ETSI TS 103 544-25 V1.3.1 (2019-10)

Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 25: Navigation Meta Data Service

CAUTION

The present document has been submitted to ETSI as a PAS produced by CCC and approved by the ETSI Technical Committee Intelligent Transport Systems (ITS).

Reference
RTS/ITS-98-25

Keywords
interface, ITS, PAS, smartphone

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Intelligent Transport Systems (ITS).

The present document is part 25 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document is part of the MirrorLink® specification which specifies an interface for enabling remote user interaction of a mobile device via another device. The present document is written having a vehicle head-unit to interact with the mobile device in mind, but it will similarly apply for other devices, which provide a color display, audio input/output and user input mechanisms.

Current MirrorLink solutions are concentrated on utilization of MirrorLink Client's main display to mirror applications or provide variety services on the MirrorLink Server. However, there are so many MirrorLink Clients which have several other displays, such as cluster display panel, Heads-up Display (HUD) and so on. Instead of applications mirroring, using these displays, the driver and the passenger can be provided with a variety meta information such as turn by turn information, photo or graphic information, meta data information of audio and video clip, text information, etc. Those Meta Information Data Services are based on the SBP (Service Binary Protocol) framework.

The present document specifies navigation meta data service based on SBP (Service Binary Protocol) framework. By receiving this data, the MirrorLink Client (e.g. a car) can provide navigation information to driver and passenger e.g. through the car's cluster display panel, or heads-up display.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 103 544-27 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 27: Basic Meta Data Service".

[2] ETSI TS 103 544-6 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 6: Service Binary Protocol".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 103 544-1 (V1.3.1): "Publicly Available Specification (PAS); Intelligent Transport Systems (ITS); MirrorLink®; Part 1: Connectivity".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

HUD	Heads-Up Display
ICD	Instrument Cluster Display
SBP	Service Binary Protocol
UTC	Coordinated Universal Time

4 Data Service Definition

4.1 Navigation Meta Data Service Version 1.0

```
/** The present document defines data objects for the Navigation Meta
 *  data service to be carried over the SBP. By receiving this data,
 *  the MirrorLink Client (i.e. car) can provide variety navigation
 *  information to driver and passenger through instrument cluster
 *  display panel, HUD, etc.
 *  The service is based on the Basic Meta Information Data Service.
 *  @version 1.0
 */
SERVICE com.mirrorlink.meta.navigation
 : com.mirrorlink.meta.basic @version 1.0 {
/** Navigation route guidance possible statuses
 */
ENUM<INT> GuidanceState {
  /** Navigation guidance state is unknown
   */
  UNKNOWN = 0x00000000,
  /** Navigation guidance has no destination set
   */
  NO_DESTINATION_SET = 0x00000001,
  /** Navigation guidance system is calculating route
   */
  CALCULATING_ROUTE = 0x00000002,
  /** Navigation guidance system is using a new route
   */
  NEW_ROUTE = 0x00000003,
  /** Navigation guidance system state has no route to destination
   */
  NO_ROUTE = 0x00000004,
  /** Navigation guidance system is in normal operation
   */
  NORMAL_OPERATION = 0x00000005,
  /** Navigation guidance system positioning info is off road.
   *  maneuverDirection DIRECTION_TO_DESTINATION information should be
   *  provided, if available
   */
  OFF_ROAD = 0x00000006,
  /** Navigation guidance system positioning info is off map.
   *  maneuverDirection COMPASS information should be provided, if
   *  available.
   */
  OFF_MAP = 0x00000007,
```

```
  /** Navigation guidance system is within the destination area.
   *  maneuverDirection DIRECTION_TO_DESTINATION information should be
   *  provided, if available
   */
  DESTINATION_AREA = 0x00000008,
  /** Navigation guidance system has reached destination.
   *  maneuverDirection FINAL_DESTINATION information shall be
   *  provided.
   */
  DESTINATION_REACHED = 0x00000009
  };
/** Navigation route guidance active possible statuses.
 */
ENUM<INT> GuidanceActive {
  /** route guidance active on sink (usually a head unit)
   */
  GUIDANCE_CLIENT = 0xffffffff,
  /** no active route guidance
   */
  GUIDANCE_NONE = 0x00000000,
  /** route guidance active on source (usually a MirrorLink app)
   */
  GUIDANCE_SERVER = 0x00000001
  };
/** Definitions for NavigationNextManeuver#nextDirection
 */
ENUM<INT> ManeuverDirection {
  /** Next Direction: No symbol defined (blank screen).
   */
  NO_SYMBOL = 0x00000000,
  /** Next Direction: No information available (current direction).
   */
  NO_INFO = 0x00000001,
  /** Next Direction: Follow the street.
   */
  FOLLOW_STREET = 0x00000002,
  /** Next Direction: Turn straight.
   */
  TURN_STRAIGHT = 0x00000003,
  /** Next Direction: Slight right turn.
   */
  TURN_SLIGHT_RIGHT = 0x00000004,
  /** Next Direction: Slight left turn.
   */
  TURN_SLIGHT_LEFT = 0x00000005,
  /** Next Direction: Turn right
   */;
  TURN_RIGHT = 0x00000006,
  /** Next Direction: Turn left;
   */
  TURN_LEFT = 0x00000007,
  /** Next Direction: Sharp right turn.
   */
  TURN_SHARP_RIGHT = 0x00000008,
  /** Next Direction: Sharp left turn.
   */
  TURN_SHARP_LEFT = 0x00000009,
  /** Next Direction: Make a U-turn to the right.
   */
  UTURN_RIGHT = 0x0000000A,
  /** Next Direction: Make a U-turn to the left
   */
  UTURN_LEFT = 0x0000000B,
  /** Next Direction: Keep right.
   */
  KEEP_RIGHT = 0x0000000C,
  /** Next Direction: Keep left.
   */
  KEEP_LEFT = 0x0000000D,
  /** Next Direction: Exit to the right.
   */
  EXIT_RIGHT = 0x0000000E,
  /** Next Direction: Exit to the left.
   */
  EXIT_LEFT = 0x0000000F,
  /** Next Direction: Slight right and slight right again.
   */
  DOUBLE_TURN_SLIGHT_RIGHT_AND_SLIGHT_RIGHT_AGAIN = 0x00000010,
```

```
/** Next Direction: Slight left and slight left again.
 */
DOUBLE_TURN_SLIGHT_LEFT_AND_SLIGHT_LEFT_AGAIN = 0x00000011,
/** Next Direction: Slight right and continue straight.
 */
DOUBLE_TURN_SLIGHT_RIGHT_AND_STRAIGHT = 0x00000012,
/** Next Direction: Slight left and continue straight.
 */
DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT = 0x00000013,
/** Next Direction: Turn right and right again.
 */
DOUBLE_TURN_RIGHT_AND_RIGHT = 0x00000014,
/** Next Direction: Turn left and left again.
 */
DOUBLE_TURN_LEFT_AND_LEFT = 0x00000015,
/** Next Direction: Turn right and then turn left.
 */
DOUBLE_TURN_RIGHT_AND_LEFT = 0x00000016,
/** Next Direction: Turn left and then turn right.
 */
DOUBLE_TURN_LEFT_AND_RIGHT = 0x00000017,
/** Next Direction: Merge.
 */
MERGE = 0x00000018,
/** Next Direction: Follow the Highway.
 */
HIGHWAY_FOLLOW = 0x00000019,
/** Next Direction: On highway, slight right.
 */
HIGHWAY_SLIGHT_RIGHT = 0x0000001A,
/** Next Direction: On highway, slight left.
 */
HIGHWAY_SLIGHT_LEFT = 0x0000001B,
/** Next Direction: On highway, slight left and then slight right.
 */
HIGHWAY_DOUBLE_TURN_SLIGHT_RIGHT_AND_SLIGHT_RIGHT = 0x0000001C,
/** Next Direction: On highway, slight left and then slight left.
 */
HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_SLIGHT_LEFT = 0x0000001D,
/** Next Direction: On highway, slight right and then straight.
 */
HIGHWAY_DOUBLE_TURN_SLIGHT_RIGHT_AND_STRAIGHT = 0x0000001E,
/** Next Direction: On highway, slight left and then straight.
 */
HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT = 0x0000001F,
/** Next Direction: Michigan turn variant 1 to the right.
 */
MICHIGAN_TURN_VARIANT_1_RIGHT = 0x00000020,
/** Next Direction: Michigan turn variant 1 to the left.
 */
MICHIGAN_TURN_VARIANT_1_LEFT = 0x00000021,
/** Next Direction: Michigan turn variant 2 to the right.
 */
MICHIGAN_TURN_VARIANT_2_RIGHT = 0x00000022,
/** Next Direction: Michigan turn variant 2 left.
 */
MICHIGAN_TURN_VARIANT_2_LEFT = 0x00000023,
/** Next Direction: Enter tunnel.
 */
TUNNEL_ENTER = 0x00000024,
/** Next Direction: Continue in tunnel
 */
TUNNEL = 0x00000025,
/** Next Direction: Exit tunnel
 */
TUNNEL_EXIT = 0x00000026,
/** Next Direction: Enter ferry.
 */
FERRY_ENTER = 0x00000027,
/** Next Direction: Stay on ferry
 */
FERRY = 0x00000028,
/** Next Direction: Exit ferry
 */
FERRY_EXIT = 0x00000029,
/** Next Direction: Continue using public transportation
 */
PUBLIC_TRANSPORTATION = 0x0000002A,
```

```
  /** Next Direction: Start walking
   */
  WALK = 0x0000002B,
  /** Next Direction: Compass; then angle to the north is given in
   *  nextAngle.
   */
  COMPASS = 0x0000002C,
  /** Next Direction: Destination is at <angle> degree. The angle is
   *  given in nextAngle.
   */
  DIRECTION_TO_DESTINATION = 0x0000002D,
  /** Next Direction: Exit roundabout to the right now.
   */
  ROUNDABOUT_RIGHT_EXIT_NOW = 0x0000002E,
  /** Next Direction: Exit roundabout to the left now.
   */
  ROUNDABOUT_LEFT_EXIT_NOW = 0x0000002F,
  /** Next Direction: Enter roundabout to the right. Exit is not
   *  known.
   */
  ROUNDABOUT_RIGHT_UNKNOWN_EXIT_NUMBER = 0x00000030,
  /** Next Direction: Enter roundabout to the right. Take exit,
   *  provided in navigationNextManeuver#index;
   */
  ROUNDABOUT_RIGHT_KNOWN_EXIT_NUMBER = 0x00000031,
  /** Next Direction: Enter roundabout to the left. Exit is not
   *  known.
   */
  ROUNDABOUT_LEFT_UNKNOWN_EXIT_NUMBER = 0x00000040,
  /** Next Direction: Enter roundabout to the left. Take exit,
   *  provided in navigationNextManeuver#index;
   */
  ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER = 0x00000041,
  /** Next Direction: Final destination.
   */
  FINAL_DESTINATION = 0x00000050,
  /** Next Direction: Final destination on the right.
   */
  FINAL_DESTINATION_ON_THE_RIGHT = 0x00000051,
  /** Next Direction: Final destination on the left.
   */
  FINAL_DESTINATION_ON_THE_LEFT = 0x00000052,
  /** Next Direction: Intermediate destination. The number of the
   *  intermediate destination is provided in
   * navigationNextManeuver#index.
   */
  INTERMEDIATE_DESTINATION = 0x00000053,
  /** Next Direction: Intermediate destination on the right.
   *  The number of the intermediate destination is provided in
   *  navigationNextManeuver#index.
   */
  INTERMEDIATE_DESTINATION_ON_THE_RIGHT = 0x00000054,
  /** Next Direction: Intermediate destination on the left.
   *  The number of the intermediate destination is provided in
   *  navigationNextManeuver#index.
   */
  INTERMEDIATE_DESTINATION_ON_THE_LEFT = 0x00000055
  };
/** The DistanceUnit enumeration defines the unit of a distance value.
 */
ENUM<INT> DistanceUnit {
  /** distance expressed in meters
   */
  METER = 0x0
  /** distance expressed in kilometer
   */
  KM = 0x1
  /** distance expressed in feet
   */
  FEET = 0x02
  /** distance expressed in yard
   */
  YARDS = 0x03
  /** distance expressed in miles
   */
  MILES = 0x04
  };
/** The LaneGuidanceArrowType enumeration contains the recommendation,
```

```
 *  whether the arrow should be shown.
 */
ENUM<BYTE> LaneGuidanceArrowType {
  /** Not shown
   */
  NOT_SHOWN = 0x00,
  /** Not recommended
   */
  NOT_RECOMMENDED = 0x01,
  /** Recommended
   */
  RECOMMENDED = 0x02,
  /** Best recommended
   */
  BEST_RECOMMENDED = 0x03
  };
/** The SpecialPurposeLane enumerations contains the defined special
 *  purpose lane types. A lane may have more than one special purpose.
 *  A regular (non-special purpose lane) shall have the value 0x00.
 */
ENUM<BYTE> SpecialPurposeLane {
  /** Non-special purpose lane: Regular lane, not having any special
   *  purpose as defined below.
   */
  REGULAR_LANE = 0x01,
  /** Special purpose lane: HOV lane for high occupancy vehicles.
   */
  HOV_LANE = 0x02,
  /** Special purpose lane: Toll lane.
   */
  TOLL_LANE = 0x03,
  /** Special purse lane: Temporary-use lane. This indicates a lane,
   *  which can be opened to traffic at certain times, e.g. during
   *  rush-hour.
   */
  TEMPOARY_LANE = 0x04
  };
/** The LaneGuidanceLineType enumeration contains defined type of
 *  lines, which are separating lanes.
 */
ENUM<BYTE> LaneGuidanceLineType {
  /** No line
   */
  NONE = 0x00,
  /** Solid Line
   */
  SOLID = 0x01,
  /** Dashed Line
   */
  DASHED = 0x02,
  /** Double solid Line
   */
  DOUBLE = 0x03,
  /** Barrier (non-crossable) or road limit
   */
  BARRIER = 0x04
  };
/** The enumeration contains the lane guidance arrow types. The values
 *  are bit mask values, i.e. a lane may show a combined arrow of two or
 *  more arrow types, e.g. TURN_RIGHT and TURN_LEFT.
 *  Each arrow type has a value defined in LaneGuidanceArrowType, which
 *  shall be bit shifted to completely fit into the bit mask.
 *  In addition, information on lane types and line types are provided.
 *  Bitfields not covered are reserved for future use.
 */
ENUM<INT> LaneGuidanceBitMask {
  /** Arrow Type: Turn straight; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType.
   */
  TURN_STRAIGHT = 0x00000003,
  /** Arrow Type: Turn slight right; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 2 (bit shift).
   */
  TURN_SLIGHT_RIGHT = 0x0000000C,
  /** Arrow Type: Turn slight left; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 4 (bit shift).
   */
  TURN_SLIGHT_LEFT = 0x00000030,
```

```
  /** Arrow Type: Turn right; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 6 (bit shift).
   */
  TURN_RIGHT = 0x000000C0,
  /** Arrow Type: Turn left; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 8 (bit shift).
   */
  TURN_LEFT = 0x00000300,
  /** Arrow Type: Turn sharp right; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 10 (bit shift).
   */
  TURN_SHARP_RIGHT = 0x00000C00,
  /** Arrow Type: Turn sharp left; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 12 (bit shift).
   */
  TURN_SHARP_LEFT = 0x00003000,
  /** Arrow Type: U turn right; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 14 (bit shift).
   */
  U_TURN_RIGHT = 0x0000C000,
  /** Arrow Type: U turn left; recommendation to show this arrow
   *  type, as defined in LaneGuidanceArrowType << 16 (bit shift).
   */
  U_TURN_LEFT = 0x00030000,
  /** Special purpose lane; if set, the lane has a special purpose as
   *  defined in SpecialPurposeLane << 20 (bit shift).
   */
  SPECIAL_PURPOSE_LANE = 0x00F00000,
  /** Contains information on the line markings between lanes. The
   *  value defines the left line of the respective lane.
   *  Possible values are defined LaneGuidanceLineType << 24 (bit
   *  shift).
   */
  LINE_TYPE = 0x0F000000
  };
/** The NavigationInfo object informs the data sink about the state of
 *  the navigation meta data source.
 *  @mandatory, @readable, @version 1.0, @uid 0xd1276567
 */
OBJECT NavigationInfo {
  /** Identifier of the current navigation application, as used within
   *  UPnP application advertisements.
   *  @mandatory, @readable, @uid 0x7dd9aa9e
   */
  INT navAppId;
  /** Navigation route guidance status.
   *  @mandatory, @uid 0x4dbc1b54
   */
  ENUM<GuidanceState> guidanceState;
  /** Status of the navigation route guidance on the MirrorLink
   *  Server. The MirrorLink Client should use the information to
   *  avoid concurrent route guidance on MirrorLink Client and Server
   *  side. The last activated one should win.
   *  @mandatory, @uid 0x99756c63
   */
  ENUM<GuidanceActive> guidanceActive;
  /** Value defines, whether all distance & speed values are
   *  represented in metric (true) or non-metric (false) system.
   *  @mandatory, @uid 0xb4e34de4
   */
  BOOLEAN metricSystem;
  /** Value defines, whether vehicles are driving on the right side of
   *  the road (true) or the left side of the road (false).
   *  @mandatory, @uid 0x59c37a70
   */
  BOOLEAN rightDriving;
  };
/** The NavigationConfig object is set from data sink, to configure
 *  the behavior of the data source.
 *  @mandatory, @writeable, @version 1.0, @uid 0xcb904e1b
 */
OBJECT NavigationConfig {
  /** Maximum number of maneuver side street angles reported in
   *  navigationNextManeuver#nextSideStreetAngles_1
   *  @mandatory, @uid 0x92f65d8b
   */
  INT maxSideStreetAngles_1;
  /** Maximum number of maneuver side street angles reported in
```

```
    *  navigationNextManeuver#nextSideStreetAngles_2
    *  @mandatory, @uid 0x92f65d8c
    */
   INT maxSideStreetAngles_2;
   /** Maximum number of maneuver side street angles reported in
    *  navigationNextManeuver#nextSideStreetAngles_3
    *  @mandatory, @uid 0x92f65d8d
    */
   INT maxSideStreetAngles_3;
   /** Maximum number of lane guidance information.
    *  @mandatory, @uid 0xec8c5a70
    */
   INT maxLaneGuidances;
   };
/** The NavigationNextManeuver object provides the data sink with
 *  details about the next upcoming maneuver.
 *  @mandatory, @readable, @version 1.0, @uid 0x1c4599e5
 */
OBJECT NavigationNextManeuver {
   /** Next maneuver direction, e.g. follow street, turn, U-turn, ...
    *  Possible maneuvers are defined in the maneuverDirection
    *  enumeration.
    *  @mandatory, @uid 0xe20e4e87
    */
   ENUM<ManeuverDirection> nextDirection;
   /** Details selected guidance instances; used in case
    *  nextDirection has one of the following values:
    *  - ROUNDABOUT_RIGHT_KNOWN_EXIT_NUMBER (mandatory), indicating the
    *    known exit number.
    *  - ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER (mandatory), indicating the
    *    known exit number.
    *  - INTERMEDIATE_DESTINATION  to
    *    INTERMEDIATE_DESTINATION_ON_THE_LEFT (optional), indicating
    *    the specific intermediate destination.
    *  Value shall be >0.
    *  @conditional, @uid 0x1e5be26d
    */
   INT index;
   /** Represents the clock-wise (turn) angle; used in case
    *  nextDirection has one of the following values:
    *  - TURN_STRAIGHT to HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT
    *    (optional)
    *  - COMPASS (mandatory)
    *  - DIRECTION_TO_DESTINATION (mandatory)
    *  - ROUNDABOUT_RIGHT_UNKNOWN_EXIT_NUMBER to
    *    ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER (mandatory)
    *  A value of -1 shall be used, if an angle is not available.
    *  @unit degrees, @range [0 .. 360], @conditional, @uid 0x7ffdc13b
    */
   INT nextAngle;
   /** Array of side streets clock-wise angles at different turn
    *  positions; used in case nextDirection has one of the following
    *  values:
    *  - TURN_STRAIGHT to HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT
    *    (optional)
    *  - ROUNDABOUT_RIGHT_UNKNOWN_EXIT_NUMBER to
    *    ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER (optional)
    *  The number of items shall not exceed the value defined in
    *  navigationStatus#maxSideStreetAngles_1.
    *  @unit, @range [0 .. 360], @conditional, @uid 0x6f927a04
    */
   ARRAY<INT> nextSideStreetAngles_1;
   /** Array of side streets clock-wise angles at different turn
    *  positions; used in case nextDirection has one of the following
    *  values:
    *  - TURN_STRAIGHT to HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT
    *    (optional)
    *  - ROUNDABOUT_RIGHT_UNKNOWN_EXIT_NUMBER to
    *    ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER (optional)
    *  The number of items shall not exceed the value defined in
    *  navigationStatus#maxSideStreetAngles_2.
    *  @unit, @range [0 .. 360], @conditional, @uid 0x6f927a05
    */
   ARRAY<INT> nextSideStreetAngles_2;
   /** Array of side streets clock-wise angles at different turn
    *  positions; used in case nextDirection has one of the following
    *  values:
    *  - TURN_STRAIGHT to HIGHWAY_DOUBLE_TURN_SLIGHT_LEFT_AND_STRAIGHT
```

```
    *    (optional)
    *  - ROUNDABOUT_RIGHT_UNKNOWN_EXIT_NUMBER to
    *    ROUNDABOUT_LEFT_KNOWN_EXIT_NUMBER (optional)
    *  The number of items shall not exceed the value defined in
    *  navigationStatus#maxSideStreetAngles_3.
    *  @unit, @range [0 .. 360], @conditional, @uid 0x6f927a06
    */
  ARRAY<INT> nextSideStreetAngles_3;
  /** Street name, where the next maneuver will happen.
    *  The name may include additional information, like the number of
    *  the exit (e.g. Exit 245) or other direction information (e.g.
    *  Interstate 280, North).
    *  Shall be empty, if the street name is unknown.
    *  @mandatory, @uid 0xa377c466
    */
  STRING nextStreetName
  /** Street name, where the current maneuver will happen.
    *  The name may include additional information, like the number of
    *  the exit (e.g. Exit 245) or other direction information (e.g.
    *  Interstate 280, North).
    *  Shall be empty, if the street name is unknown.
    *  @mandatory, @uid 0x05040ec2
    */
  STRING currentStreetName
  };
/** The SpeedLimit object provides the data sink with
 *  details about the current and upcoming speed limit.
 *  @mandatory, @readable, @version 1.0, @uid 0x580a7019
 */
OBJECT SpeedLimit {
  /** Current speed limit.
    *  - It shall be set to zero, in case no speed limit exists.
    *  - It shall be set to -1, if the speed limit is unknown.
    *  The speed limit's is provided either km/h or miles/h, dependent
    *  of the navigationInfo#metricSystem value.
    *  @mandatory, @uid 0xabb89e38
    */
  INT currentSpeedLimit;
  /** Upcoming next speed limit.
    *  - It shall be set to zero, in case no speed limit exists.
    *  - It shall be set to -1, if the speed limit is unknown.
    *  The speed limit's is provided either km/h or miles/h, dependent
    *  of the navigationInfo#metricSystem value.
    *  @mandatory, @uid 0x4a2c498c
    */
  INT nextSpeedLimit;
  /** Distance, to when the next speed limit will become effective.
    *  @optional, @uid 0x5b49a2da
    */
  INT distance;
  /** Defines the unit, which applies to distance. Shall be provided,
    *  if distance is included.
    *  @conditional, @uid 0x87079bfe
    */
  ENUM<DistanceUnit> distanceUnit;
  };
/** The NavigationNextDistance object contains the distance from the
 *  current position to the position of the next maneuver, defined in
 *  the NavigationNextManeuver object.
 *  @mandatory, @readable, @version 1.0, @uid 0xd53a7a01
 */
OBJECT NavigationNextDistance {
  /** Distance to next navigation direction.
    *  The distance shall be provided rounded in reasonable steps with
    *  respect to the defined distance unit, e.g. 200 yards, 300 m,
    *  1/4 mile or 1 km.
    *  @mandatory, @uid 0x5b49a2da
    */
  INT distance;
  /** Defines the unit, which applies to distance. Shall be provided,
    *  if distance is included.
    *  @conditional, @uid 0x87079bfe
    */
  ENUM<DistanceUnit> distanceUnit;
  /** Time to next maneuver. Note, this will be the best estimate.
    *  @unit seconds, @mandatory, @uid 0x00a0fdb2
    */
  TIME time;
```

```
  /** Percentage already traveled to the next navigation direction.
   *  - A value of 0 defines the location of the previous maneuver.
   *  - A value of 100 defines the location of the upcoming maneuver.
   *  The vehicle will typically in between position 0 and 100.
   *  Percentage is defined as
   *  @unit percentage, @range [0 .. 100], @mandatory, @uid 0x0dc4addf
   */
  INT percentage;
  };
/** The NavigationLaneGuidance object informs the data sink about
 *  available lanes, their type, arrow types and line types.
 *  @optional, @readable, @version 1.0, @uid 0xab70ecbd
 */
OBJECT NavigationLaneGuidance {
  /** Lane guidance information, for each lane (from the left to the
   *  right), constructed using the bit mask definitions outlined
   *  laneGuidanceBitMask.
   *  @mandatory, @uid 0xa01236b7
   */
  ARRAY<INT> nextLaneGuidances;
  };
/** NavigationTripInfo Object
 *  @mandatory, @readable, @version 1.0, @uid 0x3b9ec7cc
 */
OBJECT NavigationTripInfo {
  /** Name of destination.
   *  @mandatory, @uid 0xac8b0089
   */
  STRING destination
  /** Remaining distance to destination.
   *  The distance shall be provided rounded in reasonable steps with
   *  respect to the defined distance unit, e.g. 200 yards, 300 m,
   *  1/4 mile or 1 km.
   *  @optional, @uid 0x5b49a2da
   */
  INT distance;
  /** Defines the unit, which applies to distance. Shall be provided,
   *  if distance is included.
   *  @conditional, @uid 0x87079bfe
   */
  ENUM<DistanceUnit> distanceUnit;
  /** Remaining travel time to destiunation. Note, this will be the
   *  best estimate.
   *  @mandatory, @unit seconds, @uid 0x869ad938
   */
  TIME remainingTravelTime;
  /** Estimated time spend in traffic. The traffic time shall be
   *  included in remainingTravelTime. Note, this will be the
   *  best estimate.
   *  @optional, @unit seconds, @uid 0x83b85959
   */
  TIME remainingTrafficTime;
  };
};
```

5 SBP Binding

The Navigation Meta Data Services uses the following objects and their access capabilities, as defined in [1].

Table 1

Object Name	Access Type	Subscription Type	Min Interval Time	Max Interval Time
NavigationInfo	READABLE	ON_CHANGE	N/A	N/A
NavigationConfig	WRITEABLE	NONE	N/A	N/A
NavigationNextManeuver	READABLE	ON_CHANGE	N/A	N/A
SpeedLimit	READABLE	ON_CHANGE	N/A	N/A

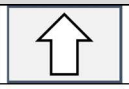
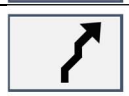
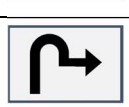
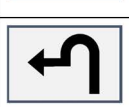
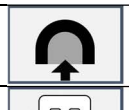
Object Name	Access Type	Subscription Type	Min Interval Time	Max Interval Time
NavigationNextDistance	READABLE	ON_CHANGE	N/A	N/A
NavigationLaneGuidance	READABLE	ON_CHANGE	N/A	N/A
NavigationTripInfo	READABLE	ON_CHANGE	N/A	N/A

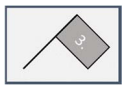
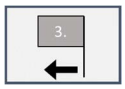
6 Theory of Operation

6.1 Maneuver Description

The following icons illustrate the different maneuvers, as defined in *ManeuverDirection*.

Table 2

Description	Example	Description	Example	Description	Example	Description	Example
No Symbol		No Info		Follow Street		Turn Straight	
Turn Slight Right		Turn Slight Left		Turn Right		Turn Left	
Turn Sharp Right		Turn Sharp Left		U-Turn Right		U-Turn Left	
Keep Right		Keep Left		Exit Right		Exit Left	
Slight Right and Slight Right again		Slight Left and Slight Left again		Turn Slight Right and Straight		Turn Slight Left and Straight	
Turn Right and Right		Turn Left and Left		Turn Right and Left		Turn Left and Right	
Merge		Highway Follow		Highway Slight Right		Highway Slight Left	
Highway Turn Slight Right and Slight Right		Highway Turn Slight Left and Slight Left		Highway Turn Slight Right and Straight		Highway Turn Slight Left and Straight	
Michigan Turn Variant 1 Right		Michigan Turn Variant 1 Left		Michigan Turn Variant 2 Right		Michigan Turn Variant 2 Left	
Tunnel Enter		Tunnel		Tunnel Exit		Compass	
Ferry Enter		Ferry		Ferry Exit		Direction to Destination	
Roundabout Right - Exit Now		Roundabout Left - Exit Now		Roundabout Right - Exit n		Roundabout Left - Exit n	
Final Destination		Final Destination on the Right		Final Destination on the Left		Public Transportation	

Description	Example	Description	Example	Description	Example	Description	Example
Intermediate Destination		Intermediate Destination on Right		Intermediate Destination on Left		Walk	

6.2 Representation of Angle

Angles are represented clock-wise from 0 to 360 degree, as shown in the following Figure 1.

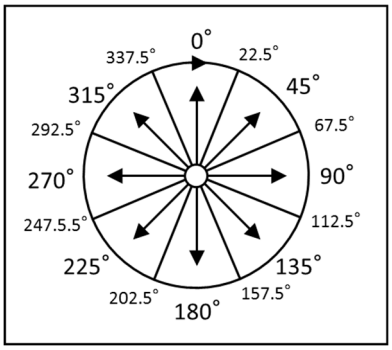


Figure 1: representation of angles

6.3 Getting Turn-by-Turn Navigation Info

Figure 2 shows how a navigation data sink retrieves meta data about the currently running navigation data source and displays turn-by-turn guidance e.g. on the Instrument Cluster Display (ICD).

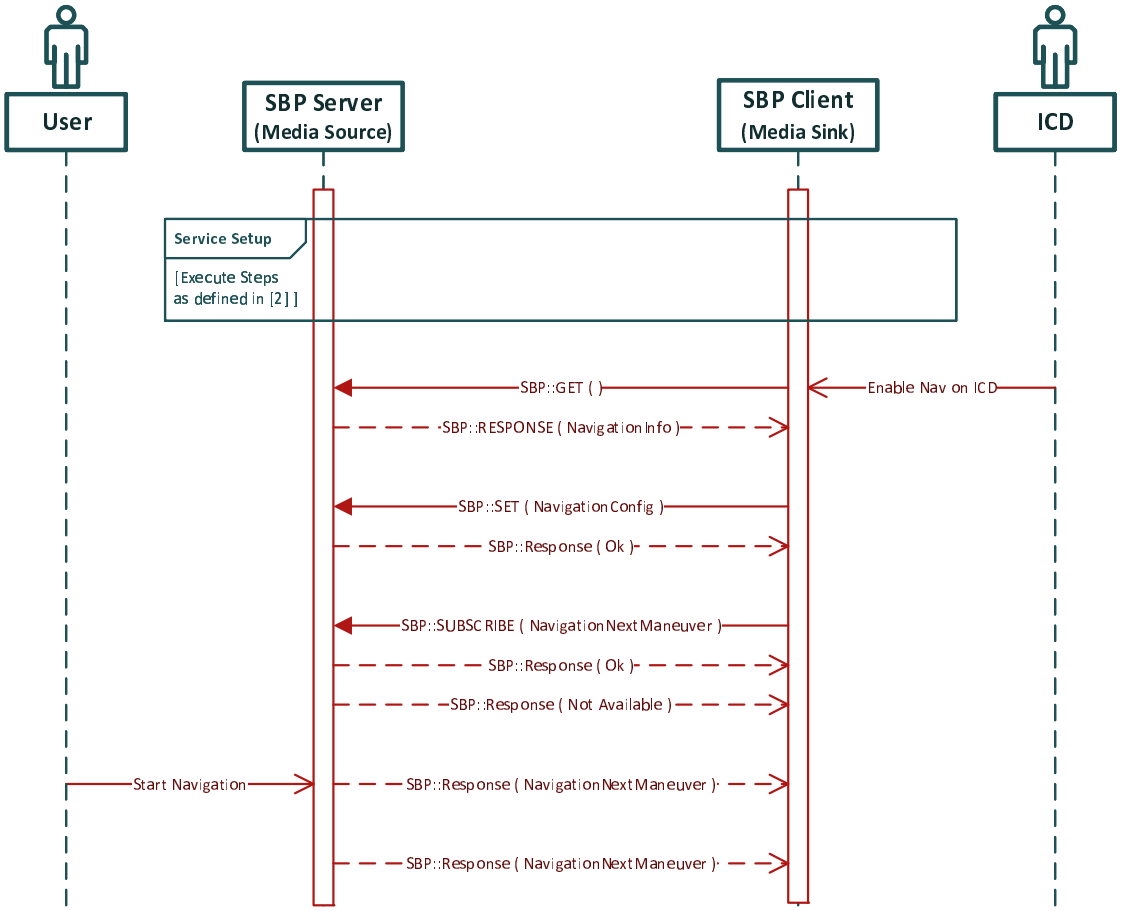


Figure 2: Message Sequence Diagram - Turn-by-Turn Navigation

It consists of the following steps, after the data service has been setup as defined in [1]:

1) Navigation Sink sends an SBP *Get* message for the *NavigationObject* object. The Navigation Source responds with the requested object value.

2) Navigation Sink sends an SBP *Set* message for the *NavigationConfig* object. The Navigation Source responds with an Ok response.

3) Navigation Sink sends an SBP *Subscribe* message for the *NavigationNextManeuver* object; subscription is ON_CHANGE. The Navigation Source responds first with an SBP *Response* message confirming the *Subscribe* message, followed by a second SBP *Response* message containing an `"Not Available"` error code, as no navigation is currently started.

NOTE: In case navigation is started prior subscription, the Navigation Sink will return a valid *NavigationNextManeuver* turn-by-turn direction object.

4) The user is starting the navigation.

5) The Navigation Source sends a *NavigationNextManeuver* turn-by-turn direction object, when the next turn is announced.

6.4 Definition of Time

The Service Binary Protocol (SBP) [2] defines TIME as a 64-bit signed integer (LONG) with the meaning of time in milliseconds since 1970-01-01-00:00 in UTC or relative time in milliseconds depending on how it is defined in each service.

The present specification defines the TIME value in the *NavigationNextDistance* object as the relative time until the next maneuver in second (i.e. not in milli-seconds).

6.5 Navigation Lane Guidance

The *NavigationLaneGuidance* object informs the data sink about available lanes. Each lane is described within a 32-Bit integer value, containing information about:

- Line Type (termination marking at the left of the lane).
- Lane Type (used for special purpose lanes).
- Directional markings on the lane.

Lane information is provided for each lane, from the left to the right (in driving direction) and stored in the *nextLaneGuidance* array. Figure 3 highlights the lane details.

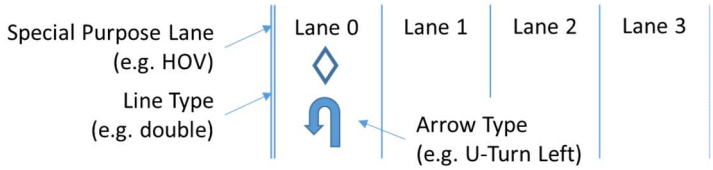


Figure 3: Lane Guidance Overview

The respective information for each lane is contained in a bit mask, as summarized in Table 3.

Table 3: Bitmask for *nextLaneGuidance* element in *NavigationLaneGuidance* Object

Bits	[27:24]	[23:20]	[17:16]	[15:14]	[13:12]	[11:10]	[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
Function	Line Type	Special Purpose Lane	U-Turn Left	U-Turn Right	Turn Sharp Left	Turn Sharp Right	Turn Left	Turn Right	Turn Slight Left	Turn Slight Right	Go Straight

Figure 4 gives an example lane setup.

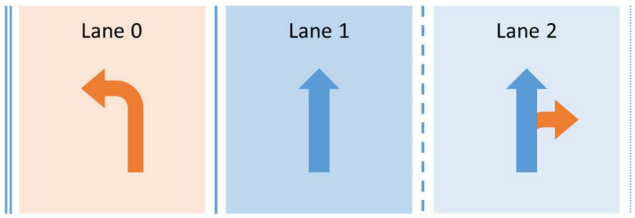


Figure 4: Lane Guidance Example

This example translates the example into the *NavigationLaneGuidance* object, given in Table 4.

Table 4: Example *NavigationLaneGuidance* Object

Bits	[27:24]	[23:20]	[17:16]	[15:14]	[13:12]	[11:10]	[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
Function	Line Type	Special Purpose Lane	U-Turn Left	U-Turn Right	Turn Sharp Left	Turn Sharp Right	Turn Left	Turn Right	Turn Slight Left	Turn Slight Right	Go Straight
Lane 0	0011b	0000b	00b	00b	00b	00b	01b	00b	00b	00b	00b

Bits	[27:24]	[23:20]	[17:16]	[15:14]	[13:12]	[11:10]	[9:8]	[7:6]	[5:4]	[3:2]	[1:0]
Lane 1	0001b	0000b	00b	00b	00b	00b	00b	00b	00b	00b	11b
Lane 2	0010b	0000b	00b	00b	00b	00b	00b	01b	00b	00b	10b

Table 2 contains the following lane features:

- Lane 0: Double Line, Turn Left (not recommended)
- Lane 1: Solid Line, Go Straight (best recommended)
- Lane 2: Dashed Line, Turn Left (not recommended),
  Go Straight (recommended)

6.6 Next Side Streets

The *NavigationNextManeuver* objects contains 3 arrays, *nextSideStreetAngles_{1,2,3}*, which represent angle values of side streets leaving from the point at next turn and prior to it:

- *nextSideStreet_1* array contains side street angles at the point of the next maneuver.
- *nextSideStreet_2* array contains side street angles at one point prior to the next maneuver.
- *nextSideStreet_3* array contains side street angles at two points prior to the next maneuver.

The *NavigationConfig* object defines the maximum number of supported side streets for each intersection point. The value is set to zero if no side streets are supported at the respective intersection. It is ok, to support only side streets at the final maneuver point (maxSideStreetAngles_1) and set the other two to zero. If *maxSideStreetAngles_2* is set to zero *maxSideStreetAngles_3* should be set to zero as well. The data source shall not provide more side street angle values, than supported from the Sink. The data source may provide less side street angle values, than supported from the Sink.

Figure 5 and Figure 6 provide examples of the *nextSideStreetAngles* use for single-step maneuver types. Figure 5 shows all 4 interception points being used.

Figure 5: Single Step Maneuver (Right Turn with 3 Intersection Points)

Figure 6 shows only a subset of interceptions points being used.

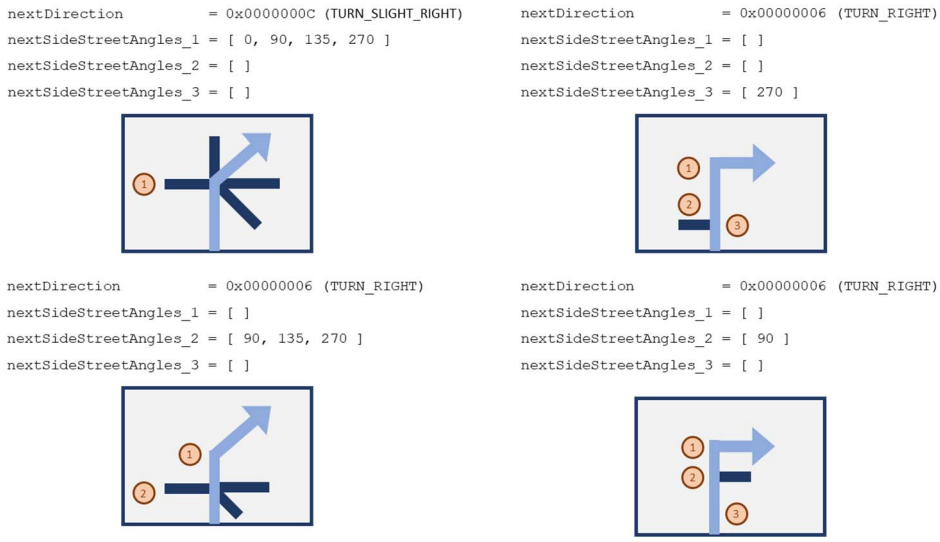


Figure 6: Single Step Maneuver (Right Turn with <3 Intersection Points)

If the next maneuver has multiple individual steps, side streets elements are references them individually. Figure 7 shows example multi-step maneuvers and the respective *nextSideStreetAngles* arrays.

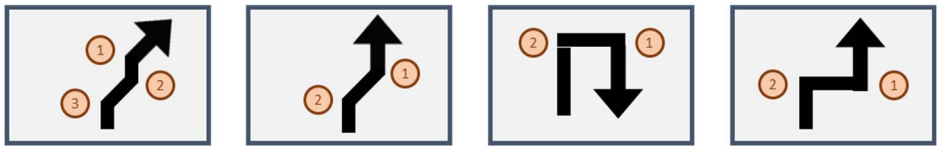


Figure 7: Example Multi-Step Maneuvers

Figure 8 provides examples, which show the use of *nextSideStreetAngles* values for different multi-step maneuvers.

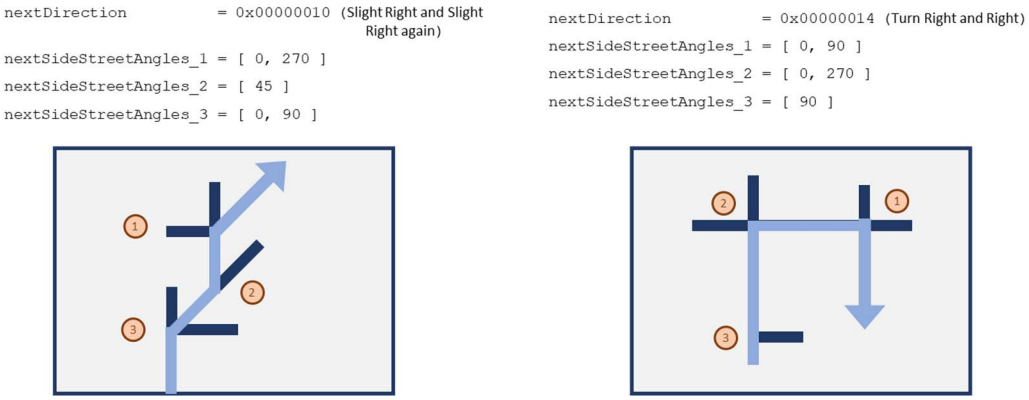
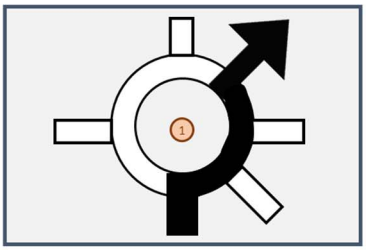


Figure 8: Multi-Step Maneuvers

A round-about is considered a single maneuver, i.e. angles *nextSideStreetAngles* array 1 are defining side streets off the round-about. The *nextSideStreetAngles* arrays 2 and 3 define side streets prior entering the round-about. Examples are shown in Figure 9.

```
nextDirection          = 0x00000031 (Round about right (45) )
nextSideStreetAngles_1 = [ 0, 90, 135, 270 ]
nextSideStreetAngles_2 = [ ]
nextSideStreetAngles_3 = [ ]
```




```
nextDirection          = 0x00000031 (Round about right (45) )
nextSideStreetAngles_1 = [ 0, 90, 135, 270 ]
nextSideStreetAngles_2 = [ 90 ]
nextSideStreetAngles_3 = [ ]
```

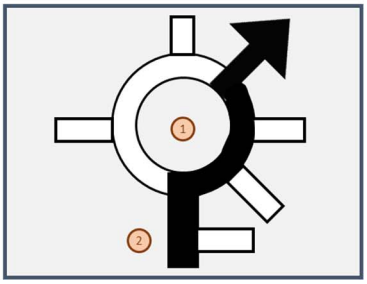


Figure 9: Round-About Maneuvers

Annex A (informative): Authors and Contributors

The following people have contributed to the present document:

Rapporteur:	Dr. Jörg Brakensiek, E-Qualus (for Car Connectivity Consortium LLC)
Other contributors:	ARam Cho, Samsung Electronics
	Kiran Vedula, Samsung Electronics
	Inyoung Shin, Samsung Electronics
	Mayur, Samsung Electronics
	Matthias Benesch, Daimler AG
	Gautier Falconnet, PSA

History

Document history		
V1.3.0	October 2017	Publication
V1.3.1	October 2019	Publication