

Australian/New Zealand Standard™

**Geographic information—Location-
based services—Tracking and
navigation**



AS/NZS ISO 19133:2006

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-004, Geographical Information/Geomatics. It was approved on behalf of the Council of Standards Australia on 29 September 2006 and on behalf of the Council of Standards New Zealand on 13 October 2006.
This Standard was published on 24 October 2006.

The following are represented on Committee IT-004:

ACT Planning and Land Authority
ANZLIC – the Spatial Information Council
Association of Aerial Surveyors Australia
Association of Crown Research Institutes, New Zealand
Australasian Fire Authorities Council
Australian Antarctic Division
Australian Bureau of Statistics
Australian Hydrographic Office
Australian Map Circle
Australian Spatial Information Business Association
CSIRO Exploration and Mining
Department of Administrative and Information Services, SA
Department of Conservation, NZ
Department of Defence
Department of Natural Resources, Mines and Water, Qld
Department of Planning and Infrastructure, NT
Department of Primary Industries, Water and Environment, Tas.
Geoscience Australia
Land Information, NZ
Land Victoria
Spatial Sciences Institute
Telecom New Zealand
Western Australian Land Information System

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.standards.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR 06311.

Australian/New Zealand Standard™

Geographic information—Location-based services—Tracking and navigation

First published as AS/NZS ISO 19133:2006.

COPYRIGHT

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia, GPO Box 476, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 7816 X

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-004, Geographical Information/Geomatics.

The objective of this Standard is to provide users and designers with a description of the data types, and operations associated with those types, for the implementation of tracking and navigation services.

This Standard is identical with, and has been reproduced from ISO 19133:2005, *Geographic information—Location-based services—Tracking and navigation*.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover
- (b) In the source text ‘this International Standard’ should read ‘this Australian/New Zealand Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO		AS	
3166	Codes for the representation of names of countries and their subdivisions	2632	Codes for the representation of names of countries and their subdivisions
3166-1	Part 1: Country codes	2632.1	Part 1: Country codes
		AS/NZS ISO	
19107	Geographic information—Spatial schema	19107	Geographic information—Spatial schema
19108	Geographic information—Temporal schema	19108	Geographic information—Temporal schema
19109	Geographic information—Rules for application schema	19109	Geographic information—Rules for application schema
19111	Geographic information—Spatial referencing by coordinates	19111	Geographic information—Spatial referencing by coordinates
19112	Geographic information—Spatial referencing by geographic identifiers	19118	Geographic information—Spatial referencing by geographic identifiers
19118	Geographic information—Encoding	19118	Geographic information—Encoding

The terms ‘normative’ and ‘informative’ have been used in this Standard to define the application of the annex to which they apply. A ‘normative’ annex is an integral part of a Standard, whereas an ‘informative’ annex is only for information and guidance.

CONTENTS

Page

1	Scope	1
2	Conformance	1
3	Normative references	2
4	Terms and definitions.....	2
5	Abbreviated terms and UML notation.....	6
5.1	Abbreviated terms	6
5.2	UML notation	6
6	Tracking	7
6.1	Semantics	7
6.2	Package: Tracking Service	7
6.3	Package: Point Estimates	21
6.4	Package: Location Transformation.....	26
6.5	Package: Measured Coordinates	27
6.6	Package: Linear Reference Systems	32
7	Navigation.....	39
7.1	Semantics	39
7.2	Cost Functions and algorithms.....	41
7.3	Package: Navigation Service	42
7.4	Package: Cost Function	55
7.5	Package: Preferences.....	68
8	Address Model	70
8.1	Semantics	70
8.2	Package: Address.....	70
8.3	Package: Address Elements.....	74
9	Network.....	85
9.1	Semantics	85
9.2	Package: Network Model	85
9.3	Package: Turn and Junction.....	89
9.4	Package: Constraint and Advisory	95
9.5	Package: Link	108
9.6	Package: Network Position.....	111
9.7	Package: Route	112
9.8	Package: Combined Networks	117
10	Basic implementation packages	120
10.1	Package: Feature Data Model.....	120
10.2	Package: New Basic Types.....	124
Annex A	(normative) Abstract test suite.....	127
Annex B	(informative) Directed weighted graphs and their algorithms	134
Annex C	(informative) View of Standard in terms of RM-ODP Services	137
Bibliography		139

Figure 1 — Tracking packages	7
Figure 2 — Context Diagram: TK_Position	8
Figure 3 — Context Diagram: TK_MobileSubscriber	9
Figure 4 — Context Diagram: TK_TrackingLocation.....	10
Figure 5 — Context Diagram: TK_TrackingService.....	11
Figure 6 — Context Diagram: TK_PositionType	12
Figure 7 — Context Diagram: TK_TrackingLocationSequence	13
Figure 8 — Context Diagram: TK_Trigger	14
Figure 9 — Context Diagram: TK_PeriodicTrigger	15
Figure 10 — Context Diagram: TK_TransitionTrigger.....	16
Figure 11 — Context Diagram: TK_TrackingLocationMetadata.....	17
Figure 12 — Context Diagram: TK_Transition	18
Figure 13 — Context Diagram: TK_QualityOfPosition	19
Figure 14 — Context Diagram: TK_Accuracy	20
Figure 15 — Context Diagram: TK_AccuracyStatement.....	20
Figure 16 — Point Estimate classes.....	21
Figure 17 — Geometric interpretations of point estimate types	22
Figure 18 — Context Diagram: EG_PointEstimateCircle	22
Figure 19 — Context Diagram: EG_PointEstimateEllipse	23
Figure 20 — Context Diagram: EG_PointEstimateArc	24
Figure 21 — Context Diagram: EG_PointEstimateSphere	25
Figure 22 — Context Diagram: EG_PointEstimateEllipsoid	26
Figure 23 — Context Diagram: LT_LocationTransformationService.....	27
Figure 24 — Measure Position.....	28
Figure 25 — Measured Coordinate Systems.....	29
Figure 26 — Context Diagram: MC_MeasurePosition.....	30
Figure 27 — Context Diagram: MC_CoordinateSystem	30
Figure 28 — Context Diagram: MC_CoordinateReferenceSystem	31
Figure 29 — LRS classes	32
Figure 30 — Context Diagram: LR_PositionExpression.....	33
Figure 31 — Context Diagram: LR_LinearReferenceMethod	35
Figure 32 — Context Diagram: LR_OffsetDirection.....	35
Figure 33 — Context Diagram: LR_ReferenceMarker	36
Figure 34 — Context Diagram: LR_Feature.....	37
Figure 35 — Context Diagram: LR_Element.....	37
Figure 36 — Context Diagram: LR_OffsetExpression.....	38
Figure 37 — Navigation Packages.....	39
Figure 38 — Example of route from one link position to another.....	40

Figure 39 — Services.....	42
Figure 40 — Context Diagram: NS_NavigationService.....	43
Figure 41 — Context Diagram: NS_RouteRequest.....	46
Figure 42 — Context Diagram: NS_Instruction.....	48
Figure 43 — Context Diagram: NS_InstructionList.....	49
Figure 44 — Context Diagram: NS_RouteResponse.....	50
Figure 45 — Context Diagram: NS_CostedTurn.....	51
Figure 46 — Context Diagram: NS_RenderingService.....	51
Figure 47 — Context Diagram: NS_RenderingRequest.....	52
Figure 48 — Context Diagram: NS_RenderingResponse.....	53
Figure 49 — Context Diagram: NS_RenderingType.....	53
Figure 50 — Context Diagram: NS_CostedLink.....	54
Figure 51 — Context Diagram: NS_CostFunctionCode.....	54
Figure 52 — Context Diagram: NS_RouteRequestType.....	55
Figure 53 — Context Diagram: NS_CostFunction.....	59
Figure 54 — Context Diagram: NS_CostElements.....	59
Figure 55 — Context Diagram: NS_MonetaryCost.....	60
Figure 56 — Context Diagram: NS_Tolls.....	61
Figure 57 — Context Diagram: NS_Fares.....	61
Figure 58 — Context Diagram: NS_Time.....	62
Figure 59 — Context Diagram: NS_TravelTime.....	62
Figure 60 — Context Diagram: NS_WaitingTime.....	63
Figure 61 — Context Diagram: NS_Counts.....	64
Figure 62 — Context Diagram: NS_NumberManeuvers.....	64
Figure 63 — Context Diagram: NS_NumberTurns.....	65
Figure 64 — Context Diagram: NS_NumberTransfers.....	65
Figure 65 — Context Diagram: NS_Distance.....	66
Figure 66 — Context Diagram: NS_WeightedCost.....	67
Figure 67 — Context Diagram: NS_CostFunctionTerm.....	68
Figure 68 — Context Diagram: NS_RoutePreferences.....	68
Figure 69 — Context Diagram: NS_AvoidList.....	69
Figure 70 — Leaf packages of the Address Model.....	70
Figure 71 — Basic Address classes.....	71
Figure 72 — Context Diagram: AD_Address.....	72
Figure 73 — Context Diagram: AD_AbstractAddress.....	72
Figure 74 — Context Diagram: AD_USAddress.....	74
Figure 75 — Context Diagram: AD_AddressElement.....	75
Figure 76 — Context Diagram: AD_Addressee.....	76
Figure 77 — Context Diagram: AD_StreetIntersection.....	76
Figure 78 — Context Diagram: AD_Street.....	78

Figure 79 — Context Diagram: AD_PostalCode	79
Figure 80 — Context Diagram: AD_StreetLocation	79
Figure 81 — Context Diagram: AD_PhoneNumber	80
Figure 82 — Context Diagram: AD_NamedPlace	81
Figure 83 — Context Diagram: AD_StreetAddress	82
Figure 84 — Context Diagram: AD_NamedPlaceClassification	82
Figure 85 — Context Diagram: AD_Building	83
Figure 86 — Context Diagram: AD_MuniQuadrant	83
Figure 87 — Context Diagram: AD_RegionCode	84
Figure 88 — Context Diagram: AD_NumberRange	85
Figure 89 — Context Diagram: AD_ListNamedPlaces	85
Figure 90 — Context Diagram: NT_Network	86
Figure 91 — Context Diagram: NT_WayPoint	87
Figure 92 — Context Diagram: NT_WayPointList	88
Figure 93 — Junction and turns	89
Figure 94 — Context Diagram: NT_Turn	92
Figure 95 — Context Diagram: NT_TurnDirection	92
Figure 96 — Context Diagram: NT_Junction	94
Figure 97 — Context Diagram: NT_JunctionType	95
Figure 98 — Context Diagram: NT_AngularDirection	95
Figure 99 — Context Diagram: NT_Constraint	96
Figure 100 — Context Diagram: NT_VehicleConstraint	98
Figure 101 — Context Diagram: NT_TemporalConstraint	99
Figure 102 — Context Diagram: NT_LaneConstraint	100
Figure 103 — Context Diagram: NT_Vehicle	101
Figure 104 — Context Diagram: NT_Advisory	102
Figure 105 — Context Diagram: NT_SpatialRelation	103
Figure 106 — Context Diagram: NT_AdvisoryCategory	104
Figure 107 — Context Diagram: NT_AdvisoryElement	104
Figure 108 — Context Diagram: NT_ExitAssociation	105
Figure 109 — Context Diagram: NT_AdvisoryDirection	106
Figure 110 — Context Diagram: NT_AdvisoryDistance	107
Figure 111 — Context Diagram: NT_AdvisorySpatialRelation	107
Figure 112 — Context Diagram: NT_Link	110
Figure 113 — Context Diagram: NT_RouteSegmentCategory	110
Figure 114 — Context Diagram: NT_LinkPosition	111
Figure 115 — Context Diagram: NT_NetworkPosition	112
Figure 116 — Context Diagram: NT_Route	114
Figure 117 — Context Diagram: NT_RouteSummary	115
Figure 118 — Context Diagram: NT_Maneuver	117

Figure 119 — Combined Networks	117
Figure 120 — Context Diagram: NT_CombinedNetwork	118
Figure 121 — Context Diagram: NT_TransferNode	119
Figure 122 — Context Diagram: NT_Transfer	119
Figure 123 — Context Diagram: NT_TransferLink	120
Figure 124 — Feature data classes	120
Figure 125 — Context Diagram: FD_Feature	121
Figure 126 — Context Diagram: FD_FeatureCollection	122
Figure 127 — Context Diagram: FD_QueryFeatureCollection	123
Figure 128 — Context Diagram: FD_FeatureName	124
Figure 129 — Context Diagram: VoiceStream	125
Figure 130 — Context Diagram: BinaryData	125
Figure 131 — Context Diagram: Map	126
Figure 132 — Context Diagram: Image	126
Figure C.1 — Conceptual architecture equating mobile and non-mobile services	137

INTRODUCTION

This International Standard is a description of the data and services needed to support tracking and navigation applications for mobile clients. The web services views of this International Standard are given in Annex C.

Geographic information — Location-based services — Tracking and navigation

1 Scope

This International Standard describes the data types, and operations associated with those types, for the implementation of tracking and navigation services. This International Standard is designed to specify web services that can be made available to wireless devices through web-resident proxy applications, but is not restricted to that environment.

2 Conformance

Conformance to this International Standard takes on two meanings dependent on the type of entity declaring conformance.

Mechanisms for the transfer of data are conformant to this International Standard if they can be considered to consist of transfer record or type definitions that implement or extend a consistent subset of the object types described within this International Standard.

Web servers for tracking and navigation are conformant to this International Standard if their interfaces implement one or more of the subtypes of service defined in this International Standard and their communications and messaging are accomplished using a conformant transfer mechanism.

Clauses 6 and 7 of this International Standard use the Unified Modeling Language (UML) to present conceptual schemas for describing the information and services for tracking and navigation. Clause 8 further describes a general schema for addresses to be used as location equivalents in three types of services. Clause 9 describes network data appropriate for these services. This International Standard concerns only externally visible interfaces and places no restriction on the underlying implementations other than what is needed to satisfy the interface specifications in the actual situation, such as

- interfaces to software services using techniques such as COM or CORBA;
- interfaces to databases using techniques such as SQL;
- data interchange using encoding as defined in ISO 19118.

Few applications will require the full range of capabilities described by this conceptual schema. This clause, therefore, defines a set of conformance classes that will support applications whose requirements range from the minimum necessary to define data structures to full object implementation. This flexibility is controlled by a set of UML types that can be implemented in a variety of manners. Implementations that define full object functionality shall implement all operations defined by the types of the chosen conformance class, as is common for UML designed object implementations. Implementations that choose to depend on external “free functions” for some or all operations, or forgo them altogether, need not support all operations, but shall always support a data type sufficient to record the state of each of the chosen UML types as defined by its member variables. Common names for “metaphorically identical” but technically different entities are acceptable. The UML model in this International Standard defines abstract types, application schemas define conceptual classes, various software systems define implementation classes or data structures, and the XML from the encoding standard (ISO 19118) defines entity tags. All of these reference the same information content. There is no difficulty in allowing the use of the same name to represent the same information content

even though at a deeper level there are significant technical differences in the digital entities being implemented.

Details of the conformance classes are given in the abstract test suite in Annex A.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO 19107, *Geographic information — Spatial schema*

ISO 19108, *Geographic information — Temporal schema*

ISO 19109, *Geographic information — Rules for application schema*

ISO 19111, *Geographic information — Spatial referencing by coordinates*

ISO 19112, *Geographic information — Spatial referencing by geographic identifiers*

ISO 19118, *Geographic information — Encoding*

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
candidate route
any **route** that satisfies all constraints of the routing request with the possible exception of optimality of the **cost function**

NOTE Navigation is the process of finding the candidate route that optimizes a chosen cost function.

4.2
cost function
function that associates a measure (cost) to a **route**

NOTE The normal mechanism is to apply a cost to each part of a **route**, and to define the total **route** cost as the sum of the cost of the parts. This is necessary for the operation of the most common navigation algorithms. The units of cost functions are not limited to monetary costs and values only, but include such measures as time, distance, and possibly others. The only requirement is that the function be additive and at least non-negative. This last criteria can be softened as long as no zero or less cost is associated with any loop in the network, as this will prevent the existence of a “minimal cost” **route**.

4.3
Dijkstra graph
positively weighted directed graph appropriately configured to execute a shortest path search

NOTE The term comes from the most commonly known algorithm for finding a shortest path in a positively weighted graph, from E. Dijkstra’s paper [7]. Although this algorithm is not the only one in use, the requirements for the graph are common to most. The most common relaxation of the requirement is the “positive weights”, which are not needed in the Bellman–Ford algorithm [4], [8].

4.4

geocoding

translation of one form of **location** into another

NOTE Geocoding usually refers to the translation of “address” or “intersection” to “direct position”. Many service providers also include a “reverse geocoding” interface to their geocoder, thus extending the definition of the service as a general translator of **location**. Because routing services use internal **location** encodings not usually available to others, a geocoder is an integral part of the internals of such a service.

4.5

instantiate

to represent (an abstraction) by the creation of a concrete instance or to create the ability to create an instance

NOTE A class or data element definition instantiates a type if it creates the ability to create objects or data elements, respectively, that can represent the concepts (instance data and/or operations) defined by that type. A class is instantiated by an object if the class defines that object’s structure and function. A data schema is instantiated by a data element if the data schema defines that element’s structure.

4.6

junction

single topological node in a **network** with its associated collection of **turns**, incoming and outgoing **links**

NOTE Junction is an alias for node.

4.7

linear referencing system

linear positioning system [ISO 19116]

positioning system that measures distance from a reference point along a route (feature)

NOTE The system includes the complete set of procedures for determining and retaining a record of specific points along a linear feature such as the **location** reference method(s) together with the procedures for storing, maintaining, and retrieving **location** information about points and segments on the highways. [NCHRP Synthesis 21, 1974]

4.8

link

directed topological connection between two nodes (**junctions**), consisting of an edge and a direction

NOTE Link is an alias for directed edge.

4.9

link position

position within a network on a **link** defined by some strictly monotonic measure associated with that **link**

NOTE Link positions are often associated with a target feature that is not part of the network. The most common link measures used for this are the distance from start node or address. The most common use of a link position is to geolocate an “address”.

4.10

location

identifiable geographic place

[ISO 19112]

NOTE A location is represented by one of a set of data types that describe a **position**, along with metadata about that data, including coordinates (from a coordinate reference system), a measure (from a **linear referencing system**), or an address (from an address system).

4.11 location-based service LBS

service whose return or other property is dependent on the **location** of the client requesting the service or of some other thing, object or person

4.12 location-dependent service LDS

service whose availability is dependent upon the **location** of the client

4.13 main-road rule

set of criteria used at a **turn** in lieu of a **route instruction**; default instruction used at a node

NOTE This rule represents what is “most natural” to do at a node (intersection), given the entry link used. The most common version is “as straight as possible”, or to exit a **turn** on the most obvious extension of the entry street, which is usually, but not always, the same named street that was the entry. Every node in a route is either associated with an instruction or can be navigated by the main-road rule.

4.14 maneuver manœuvre

collection of related **links** and **turns** used in a route in combination

NOTE Maneuvers are used to cluster **turns** into convenient and legal combinations. They may be as simple as a single **turn**, a combination of quick **turns** (“jogs” in the American mid-west consisting of a **turn** followed immediately by a **turn** in the opposite direction) or very complex combinations consisting of entry, exit, and connecting roadways (“magic roundabouts” in the UK).

4.15 navigation combination of **routing**, **route transversal** and **tracking**

NOTE This is essentially the common term navigation, but the definition decomposes the process in terms used in the packages defined in this International Standard.

4.16 navigation constraint constraint

restriction on how a **link** or **turn** may be traversed by a **vehicle**, such as **vehicle** classification, physical or temporal constraint

4.17 network

abstract structure consisting of a set of 0-dimensional objects called **junctions**, and a set of 1-dimensional objects called **links** that connect the **junctions**, each **link** being associated with a start (origin, source) **junction** and end (destination, sink) **junction**

NOTE The **network** is essentially the universe of discourse for the **navigation** problem. **Networks** are a variety of 1-dimensional topological complex. In this light, **junction** and topological node are synonyms, as are **link** and directed edge.

4.18 position

data type that describes a point or geometry potentially occupied by an object or person

NOTE A direct position is a semantic subtype of position. Direct positions as described can only define a point and therefore not all positions can be represented by a direct position. That is consistent with the “is type of” relation. An ISO 19107 geometry is also a position, just not a direct position.

4.19**route**

sequence of **links** and/or partial **links** that describe a path, usually between two **positions**, within a **network**

4.20**route instruction**

information needed at a point along a **route** in a **network** that allows that **route** to be traversed

NOTE To minimize the number of **instructions** needed to complete a **route traversal**, a default instruction can be assumed at **junctions** without specifically associated **instructions**. This default is called the **main-road rule**.

4.21**route traversal**

process of following a **route**

4.22**routing**

finding of optimal (minimal **cost function**) **routes** between **locations** in a **network**

4.23**slope**

rate of change of elevation with respect to curve length

4.24**tracking**

monitoring and reporting the **location** of a **vehicle**

4.25**traveller**

person subject to being navigated or tracked

cf. vehicle

NOTE Includes pedestrians. See ISO 14825. In this International Standard, traveller can be replaced by **vehicle** without any change of intent.

4.26**traversable**

condition of a **link** or **turn** that allows or restricts all traffic's traversal, as opposed to a more detailed **navigation constraint**

NOTE Traversability is usually a function of physical, cultural, or legal conditions. If traversable is false, then the object cannot be navigated. This effectively removes a **link** from the usable network. In the case of a node, it effectively removes the node and all associated **links** from the useable network. In the case of a **turn**, it simply removes it from any viable **route**. Non-traversable entities are not included in **maneuvers** or **routes**.

4.27**turn**

part of a **route** or **network** consisting of a **junction** location and an entry and exit **link** for that **junction**

4.28**vehicle**

object subject to being navigated or tracked

cf. traveller

NOTE Includes pedestrians. See ISO 14825. In this International Standard, vehicle can be replaced by **traveller** without any change of intent.

4.29 vehicle classification

type of **vehicle**, based on the nature of its construction or intended purpose

NOTE Classifications based on construction include automobile, truck, bus, bicycle, etc. Classifications based on purpose include taxi, emergency vehicle, etc. Vehicle classification can be used to determine the application of **navigation** constraints.

4.30 waypoint

location on the **network** that plays a role in choosing **candidate routes** potentially satisfying a routing request

5 Abbreviated terms and UML notation

5.1 Abbreviated terms

CRS	Coordinate Reference System
CSL	Conceptual Schema Language
ECCMA	Electronic Commerce Code Management Association
GDF	Geographic Data Files
GML	Geography Markup Language
GPS	Global Positioning System
IAEC	International Address Element Code
LBS	Location Based Service
LDS	Location Dependent Service
LRM	Linear Referencing Method
LRS	Linear Reference System
OCL	Object Constraint Language
PDA	Personal Digital Assistant
UML	Unified Modeling language
XML	eXtensible Markup Language

5.2 UML notation

The UML notation used in this International Standard is described in ISO 19107, and differs from standard UML only in the existence and interpretation of some special stereotypes, in particular “CodeList” and “Union”.

The term “context diagram” used extensively in the naming of figures in this International Standard means a diagram that illustrates the context of a specified central type, meaning the types of its attributes, operations and association targets. This is the information most useful to the implementer of this central class.

6 Tracking

6.1 Semantics

The package “Tracking” contains other packages that are used in tracking services and related functions (see Figure 1).

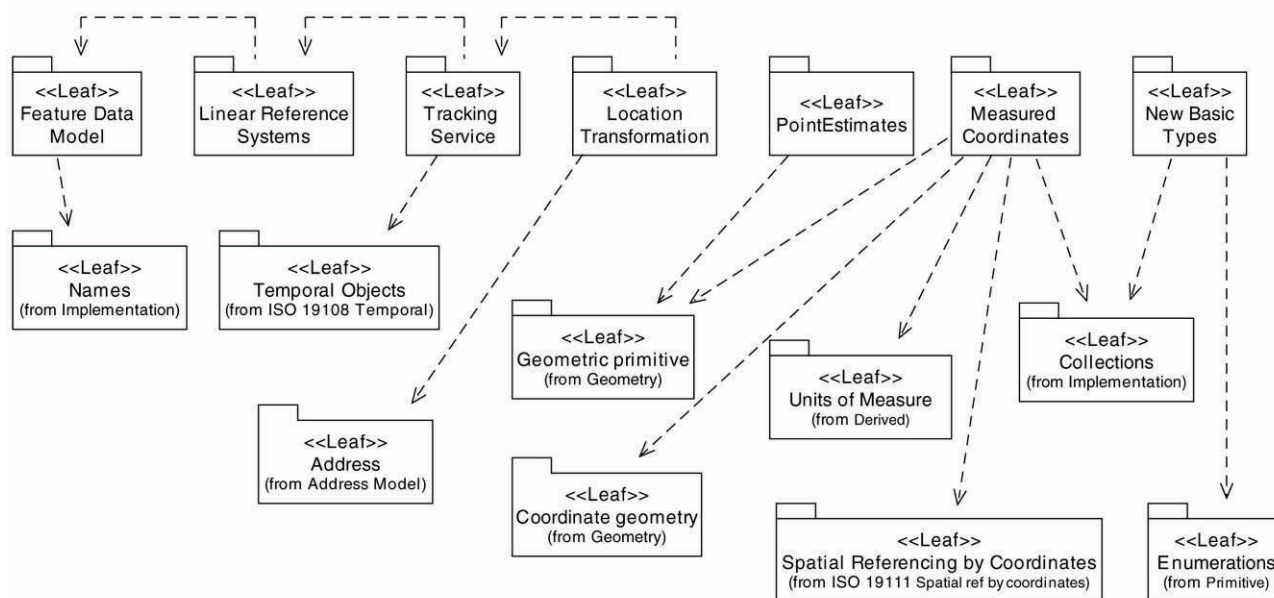


Figure 1 — Tracking packages

6.2 Package: Tracking Service

6.2.1 Semantics

The package “Tracking Service” contains types and classes useful in creating a tracking service. Since this is the core of many of the navigation functions described elsewhere in this International Standard, this package contains some important types universal to most if not all location-based services.

6.2.2 TK_Position

6.2.2.1 Semantics

The union class “TK_Position” is used to represent positions in tracking and associated applications. An instance of this class locates a position or place within the network. It should always be interpretable as a direct position (coordinate in a reference system), an address or as a network position. The discriminators for the union, and their associated types are as follows:

directPosition:	DirectPosition
placeName:	SI_LocationInstance
featureID:	FD_FeatureName
linearReference:	LR_PositionExpression
networkPosition:	NT_NetworkPosition
address:	AD_AbstractAddress
phone:	CharacterString

The operations on this type are all cast operators that allow the programmers to determine the position in the form most useful to them. The UML for TK_Position is given in Figure 2.

6.2.2.2 Operation: asPosition

The operator “asPosition” derives the coordinates in the CRS of the data of this location.

```
TK_Position :: asPosition() : DirectPosition
```

6.2.2.3 Operation: asNetworkPosition

The operator “asNetworkPosition” derives the network position of this location.

```
TK_Position :: asNetworkPosition() : NT_NetworkPosition
```

6.2.2.4 Operation: asAddress

The operator “asAddress” derives the address of this location.

```
TK_Position :: asAddress() : AD_Address
```

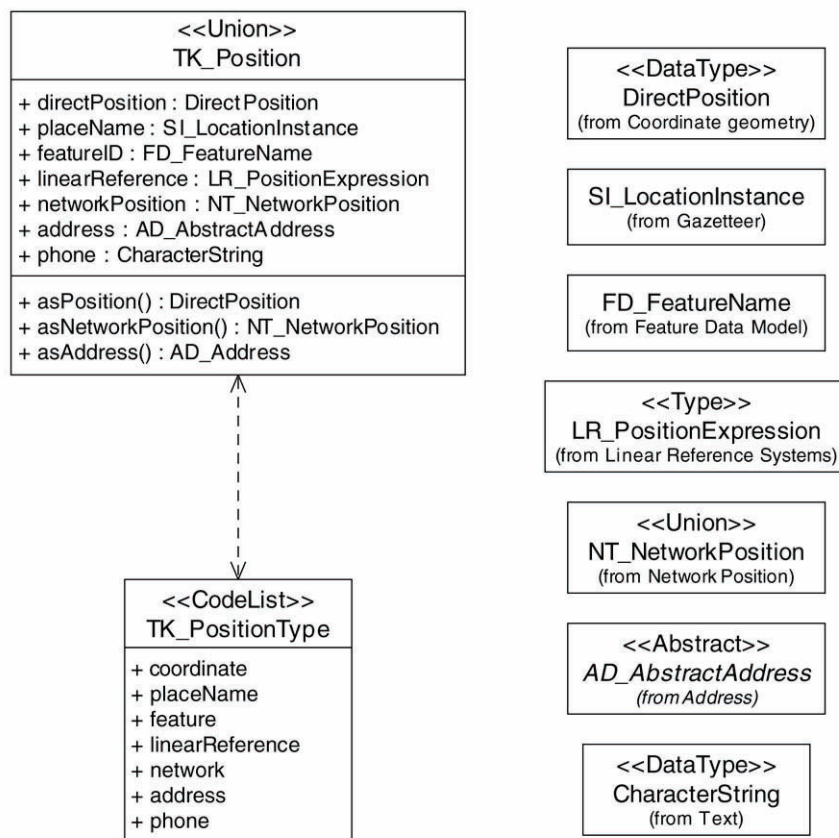


Figure 2 — Context Diagram: TK_Position

6.2.3 TK_MobileSubscriber

6.2.3.1 Semantics

The type “TK_MobileSubscriber” is used to model the items being tracked within the tracking service. The most common usage is the subscriber to a mobile service such as cell phone, where the service has the ability to determine the location of the device. The interfaces for the tracking services are also applicable to services that query the device for its location and would thus be applicable to GPS equipped devices as well. The service is a UML <<Type>> as opposed to a UML <<Interface>> because of the need to associate subscribers with their service.

There are deep issues of authentications between the service and the client, especially if the client is different from the mobile subscriber (in which case there are privacy issues). This International Standard assumes that those issues have been solved outside of the tracking interfaces (possible through a “verification” service, not dependent on geography, and therefore outside the scope of this International Standard). The UML for TK_MobileSubscriber is given in Figure 3.

6.2.3.2 Attribute: id : CharacterString

The attribute “id” is the identifier by which the associated tracking service knows this subscriber.

```
TK_MobileSubscriber :: id : CharacterString
```

6.2.3.3 Attribute: location : TK_TrackingLocation

The derived attribute “location” is the location of the subscriber at the time the attribute is accessed. This is essentially another access to the tracking service in many cases. In the case where the subscriber is a GPS equipped device, accessing this attribute would not necessarily activate the tracking service.

```
TK_MobileSubscriber :: location : TK_TrackingLocation
```

6.2.3.4 Role: trackingService[1..*] : TK_TrackingService

The association role “trackingService” links the subscribers to the tracking service that supports tracking them.

```
TK_MobileSubscriber :: trackingService[1..*] : TK_TrackingService
```

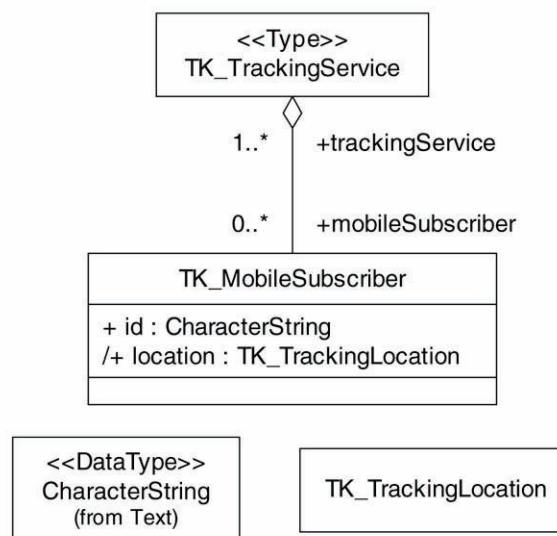


Figure 3 — Context Diagram: TK_MobileSubscriber

6.2.4 TK_TrackingLocation

6.2.4.1 Semantics

The type “TK_TrackingLocation” is used to represent objects describing a location and its associated metadata, such as time of measure, or direction of travel. The UML for TK_TrackingLocation is given in Figure 4.

6.2.4.2 Attribute: position : TK_Position

The attribute “position” describes the measurements that represent the position of this object.

```
TK_TrackingLocation :: position : TK_Position
```

6.2.4.3 Attribute: time[0..1] : TM_Primitive

The optional attribute “time” describes the time of the measurements that represent the position of this object.

```
TK_TrackingLocation :: time[0..1] : TM_Primitive
```

6.2.4.4 Attribute: direction[0..1] : Bearing

The attribute “direction” describes the direction of travel of the subscriber being tracked.

```
TK_TrackingLocation :: direction[0..1] : Bearing
```

6.2.4.5 Attribute: speed[0..1] : Velocity

The attribute “speed” describes the velocity of travel of the subscriber being tracked.

```
TK_TrackingLocation :: speed[0..1] : Velocity
```

6.2.4.6 Role: metadata[0..*] : TK_TrackingLocationMetadata

The association role “metadata” aggregates optional metadata elements associated with this location.

```
TK_TrackingLocation :: metadata[0..*] : TK_TrackingLocationMetadata
```

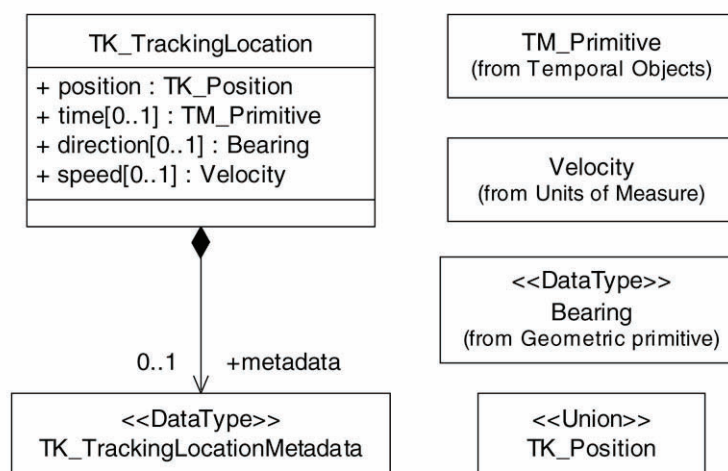


Figure 4 — Context Diagram: TK_TrackingLocation

6.2.5 TK_TrackingService

6.2.5.1 Semantics

The type “TK_TrackingService” defines interfaces and associations for tracking services. The UML for TK_TrackingService is given in Figure 5.

6.2.5.2 Role mobileSubscriber[0..*] : TK_MobileSubscriber

The association role “mobileSubscriber” aggregates items (vehicle or traveller) that may be tracked by using this service.

```
TK_TrackingService :: mobileSubscriber[0..*] : TK_MobileSubscriber
```

6.2.5.3 Operation: locate

The operation “locate” returns a single location for a tracked item.

```
TK_TrackingService ::
  locate(ms : TK_MobileSubscriber, type : TK_PositionType) :
    TK_TrackingLocation
```

6.2.5.4 Operation: track

The operation “track” returns a sequence of locations for a tracked item. The trigger types that cause a new location to be placed in the sequence are specified in the operation protocol.

```
TK_TrackingService ::
  track(ms : TK_MobileSubscriber, triggerType[1..*] : TK_Trigger) :
    TK_TrackingLocationSequence
```

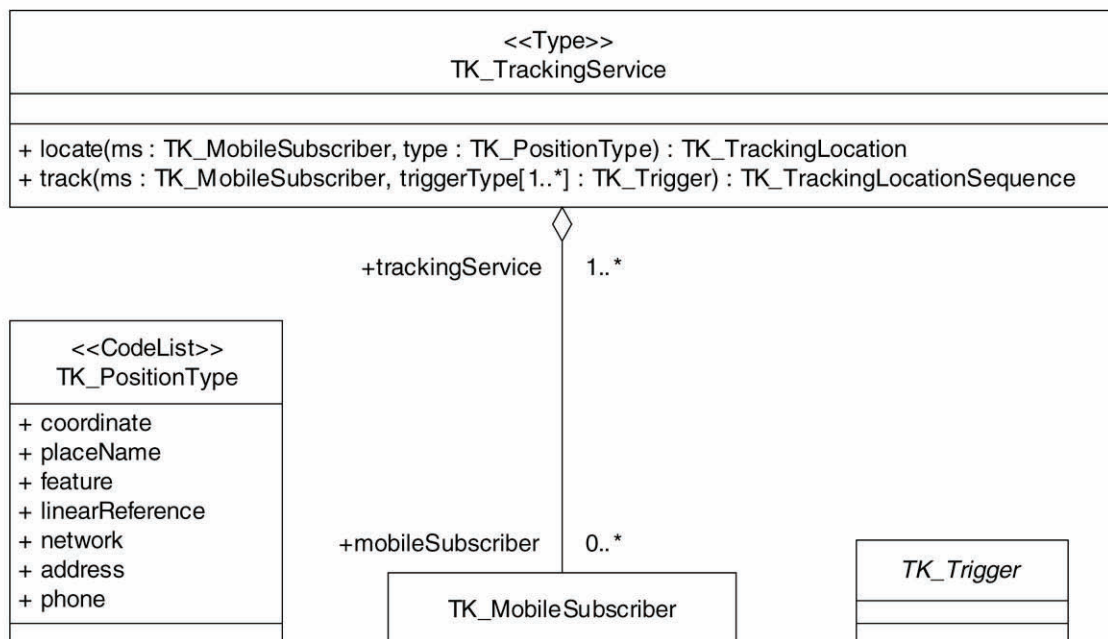


Figure 5 — Context Diagram: TK_TrackingService

6.2.6 TK_PositionType

The code list “TK_PositionType” lists the types of position available to the application. The initial members of this list include: “coordinate”, “placeName”, “feature”, “linearReference”, “network”, “address”, and “phone”. The UML for TK_PositionType is given in Figure 6.

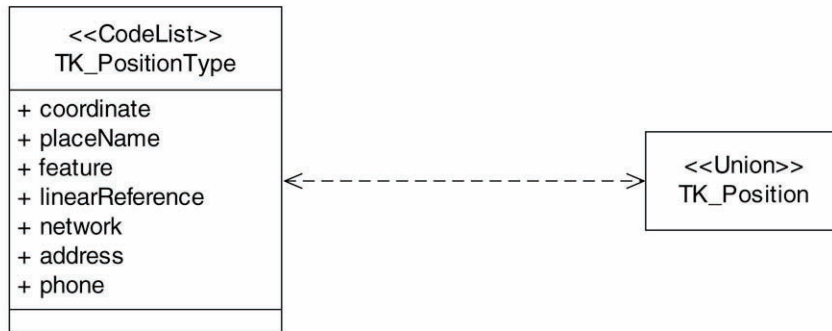


Figure 6 — Context Diagram: TK_PositionType

6.2.7 TK_TrackingLocationSequence

6.2.7.1 Semantics

The type “TK_TrackingLocationSequence” is an access mechanism for the return of a continuous tracking of an item. The UML for TK_TrackingLocationSequence is given in Figure 7.

6.2.7.2 Attribute: ms : TK_MobileSubscriber

The attribute “ms” gives the identity of the item being tracked.

```
TK_TrackingLocationSequence :: ms : TK_MobileSubscriber
```

6.2.7.3 Attribute: sequenceID : CharacterString

The attribute “sequenceID” gives the identity of the location sequence so that a client can repeatedly access the location sequence.

```
TK_TrackingLocationSequence :: sequenceID : CharacterString
```

6.2.7.4 Operation: next

The operation “next” returns the next location in the sequence.

```
TK_TrackingLocationSequence :: next() : TK_TrackingLocation
```

6.2.7.5 Operation: suspend

The operation “suspend” stops temporarily the stream of locations in the sequence.

```
TK_TrackingLocationSequence :: suspend() : Boolean
```

6.2.7.6 Operation: restart

The operation “restart” restarts the stream of locations in the sequence that has been stopped by “suspend”.

```
TK_TrackingLocationSequence :: restart() : TK_TrackingLocation
```

6.2.7.7 Operation: terminate

The operation “terminate” stops permanently the stream of locations in the sequence.

```
TK_TrackingLocationSequence :: terminate() : Boolean
```

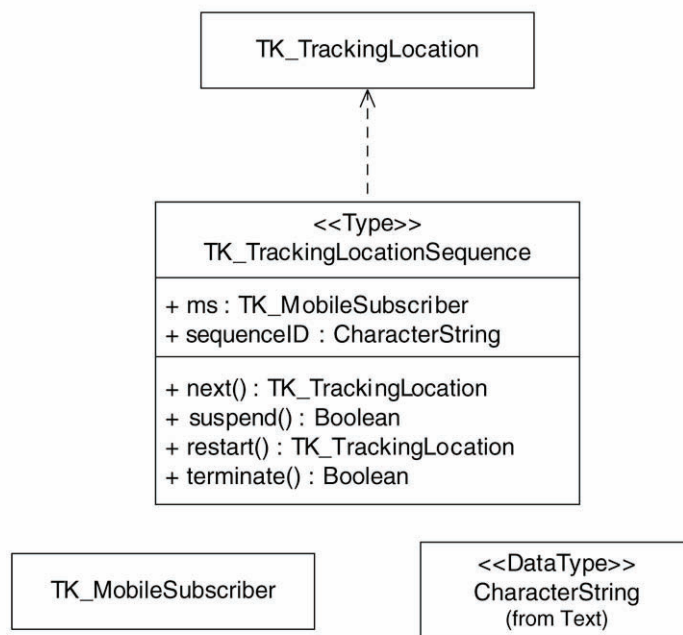


Figure 7 — Context Diagram: TK_TrackingLocationSequence

6.2.8 TK_Trigger

The abstract class “TK_Trigger” acts as a root class for trigger types to be used in controlling a location sequence. There are generally two types: triggered by an event, or triggered by the passage of time. The UML for TK_Trigger is given in Figure 8.

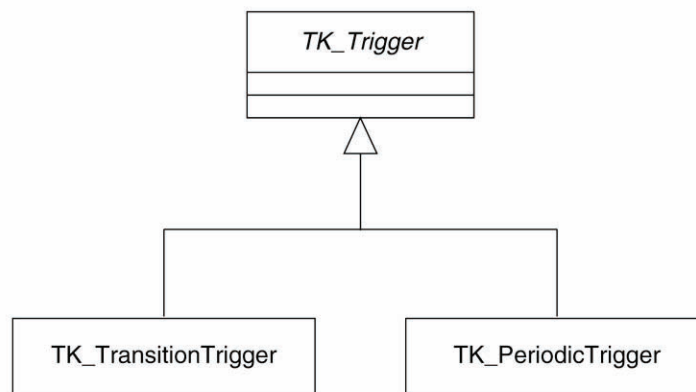


Figure 8 — Context Diagram: TK_Trigger

6.2.9 TK_PeriodicTrigger

6.2.9.1 Semantics

The class “TK_PeriodicTrigger” is used to control location sequences by setting up temporal limits on how far apart in time tracking samples are taken. Either minTime or maxTime shall be present for a valid TK_PeriodicTrigger. The UML for TK_PeriodicTrigger is given in Figure 9.

6.2.9.2 Attribute: minTime[0..1] : TM_Primitive

The attribute “minTime” is the minimum amount of time between tracking samples. In the absence of other criteria, the tracking service will begin the process to take a sample when the “minTime” has passed. Assuming no lag time is involved, the samples will be “minTime” apart.

```
TK_PeriodicTrigger :: minTime[0..1] : TM_Primitive
```

6.2.9.3 Attribute: maxTime[0..1] : TM_Primitive

The attribute “maxTime” is the maximum amount of time between tracking samples. In the absence of other criteria, the tracking service will attempt to assure that no two samples in the sequence are further apart than the “maxTime” parameter.

```
TK_PeriodicTrigger :: maxTime[0..1] : TM_Primitive
```

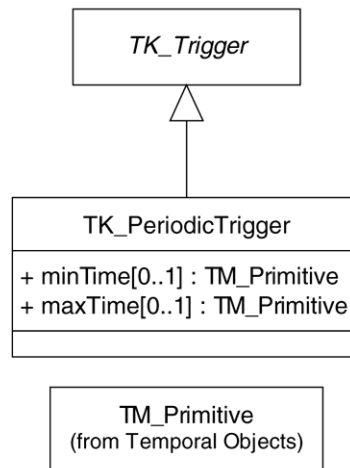


Figure 9 — Context Diagram: TK_PeriodicTrigger

6.2.10 TK_TransitionTrigger

6.2.10.1 Semantics

The class “TK_TransitionTrigger” models transitions relative to the item being tracked that will trigger the taking of a new location sample. In a TK_TransitionTrigger, at least one of the attributes shall be non-null. The UML for TK_TransitionTrigger is given in Figure10.

6.2.10.2 Attribute: type[0..*] : TK_TransitionType

The attribute “type” describes the type or types of transitions being watched. The various types of transitions are listed in the code list associated with the service.

```
TK_TransitionTrigger :: type[0..*] : TK_TransitionType
```

6.2.10.3 Attribute: deltaDirection[0..1] : Angle

The attribute “deltaDirection” describes the magnitude of the change of direction that triggers a new location sample.

```
TK_TransitionTrigger :: deltaDirection[0..1] : Angle
```

6.2.10.4 Attribute: deltaPosition[0..1] : Distance

The attribute “deltaPosition” describes the magnitude of the change of location (distance) that triggers a new location sample.

```
TK_TransitionTrigger :: deltaPosition[0..1] : Distance
```

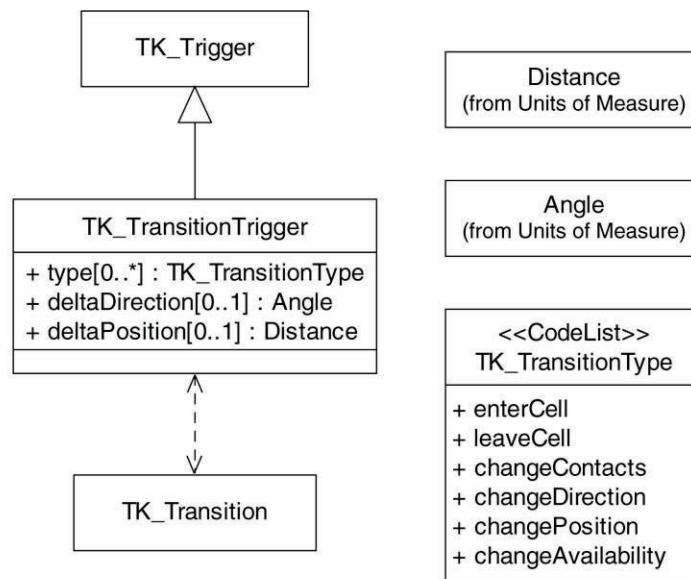


Figure 10 — Context Diagram: TK_TransitionTrigger

6.2.11 TK_TransitionType

The code list “TK_TransitionType” enumerates the types of transition tracked by this service. The initial values of the list include the following:

“enterCell”	mobile subscriber enters a cell associated with the network, e.g. cell phone network
“leaveCell”	mobile subscriber leaves a cell associated with the network
“changeContacts”	mobile subscriber changes the receivers with which it has contact
“changeDirection”	mobile subscriber changes the direction of its motion greater than the deltaDirection parameter specifies
“changePosition”	mobile subscriber changes its position greater than the deltaPosition parameter allows
“changeAvailability”	contact with the mobile subscriber is either lost or gained. If lost, the position just prior to loss is given. If gained, the first position calculated for the mobile subscriber after the new contact is given

6.2.12 TK_TrackingLocationMetadata

6.2.12.1 Semantics

The data type “TK_TrackingLocationMetadata” contains the metadata that is used to describe the position returned by the tracking service. As such, this data type shall always be contained in a TK_TrackingLocation, or other location type. The UML for TK_TrackingLocationMetadata is given in Figure 11.

6.2.12.2 Attribute: ms[0..1] : TK_MobileSubscriber

The attribute “ms” gives the identity of the item being tracked. If this attribute is not present, it should be derivable from context through the ownership of the containing TK_TrackingLocation.

```
TK_TrackingLocationSequence :: ms[0..1] : TK_MobileSubscriber
```

6.2.12.3 Attribute: quality : TK_QualityOfPosition

The attribute “quality” describes the quality of the position.

```
TK_TrackingLocationSequence :: quality : TK_QualityOfPosition
```

6.2.12.4 Attribute: time : TM_Primitive

The attribute “time” gives the time the position was measured.

```
TK_TrackingLocationSequence :: time : TM_Primitive
```

6.2.12.5 Attribute: clientID : CharacterString

The attribute “clientID” gives the identity of the client requesting the tracking service.

```
TK_TrackingLocationSequence :: clientID : CharacterString
```

6.2.12.6 Attribute: trigger[0..*] : TK_Transition

The attribute “trigger” gives the description of the transition that triggered this location measurement.

```
TK_TrackingLocationSequence :: trigger[0..*] : TK_Transition
```

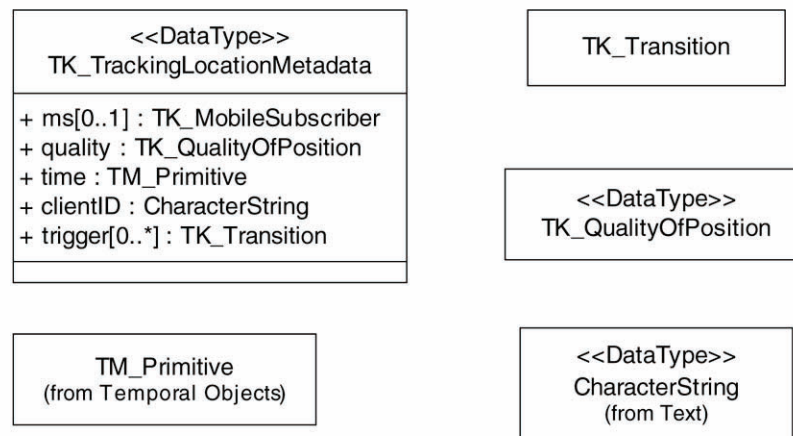


Figure 11 — Context Diagram: TK_TrackingLocationMetadata

6.2.13 TK_Transition

6.2.13.1 Semantics

The class “TK_Transition” describes a particular event or transition that triggered a location measure. The UML for TK_Transition is given in Figure 12.

6.2.13.2 Attribute: type[1..*] : TK_TransitionType

The attribute “type” describes the type or types of transitions.

```
TK_Transition :: type[1..*] : TK_TransitionType
```

6.2.13.3 Attribute: time : TM_Primitive

The attribute “time” describes the time that the transition occurred. Normally, this should be approximately the same time as in the location metadata.

```
TK_Transition :: time : TM_Primitive
```

6.2.13.4 Attribute: deltaDirection[0..1] : Angle

The attribute “deltaDirection” describes the magnitude of the change of direction that triggered this new location sample.

```
TK_Transition :: deltaDirection[0..1] : Angle
```

6.2.13.5 Attribute: deltaPosition[0..1] : Distance

The attribute “deltaPosition” describes the magnitude of the change of location (distance) that triggered this location sample.

```
TK_Transition :: deltaPosition[0..1] : Distance
```

6.2.13.6 Attribute: availability[0..1] : Boolean

The attribute “availability” describes whether the location of the item being tracked is currently available to be measured.

```
TK_Transition :: availability[0..1] : Boolean
```

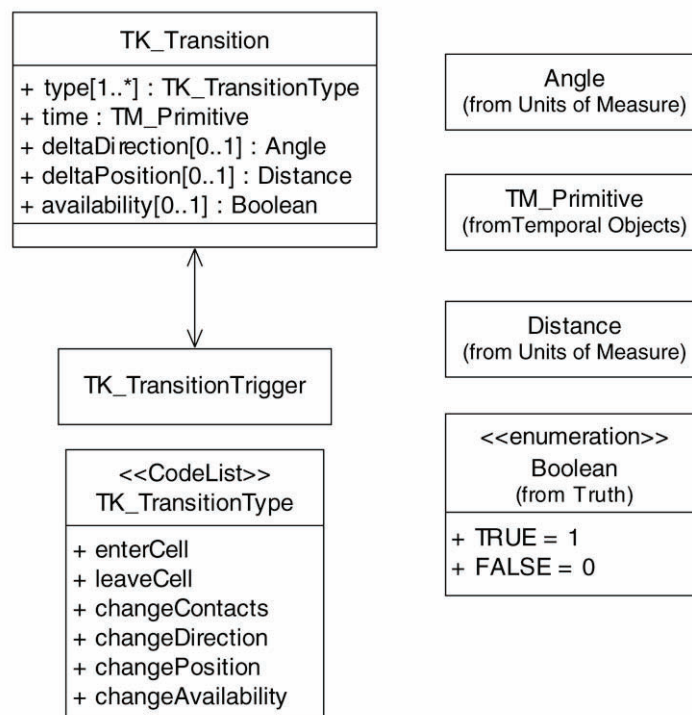


Figure 12 — Context Diagram: TK_Transition

6.2.14 TK_QualityOfPosition

6.2.14.1 Semantics

The data type “TK_QualityOfPosition” is used to describe the quality of position measurements. The UML for TK_QualityOfPosition is given in Figure 13.

6.2.14.2 Role: accuracyStatement[1..*] : TK_AccuracyStatement

The association role “accuracyStatement” aggregates statements about the accuracy of the position.

```
TK_QualityOfPosition :: accuracyStatement[1..*] : TK_AccuracyStatement
```

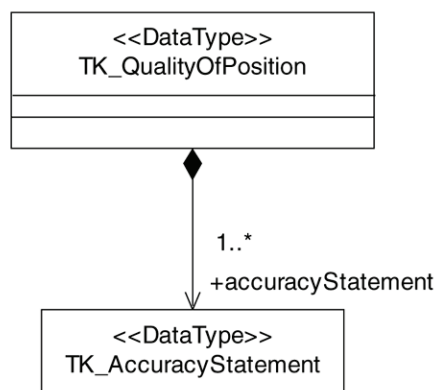


Figure 13 — Context Diagram: TK_QualityOfPosition

6.2.15 TK_Accuracy

6.2.15.1 Semantics

The class “TK_Accuracy” is used to describe accuracy of a measure. The numerical values are in the same units as the measure. The UML for TK_Accuracy is given in Figure 14.

6.2.15.2 Attribute: accuracyType : TK_AccuracyType

The attribute “accuracyType” gives the type of accuracy measure.

```
TK_Accuracy :: accuracyType : TK_AccuracyType
```

6.2.15.3 Attribute: sigmaLevel[0..1] : Number

The attribute “sigmaLevel” describes the accuracy of the measure based on a multiple of the standard deviation.

```
TK_Accuracy :: sigmaLevel[0..1] : Number
```

6.2.15.4 Attribute: sigma[0..1] : Number

The attribute “sigma” describes the standard deviation.

```
TK_Accuracy :: sigma[0..1] : Number
```

6.2.15.5 Attribute: units[0..1]: UnitOfMeasure

The attribute “units” gives the unit of measure for the accuracy for the measurement as needed.

```
TK_AccuracyStatement :: unit[0..1] : UnitOfMeasure
```

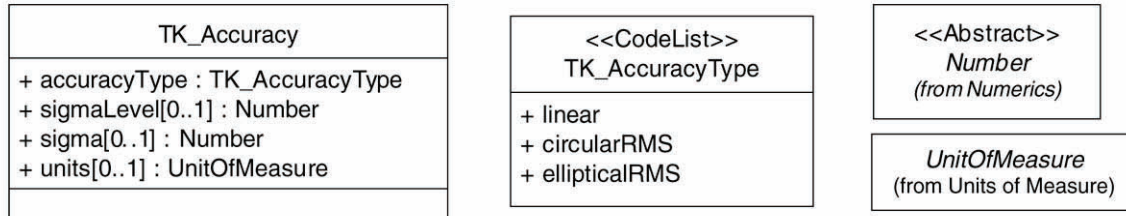


Figure 14 — Context Diagram: TK_Accuracy

6.2.16 TK_AccuracyType

The code list “TK_AccuracyType” lists the type of accuracy measure used by this service. The initial values in the value domain are “linear”, “circularRMS”, and “ellipticalRMS”.

6.2.17 TK_AccuracyStatement

6.2.17.1 Semantics

The data type “TK_AccuracyStatement” is used to make statements about metric accuracy of measurements. The UML for TK_AccuracyStatement is given in Figure 15.

6.2.17.2 Attribute: axis[1..*] : CharacterString

The attribute “axis” gives the axis or axes about which the statement applies. This name shall correspond to axes in a given coordinate reference system for the geometry of the given position.

```
TK_AccuracyStatement :: axis[1..*] : CharacterString
```

6.2.17.3 Attribute: accuracy : TK_Accuracy

The attribute “accuracy” gives accuracy for a measurement.

```
TK_AccuracyStatement :: accuracy : TK_Accuracy
```

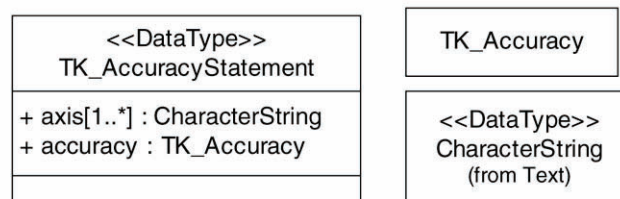


Figure 15 — Context Diagram: TK_AccuracyStatement

6.3 Package: Point Estimates

6.3.1 Semantics

The package “Point Estimates” contains geometry object types, all subclasses of “GM_Point”, used in describing direct position estimates seen in tracking services. The UML for the inheritance hierarchy for the point estimate classes is given in Figure 16. A graphic of their form, showing the estimate with its error geometry, is given in Figure 17.

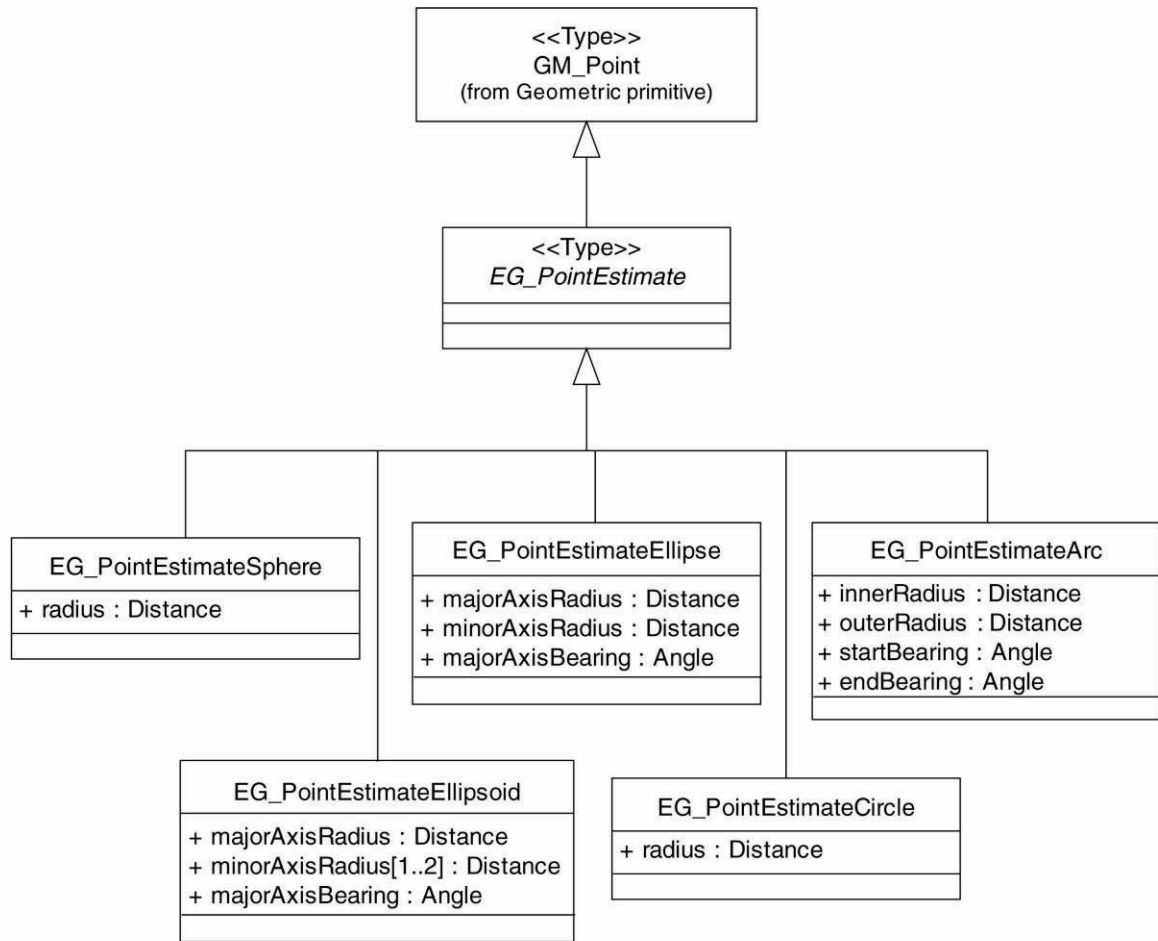


Figure 16 — Point Estimate classes

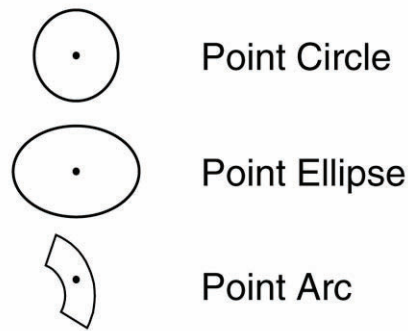


Figure 17 — Geometric interpretations of point estimate types

6.3.2 EG_PointEstimate

The type “EG_PointEstimate” is the abstract root of all the error estimate geometry objects. It is a subtype of point. The best estimate of the location is the centre of the given geometry. The coordinates of the point specify the primary constructive location of the geometry, and will be the best estimate of the location if and only if it is also the centre of the geometry specified. The UML for EG_PointEstimate is given in Figure 16.

6.3.3 EG_PointEstimateCircle

6.3.3.1 Semantics

The class “EG_PointEstimateCircle” is used to describe a point with an error estimate that is a metric circle of a given radius. It should be noted that this circle is not a circle in the coordinate reference system of the point unless the reference system is equivalent a local engineering one. The point coordinates (inherited from GM_Point) specify the centre of the circle, and the best estimate of position. The UML for EG_PointEstimateCircle is given in Figure 18.

6.3.3.2 Attribute: radius : Distance

The attribute “radius” is the radius of the error circle used in the point estimate. The usual semantics is to return a fixed “sigma” level that is a fixed multiple of the standard deviation of the measurement procedure. Unless otherwise specified by a profile, a 1,0-sigma level is given.

```
EG_PointEstimateCircle :: radius : Distance
```

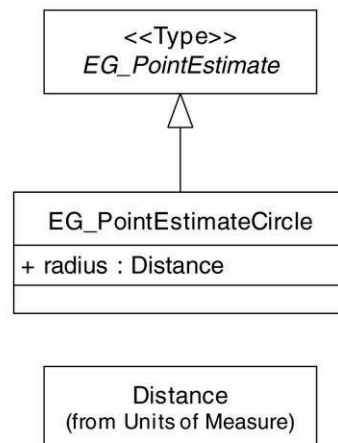


Figure 18 — Context Diagram: EG_PointEstimateCircle

6.3.4 EG_PointEstimateEllipse

6.3.4.1 Semantics

The point ellipse is the same as a point circle except that an ellipse defined by the parameters given specifies the error. The point coordinates (inherited from GM_Point) specify the centre of the ellipse, and the best estimate of position. The UML for EG_PointEstimateEllipse is given in Figure 19.

6.3.4.2 Attribute: majorAxisRadius : Distance

The attribute “majorAxisRadius” is half the length of the major axis of the ellipse.

```
EG_PointEstimateEllipse :: majorAxisRadius : Distance
```

6.3.4.3 Attribute: minorAxisRadius : Distance

The attribute “minorAxisRadius” is half the length of the minor axis of the ellipse.

```
EG_PointEstimateEllipse :: minorAxisRadius: Distance
```

6.3.4.4 Attribute: majorAxisBearing : Angle

The attribute “majorAxisBearing” is bearing of the major axis of the ellipse, given as an angle from true north. The minor axis is perpendicular to the major axis.

```
EG_PointEstimateEllipse :: majorAxisBearing : Angle
```

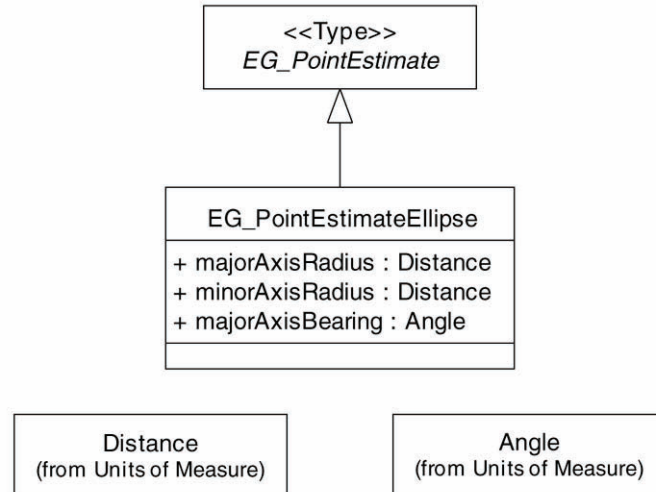


Figure 19 — Context Diagram: EG_PointEstimateEllipse

6.3.5 EG_PointEstimateArc

6.3.5.1 Semantics

The point arc is the same as a point circle except that a pair of arcs between two given radii and two given bearing angles gives the area estimate error. The point coordinates specify the centre of the two arcs. The best estimate of position is the along the centreline of the two arcs and halfway between the two radii. The UML for EG_PointEstimateArc is given in Figure 20.

6.3.5.2 Attribute: innerRadius : Distance

The attribute “innerRadius” is the radius of the smaller of the two arcs.

```
EG_PointEstimateArc :: innerRadius : Distance
```

6.3.5.3 Attribute: outerRadius : Distance

The attribute “outerRadius” is the radius of the larger of the two arcs.

```
EG_PointEstimateArc :: outerRadius: Distance
```

6.3.5.4 Attribute: startBearing : Angle

The attribute “startBearing” is the bearing of one side of the arcs (left-hand end when viewed from the arc centre).

```
EG_PointEstimateArc :: startBearing : Angle
```

6.3.5.5 Attribute: endBearing : Angle

The attribute “endBearing” is the bearing of the other side of the arcs (right-hand end when viewed from the arc centre).

```
EG_PointEstimateArc :: endBearing: Angle
```

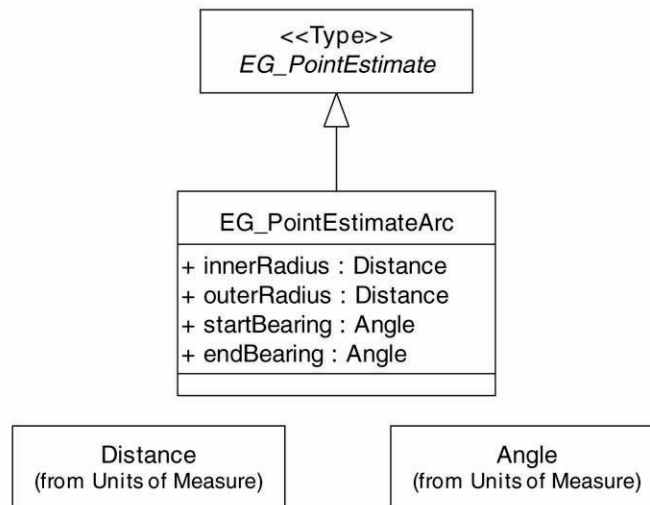


Figure 20 — Context Diagram: EG_PointEstimateArc

6.3.6 EG_PointEstimateSphere

6.3.6.1 Semantics

The class “EG_PointEstimateSphere” is used to describe a point with an error estimate that is a metric sphere of a given radius. It should be noted that this sphere is not a sphere in the coordinate reference system of the point unless the reference system is equivalent to a local engineering one. The point coordinates (inherited from GM_Point) specify the centre of the sphere, and the best estimate of position. The UML for EG_PointEstimateSphere is given in Figure 21.

6.3.6.2 Attribute: radius : Distance

The attribute “radius” is the radius of the error sphere used in the point estimate. The usual semantics is to return a fixed “sigma” level that is a fixed multiple of the standard deviation of the measurement procedure. Unless otherwise specified by a profile, a 1,0-sigma level is given.

```
EG_PointEstimateSphere :: radius : Distance
```

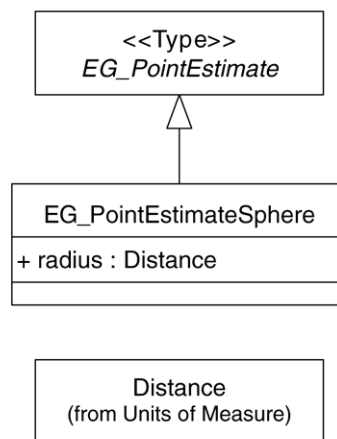


Figure 21 — Context Diagram: EG_PointEstimateSphere

6.3.7 EG_PointEstimateEllipsoid

6.3.7.1 Semantics

The point ellipsoid is the same as a point sphere except that an ellipsoid defined by the parameters given specifies the error. The point coordinates (inherited from `GM_Point`) specify the centre of the ellipsoid, and the best estimate of position. The UML for `EG_PointEstimateEllipsoid` is given in Figure 22.

6.3.7.2 Attribute: majorAxisRadius : Distance

The attribute “majorAxisRadius” is half the length of the major axis of the ellipsoid. In this case, major means “primary” or “central”. It may not necessarily be the longest of the three axes. It is the axis of rotation in the case where the remaining axes are equal.

```
EG_PointEstimateEllipsoid :: majorAxisRadius : Distance
```

6.3.7.3 Attribute: minorAxisRadius : Distance

The attribute “minorAxisRadius” is half the length of the minor axis of the ellipsoid. If one axis radius is given, the ellipsoid is a figure of rotation about the major axis, with this radius at the central “equator”. If two axes are given, then the figure is a fully general tri-axis ellipsoid.

```
EG_PointEstimateEllipsoid :: minorAxisRadius : Distance
```

6.3.7.4 Attribute: majorAxisBearing : Angle

The attribute “majorAxisBearing” is bearing of the major axis of the ellipsoid, given as an angle from true north. The minor axis is perpendicular to the major axis. If a second angle is given, this is an elevation angle from the local horizontal.

```
EG_PointEstimateEllipsoid :: majorAxisBearing[1..2] : Angle
```

6.3.7.5 Attribute: minorAxisBearing : Angle

The attribute “minorAxisBearing” is bearing of the minor axis of the ellipsoid, given as an angle from true north. The minor axis is perpendicular to the major axis. If no angle is given, then the minor axis is in the horizontal plane, perpendicular to the major axis. If a second angle is given, this is an elevation angle from the local horizontal. The third axis (second minor axis, if needed) is perpendicular to both other axes and therefore its direction is fixed by the values already given. Its unit vector direction in 3D is the cross product of the unit vector directions of the other two axes.

```
EG_PointEstimateEllipsoid :: minorAxisBearing[0..2] : Angle
```

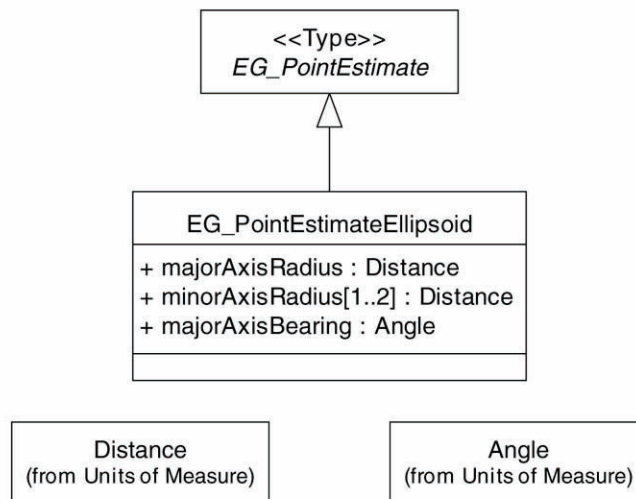


Figure 22 — Context Diagram: EG_PointEstimateEllipsoid

6.4 Package: Location Transformation

6.4.1 Semantics

The package “Location Transformation” contains classes to support the transformation of the various forms of location into one another.

6.4.2 LT_LocationTransformationService

6.4.2.1 Semantics

The abstract type “LT_LocationTransformationService” provides interfaces for services based upon the transformation of one form of location into another. The most common example of such a service is a geocoder, which maps addresses to geographic locations and vice versa. The UML for LT_LocationTransformationService is given in Figure 23.

6.4.2.2 Operation: cast

The operation “cast” takes one form of location and creates another form of the same location.

```
LT_LocationTransformationService ::
  cast(p : TK_TrackingLocation, asType : TK_PositionType) :
    TK_TrackingLocation
```

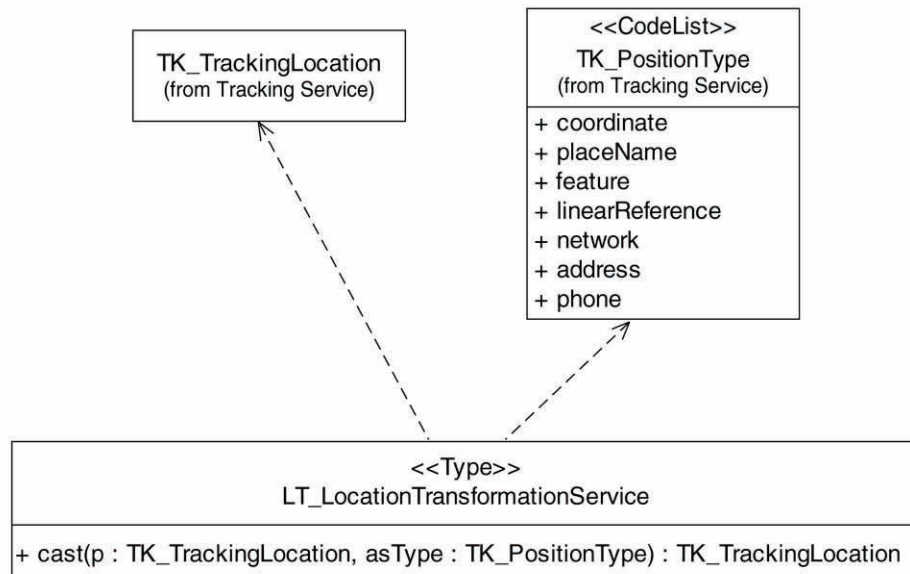


Figure 23 — Context Diagram: LT_LocationTransformationService

6.5 Package: Measured Coordinates

6.5.1 Semantics

The package “Measured Coordinates” contains classes for associating measures to coordinate positions. This is most commonly done in the creation of linear reference systems. This is done by modifying the coordinate reference systems defined in ISO 19111, by adding extra coordinate axes to the systems that allow for the storage of dependent coordinate measures, such as length measures.

This concept is illustrated in Figure 24. This illustrates how direct positions, the foundation of coordinate geometries in ISO 19107, are modified to carry measure information. Figure 25 shows the mechanism for altering a coordinate reference system to allow for the additional axes to be added allowing for the measure. Normally, only one measure is added (usually a nominal arc distance), and other measures are given as functions of this basic one.

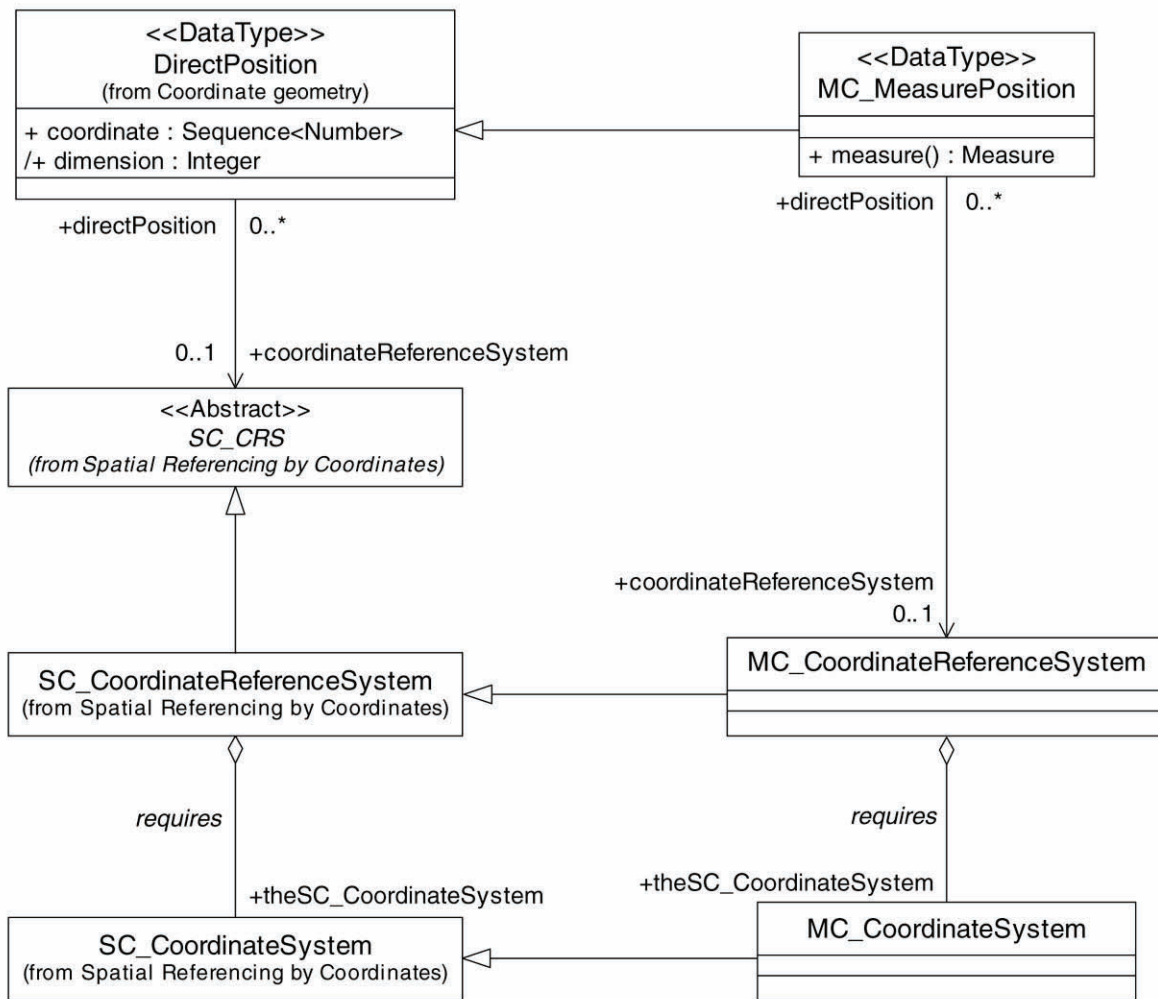


Figure 24 — Measure Position

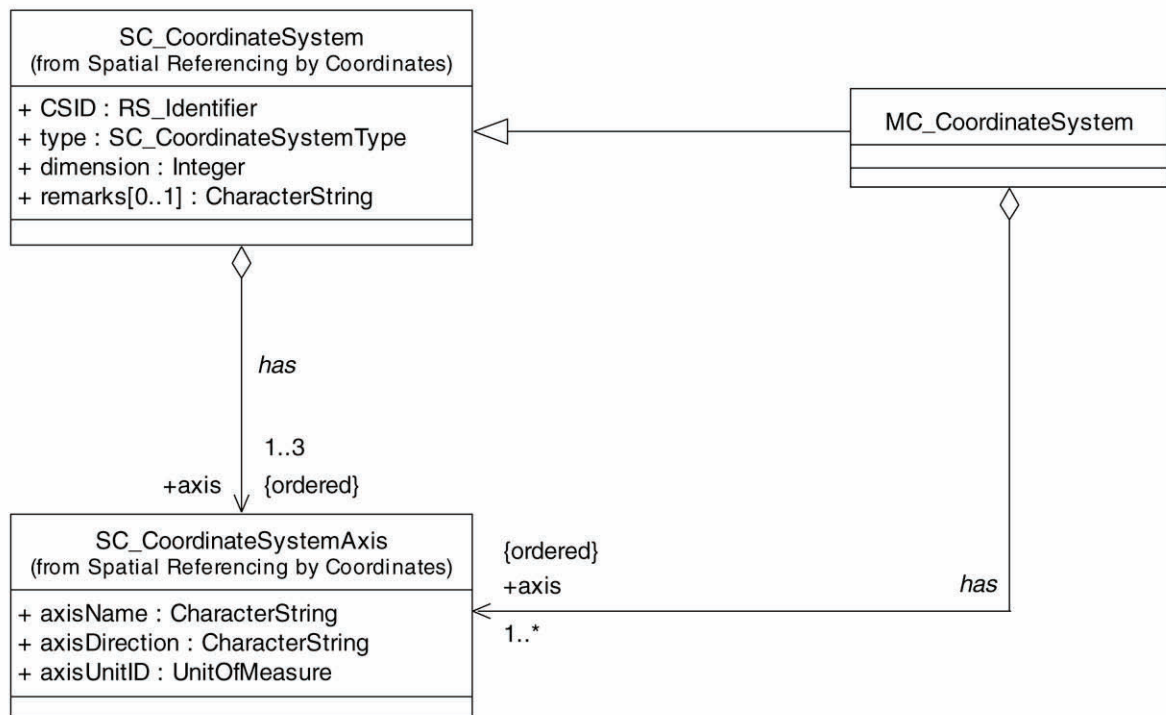


Figure 25 — Measured Coordinate Systems

6.5.2 MC_MeasurePosition

6.5.2.1 Semantics

The data type “MC_MeasurePosition” extends the concept of a DirectPosition to include additional measures as ordinates. The UML for MC_MeasurePosition is given in Figure 26.

6.5.2.2 Role: coordinateReferenceSystem[0..1] : MC_CoordinateReferenceSystem

The optional association role “CoordinateReferenceSystem” inherited from DirectPosition, will now reference a measured coordinate system. If this role is empty, then, as in the case of direct position, there shall be a larger context that implies the coordinate reference system.

```
MC_MeasurePosition :: CoordinateReferenceSystem[0..1] :
    MC_CoordinateReferenceSystem
```

6.5.2.3 Operation: measure

The operation “measure” gives the measured position of a projection onto its measure component.

```
MC_MeasurePosition :: measure() : Measure
```

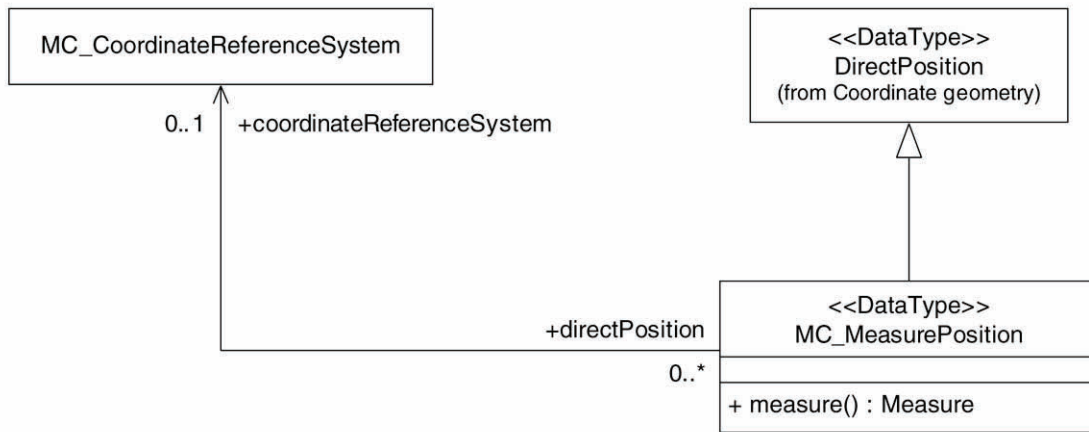


Figure 26 — Context Diagram: MC_MeasurePosition

6.5.3 MC_CoordinateSystem

6.5.3.1 Semantics

The class “MC_CoordinateSystem” describes coordinate systems that have been augmented with measures. The UML for MC_CoordinateSystem is given in Figure 27.

6.5.3.2 Role: axis[1..*] : SC_CoordinateSystemAxis

The role “axis” aggregates the axes for the coordinate reference system. This association is inherited from ISO 19111, but is modified here to accommodate additional axes for the measures. Normally, only one strictly monotonic measure axis is needed, as other measures can be implemented as functions of this one. The most common measure for this is some version of arc length for curves.

```
MC_CoordinateSystem :: axis[1..*]: SC_CoordinateSystemAxis
```

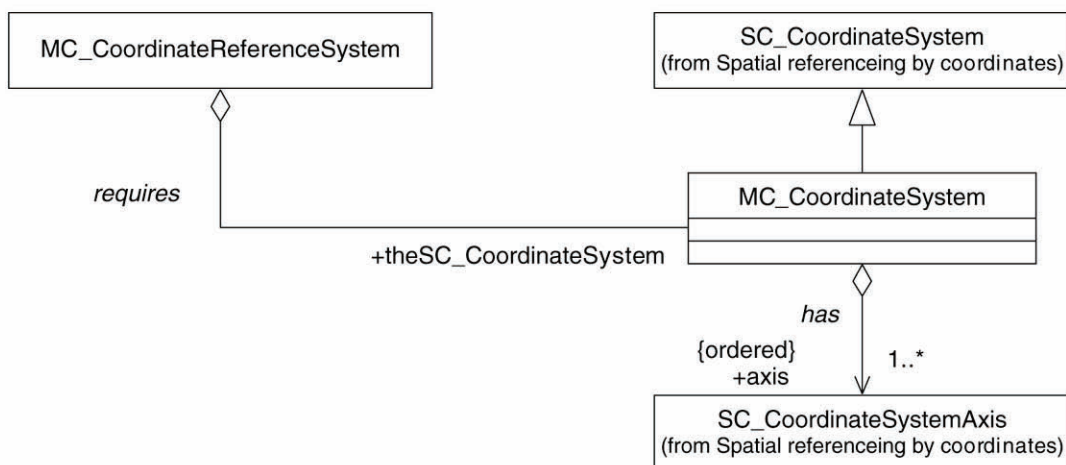


Figure 27 — Context Diagram: MC_CoordinateSystem

6.5.4 MC_CoordinateReferenceSystem

6.5.4.1 Semantics

The class “MC_CoordinateReferenceSystem” extends the class “SC_CoordinateReferenceSystem” defined in ISO 19111 to allow for additional coordinate axes to accommodate measures. The UML for MC_CoordinateReferenceSystem is given in Figure 28.

6.5.4.2 Role: theSC_CoordinateSystem : MC_CoordinateSystem

The association role “theSC_CoordinateSystem” inherited from ISO 19111 is extended here to allow for target class changes.

```
MC_CoordinateReferenceSystem ::
    theSC_CoordinateSystem : MC_CoordinateSystem
```

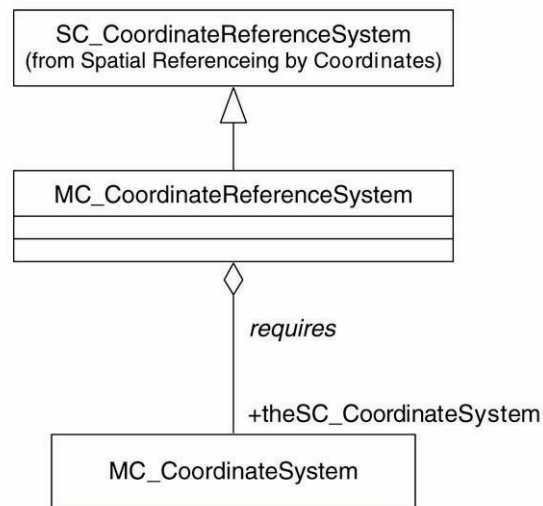


Figure 28 — Context Diagram: MC_CoordinateReferenceSystem

6.6 Package: Linear Reference Systems

6.6.1 Semantics

The package “Linear Reference Systems” supplies classes and types to the definition of linear reference systems. Linear reference systems are in wide use in transportation. They allow for the specification of positions along curvilinear features by using measured distances from known positions, usually represented by physical markers along the right-of-way of the transportation feature. The classes for this system and their relationships are depicted in Figure 29.

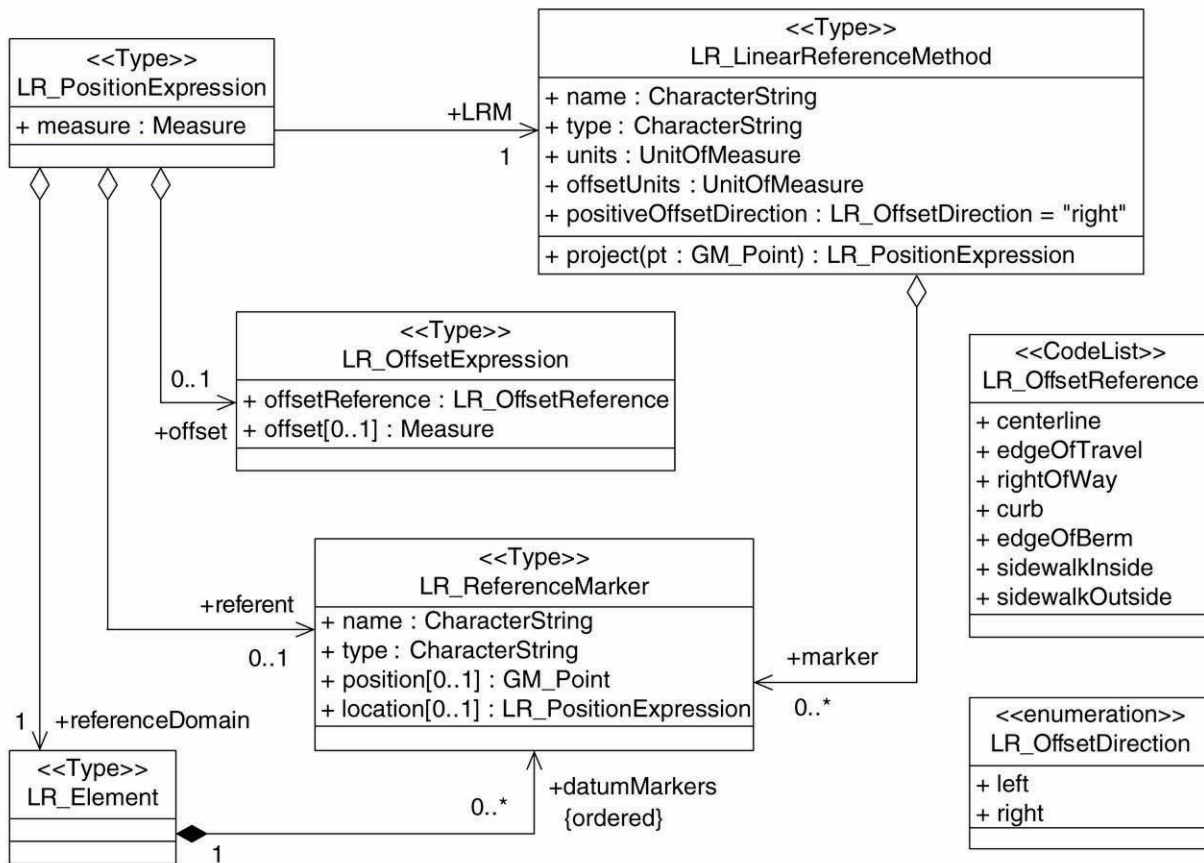


Figure 29 — LRS classes

6.6.2 LR_PositionExpression

6.6.2.1 Semantics

The class “LR_PositionExpression” is used to describe position given by a measure value, a curvilinear element being measured, and the method of measurement. The UML for LR_PositionExpression is given in Figure 30.

6.6.2.2 Attribute: measure : Measure

The attribute “measure” gives the measure (usually a distance) of this position expression.

```
LR_PositionExpression :: measure : Measure
```

6.6.2.3 Role: LRM : LR_LinearReferenceMethod

The role “LRM” gives the linear reference method used for this position expression.

```
LR_PositionExpression :: LRM : LR_LinearReferenceMethod
```

6.6.2.4 Role: referent [0..1] : LR_ReferenceMarker

The optional association role “referent” gives the marker or known position from which the measure is taken for the linear reference method used for this position expression. If the referent is absent, the measurement is made from the start of the LR_element.

```
LR_PositionExpression :: referent [0..1]: LR_ReferenceMarker
```

6.6.2.5 Role: referenceDomain : LR_Element

The role “referenceDomain” gives the linear object upon which the measure is taken for the linear reference method used for this position expression.

```
LR_PositionExpression :: referenceDomain : LR_Element
```

6.6.2.6 Role: offset[0..1] : LR_OffsetExpression

The optional association role “offset” gives perpendicular distance offset of this position expression. If the offset is absent, then the position is on the LR_element.

```
LR_PositionExpression :: offset[0..1] : LR_OffsetExpression
```

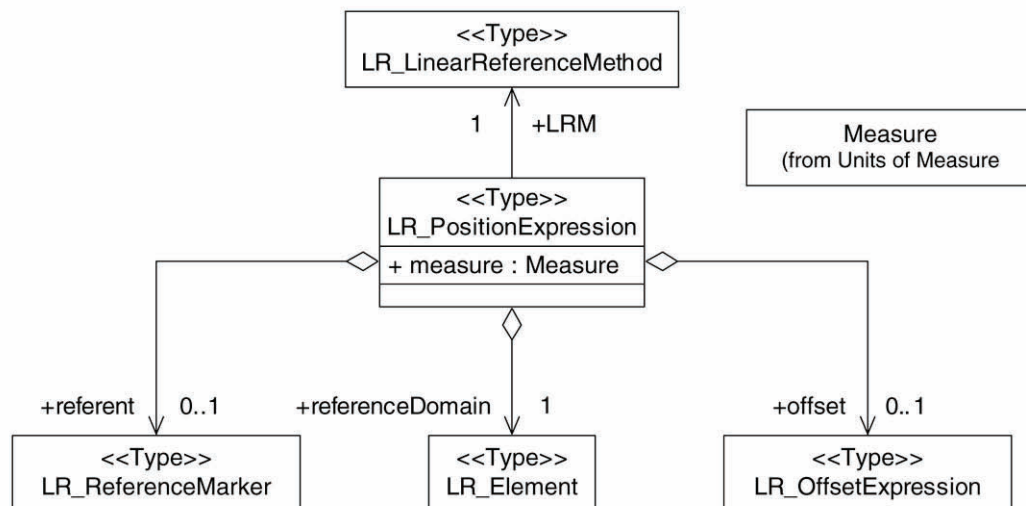


Figure 30 — Context Diagram: LR_PositionExpression

6.6.3 LR_LinearReferenceMethod

6.6.3.1 Semantics

The type “LR_LinearReferenceMethod” describes the manner in which measurements are made along (and optionally laterally offset from) a curvilinear element. The UML for LR_LinearReferenceMethod is given in Figure 31.

6.6.3.2 Attribute: name : CharacterString

The attribute: “name” gives the name of this linear reference method.

```
LR_LinearReferenceMethod :: name : CharacterString
```

6.6.3.3 Attribute: type : CharacterString

The attribute: “type” gives the type of this linear reference method.

```
LR_LinearReferenceMethod :: type : CharacterString
```

6.6.3.4 Attribute: units : UnitOfMeasure

The attribute: “units” gives the units of measure used for this linear reference method for measures along the base elements.

```
LR_LinearReferenceMethod :: units : UnitOfMeasure
```

6.6.3.5 Attribute: offsetUnits : UnitOfMeasure

The attribute: “offsetUnits” gives the units of measure used for this linear reference method for measures perpendicular to the base elements.

```
LR_LinearReferenceMethod :: offsetUnits : UnitOfMeasure
```

6.6.3.6 Attribute: positiveOffsetDirection : LR_OffsetDirection = "right"

The attribute: “positiveOffsetDirection” gives the direction used as positive for this linear reference method for measures perpendicular to the base elements. The default value is right for positive, left for negative.

```
LR_LinearReferenceMethod ::  
  positiveOffsetDirection : LR_OffsetDirection = "right"
```

6.6.3.7 Role: marker[1..*] : LR_ReferenceMarker

The association role “marker” aggregates all reference markers used by the linear reference methods. Normally, this will be grouped by linear element.

```
LR_LinearReferenceMethod :: marker[0..*] : LR_ReferenceMarker
```

6.6.3.8 Role: referenceElement[1..*] : LR_Element

The role: “referenceElement” aggregates all the linear elements along which this method is supported.

```
LR_LinearReferenceMethod :: referenceElement[1..*] : LR_Element
```

6.6.3.9 Operation: project

The operation “project” will find the measure of the point on a base element closest to the given point, and then express the point as a position expression for the linear reference method. If the point is precisely on one of the linear elements, then the offset will be zero there is no offset expression.

```
LR_LinearReferenceMethod ::  
  project(pt : GM_Point) : LR_PositionExpression
```

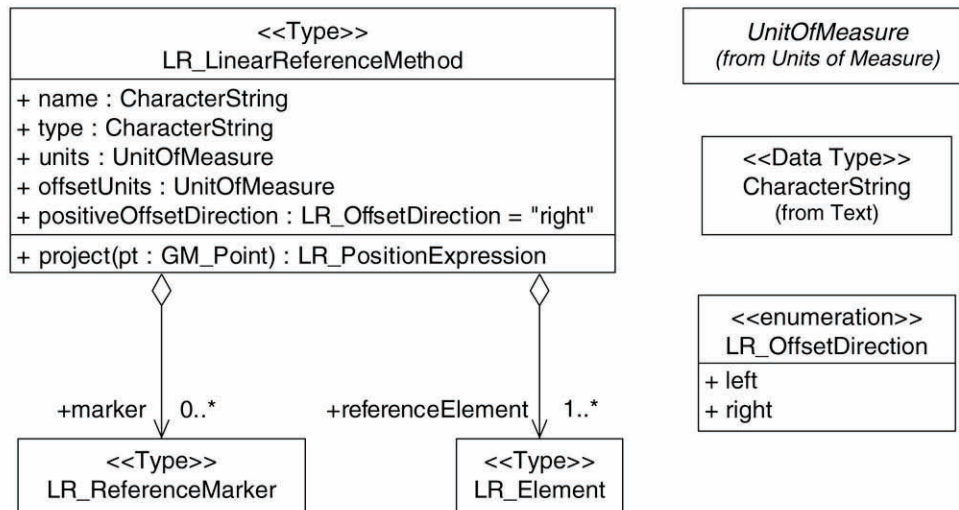


Figure 31 — Context Diagram: LR_LinearReferenceMethod

6.6.4 LR_OffsetDirection

The enumeration “LR_OffsetDirection” gives the two options for offset measure. The values are left and right. This offset direction is as viewed from above the linear element facing in the direction of increasing measure. The UML for LR_OffsetDirection is given in Figure 32.

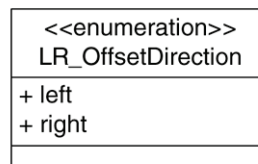


Figure 32 — Context Diagram: LR_OffsetDirection

6.6.5 LR_ReferenceMarker

6.6.5.1 Semantics

The type “LR_ReferenceMarker” is used to describe reference markers used in linear reference systems. At least one of the attributes “position” or “location” shall be given. If both are given they shall refer to the same physical location. The UML for LR_ReferenceMarker is given in Figure 33.

6.6.5.2 Attribute: name : CharacterString

The attribute “name” is the identifier used for this marker.

```
LR_ReferenceMarker :: name : CharacterString
```

6.6.5.3 Attribute: type : CharacterString

The attribute “type” is the type of this marker.

```
LR_ReferenceMarker :: type : CharacterString
```

6.6.5.4 Attribute: position[0..1] : GM_Point

The optional attribute “position” is the position of this for this marker, given in some coordinate system. If this attribute is not given, then the “location” shall be given.

```
LR_ReferenceMarker :: position[0..1] : GM_Point
```

6.6.5.5 Attribute: location[0..1] : LR_PositionExpression

The optional attribute “location” is the location of this marker given as a linear reference measure along and from the start of the underlying linear element.

```
LR_ReferenceMarker :: location[0..1] : LR_PositionExpression
```

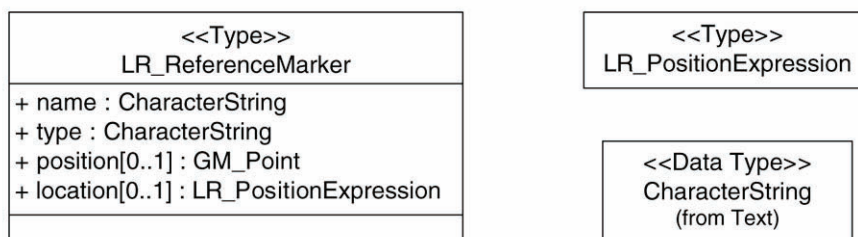


Figure 33 — Context Diagram: LR_ReferenceMarker

6.6.6 LR_OffsetReference

The code list “LR_OffsetReference” enumerates the offset reference types used for this linear reference method (see Figure 36). The initial value domain included:

- 1) “centerline” centre of the structure of the highway, or reference line for the highway
- 2) “edgeOfTravel” outside edge of all travel lanes
- 3) “edgeOfPavement” outside edge of travel-lane quality paved surface
- 4) “rightOfWay” edge of the legal right of way
- 5) “curbFace” curb (the roadway must be curbed for this to be used)
- 6) “curbBack” curb (the roadway must be curbed for this to be used)
- 7) “edgeOfShoulder” outside edge of all hardened surface (paved or gravel) surface
- 8) “edgeOfBerm” outside edge of levelled land for the road structure
- 9) “walkwayInside” sidewalk edge closest to travel lanes
- 10) “walkwayOutside” sidewalk edge furthest from travel lanes