cone lateral radius
rS1 top cone longitudinal radius
rT1 top cone lateral radius

Link.<id>.Bump.<bId>.Param = *s lonR t latR reg fric ang nU nV [zScale [uScale [vScale]]]*

Mesh:
s mesh start position
lonR integer value for longitudinal reference of s0 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
reg 1 for rectangular, 0 for following reference line
fric local friction
ang only if reg = 1; defines rotation relative to the reference line
nU,nV number of nodes in u, v.
zScale optional z scale factor (Default: 1)
uScale optional u scale factor (Default: 1)
vScale optional v scale factor (Default: 1)

Link.<id>.Bump.<bId>.Param = *s0 lonR0 s1 lonR1 t latR mirror rampOnS rampOnTType rampOffS rampOffTType fric*

LatProfile (LatProfileRSL, LatProfileRSR for version 5.1 and lower)
s0 longitudinal profile start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 longitudinal profile end position
lonR1 integer value for longitudinal reference of s1 (see Table 1.3:)
t lateral offset (unused)
latR integer value for lateral reference for offset
(only -1 and 1 allowed, see Table 1.3:)
(no meaning for version 5.1 and lower)
mirror flips the profile on the s-axis
rampOnS transition length at start in s direction
rampOnTType start transition type (see)
rampOffS transition length at end in s direction
rampOffTType end transition type (see)
fric local friction

Link.<id>.Bump.<bId>.Material.sTex = *sTex*

Optional. Allows to set the lateral profile texture length in s direction (Default:1)

Link.<id>.Bump.<bId>.Profile:

dt0	**dz0**
dt1	**dz1**
.	**.**
.	**.**
.	**.**
dtn	**dzn**

Defines the lateral profile in t-direction: *dt*-values define the distance between two nodes and *dz*-values stand for the z difference to the road border.

Link.<id>.TreeStrip.<treeId> = *s0 lonR0 s1 lonR1 t latR width spacing scaleH scaleV randH randW*

Tree strip ids must start with 0 and increase strictly monotone.
s0 tree stript start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 tree strip end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.3:)
t lateral offset
width width of Tree strip
spacing average spacing between 2 trees in meters
scaleH, scaleV horizontal and vertical scaling of trees
randH,randW random factors for tree size variety

Example
```
Link.1.TreeStrip.3 = 55 2 0 3 -5.25 -1 50 3 2.1 1.9 0.1 0.1
```

Link.<id>.Bump.<bId>.Param = *s0 lonR0 s1 lonR1 t latR mirror rampOnS rampOnTType rampOffS rampOffTType fric*

LatProfileRSL,LatProfileRSR:
s0 lateral profile start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 lateral profile end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset (currently unused)
latR integer value for lateral reference for offset (see Table 1.3: , currently only -1 or 1 allowed)
mirror flips the profile on the s-axis
rampOnS transition length at start in s direction
rampOnTType start transition type, see Table 1.1:
rampOffS transition length at end in s direction
rampOffTTyp end transition type, see Table 1.1:
fric local friction

Link.<id>.Bump.<bId>.Profile:
dt0 dz0
dt1 dz1
. .
dtn dzn

LatProfileRSL,LatProfileRSR only. List of relative distances and z-values starting at the roadside.

SensorObjects:

Link.<id>.GuidePost.<*gpId*> = *s0 lonR0 s1 lonR1 t latR spacing*

Guide post ids must start with 0 and increase strictly monotone.
s0 guide post start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 guide post end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
spacing distance between the guide post

Example
```
Link.0.GuidePost.1 = 0 0 0 1 0.5 -1 50
```

Link.<id>.GeoObjects.<*goId*> = *s0 lonR0 s1 lonR1 t latR spacing rel2Ground rotX rotY rotZ scaleX scaleY scaleZ zOffset file*

Geometry object ids must start with 0 and increase strictly monotone.
s0 geometry object start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 geometry object end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
spacing distance between the guide post
rel2Ground 1 -> Orientation perpendicular to the ground
0 -> Vertical orientation
rotX, rotY, rotZ twist angles in deg
scaleX, scaleY,scaleZ scaling factors
zOffset offset in Z direction
file file containing geometry object

Link.<id>.RoadPainting.<*rpId*> = *s lonR t latR u v ang file*

Road painting ids must start with 0 and increase strictly monotone.
s road painting position
lonR integer value for longitudinal reference for s (see Table 1.2:)
t t offset (t<0 -> Left)
l*atR* integer value for lateral reference for t (see Table 1.3:)
u,v image size (height, width -> s, t)
ang twist angle in deg
file texture file

Link.<id>.SignPlate.<*spId*> = *s lonR t latR height width ang poleHeight poleDist file*

Sign plate ids must start with 0 and increase strictly monotone.
s sign plate position
lonR integer value for longitudinal reference for s (see Table 1.2:)
t t offset (t<0 -> Left)
l*atR* integer value for lateral reference for t (see Table 1.3:)
width, height image size
ang angle of twist [deg]
poleHeight height over the ground
poleDist distance between poles, if 0 only one pole is created
file texture file

Link.<id>.RoadMarking<rmId> = s0 lonR0 s1 lonR1 t latR width height type kind userId dash space detectability visibility priority mat

Road marking ids must start with 0 and increase strictly monotone.
s0 road marking start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 road marking end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference (see Table 1.3:)
width width of road marking
height height (No physical effect, but can be 'seen' by the line sensor)
type line type (see Table 1.5:)
kind kind (see Table 1.9:)
userId custom indexing
dash length of dashes in dashed lines
space length of spaces between dashes in dashed lines
detectability factor for probability of detection (Value 0..1)
visibility controls whether geometry is visible in IPGMovie
priority controls visibility in overlapping lines
mat specifies texture file or color

Example
```
Link.2.RoadMarking.1 = 0 2 33 2 0 0 0.12 0 2 0 0 2 2 1 1 2 ""
```

Link.<id>.RoadMarking.<rmId>.Pointlist:
s0 t0
s1 t1
. .
sn tn

Pointlist with s,t coordinates needed if RoadMarking kind is *spline.*

Link.<id>.TrfBarrier.<*tbId*> = *s0 lonR0 s1 lonR1 t latR width height type kind interval detectability visibility mat0 mat1*

Traffic barrier ids must start with 0 and increase strictly monotone.
s0 traffic barrier start position
lonR0 integer value for longitudinal reference of s0 (see Table 1.2:)
s1 traffic barrier end position
lonR1 integer value for longitudinal reference for s1 (see Table 1.2:)
t lateral offset
width width of Roadmarking
height height (no physical effect, but can be 'seen' by the line sensor)
type traffic barrier type (see Table 1.6:)
kind kind (see Table 1.9:)
interval meaning depends on *type*
detectability factor for probability of detection (value 0..1)
visibility controls whether geometry is visible in the IPGMovie
priority controls visibility in overlapping lines
mat0 specifies texture file or color
mat1 specifies texture file or color

Example
```
Link.2.TrfBarrier.1 = 0 2 33 2 0 0 0.12 0 2 0 0 2 2 1 1 2 ""
```

Link.<id>.TrfBarrier.<tbId>.Pointlist:
s0 t0
s1 t1
. .
sn tn

Pointlist with s,t coordinates needed if TrfBarrier kind is *spline.*

Link.<id>.Mount.<mountId> = *s lonR t latR nPart type rotX rotY rotZ*

Mount ids must start with 0 and increase strictly monotone.
s mount position
lonR integer value for longitudinal reference of s0 (see Table 1.2:)
t lateral offset
latR integer value for lateral reference for offset (see Table 1.3:)
nPart number of parts mounted to this mount
type mount type
rotX,rotY,rotZ twist angles in deg

Link.<id>.Mount.<mountId>.<*partId*> = *0 hOff vOff rzOff latR facing detectability mainsign mainsize mainattr mainval0 mainval1 isTwoSided sub0sign sub0size sub0attr sub0val0 sub0val1 sub1sign sub1size sub1attr sub1val0 sub1val1*

Part ids must start with 0 and increase strictly monotone.
First parameter is 0 -> Traffic Sign

*hOff*horizontal offset which depends on mount type
*vOff*vertical offset which depends on mount type
*rzOff*rotation relative to mount
*latR*lateral reference
*facing*acts in link direction or direction opposite to link
*detectability*probability factor for sign detection

main,sub0,sub1:
sign name of the sign
size string specifying the size according to the defintions in the sign file
*attr*additional attributes
*val0*optional value is displayed on the label
*val1*otional value is displayed on the label

*isTwoSided*Specifies whether the sign is diplayed on both the sides or not

Example
```
Link.4.Mount.1.1 = 0 0 2.1 0 -1 1 1 SpeedLimit M "" 30 0 0 - - - 0 0 0 - - - 0 0 0
```

Link.<id>.Mount.<mId>.<partId>.Size = *<height> <width>*

Optional. Indicates overall height and width of the sign (including sub signs). The size affects the positioning of the sign on the mount.

Link.<id>.Mount.<mId>.<*partId*> = *1 hOff vOff rzOff name latR facing type startcond initialphase isDrvCtrl timing0 timing1 timing2 timing3 timing4>*

Part ids must start with 0 and increase strictly monotone.
First parameter is 1 -> Traffic Light

*hOff*horizontal offset which depends on Mount Type
*vOff*vertical offset which depends on Mount Type
*rzOff*rotation relative to mount
*latR*lateral reference (see Table 1.3:)
*facing*acts in link direction or direction opposite to link
*type*traffic light type (see Table 1.8:)
*startcond*text with initial condition
*initialphase*signal phase at the start of simulation
*isDrvCtrl*defines whether traffic light is recognized or ignored by drivers
*timingx*traffic signal timing

Link.<id>.Bridge.<bridgeId>= *s0 lonR0 s1 lonR1 h bwl bwr ang0 ang1 type mat [pillar0 [pillar1]]*

Bridge ids must start with 0 and increase strictly monotone.
*s0*bridge start position
*lonR0*integer value for longitudinal reference of s0 (see Table 1.2:)
*s1*bridge end position
*lonR1*integer value for longitudinal reference for s1 (see Table 1.2:)
*h*height in meters.
*bwl*distance to the left of road in meters.
*bwr*distance to the right of road in meters.
*ang0*angle in degrees with s0 (currently without effect).
*ang1*angle in degrees with s1 (currently without effect).
*type*string, currently only "Standard"
*mat*texture
*pillar0*optional. string: "None" Or "Standard" (Default "Standard")
*pillar1*optional. string: "None" Or "Standard" (Default "Standard")

The optional parameters *pillar0* and *pillar1* allow to suppress the illustration of the bridge pillars in IPGMovie.

```
Link.11.Bridge.0 = 5 0 35 0 6 0 0 0 0 Standard "Concrete_1.jpg" None None
```

Link.<id>.Tunnel.<tunnelId>= *s0 lonR0 s1 lonR1 h bwl bwr ang0 ang1 type mat*

Tunnel ids must start with 0 and increase strictly monotone.
*s0*tunnel start position
*lonR0*integer value for longitudinal reference of s0 (see Table 1.2:)
*s1*tunnelend position
*lonR1*integer value for longitudinal reference for s1 (see Table 1.2:)
*h*height in meters.
*bwl*distance to the left of road in meters.
*bwr*distance to the right of road in meters.
*ang0*angle in degrees with s0 (Currently without effect).
*ang1*angle in degrees with s1 (Currently without effect).
*type*string, currently only "Standard"
*mat*texture

```
Link.11.Tunnel.0 = 5 0 35 0 6 0 0 0 0 Standard "Concrete_1.jpg"
```

1.4 Route Parameters

Route.<*id*>:
linkId0
linkId1
.
.
linkIdn

Route id must start with 0 and increase strictly monotone. A Route is defined as a series of links represented by their ids.

Example

```
Route.0:
    4
    1
    3
```

Route. <id> .Name = *<name>*

Optional. Allows to define a name.

Route. <id> .reverse = *rev*

If only one link exists in a route, then *rev* specifies the direction in which the link is passed .

Route.<id>.Path.<secId0>:
s0 latR0 tOff0
s1 latR1 tOff1
. . .
si latRi tOffi

Route.<id>.Path.<secId1>:
s0 latR0 tOff0
s1 latR1 tOff1
. . .
sj latRj tOffj

Route.<id>.Path.<secId2>:
s0 latR0 tOff0
s1 latR1 tOff1
. . .
sk latRk tOffk

Path definition is required for each specified link. A section consists of a link with subsequent intersection. Therefore the number of sections must be the same as the number of route links.

ss-coordinate start relative to section start
latRlateral reference (see Table 1.3:)
tOffoffset in t-direction

Example

```
Route.0.Path.0:
    0.0000 0 0
    45.0000 0 0.5
    85.0000 0 -0.3
    95.0000 0 0
Route.0.Path.1:
    0.0000 -3 0
    350.0000 -3 0
    372.0000 -3 0
    392.0000 -2 0
    408.0000 -2 0
Route.0.Path.2:
    0.0000 -2 0
```

Appendix A

Numerical Codes

Table 1.1: Transition Types

	meaning	c-api enum	
0	cubic interpolation	TT_Cubic	
1	linear interpolation	TT_Linear	
2	keep last value	TT_Step	
3		TT_None	

Table 1.2: Longitudinal References

	meaning	c-api enum	
0		LR_Node0	
1		LR_JE0	
2		LR_Node1	
4		LR_JE1	

Table 1.3: Lateral References

	meaning	**c-api enum**	
-11	right lane 9	LR_R9	
-10	right lane 8	LR_R8	
:	:	:	
-3	right lane 1	LR_R1	
-2	right lane 0	LR_R0	
-1	right border	LR_Right	
0	center line	LR_Center	
1	left border	LR_Left	
2	left lane 0	LR_L0	
3	left lane 1	LR_L1	
:	:	:	
10	left lane 8	LR_L8	
11	left lane 9	LR_L9	

Table 1.4: Lane Types

	meaning	**c-api enum**	
0	drivable road	LT_Road	
4	border lane	LT_Border	
5	road side	LT_RoadSide	
10	bicycle lane	LT_Bicycle	
11	pedestrian lane	LT_Pedestrian	
12	traffic island	LT_TrafficIsland	
13	parking area	LT_ParkingArea	

Table 1.5: Line Types

	meaning	**c-api enum**	
0		LT_UnknownLine	
1		LT_Line	
2		LT_LineDshd	
3		LT_LineSqrDot	
4		LT_DblLine	
5		LT_DblLineDshd	
6		LT_DblLineSqrDot	
7		LT_DblLineDshdL	
8		LT_DblLineDshdR	
9		LT_DblLineSqrDotL	
10		LT_DblLineSqrDotR	

Table 1.5: Line Types

	meaning	c-api enum	
11		LT_RoadBoundary	internal use only
12		LT_GuardRail	internal use only

Table 1.6: Traffic Barrier Types

	meaning	c-api enum	
0		TBT_Unknown	
1	single guard rail	TBT_GuardRail1	
2	double guard rail	TBT_GuardRail2	
3	triple gaurd rail	TBT_GuardRail3	
4	concrete barrier	TBT_JerseyBarrier1	
5	concrete barrier	TBT_JerseyBarrier2	
6		TBT_Wall1	
7		TBT_Wall2	

Table 1.7: Mount Types

	meaning	c-api enum	
0		MNT_None	
1		MNT_Pole	
2		MNT_DblPole	
3		MNT_Frame	
4		MNT_Road	
5		MNT_PoleTL	
6		MNT_Gantry	
7		MNT_HalfGantry	
8		MNT_GantryTL	
9		MNT_HalfGantryTL	

Table 1.8: Traffic Light Types

	meaning	c-api enum[a]	
0		TLT_200_RYG	
1		TLT_200_RYG_Straight	
2		TLT_200_RYG_Left	
3		TLT_200_RYG_Right	
4		TLT_200_RYG_StraightLeft	
5		TLT_200_RYG_StraightRight	
6		TLT_100_RYG	
7		TLT_200_RY	

Table 1.8: Traffic Light Types

	meaning	c-api enum[a]	
8		TLT_200_YG	
9		TLT_200_YG_Left	
10		TLT_200_YG_Right	
11		TLT_300_YG_Left	
12		TLT_300_YG_Right	
13		TLT_300_R	
14		TLT_Ped_R	
15		TLT_Pedestrian	

a. Notation: TLT_<diameter in mm>_<top down colors present>[_<direction>]

Table 1.9: Line Kinds (used by RoadMarkings and Traffic Barriers)

	meaning	c-api enum	
0		LK_Offset	
1		LK_Polyline	
2		LK_Spline	
3		LK_Unknown	

Table 1.10: Meaning of segment parameters with respect to segment type

p	Straight	Turn-Left	Turn-Right	Cloth-Left	Cloth-Right	PointList	File	Connect
0	len	radius	radiuse	start radius	start radius	suv1x	isAbs	0
1	0	angle	angle	end radius	end radius	suv1y	smooth xy (0..1)	0
2	0	0	0	angle	angle	isAbs	smooth z (0..1)	0
3	0	0	0	0	0	smooth XY (0..1)	smooth Q (0..1)	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	zScale	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0