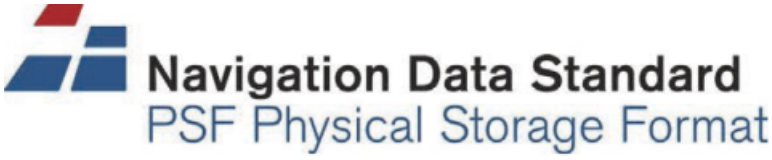
Navigation Data Standard

Location Documentation

FTX version 2.0



Intellectual Property Rights

This document of a specification is released by the Navigation Data Standard (NDS) e.V., in the following called NDS e.V. It is released as a development partnership and intended for the purpose of information only. The NDS e.V. will not be liable for any use of this specification. Following the completion of the development of the Navigation Data Standard PSF specifications, commercial exploitation licenses will be made available to end users by way of written License Agreement only.

Navigation Data Standard PSF and the associated specification documents are subject to change and are continually updated as the development of Navigation Data Standard PSF progresses. The responsibility for maintaining and interpreting the Navigation Data Standard PSF specification documents lies with the bodies of the NDS e.V. Navigation Data Standard PSF development is driven by a well-defined process for releasing documented versions of the standard.

No part of this document shall be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from NDS e.V.

Specification documents for the Navigation Data Standard PSF may contain exemplary items (exemplary reference models, “use cases”, and/or references to exemplary technical solutions, devices, processes or software). Any such exemplary items are contained in the specification documents for illustration purposes only, and they themselves are not part of Navigation Data Standard PSF. Neither their presence in such specification documents, nor any later documentation of standard conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to Navigation Data Standard PSF.

Navigation Data Standard PSF is under constant development. For this reason, the description in this document may deviate from the physical implementation. If this is the case, the physical implementation in DataScript shall prevail.

Copyright © 2020 – Navigation Data Standard (NDS) e.V. All Rights Reserved.

Document version 2.0

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Table of Contents

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

List of Figures

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

List of Tables

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

1 About this Document

This chapter gives an overview of the document's history, purpose, subjects and target audience, followed by abstracts of the subsequent chapters.

1.1 Purpose and Subject

This document describes how to extend NDS databases with fast track extensions. NDS is a standardized physical storage format for navigation systems. The format has been developed by NDS e.V., a registered society of car manufacturers, navigation system suppliers, and map suppliers.

The standardized binary format for navigation data as specified in the *NDS – Format Specification* offers new possibilities for extending navigation databases.

1.2 Target Audience

This document is provided for IT professionals who want to get familiar with fast track extensions for Navigation Data Standard. This can be, for example, software development managers, product managers specifying navigation systems, or engineers developing such systems.

The following knowledge is assumed:

- Familiarity with digital maps and navigation systems
- Solid know-how regarding the structure and functionality of databases

1.3 NDS Documentation Overview

The following documentation on Navigation Data Standard (NDS) is available:

- *NDS – Physical Model Description*: This documentation comprises the NDS DataScript implementation. It also contains the comments which describe the structure and properties of the NDS data types.
- *NDS – Format Specification*: This document contains detailed descriptions of the data types of NDS databases, meaning features and their attributes, as well as metadata.
- *NDS – Update Specification*: This document describes the general concepts and processes for updating NDS databases.
- FTX documentation: These documents describe how to extend NDS databases with fast track extensions

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

- *NDS – Certification Process Specification*: This document describes the certification process and the requirements for certification of navigation databases complying with Navigation Data Standard.
- *NDS – Certification Bench User Guide*: This document describes the steps and tools that are required for certifying navigation databases complying with Navigation Data Standard.
- *SQLite Information*: Overview on the SQLite reference engine.
- *RDS User Guide*: Guide for developers and users of the RDS tool.

1.4 Use of Modal Verbs

For compliance with the NDS standard, database suppliers need to be able to distinguish between mandatory requirements, recommendations, permissions, as well as possibilities and capabilities. This is supported by the rules for use of modal verbs that are followed in the NDS specification documents to express the four kinds of provisions.

Table 1-1: Rules for use of modal verbs

Provision	Verbal form
Requirement Requirements shall be followed strictly in order to conform to the standard. Deviations are not allowed.	shall shall not
Recommendation Recommendations indicate that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others.	should should not
Permission Permissions indicate a course of action permissible within the limits of NDS deliverables.	may need not
Possibility and capability These verbal forms are used for stating possibilities or capabilities, being technical, material, physical, or others.	can cannot
Obligation and necessity These verbal forms are used to describe legal, organizational, or technical obligations and necessities that are not regulated or enforced by the NDS standard.	must must not

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

1.5 Typographical Conventions

This documentation uses the following typographical conventions:

- `Code elements`: This format is used for code elements, such as technical names of classes and attributes, as well as attribute values.
- `File and path names`: This format is used for the names of files and folders, as well as for directory paths.
- *Cross-references*: This format is used for cross-references to topics in the current document, to topics in other specifications that are part of Navigation Data Standard, or to external documents.
- *Terms*: This format is used to introduce glossary terms, new terms and emphasize particular terms.
- <Variables>: For placeholders that need to be replaced by actual values, angle brackets are used. Example: The variable `<installation directory>` needs to be replaced by, for example, `C:\Program Files`.

The following types of note are used:

Information	Notes of type "Information" offer background knowledge, show alternative ways of performing a task or provide information to make your work more efficient.
Note	Notes of type "Note" provide important information that helps you avoid errors and problems.
Caution	Notes of type "Caution" contain essential information that you must follow to avoid malfunctions, incorrect data and other major problems.
Example	Notes of type "Example" contain examples of the concept described in the current topic.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2 Location Extension

The Location extension provides data structures to define and attribute map-agnostic locations that support multiple use cases, such as:

- Mapping of real-world locations to base links and road geometry lines of the Routing building block. This allows to map different topologies, such as SD and HD topologies.
- Providing flexible attributes with short and long lifespans that override and complement existing map attributes. This approach is an alternative to the Volatile Data building block. That way, data from multiple dynamic data providers can be integrated and the providers only need to know the predefined source location IDs, but do not need to have full knowledge of the preinstalled map.

Applications for highly automated driving can use location information to derive likely behavior of other traffic.

2.1 Definition of Source Locations

A source location is a logical entity that describes a stretch of a real-world road. The Location extension provides location features for storing source locations and for defining their relations to base links or road geometry lines of the Routing building block.

Source locations are supported under the following conditions:

- Unique ID
  The ID is globally unique.
- Direction
  Start and end position define the direction of the source location.
- Continuity
  There are no holes or gaps.
- Unambiguous path
  In one travel direction, there are no adjacent parts.
- Recommendation: No loops
  One source location does not cross or reconnect to itself.
- Recommendation: No splits or merges
  There are no splits or merges.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2.2 Data Structures for Location Features

The Location extension adds the following data structures to the product database:

- Location tile table

  The `LocationTileTable` stores location features. It defines references from source location IDs to base links and road geometry lines of an update region.

  DataScript location: `nds.ftx.location.main`

- Location lookup table

  The `LocationId2TileTable` identifies all tiles that contain the location feature.

  DataScript location: `nds.ftx.location.main`

- Location metadata

  The following metadata tables are available in the Location extension:

  - `LocationProviderMetadataTable`: Contains metadata of all providers in the Location extension.
  - `LocationSourceLocationSetTable`: Contains metadata of all sets of source locations that are used in the database.
  - `LocationDynamicDataServicesTable`: Contains metadata of dynamic data services that supply data for the source locations.
  - `LocationStaticDataMappingTable`: Contains metadata of the mapping of static location data to the database.

  DataScript location: `nds.ftx.location.main`

- Attribute tables

  The attribute tables define flexible attributes that are assigned to a location feature. The following tables are available:

  - `LocationAttrTileTable`: Contains tile-based flexible attributes that have a long lifespan. The table entries can be versioned. If an attribute applies to multiple tiles, then it shall be stored redundantly in each tile.
  - `LocationDynamicDataAttrTileTable` and `LocationDynamicDataAttributeTable`: Both tables contain flexible attributes that have a short lifespan. All location attributes are stored with a timestamp and they can have a start time and an expiration time.

    The `LocationDynamicDataAttrTileTable` stores flexible attributes per tile and source location.

    The `LocationDynamicDataAttributeTable` stores flexible attributes for a complete source location or part of a source location, for a sequence of source locations, or for a transition between two source locations.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

For example, the attributes can be used to reverse transition permissions between two source locations. This dynamic data does not act as full replacement of the corresponding primary attributes in the preinstalled map, but serves as an additional source of information for the in-car application.

Caution If the `LocationDynamicDataAttrTileTable` is used to assign the same attribute value to a source location in multiple tiles, then the attribute shall be stored redundantly for each tile.

Note Which table to use, depends on the use case. If the amount of data is small but the data covers a large geographic area, then the `LocationDynamicDataAttributeTable` can be more suitable. The advantage of the `LocationDynamicDataAttrTileTable` is that the tiles are self-contained and can be distributed individually.

DataScript location: `nds.ftx.location.flexattr`

- Location messages

  Location messages include one or more dynamic attributes from a dynamic data service. In addition, location messages contain the ID of the dynamic data service so that messages from different dynamic data providers can be logically separated. The following message types are available:

  - `LocationDynamicDataAttributeMessage`: Contains a list of flexible attributes with dynamic data. The attributes are assigned to a specific source location including an assignment direction. The dynamic location attributes have a start time and expiration time, as well as a reference to a source location version.

  - `LocationDynamicDataAttrTileMessage`: Contains attribute tiles with dynamic data. Attribute tiles have a start time and expiration time, as well as a reference to a source location version. If an attribute applies to multiple tiles, then it shall be stored redundantly in each tile.

  DataScript location: `nds.ftx.location.messages`

When the Location extension is integrated, a dedicated entry is created in the `buildingBlockTable` in the product database contains . The building block type of the integrated Location extension is `EXTENSION`. The building block type contains the following properties:

```
buildingBlockType EXTENSION
ExtensionData (BLOB)
{
 ndsDbSupplierId = NDS_SUPPLIER_ID
 ndsFtxId = FTXID_LOCATION (11)
}
```

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2.3 Location Features

Location tiles contain several location features. Each location feature has a unique `locationId`, which is identical to the ID of the source location.

The `LocationFeatureRefList` defines all references of a location feature to base links or road geometry lines of the Routing building block. Each feature reference contains information about the direction of the location feature, which is defined in relation to the corresponding routing features. For more information, see *Section 2.6: "References of Location Features"* on page 15.

A location feature defines one or more location branches. For more information, see *Section 2.4: "Location Branches"* on page 12.

The following values define the length of a location feature:

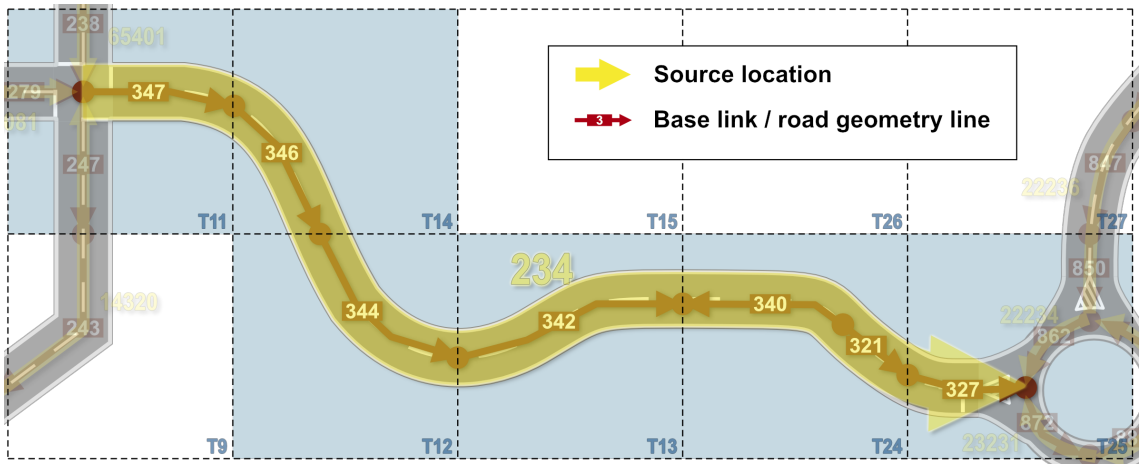
- `totalLengthInCm` stores the total length of the location feature.
- `tileLengthInCm` stores the length within the specific tile.

The `CoveredTileList` specifies all tiles that the location feature belongs to.

DataScript location: `nds.ftx.location.main` > `LocationFeature`

Figure 2-1 on page 12 illustrates a source location that spans several tiles. It is represented by one location feature.

Figure 2-1: Source location

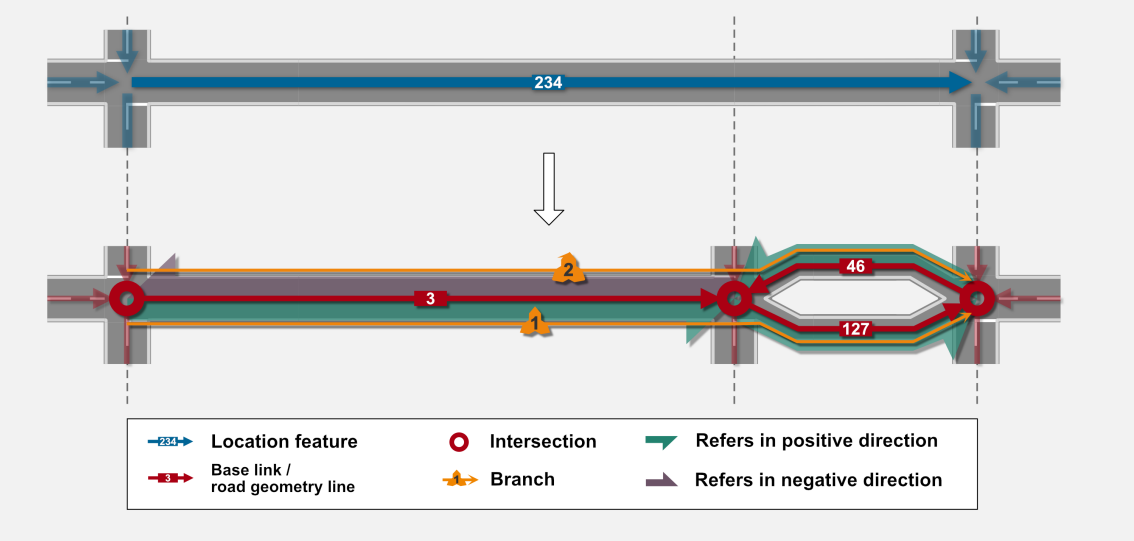


2.4 Location Branches

Each location feature defines one or more location branches. *Figure 2-2* on page 13 illustrates one location feature with multiple location branches.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

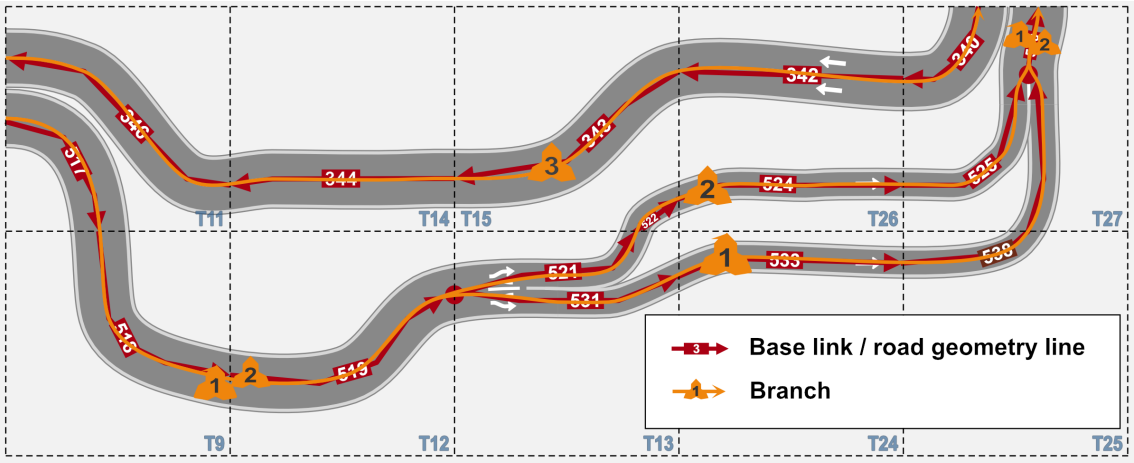
Figure 2-2: Location branches



NDS recommends that one location feature defines a maximum of two location branches. For complex source locations as depicted in *Figure 2-3* on page 13, the compiler can define multiple location features:

- Location feature 1 for the two location branches 1 and 2
- Location feature 2 for location branch 3

Figure 2-3: Complex location branches



2.5 Provider Data

The source locations themselves and the attribute values for these locations may be provided by different providers:

- The *source location provider* defines a set of source locations for a map and makes them available to the dynamic data providers and the database provider.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

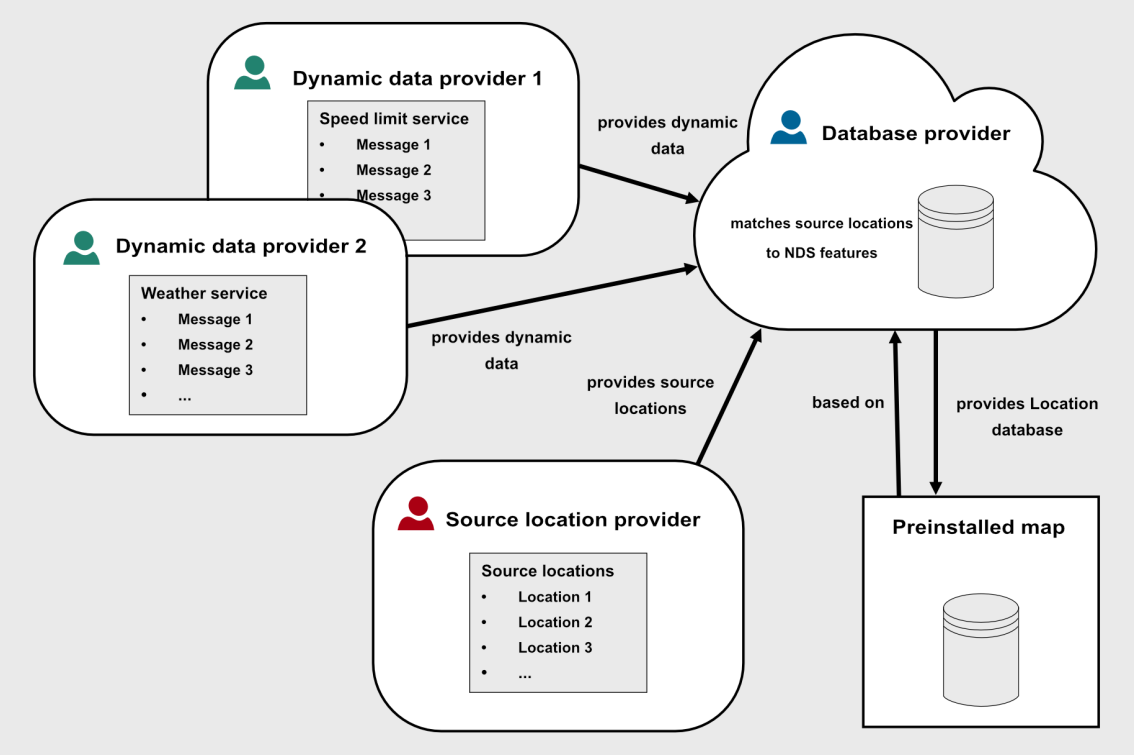
- The *dynamic data provider* sends attribute values for the defined set of source locations via dynamic data services. Each location attribute shall be stored with the corresponding dynamic data service ID. The dynamic data provider does not need to know about the attribution in the preinstalled map.
- The *database provider* matches the source locations to NDS features in the preinstalled map. There shall only be one database provider per product.

The following rules apply:

- Data from multiple dynamic data providers can be combined in the same database.
- Each instance of the Location database shall only contain source locations from one source location provider. Multiple instances of the databases may be provided.

Figure 2-4 on page 14 shows an example of how dynamic location data by different providers is supplied and processed. The database provider aggregates dynamic location data and source locations in the Location database.

Figure 2-4: Cooperation between provider types of the Location extension



If source locations are defined by multiple source location providers, then these locations shall be stored in separate instances of the Location extension.

The `locationSrcLocationSetTable`, `locationDynDataServicesTable`, and `locationStaticDataMappingTable` of the preinstalled map contain the IDs of all source location providers, dynamic data providers, and the database provider, respectively.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2.6 References of Location Features

For each reference of a location feature, the `sameDirectionAsSource` flag indicates whether the location feature refers to the positive or negative digitization direction of the corresponding base link or road geometry line.

Figure 2-5 on page 15 illustrates how to set the `sameDirectionAsSource` flag to model the direction of a location feature in relation to a single base link or road geometry line. In the example, all feature references apply to both digitization directions.

Figure 2-5: Direction of location feature

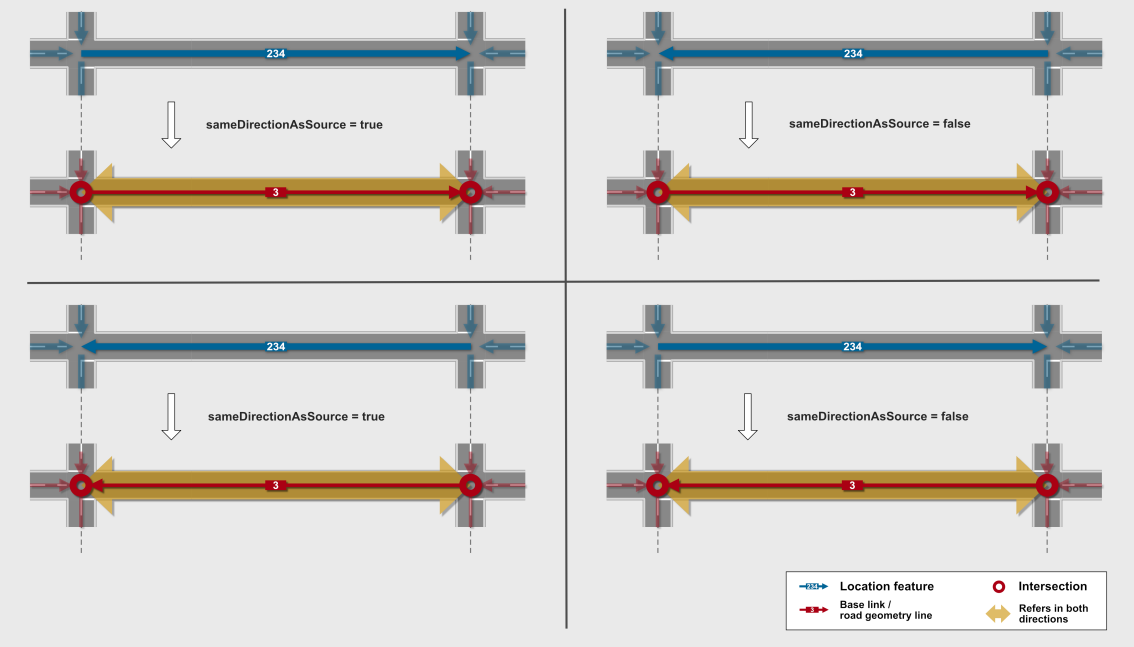
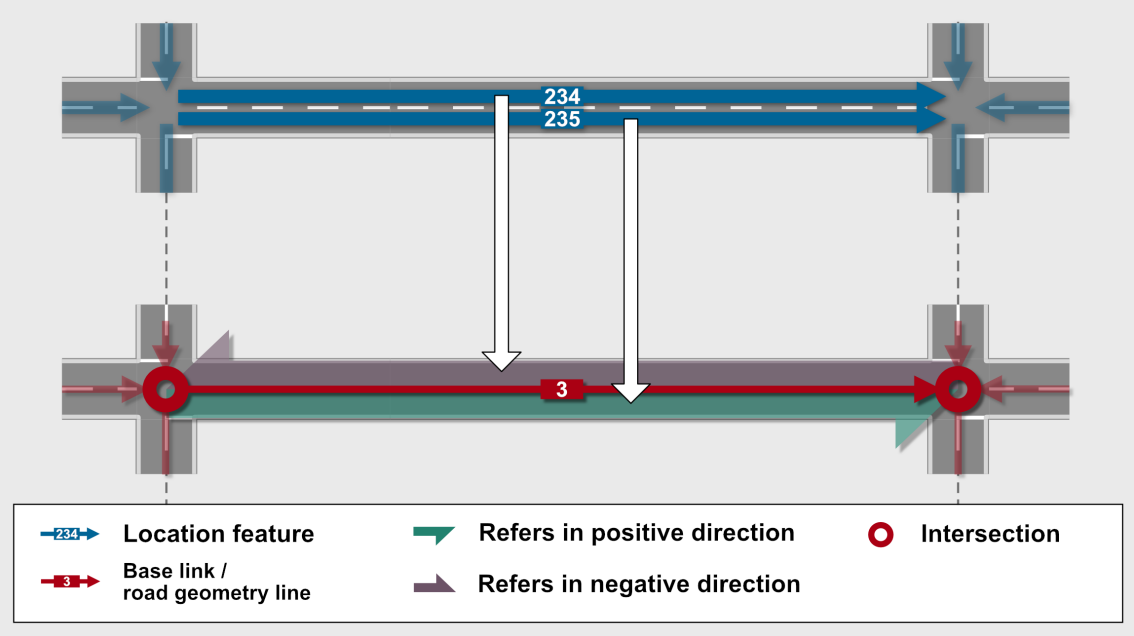


Figure 2-6 on page 16 illustrates how to model feature references that apply to different directions of the base link or road geometry line. In the example, the `sameDirectionAsSource` flag is set to `true` for source location ID 234 and source location ID 235, but the corresponding feature references apply to different directions.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 2-6: Direction of feature reference



2.7 Reversal of Prohibitions

To invalidate prohibitions on a base link, road geometry line, lane, or transition in the preinstalled map, it is necessary to reverse such prohibitions. Flexible attributes for prohibitions that are assigned in the preinstalled map cannot simply be taken away. Therefore, the Location extension defines new flexible attributes, which reverse the meaning of these prohibitions. For example:

- `FTX_LOCATION_ALLOWED_PASSAGE`: Reverses the meaning of the flexible attribute `PROHIBITED_PASSAGE`, transition rules, and the fixed attribute `travelDirection`.
- `FTX_LOCATION_INVALIDATE_SPEED_LIMIT`: Invalidates a speed limit in the preinstalled map for the defined range on a routing feature.
- `FTX_LOCATION_INVALIDATE_OVERTAKING_PROHIBITION`: Invalidates an overtaking prohibition in the preinstalled map for the defined range on a routing feature.

Example: Allow Passage on a Feature that is Closed for Traffic

To allow passage on a base link, road geometry line, lane, or transition that is closed for traffic in the preinstalled map, the flexible attribute `FTX_LOCATION_ALLOWED_PASSAGE` can be used. The dynamic data provider does not need to know about the `PROHIBITED_PASSAGE` value of the affected base link, road geomety line, lane, or transition:

- If `PROHIBITED_PASSAGE` is assigned in the preinstalled map and `FTX_LOCATION_ALLOWED_PASSAGE` is sent, then the meaning

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

of PROHIBITED_PASSAGE shall be overridden with the new FTX_LOCATION_ALLOWED_PASSAGE value.

- If PROHIBITED_PASSAGE is not assigned in the preinstalled map and FTX_LOCATION_ALLOWED_PASSAGE is sent, then nothing changes. The corresponding base link, road geometry line, lane, or transition remains open.

The dynamic data provider shall ensure that the supplied attributes contain all relevant groupings. For example, if a shoulder is opened temporarily for cars and trucks, then FTX_LOCATION_ALLOWED_PASSAGE is grouped with the corresponding FREQUENTLY_USED_VEHICLE_TYPES values.

2.8 Using Locations

The following sections illustrate how locations can be used to map routing features of different maps to each other or as an alternative method to provide updated map attributes.

2.8.1 Mapping Location Features to Routing Features

Figure 2-7 on page 17 illustrates how to map location features to base links or road geometry lines of the Routing building block.

Figure 2-7: Mapping location features to routing features

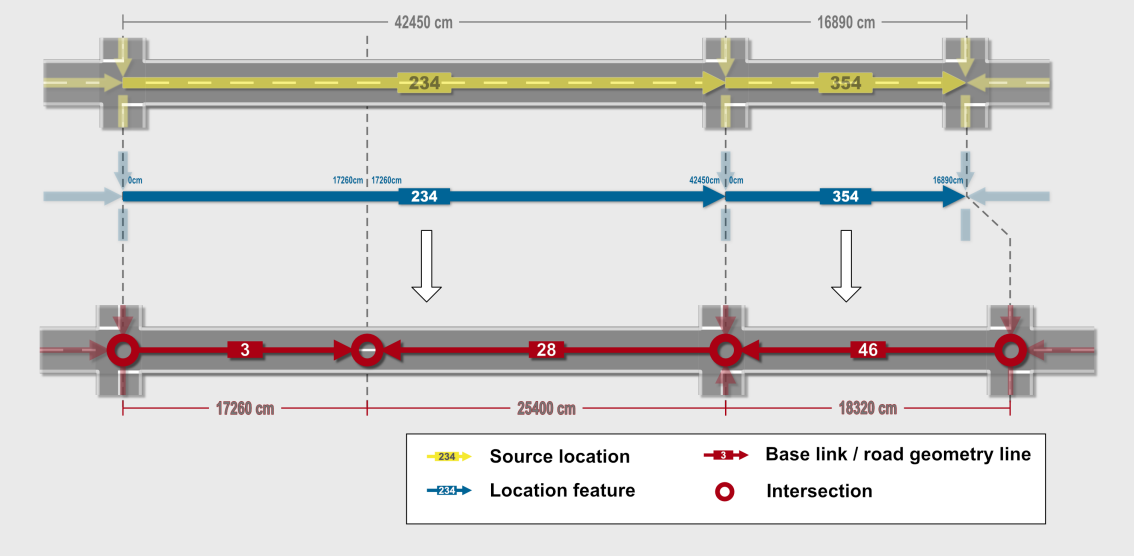


Table 2-1 on page 17 shows how to define location features in the LocationTileTable.

Table 2-1: Filling of LocationTileTable

locationId	sameDirectionAsSource	featureRefList
354	false	link 46

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

locationId	sameDirectionAsSource	featureRefList
		(feature reference points to both directions)
234	true	link 3 (feature reference points to both directions)
	false	link 28 (feature reference points to both directions)

2.8.2 Mapping One Location to Multiple Routing Features

Figure 2-8 on page 18 illustrates how to map one location feature to multiple base links or road geometry lines of the Routing building block.

Figure 2-8: Mapping one location feature to two routing features

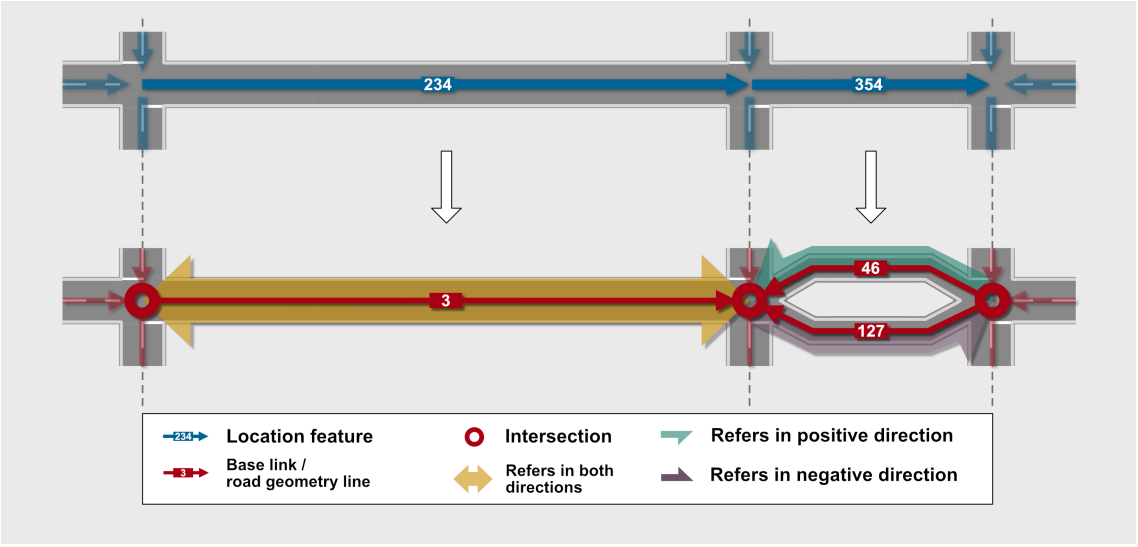


Table 2-2 on page 18 shows how to define location features in the `LocationTileTable`.

Table 2-2: Filling of `LocationTileTable`

locationId	sameDirectionAsSource	featureRefList
234	true	link 3 (feature reference points to both directions)
354	false	link 46

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

locationId	sameDirectionAsSource	featureRefList
		(feature reference points to positive direction)
	false	link 127 (feature reference points to negative direction)

2.8.3 Mapping Multiple Locations to One Routing Feature

Figure 2-9 on page 19 illustrates how to map multiple location features to a single base link or road geometry line of the Routing building block.

Figure 2-9: Mapping two location features to one routing feature

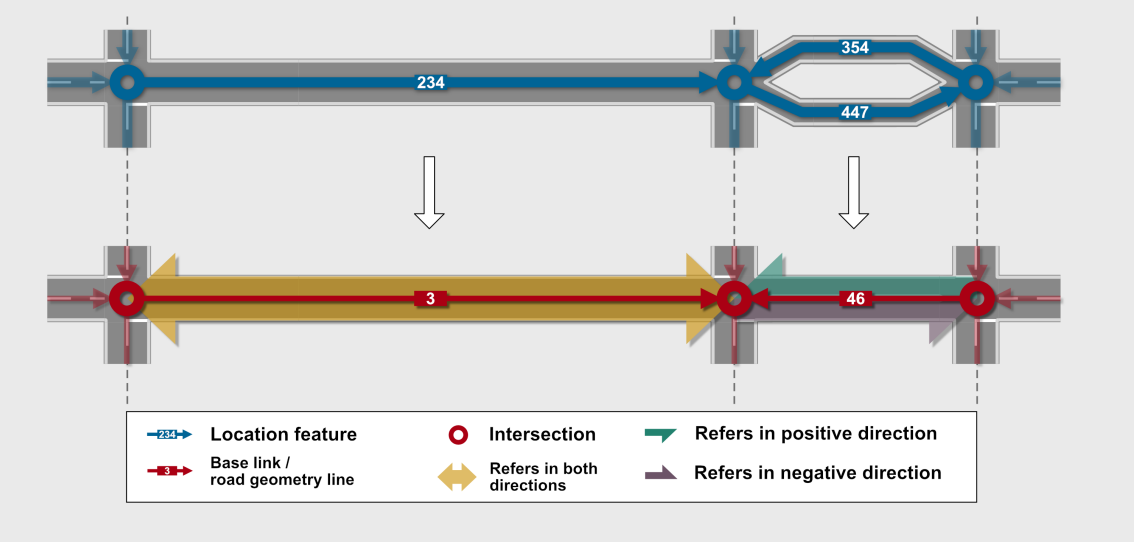


Table 2-3 on page 19 shows how to define location features in the `LocationTileTable`.

Table 2-3: Filling of `LocationTileTable`

locationId	sameDirectionAsSource	featureRefList
234	true	link 3 (feature reference points to both directions)
354	true	link 46 (feature reference points to positive direction)
447	false	link 46

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

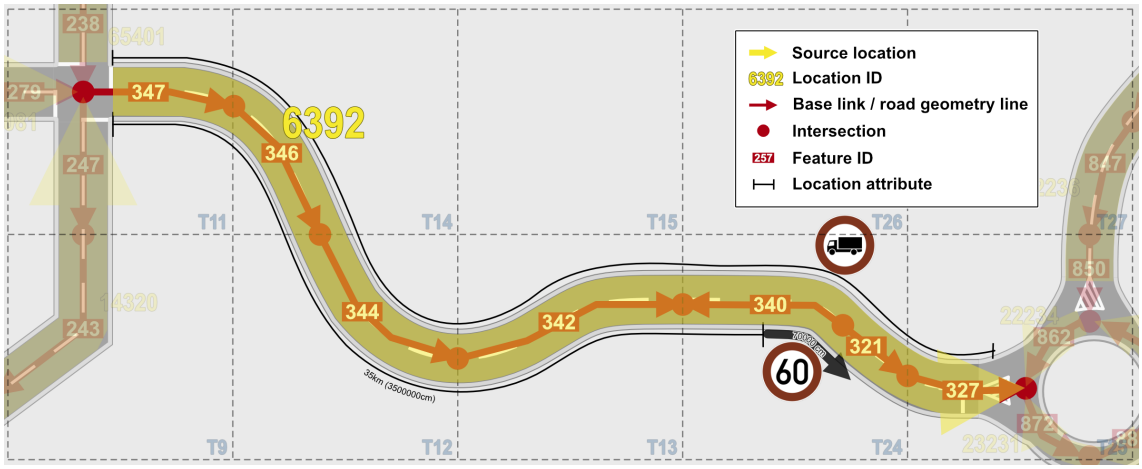
locationId	sameDirectionAsSource	featureRefList
		(feature reference points to negative direction)

2.8.4 Assigning Dynamic Attributes to Source Locations

Two different dynamic data providers supply attributes for the same source location. One provider supplies information about road closures while the other supplies updated speed limits.

Figure 2-10 on page 20 shows a source location that is completely closed for trucks in one direction and has a new speed limit for a short stretch in the other direction.

Figure 2-10: Location with speed limit and prohibition for trucks



Example 1: Truck prohibition

The dynamic data provider ABC provides the data for the truck prohibition using the `locationDynamicDataAttributeTable`:

Table 2-4: Location data for source location 6392

Data structure	Value	Description
`locId`	`6392`	The attributes apply in negative direction on source location 6392. The direction is relative to the location direction.
`assignmentDirection`	`IN_NEGATIVE_DIRECTION`	
`branchId`	`0`	ID of the branch that the attributes apply to. In this example, the location only has one branch.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Data structure	Value	Description
serviceId	56	ID of the dynamic data service that dynamic data provider ABC uses to provide the truck prohibition.
timestamp	"2019-12-09 11:23:00"	The changed attribute values were captured on 09 December 2019 at 11:23 a.m.
startTime	"2019-12-24 00:00:00"	The truck prohibition is valid from 24 December 2019 to the end of 01 January 2020.
expirationTime	"2020-01-01 23:59:59"	
srcLocationVersion	01	Version 01 of the corresponding source location set is used.
Filling of `locationAttrGroupValues` > `rangedAttributeGroup[0]`		
completeLocation	true	The attributes apply to the complete location.
numberOfRelated Locations	0	The attributes apply to one location.
locationAttributeGroup	PROHIBITED_PASSAGE FREQUENTLY_USED_ VEHICLE_TYPES with: ■ isTruck = true ■ isInclusive = true	Trucks are not permitted to use the road in negative direction.

Note

The dynamic data provider supplies the dynamic data using the `locationDynamicDataAttrTileTable`.

Alternatively, the `locationDynamicDataAttrTileTable` can be used. In that case, the `PROHIBITED_PASSAGE` attribute needs to be stored for tiles T11, T12, T13, T14, and T24 separately. The advantage of this method is that each tile is self-contained and the data for each tile can be provided independently.

Example 2: New Speed Limit

The dynamic data provider DEF provides the data for the speed limit using the `locationDynamicDataAttrTileTable`.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

*Table 2-5: Location data for location 6392 in tile _T24_*

Data structure	Value	Description
id	24	Tile ID.
serviceId	1823	ID of the dynamic data service that dynamic data provider DEF uses to provide speed limits.
timestamp	"2019-12-09 11:23:10"	The changed attribute values were captured on 09 December 2019 at 11:23 a.m.
startTime	NULL	The new speed limit is valid immediately.
expirationTime	"2020-01-01 23:59:59"	The new speed limit is valid until 01 January 2020.
srcLocationVersion	02	Version 02 of the corresponding source location set is used.
Filling of `attributeMapList` > `attrMap[0]`		
locationReference	assignmentDirection = IN_POSITIVE_DIRECTION locationId = 6392 branchId = 0	The attributes apply in positive location direction on source location 6392. The source location only has one branch.
Filling of `locationAttrGroupValues` > `rangedAttributeGroup[0]`		
completeLocation	false	The attributes apply to a validity range on source location 6392. Start and end are calculated in cm from the start of the source location.
validityRange	startPosition = 3500000 endPosition = 3570000	
numberOfRelated Locations	0	The attributes apply to one source location.
locationAttributeGroup	SPEED_LIMIT = 60	A speed limit of 60 kmph applies in the validity range.

2.8.5 Overriding Prohibited Passage

Turning right from a major road to a minor road is normally allowed. Because of ongoing construction works that cause a high traffic load in North-South direction, it is decided

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

to prohibit the turn for a while. The major road and the minor road are part of different source locations.

The `locationDynamicDataAttrTileTable` is used to prohibit the transition from one source location to the other.

DataScript locations:

- `nds.ftx.location.flexattr`
- `nds.ftx.location.db`

Figure 2-11 on page 23 shows the intersection with the prohibited transition.

Figure 2-11: Intersection with prohibited right-turn

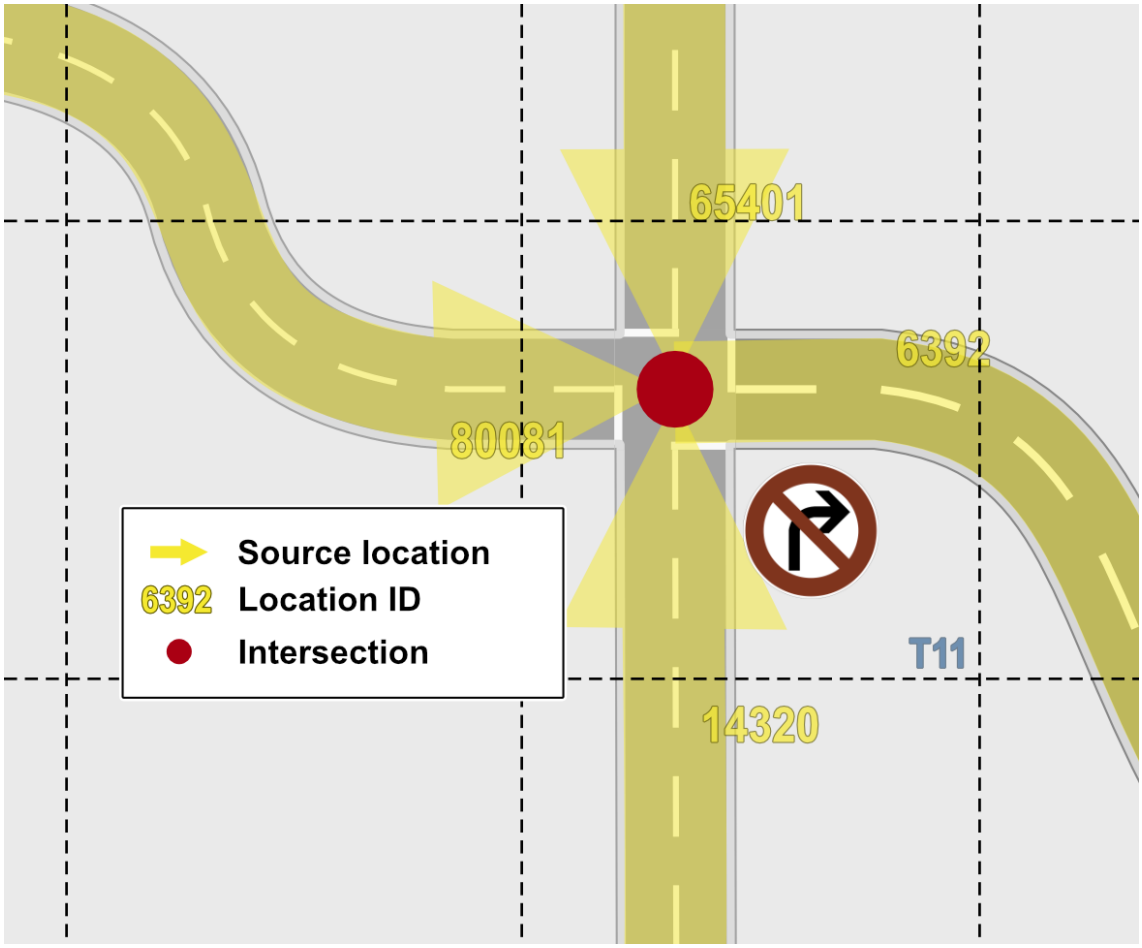


Table 2-6: Location data for `_T11_` in the `locationDynamicDataAttrTileTable`

Data structure	Value	Description
`id`	`11`	Tile ID.
`serviceId`	`585`	ID of the dynamic data service.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Data structure	Value	Description
timestamp	"2019-12-10 21:03:19"	The location data was captured on 10 December 2019.
startTime	NULL	The location data is valid immediately.
expirationTime	"2020-02-02 20:30:00"	The location data is valid until 02 February.
srcLocationVersion	01	Version 01 of the corresponding source location set is used.
Filling of `attributeMapList` > `attrMap[0]`		
locationReference	assignmentDirection = IN_POSITIVE_DIRECTION locationId = 14320 branchId = 0	The attributes apply in positive location direction on source location 14320. This is the start location. The source location only has one branch.
Filling of `locationAttrGroupValues` > `rangedAttributeGroup[0]`		
completeLocation	false	The attributes only apply to the end of start location 14320. Start and end are calculated in cm from the start of the location.
validityRange	startPosition = 482990 endPosition = 482990	
numberOfRelated Locations	1	The attributes apply to a transition between two source locations.
relatedLocation[0]	assignmentDirection = IN_POSITIVE_DIRECTION completeLocation = false validityRange: ■ startPosition = 0 ■ endPosition = 0 locationId = 6392	Directed reference to location 6392. This is the end location. The attributes apply in positive location direction. The attributes only apply to the start of the location. Start and end are calculated in cm from the start of the location.
locationAttributeGroup	PROHIBITED_PASSAGE	The transition is prohibited.

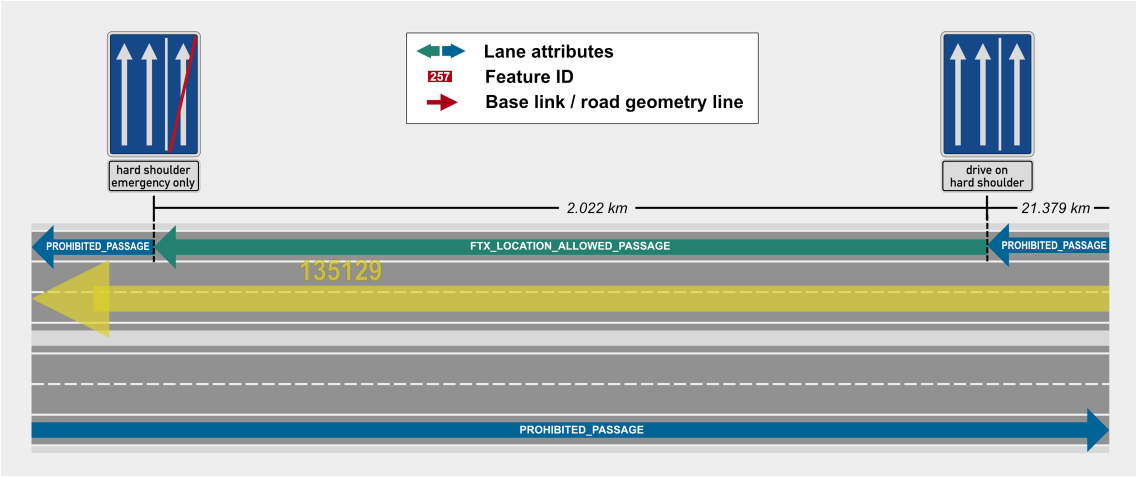
Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2.8.6 Temporarily Allowing Passage on the Shoulder of a Motorway

Hard shoulders on motorways in Germany usually have `PROHIBITED_PASSAGE` for all vehicle types. In times of heavy traffic, it is often allowed to drive on the shoulder. A sign allows the drivers to use the shoulder.

Figure 2-12 on page 25 shows a motorway with a shoulder that is usually closed but can be opened if needed.

Figure 2-12: Example: Allow passage for trucks on road A



The `LocationDynamicDataAttrTileTable` is used to store `FTX_LOCATION_ALLOWED_PASSAGE` for all vehicle types that are allowed to drive on a motorway. The source location is located completely in one tile.

Table 2-7: Location data in the `locationDynamicAttrTileTable`

Data structure	Value	Description
`serviceId`	`67`	ID of the dynamic data service.
`timestamp`	`"2019-12-13 15:00:00"`	The changed attribute values are captured on 13 December 2019 at 3 p.m.
`startTime`	`NULL`	The attributes are valid immediately.
`expirationTime`	`"2020-12-13 19:00:00"`	The attributes are valid for four hours, until 13 December 2019, 7 p.m.
`srcLocationVersion`	`02`	Version 02 of the corresponding source location set is used.
Filling of `attributeMapList` > `attrMap[0]`		

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Data structure	Value	Description
locationReference	assignmentDirection = IN_POSITIVE_DIRECTION locationId = 135129 branchId = 0	The attributes apply in positive location direction on source location 135129. The source location only has one branch.
Filling of locationAttrGroupValues > rangedAttributeGroup[0]		
completeLocation	false	The attributes apply to a validity range on source location 135129. Start and end are calculated in cm from the start of the source location.
validityRange	startPosition = 2137900 endPosition = 2340100	
locationAttributeGroup	FTX_LOCATION_ALLOWED_PASSAGE LANE_RANGE_MASK with type = COMPLETE_RANGE FREQUENTLIY_USED_VEHICLE_TYPES with: ■ isMotorizedVehicle = true ■ isPersonalCar = true ■ isTruck = true ■ isMotorcycle = true ■ isNotLightMotorizedVehicle = true ■ isInclusive = true	PROHIBITED_PASSAGE is reversed for lane 0. The new values only apply to specific motorized vehicle types.

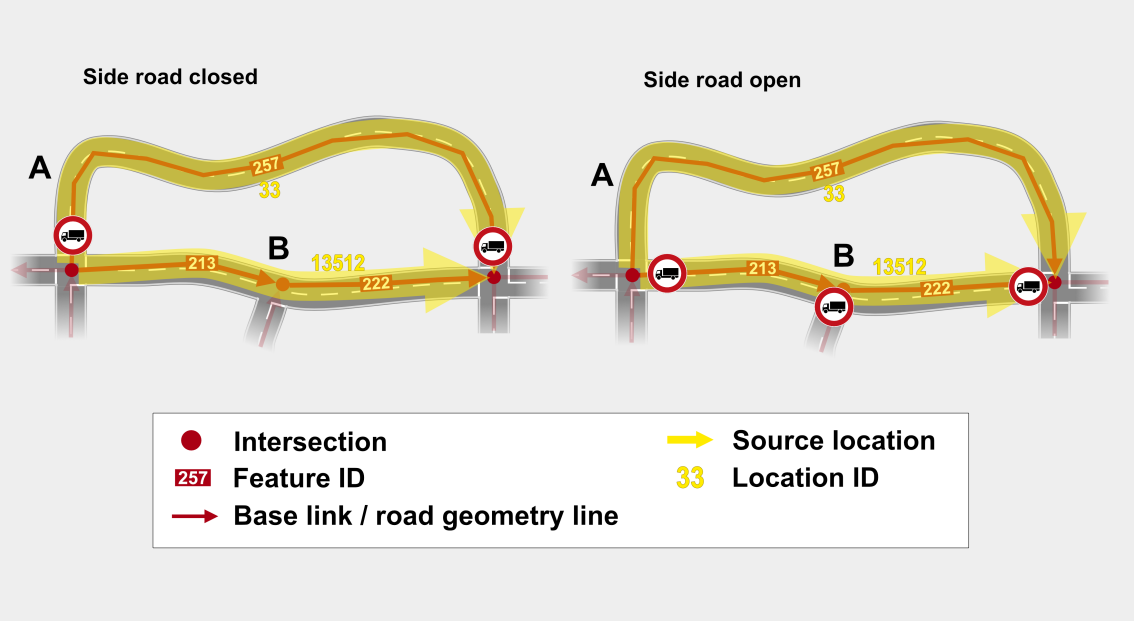
2.8.7 Allowing Passage for Trucks on Side Road

A side road is usually closed for trucks using PROHIBITED_PASSAGE for that vehicle type. Because of construction works on the main road, trucks are allowed to use the side road for a week.

Figure 2-13 on page 27 shows the scenario before and after the closure of the main road for trucks. Road A is the side road that is usually closed for trucks. Road B is the main road that is usually open for trucks.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 2-13: Example: Allow passage for trucks on road A



Both roads are located in the same tile. The dynamic data provider supplies the following data for the `locationDynamicDataAttrTileTable`:

Table 2-8: Location data in the `locationDynamicDataAttrTileTable`

Data structure	Value	Description
`serviceId`	`32`	ID of the dynamic data service.
`timestamp`	`"2020-01-01 23:59:59"`	The changed attribute values are captured on 01 January 2020.
`startTime`	`NULL`	The attributes are valid immediately.
`expirationTime`	`"2020-01-08 23:59:59"`	The attributes are valid until 08 January, 2020.
`srcLocationVersion`	`02`	Version 02 of the corresponding source location set is used.

Table 2-9: Filling of `attributeMapList` > `attrMap[0]` for road A

Data structure	Value	Description
`locationReference`	`assignmentDirection = IN_BOTH_DIRECTIONS` `locationId = 33`	The attributes apply to road A, that is, location 33, in both directions. The source location only has one branch.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Data structure	Value	Description
	branchId = 0	
Filling of locationAttrGroupValues > rangedAttributeGroup[0]		
completeLocation	true	The attributes apply to the complete source location.
locationAttributeGroup	FTX_LOCATION_ALLOWED_PASSAGE	PROHIBITED_PASSAGE is reversed.
Filling of locationAttrGroupValues > rangedAttributeGroup[1]		
completeLocation	true	The attributes apply to the complete source location.
locationAttributeGroup	FREQUENTLIY_USED_VEHICLE_TYPES with isTruck = true and isInclusive = true	The reversal of PROHIBITED_PASSAGE applies to trucks.

Table 2-10: Filling of attributeMapList > attrMap[1] for road B

Data structure	Value	Description
locationReference	assignmentDirection = IN_BOTH_DIRECTIONS locationId = 13512	The attributes apply to road B, that is, location 13512, in both directions.
Filling of locationAttrGroupValues > rangedAttributeGroup[0]		
completeLocation	true	The attributes apply to the complete location.
locationAttributeGroup	PROHIBITED_PASSAGE	The passage is prohibited.
Filling of locationAttrGroupValues > rangedAttributeGroup[1]		
completeLocation	true	The attributes apply to the complete location.
locationAttributeGroup	FREQUENTLIY_USED_VEHICLE_TYPES with isTruck = true and isInclusive = true	The prohibition applies to trucks.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

3 Appendix - Integration User Guide for Fast Track Extensions

NDS fast track extensions add functionality to existing NDS versions without changing the released specification. NDS partners develop and test the fast track extensions. This way, the implementation can become mature and stable before it is added to the NDS standard as an official functionality. Fast track extensions both encourage innovation in NDS and increase stability of NDS versions.

Example An example of a fast track extension is the Auto Drive functionality.

The fast track extension concept ensures compatibility of extensions with the NDS releases. It provides sufficient information to automatically integrate extensions into a specific NDS release.

3.1 Architecture of Fast Track Extensions

The following data structures in the NDS physical model enable fast track extensions:

- FTX compatibility list file
- Integration file for each extension
- Package `nds.ftx.all`
- Extension packages

3.1.1 FTX Compatibility List File

The FTX compatibility list file contains versioning information for all fast track extensions supported by the given NDS release. The FTX compatibility list files for all NDS versions are maintained in the following repository on Stash: Infrastructure Software. For each NDS release, there is a subfolder in `ndsbuildtools/conf` that contains the corresponding file.

Example: https://stash.nds-association.org/projects/TOOL/repos/infrastructure-software/browse/ndsbuildtools/conf/2.5.3/nds-ftx-compatibility-2.5.3-ftxlist.xml

Table 3-1 on page 30 describes the elements of the FTX compatibility list file.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Table 3-1: Elements of the FTX compatibility list file

Element	Parent element	Cardinal quantifier	Description
`NdsFtxList`	n/a	1,1	Root element. Attributes: ■ `@ndsrelease`: Version of the NDS release.
`NdsFtxRef`	`NdsFtxList`	1,M	Version information of a compatible fast track extension. Attributes: ■ `@id`: Unique identifier of the fast track extension as assigned by the NDS consortium. ■ `@name`: Name of the fast track extension. `@name` shall be identical to the name of the main directory of the fast track extension.
`url`	`NdsFtxRef`	1,1	URL pointing to the NDS repository where the fast track extension is developed.
`version`	`NdsFtxRef`	1,M	Available version of the fast track extension. Multiple versions may be listed.

Table 3-1 on page 30 shows the FTX compatibility list file for NDS 2.5.3.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 3-1: Example of FTX compatibility list file

```
<NdsFtxList ndsrelease="2.5.3" ndsftxrelease="2.5.3.0">

    <NdsFtxRef id="0" name="nds-standard">
        <url>https://stash.nds-association.org/scm/nds/nds-standard.git</url>
        <version>2.5.3</version>
    </NdsFtxRef>

    <NdsFtxRef id="7" name="locobj">
        <url>https://stash.nds-association.org/scm/nds/ftx-localization-objects.git</url>
        <version>2.2</version>
    </NdsFtxRef>

    <NdsFtxRef id="8" name="obstacles">
        <url>https://stash.nds-association.org/scm/nds/ftx-obstacles.git</url>
        <version>1.0</version>
        <version>2.0</version>
        <version>2.1</version>
        <version>2.2</version>
    </NdsFtxRef>

    <NdsFtxRef id="10" name="occupancy">
        <url>https://stash.nds-association.org/scm/nds/ftx-occupancy.git</url>
        <version>0.1</version>
        <version>1.0</version>
    </NdsFtxRef>
```

3.1.2 Integration File

The integration file contains information on how to integrate an extension into an NDS release. The integration of extensions may differ between NDS releases. This is why the integration file depends on the NDS release version. The integration file is part of the extension release. There must be a dedicated integration file for each extension.

File location: `<extension_dir>/ftxintegration.xml`

Table 3-2 on page 31 describes the elements of the integration file.

Table 3-2: Elements of the integration file

Element	Parent element	Cardinal quantifier	Description
`NdsFtx`	n/a	1,1	Root element of the integration file that defines main properties of the extension. Attributes: ■ `@id`: Unique identifier of the fast track extension as assigned by the NDS consortium. ■ `@version`: Version of the fast track extension.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Element	Parent element	Cardinal quantifier	Description
			■ `@name`: Name of the fast track extension, which shall be identical to the name of the main directory of the fast track extension.
`DsCode`	`NdsFtx`	1,M	Defines structure and integration of the extension. `DsCode` contains the extension's DataScript code. For more informationa, refer to *Table 3-3* on page 32.

Table 3-3 on page 32 shows the attributes of the `DsCode` element.

Table 3-3: Attributes of DsCode

Attribute	Description
`@type`	Type of the integration element: ■ `const`: Definition of a constant. ■ `import`: Definition of an import command. DataScript package that is modified by the extension. ■ `choicecase`: Definition of a case. ■ `enumval`: Definition of an enumeration value. ■ `enumannotation`: Definition of a flexible attribute annotation.
`@package`	
`@typename`	DataScript type that is modified by the extension. This attribute is needed for the `@type` values `choicecase`, `enumval`, and `enumannotation`.
`@tag`	Defines an insertion marker within the DataScript package specified in `@package` and the DataScript structure specified in `@typename`. The insertion marker is a line in the DataScript package after which the content of the `DsCode` element is inserted. The insertion marker line is a DataScript comment in the form: `//@ftx <tag>`, for example, `//@ftx main.import` The special tags "`{`" and "`}`" indicate that the DataScript code is inserted at the start or end of the structure, respectively. An empty tag defaults to "`{`".
`@flexattr`	Defines the name of the flexible attribute that is modified. This attribute is needed for `@type` = `enumannotation`.

Figure 3-2 on page 33 shows an integration file for the Auto Drive extension.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 3-2: Example of integration file

```
<NdsFtx id="2" version="2.0" name="AutoDrive">

  <DsCode type="const" package="nds.ftx.all" tag="ftx.id">
    /** @see AutoDriveDatabase */
    const NdsFtxId FTXID_AUTO_DRIVE = 2;

    /** @see AutoDriveDatabase */
    AutoDriveDataModelVersion
    {
       function DataModelVersion value()
       {
          return 13719;
       }
    };
  </DsCode>

  <DsCode type="import" package="nds.ftx.all" tag="main.import">
    import nds.ftx.autodrive.db.*;
    import nds.common.DataModelVersion;
  </DsCode>

  <DsCode type="import" package="nds.common.flexattr.valuecodes" tag="fa.import">
    import nds.ftx.autodrive.attrdefs.*;
  </DsCode>

  <DsCode type="enumval" package="nds.common.flexattr.valuecodes" typename="AttributeTypeCode" tag="fa.const">
 /**! FTX_ABS_GEO_EXTENT_3D:1000
   *
   * description:
   *  "Defines absolute geographic 3D position and extents of a feature.";
   *
   * default:
   *  "If not assigned to a feature then a feature it's absolute geographic
   *   3D position and extents are unknown.";
   *
   *  category:
   *    @dse Category.GUIDANCE;
   *  group_role:
   *    @dse GroupRole.SECONDARY;
   *  feature:
   *    @dse ReferenceType.ALL_TILE_FEATURES;
   *    @dse ReferenceType.ROUTING_LINK_BOTH_DIRECTIONS; #(baselink)
   *    @dse ReferenceType.ROUTING_LINK_DIRECTED; #(baselink)
   *    @dse ReferenceType.ROUTING_ROAD_GEO_LINE_BOTH_DIRECTIONS;
   *    @dse ReferenceType.ROUTING_ROAD_GEO_LINE_DIRECTED;
   *    @dse ReferenceType.ROUTING_SIMPLE_INTERSECTION; #(if the arriving links have a traffic light for all transitions)
   *    @dse ReferenceType.ROUTING_SIMPLE_INTERSECTION_TRANSITION; #(if the traffic light is relevant for the transition only)
   *  grouping:;
   *  layer:
   *    @dse autoDriveTileTable.ndsData.attributeMaps[13];
   */
  FTX_ABS_GEO_EXTENT_3D = 1000,
```

For a detailed description of the integration process, refer to *Section 3.2: "Integrating Fast Track Extensions"* on page 36.

3.1.3 Package `nds.ftx.all`

The package `nds.ftx.all` imports the extensions to NDS. The package contains main import entries, DataScript constant values that identify the extensions, and the automatic retrieval function for the NDS data model version.

The NDS standard package `nds.all` imports `nds.ftx.all` through import entries.

By default, `nds.ftx.all` does not contain any import entries, only a list of possible extensions (see *Section 3.1.2: "Integration File"* on page 31). No extensions are integrated by default.

Figure 3-3 on page 34 shows `nds.ftx.all` before the integration of extensions.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 3-3: Default `nds.ftx.all` package

```
83  ////////////////////////////////////////////////////////////////////////////
84  // Top level imports for extensions
85  ////////////////////////////////////////////////////////////////////////////
86
87
88  //@ftx main.import
89
90
91  ////////////////////////////////////////////////////////////////////////////
92  // List of identifiers for extensions
93  ////////////////////////////////////////////////////////////////////////////
94  /** Id of an FTX. It is assigned by the NDS consortium.*/
95  subtype uint32 NdsFtxId;
96
97
98  //@ftx ftx.id
99
```

After the integration of extensions, `nds.ftx.all` contains corresponding import entries and constant values.

Figure 3-4 on page 35 shows `nds.ftx.all` after the integration of the Auto Drive extension.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Figure 3-4: Filled `nds.ftx.all` package

```
83  ////////////////////////////////////////////////////////////////////////////
84  // Top level imports for extensions
85  ////////////////////////////////////////////////////////////////////////////
86
87
88  //@ftx main.import
89  import nds.ftx.autodrive.db.*;//# ftx: AutoDrive v2.0
90      import nds.common.DataModelVersion;//# ftx: AutoDrive v2.0
91
92
93  ////////////////////////////////////////////////////////////////////////////
94  // List of identifiers for extensions
95  ////////////////////////////////////////////////////////////////////////////
96  /** Id of an FTX. It is assigned by the NDS consortium.*/
97  subtype uint32 NdsFtxId;
98
99
100 //@ftx ftx.id
101 /** @see AutoDriveDatabase */
102     const NdsFtxId FTXID_AUTO_DRIVE = 2;//# ftx: AutoDrive v2.0
103
104     /** @see AutoDriveDatabase */
105     AutoDriveDataModelVersion
106     {
107        function DataModelVersion value()
108        {
109           return 13702;//# ftx: AutoDrive v2.0
110        }
111     };//# ftx: AutoDrive v2.0
112
```

3.1.4 Package `nds.common.flexattr.valuecodes`

The NDS standard package `nds.common.flexattr.valuecodes` contains the DataScript structure `AttributeTypeCode`. Each entry consists of the attribute type code and the corresponding annotation.

The attribute type codes are assigned as follows:

- Attribute type codes 0 – 999

  Reserved for copies of NDS standard attributes. These copies enable extensions to reuse definitions of standard attributes, including grouping, layer assignment rules, etc. They also enable grouping of standard attributes with extension attributes.

- Attribute type codes 1000 – 65535

  May be assigned to extension attributes by the NDS consortium.

By default, `nds.common.flexattr.valuecodes` is filled with standard attribute codes of the associated NDS release. No extension attributes are integrated by default.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

After the integration of extensions, `nds.common.flexattr.valuecodes` contains attribute definitions of the extension in `AttributeTypeCode` and `AttributeValue`.

Note Extensions shall not change the standard attribute type codes and cases in the range 0 - 999. Extensions may, however, change annotations of standard attributes according to the integration rules provided in the `DsCode` element of type `DsCode.type == enumannotation`. This `DsCode` element is stored in the extension integration file (see *Section 3.1.2: "Integration File"* on page 31). Modified annotations enable compilers to group standard attributes with extension attributes, assign standard attribute to extension layers, etc.

3.1.5 Extension Packages

Extension packages define extension structures. The following rules apply to extension definitions:

- Each extension package shall have the prefix `nds.ftx.<name>`. In the extension integration file, the element `NdsFtx.name` defines `<name>`. For example, `nds.ftx.autodrive`.
- The extension shall contain at least one package whose name is identical to the definition in the `DsCode` element with the type `DsCode.type == import` and the tag `DsCode.tag ==` "main.import".

Example of Extension Package

In case of *Figure 3-5* on page 36, the package `nds.ftx.autodrive.db` for the Auto Drive extension must exist. This is the main package of the extension. It can import further packages of the extension.

Figure 3-5: Import of extension packages example

```
63  <DsCode type="import" package="nds.ftx.all" tag="main.import">
64    import nds.ftx.autodrive.db.*;
65    import nds.common.DataModelVersion;
66  </DsCode>
```

3.2 Integrating Fast Track Extensions

The integration of a fast track extension into NDS consists of the following procedures:

1. Check whether the fast track extension that you want to use is compatible with your NDS version, see *Verifying the Compatibility of a Fast Track Extension with an NDS Version* on page 37.
2. Perform a custom build of the fast track extension and download the files, see *Performing a Custom Build of a Fast Track Extension* on page 37.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Verifying the Compatibility of a Fast Track Extension with an NDS Version

1. In your browser, log in to Stash and navigate to https://stash.nds-association.org/projects/TOOL/repos/infrastructure-software/browse/ndsbuildtools/conf/.
2. Open the FTX compatibility list file for your NDS version, for example, `2.5.3 > nds-ftx-compatibility-2.5.3-ftxlist.xml`.
3. Take a note of the ID and the version that you want to use.
   Example: For the Obstacles FTX, the ID is 8 and the latest version is 2.2.

```
<NdsFtxRef id="8" name="obstacles">
    <url>https://stash.nds-association.org/scm/nds/ftx-obstacles.git</url>
    <version>1.0</version>
    <version>2.0</version>
    <version>2.1</version>
    <version>2.2</version>
</NdsFtxRef>
```

Note If you cannot find the required FTX in the list, contact the NDS support team.

Performing a Custom Build of a Fast Track Extension

1. In your browser, log in to Bamboo at https://bamboo.nds-association.org/.
2. In the project "NDS", click **Custom FTX Integration**.
3. Select **Run** > **Run customized**.
4. In the **Revision** field, enter the NDS version that you want to use, for example, "2.5.3".
5. Click **Override a variable**, select **ftxlist** and enter the ID and version number of the FTX, for example, "8:2.2".

Note You can build multiple fast track extensions at the same time by entering multiple values in the **ftxlist** value field.
Example: Enter "8:2.2 12:1.0" to build the Obstacles and the Signatures extensions.

6. Click **Override a variable** again, select **ndsversion** and enter the NDS version number again, for example, "2.5.3".
7. Click `Run`.
   A status bar indicates that the build is active.
8. When the build has finished, click **Artifacts** and then **Physical Model + HTML Documentation**.
   The integrated product with the fast track extension is downloading to your computer.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

Example of Integrated Auto Drive Extension

The following tables illustrate how the compiler fills the entries for the Auto Drive extension version 2.0 in the `buildingBlockTable`, `urBuildingBlockVersionTable`, and `dataModelVersionTable` of an NDS product.

Table 3-4: Content of the `buildingBlockTable`

buildingBlockId	buildingBlockName	buildingBlockType	buildingBlockDetailedType	globalBuildingBlockMetadata
0	Shared Data	0	[...]	[...]
[...]	[...]	[...]	[...]	[...]
10	Auto Drive FTX	255	BLOB	[...]

Table 3-5: Content of the `urBuildingBlockVersionTable`

updateRegionId	BuildingBlockId	versionId	[...]	uri	dataModelVersionId
7	0	1	[...]	UR_7/SHARED.NDS	1
[...]	[...]	[...]	[...]	[...]	[...]
7	10	1	[...]	UR_7/AUTODRV.NDS	2

Table 3-6: Content of the `dataModelVersionTable`

dataModelVersionId	dataModelVersion	datascriptVersionName	hasAdaptation	uri
1	12738	NDS 2.4.2	0	[...]
2	13719	AutoDrive FTX 2.0	0	[...]

3.2.1 Using the Search-and-Insert Method

The search-and-insert method helps the compiler to identify the DataScript files that require import commands. The method is used to integrate extensions into NDS.

1. Open the `ftxintegration.xml` file of the fast track extension.

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

2. Check the following attributes of the next `DsCode` element:
   - `@package`: Shows the name of the NDS DataScript file where the content of the `DsCode` element shall be inserted.
   - `@tag`: Shows the name of the insertion marker in the NDS DataScript file.
3. Open the NDS DataScript file.
4. In the NDS DataScript file, navigate to the insertion marker.
5. Copy and paste the content of the `DsCode` element below the insertion marker.
6. Add the name and version of the fast track extension after the inserted `DsCode` element.
   Get the values from the `@name` and `@version` attributes of the `NdsFtx` element and add them as follows: `//# ftx: <name> v<version>`.
7. Repeat steps 2 to 6 for each `DsCode` element of the `ftxintegration.xml` file.
8. Save and close the NDS DataScript files.

Example of the Search-and-Insert Method

The following shows an example of the search-and-insert method based on the Auto Drive extension.

Figure 3-6 on page 39 shows the element `NdsFtx` and one `DsCode` element from the Auto Drive extension including all attributes.

Figure 3-6: Example DsCode

```
47  <NdsFtx id="2" version="2.0" name="AutoDrive">
48
49    <DsCode type="const" package="nds.ftx.all" tag="ftx.id">
50      /** @see AutoDriveDatabase */
51      const NdsFtxId FTXID_AUTO_DRIVE = 2;
52
53      /** @see AutoDriveDatabase */
54      AutoDriveDataModelVersion
55      {
56          function DataModelVersion value()
57          {
58              return 13719;
59          }
60      };
61    </DsCode>
62
63    <DsCode type="import" package="nds.ftx.all" tag="main.import">
64      import nds.ftx.autodrive.db.*;
65      import nds.common.DataModelVersion;
66    </DsCode>
```

The attributes have the following values:

- `@package` = "nds.ftx.all"
- `@tag` = "main.import"
- `@version` = "2.0"

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.

- `@name` = "autodrive"

Proceed as follows:

1. Open the file `nds/ftx/all.ds`.
2. Navigate to the insertion marker `// @ftx.main.import` as shown in *Figure 3-7* on page 40.

Figure 3-7: Example before import

```
83  ////////////////////////////////////////////////////////////////////////////
84  // Top level imports for extensions
85  ////////////////////////////////////////////////////////////////////////////
86
87
88  //@ftx main.import
89
```

3. Copy the content of `DsCode` (`import nds.ftx.autodrive.db.*;`) and paste it below the insertion marker (`//@ftx main.import`).
4. Add name and version of the fast track extension after the content of `DsCode` as shown in *Figure 3-8* on page 40.

Figure 3-8: Example after import

```
83  ////////////////////////////////////////////////////////////////////////////
84  // Top level imports for extensions
85  ////////////////////////////////////////////////////////////////////////////
86
87
88  //@ftx main.import
89  import nds.ftx.autodrive.db.*;//# ftx: AutoDrive v2.0
90      import nds.common.DataModelVersion;//# ftx: AutoDrive v2.0
91
```

Copyright 2020 © - NDS e.V. All Rights Reserved.
No part of this document may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm without permission in writing from the author.