



GENIVI MapViewer API

Release 3.0.2
Status: Approved

30 June 2014

Accepted for release by:

This document has been accepted for the GENIVI Gemini Release by the Expert Group Location Based Services (EG-LBS).

Abstract:

This document describes the API of the MapViewer Abstract Component.

Keywords:

NavigationAPIs, MapViewer.

SPDX-License-Identifier: CC-BY-SA-4.0

Copyright (C) 2014, BMW Car IT GmbH, Continental Automotive GmbH, Elektrobit Automotive GmbH, Neusoft Technology Solutions GmbH, PCA Peugeot Citroën, TomTom International B.V., XS Embedded GmbH

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License

To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/4.0/>

or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

Table of contents

1	Change History	4
2	Introduction	5
3	Terminology	6
4	Requirements	7
5	Architecture.....	8
5.1	Interfaces.....	8
5.2	Interaction with other Components.....	9
6	API	10
6.1	D-Bus	10
6.2	Git Repository	10
6.3	Naming Convention	10
6.4	Data Types Convention.....	11
6.5	Errors	12
6.6	Sequence Diagrams.....	13
6.6.1	navigation application browses map	13
6.6.2	navigation application creates map session.....	14
6.6.3	navigation application sets center.....	15
6.6.4	navigation application sets map zoom by delta.....	16
6.6.5	navigation application shows route	17
6.7	Interfaces.....	18

1 Change History

Version	Date	Author	Change
0.1	27 Feb 2012	Marco Residori (XS Embedded)	Document Created
0.2	19 Mar 2012	Marco Residori (XS Embedded)	Updated sequence diagrams. Updated <i>Interfaces</i> chapter.
0.3	21 Mar 2012	Marco Residori (XS Embedded)	Updated <i>Interfaces</i> chapter.
1.0	22 Mar 2012	Marco Residori (XS Embedded)	System Architecture Team (SAT) approval.
2.0 (beta)	07 Jun 2013	Marco Residori (XS Embedded)	Updated API description. API Version 2.0.
2.0	17 Jun 2013	Marco Residori (XS Embedded)	Updated API description. API fixes: GT-2651. API Version 2.0 (<i>emini-final</i> tag)
3.0.0	21 Jan 2014	Marco Residori (XS Embedded)	Updated API description. API Version 3.0.0.
3.0.1	22 May 2014	Marco Residori (XS Embedded)	Updated copyright notes.
3.0.2	30 June 2014	Marco Residori (XS Embedded)	Updated contributors list.

2 Introduction

This document describes the MapViewer API.

3 Terminology

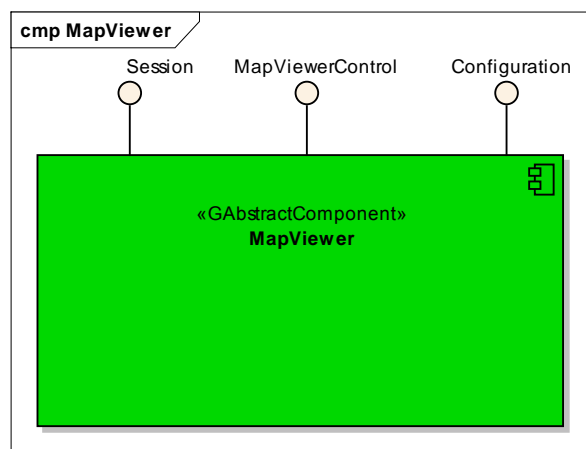
<i>Term</i>	<i>Description</i>
TargetPoint	Point the camera looks at. If the map viewer is set to follow the car position, it coincides with the vehicle position.

4 Requirements

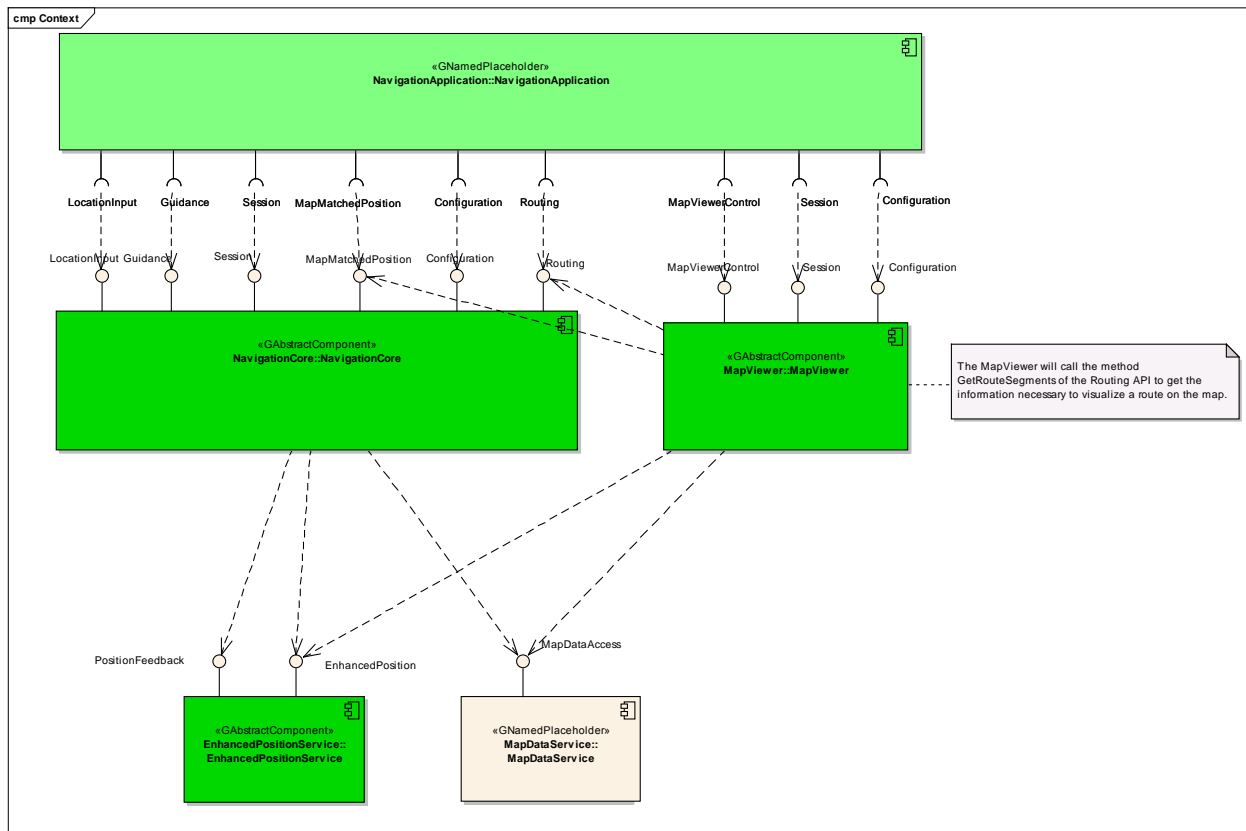
- Ease of Development
- Extensibility
- Multi-client Behavior
- Simplicity

5 Architecture

5.1 Interfaces



5.2 Interaction with other Components



6 API

6.1 D-Bus

The MapViewer interfaces are D-Bus interfaces. They are defined using the D-Bus introspection data format, which is nothing but an IDL expressed in XML format.

For more information about the D-Bus data types please refer to the following website:

<http://dbus.freedesktop.org/doc/dbus-specification.html#message-protocol-signatures>

For more information about the D-Bus introspection data format, please refer to the following website:

<http://dbus.freedesktop.org/doc/dbus-specification.html#introspection-format>

6.2 Git Repository

The MapViewer interfaces can be found in the GENIVI Git repository at:

<https://git.genivi.org/git/gitweb.cgi?p=navigation;a=tree;f=MapView/api>

6.3 Naming Convention

Element	Description	Example
Interface File	genivi.<component name in lowercase character>.<interface name in lowercase characters>	genivi.navigationcore.mapviewer control.xml
Methods/Signal/Properties	Camel case naming convention. First letter uppercase	CreateMapViewInstance
Arguments	Camel case naming convention. First letter lowercase	scaleID

6.4 Data Types Convention

D-bus types code are used. Please refer to the following webpage for more information:

<http://dbus.freedesktop.org/doc/dbus-specification.html>

Element	D-Bus Data Type Code	Example
Enumerators	q (uint16)	
Handles	y (uint8)	
Maps	a{qv}	Dictionary of tuples (key, value) The key is expressed as an enumerator

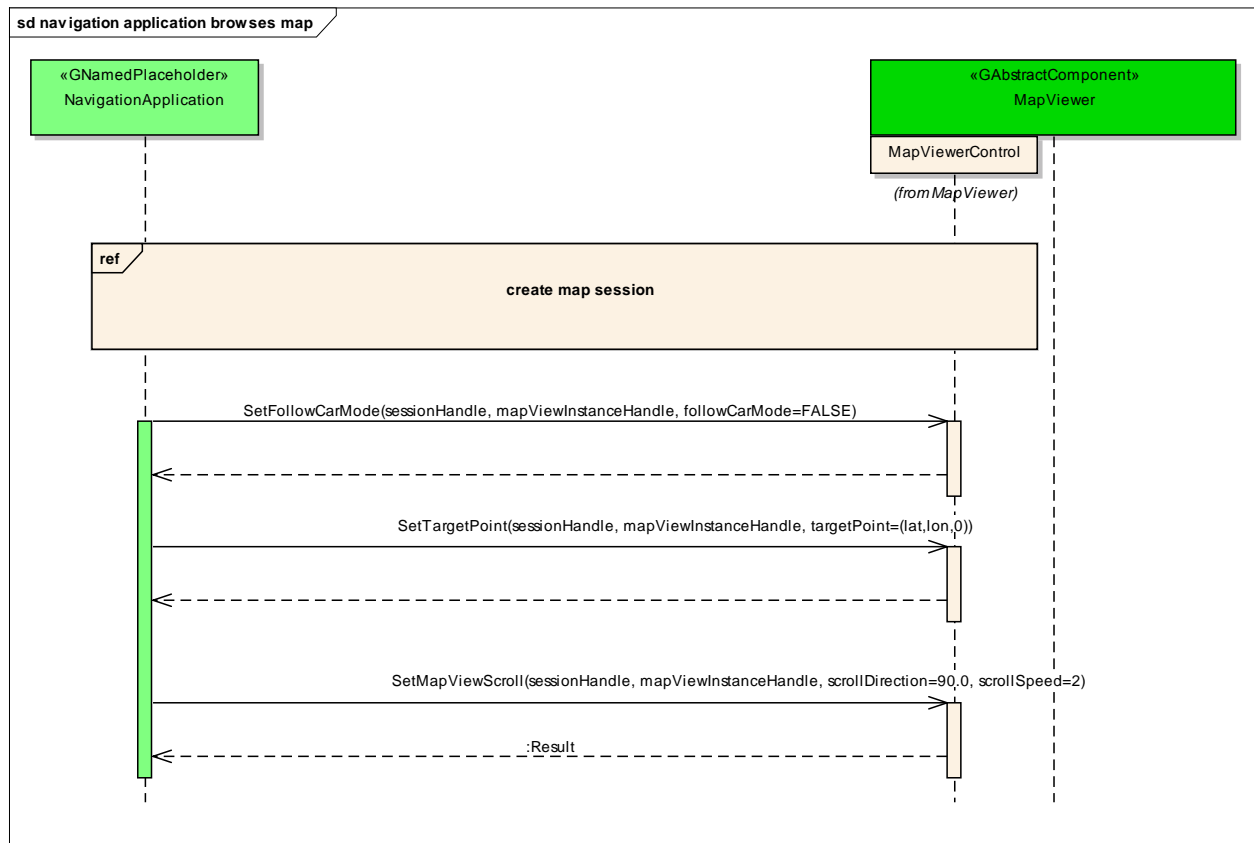
6.5 Errors

Error Type	Description	Example	Error Documentation	Note
User Error	Error caused by user actions	The user tries to start route guidance, although guidance is already running	Application specific error string documented in the XML file	Can occur in final product
Hardware Error	Error related to hardware/database related problems	No map data	Application specific error string documented in the XML file	Can occur in final product
Protocol Error	Error caused by wrong sequence of commands	Wrong sequence of commands to enter destination	Standard D-Bus error string	Should not occur in final product
Bus Error	D-Bus communication error	Bus busy	Standard D-Bus error string	Can occur in final product
Programming Error	Programming Error	Invalid parameters	Standard D-Bus error string and debug messages	Should not occur in production code

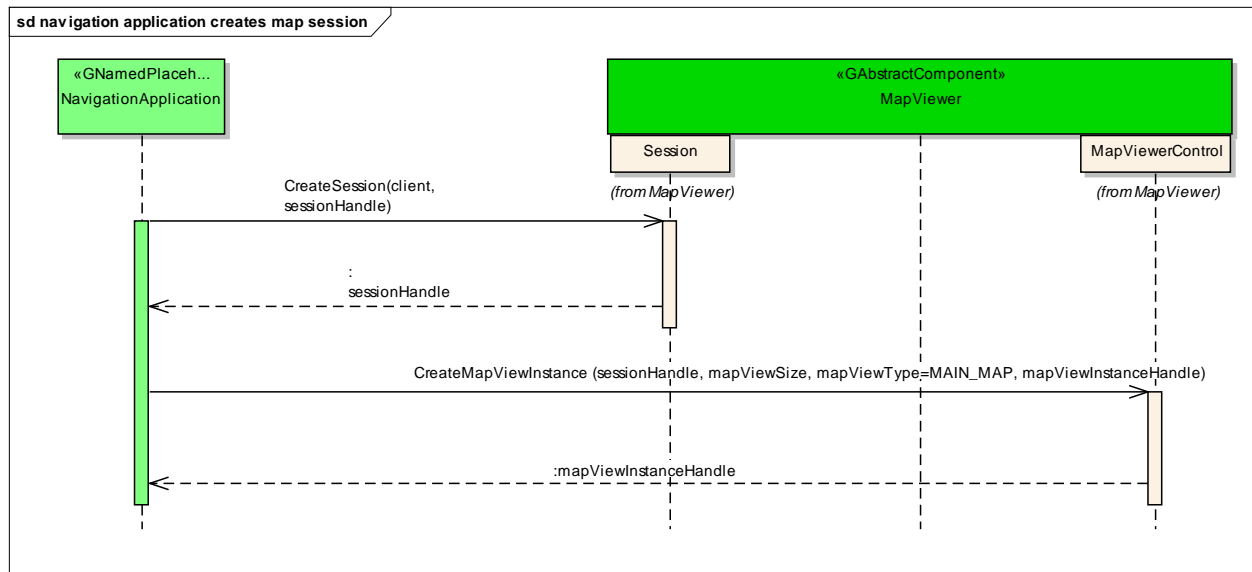
Only application-specific errors are documented directly in the interfaces (XML files). For all other errors, standard D-Bus strings are used. These kinds of strings are not documented in the interfaces. It is implicitly assumed that every method may return a standard D-Bus error string.

6.6 Sequence Diagrams

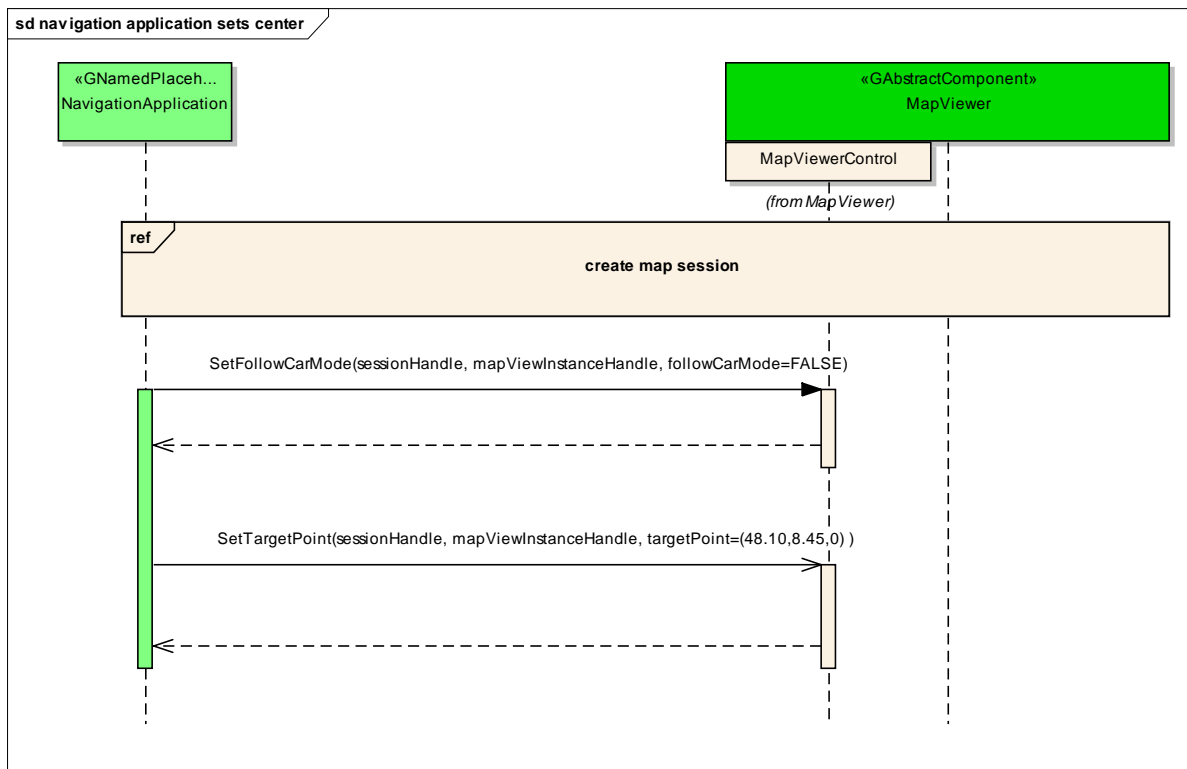
6.6.1 navigation application browses map



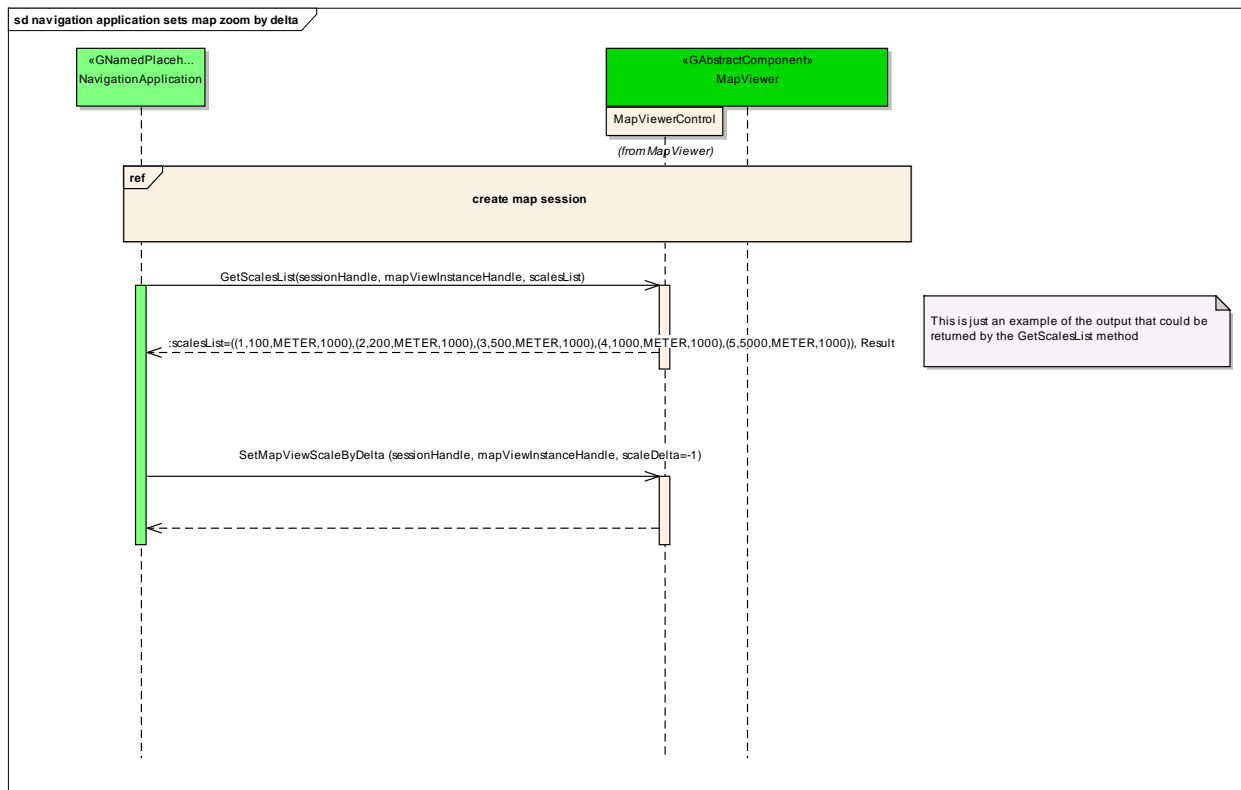
6.6.2 navigation application creates map session



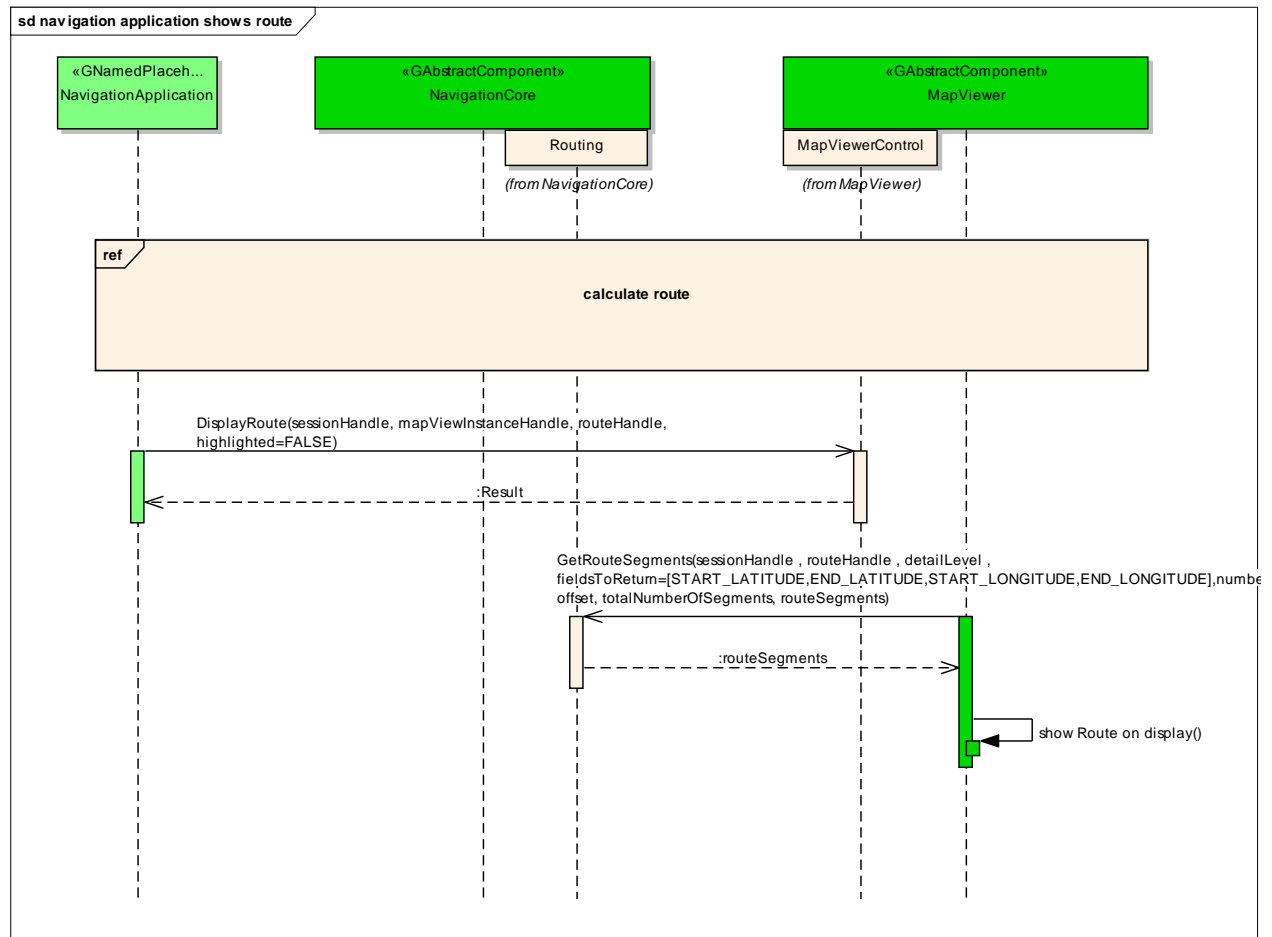
6.6.3 navigation application sets center



6.6.4 navigation application sets map zoom by delta



6.6.5 navigation application shows route



6.7 Interfaces

interface

org.genivi.mapviewer.MapViewerControl

version 3.1.0-alpha (03-03-2014)

MapViewControl = This interface offers functions to control the MapViewer

GetVersion = This method returns the API version implemented by the server application

method GetVersion

version = struct(major,minor,micro,date)

major = when the major changes, then backward compatibility with previous releases is not granted

minor = when the minor changes, then backward compatibility with previous releases is granted, but something changed in the implementation of the API (e.g. new methods may have been added)

micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or documentation modifications)

date = release date (e.g. 21-06-2011)

out (qqqs) version

CreateMapViewInstance = This method creates a new map instance

Note: when a map instance is created, it is set to 'not visible' by default

method CreateMapViewInstance

sessionHandle = Session handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewSize = struct(horizontalSize,verticalSize) of the map instance in pixels.

horizontalSize = horizontal size of the map instance in pixels

verticalSize = vertical size of the map instance in pixels

in (qq) mapViewSize

mapViewType = enum(INVALID,MAIN_MAP,SPLIT_SCREEN, ...)

Note: to be used in case that a specific position of the map instance with respect to the display viewport is required

For example, if the instance 1 is of type MAIN_MAP and the instance 2 is of type SPLIT_SCREEN, the offset of the map instances

with respect to the display viewport could be (0;0) and (hres/2;0) respectively

in q mapViewType

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

This error is generated if no more map view instance handles are available

error org.genivi.mapviewer.MapViewerControl.Error.NoMoreMapViewInstanceHandles

ReleaseMapViewInstance = This method releases (i.e. destroys) a given map instance. Only invisible map instances can be released

method ReleaseMapViewInstance

sessionHandle = Session handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

This error is generated if an application tries to delete a map view instance handle that is not available

error *org.genivi.mapviewer.MapViewerControl.Error.MapViewInstanceNotAvailable*

GetMapViewType = This method returns the map type of a map instance as it was set using CreateMapViewInstance

method GetMapViewType

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

mapViewType = enum(INVALID,MAIN_MAP,SPLIT_SCREEN, ...)

out q mapViewType

GetSupportedMapViewTypes = This method retrieves the supported map view types

method GetSupportedMapViewTypes

mapViewTypeList = array[mapViewType]

mapViewType = enum(INVALID,MAIN_MAP,SPLIT_SCREEN, ...)

out aq mapViewTypeList

SetTargetPoint = This method sets the position of the point the camera is always aimed at

Note: the target point is typically visualized in the center of the map

method SetTargetPoint

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

targetPoint = (lat,lon,alt)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

alt = altitude to the ground in meters

in (ddi) targetPoint

GetTargetPoint = This method retrieves the target point position

Note: if the FollowCar mode is active, the this method will return the current vehicle position

method GetTargetPoint

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

targetPoint = (lat,lon,alt)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

alt = altitude to the ground in meters

out (ddi) targetPoint

SetFollowCarMode = This method sets the FollowCar mode

Note: if the FollowCar is activated, the current car position is interpreted as target point position

method SetFollowCarMode

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

followCarMode = flag. If true, the current car position is interpreted as position of the point the camera must look at

in b followCarMode

GetFollowCarMode = This method returns the current FollowCar-mode

method GetFollowCarMode

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

followCarMode = flag. If true, the current car position is interpreted as position of the point the camera must look at

out b followCarMode

FollowCarModeChanged = This signal is emitted when the follow-car-mode changes

signal FollowCarModeChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

followCarMode = flag. If true, the current car position is interpreted as position of the point the camera must look at

out b followCarMode

SetCameraPosition = This method sets the coordinates of the point at which the camera must be positioned

Note: the camera heading will be automatically set in such a way, that the camera is aimed at the view point

method SetCameraPosition

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

position = (lat,lon,alt)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

alt = altitude to the ground in meters

in (ddi) position

GetCameraPosition = This method returns the coordinates of the point at which the camera is positioned

method GetCameraPosition

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

position = (lat,lon,alt)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

alt = altitude to the ground in meters

out (ddi) position

CameraPositionChanged = This signal is emitted when the position of the camera changes

signal CameraPositionChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

targetPoint = (lat,lon,alt)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

alt = altitude to the ground in meters

out (ddi) targetPoint

SetCameraHeadingAngle = This method sets the map view heading angle

Note: the camera position will be automatically set in such a way, that it looks at the currently selected target point

method SetCameraHeadingAngle

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

heading = heading angle in degrees. Range [0:360]

The angle in degrees between projection on the ground plane of the line through the center of the screen and the top-centre of the screen, and the North direction

0 degrees means that the map view is oriented such that North is at the top of the screen

Degrees are measured clockwise such that 90 degrees correspond to the East direction

in i heading

SetCameraHeadingToTarget = This method sets the camera heading in such a way, that the camera always looks at a given target

Note: the camera position will be automatically set in such a way, that it aims at the current view point

method SetCameraHeadingToTarget

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

target = struct(lat,lon)

lat = latitude of the point towards which the map vertical axis must be aligned in format %3.6f. Range[-90:+90]

lon = longitude of the point towards which the map vertical axis must be aligned in format %3.6f. Range[-180:+180]

in (dd) target

SetCameraHeadingTrackUp = This method sets the camera heading in such a way, that the camera always looks in the direction in which the car is moving

method SetCameraHeadingTrackUp

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

GetCameraHeading = This method returns the current camera heading

method GetCameraHeading

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

headingType = enum(INVALID,CONSTANT_ANGLE,TRACK_UP,TOWARDS_TARGET, ...)

out q headingType

headingAngle = heading angle in degrees measured from the North axis clockwise. Range[0:360]

Note: meaningful only if heading != TOWARDS_TARGET, otherwise it is just a dummy value and must be set to 0

out i headingAngle

headingTarget = struct(lat,lon)

lat = latitude of the point towards which the map vertical axis must be aligned in format %3.6f. Range[-90:+90]

lon = longitude of the point towards which the map vertical axis must be aligned in format %3.6f. Range[-180:+180]

Note: meaningful only if heading = TOWARDS_TARGET, otherwise it is just a dummy value and must be set to (0,0)

out (dd) target

CameraHeadingAngleChanged = This signal is emitted when the heading angle of the camera changes

signal CameraHeadingAngleChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

headingType = enum(INVALID,CONSTANT_ANGLE,TRACK_UP,TOWARDS_TARGET, ...)

out q headingType

headingAngle = heading angle in degrees. Range [0:360]

The angle in degrees between projection on the ground plane of the line through the center of the screen and the top-centre of the screen, and the North direction

0 degrees means that the map view is oriented such that North is at the top of the screen

Degrees are measured clockwise such that 90 degrees correspond to the East direction

out i headingAngle

SetCameraTiltAngle = This method sets the camera tilt angle

method SetCameraTiltAngle

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

tilt = tilt angle in degrees. Range [-180:180]

Example: 0 = down (straight down to the ground plane), 90 = front (parallel to the ground plane), 180 = top (straight up to the sky)

Negative angles [0:-180] result in an upside down view

in i tilt

GetMapViewTiltAngle = This method returns the camera tilt angle

method GetCameraTiltAngle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

tilt = tilt angle in degrees. Range [-180:180]

Example: 0 = down (straight down to the ground plane), 90 = front (parallel to the ground plane), 180 = top (straight up to the sky)

Negative angles [0:-180] result in an upside down view

out i tilt

CameraTiltAngleChanged = This signal is emitted when the camera tilt angle changes

signal CameraTiltAngleChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

tilt = tilt angle in degrees. Range [-180:180]

Example: 0 = down (straight down to the ground plane), 90 = front (parallel to the ground plane), 180 = top (straight up to the sky)

Negative angles [0:-180] result in an upside down view

out i tilt

[Optional]

SetCameraRollAngle = This method sets the camera roll angle

method SetCameraRollAngle

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

roll = roll angle in degrees. Range [-180:180]

in i roll

[Optional]

GetCameraRollAngle = This method returns the camera roll angle

method GetCameraRollAngle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

roll = roll angle in degrees. Range [-180:180]

out i roll

SetCameraDistanceFromTargetPoint = This method sets the mode and the camera distance from the target point

Note: this method can be used to zoom in and out

method SetCameraDistanceFromTargetPoint

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

distance = distance from the view point in meters

in u distance

GetCameraDistanceFromTargetPoint = This method gets the mode and the camera distance from the target point

method GetCameraDistanceFromTargetPoint

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

distance = distance from the view point in meters

out u distance

CameraDistanceFromTargetPointChanged = This signal is emitted when the camera distance from the target point changes

signal CameraDistanceFromTargetPointChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

distance = distance from the view point in meters

out u distance

SetMapViewScaleMode = This method sets the scaling mode.

method SetMapViewScaleMode

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

scaleMode = enum(AUTOMATIC,MANUAL,HYBRID)

scaleMode = AUTOMATIC, adjusts the camera distance automatically (e.g. depending on the speed)

scaleMode = MANUAL, the camera distance is specified by the argument 'distance'

scaleMode = HYBRID, e.g. AUTOMATIC depending on the proximity to the target, MANUAL otherwise

in q scaleMode

GetMapViewScaleMode = This method gets the scaling mode.

method GetMapViewScaleMode

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

scaleMode = enum(AUTOMATIC,MANUAL,HYBRID)

scaleMode = AUTOMATIC, adjusts the camera distance automatically (e.g. depending on the speed)

scaleMode = MANUAL, the camera distance is specified by the argument 'distance'

scaleMode = HYBRID, e.g. AUTOMATIC depending on the proximity to the target, MANUAL otherwise

out q scaleMode

GetSupportedMapViewScaleModes = This method gets the supported scaling modes.

method GetSupportedMapViewScaleModes

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

scaleModeList = array[scaleMode]

scaleMode = enum(AUTOMATIC,MANUAL,HYBRID)

scaleMode = AUTOMATIC, adjusts the camera distance automatically (e.g. depending on the speed)

scaleMode = MANUAL, the camera distance is specified by the argument 'distance'

scaleMode = HYBRID, e.g. AUTOMATIC depending on the proximity to the target, MANUAL otherwise

out aq scaleModeList

MapViewScaleChanged = This signal is emitted when the mapview scale changes

signal MapViewScaleChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

scaleID = scale identifier. Range[0:256]

out y scaleID

isMinMax = enum(INVALID,MIN,MAX,MID, ...)

MIN = scale ID minimal value

MID = scale ID intermediate value

MAX = scale ID maximal value

out q isMinMax

AddMapViewScaleChangeListener = This method adds a listener which is notified when map view scale changes.

method AddMapViewScaleChangeListener

RemoveMapViewScaleChangeListener = This method removes a listener which is notified when map view scale changes.

method RemoveMapViewScaleChangeListener

SetCameraHeight = This method sets the camera height

Note: this method is a subset of SetCameraPosition

method SetCameraHeight

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

height = height from the ground in meters

in u height

GetCameraHeight = This method gets the camera height

Note: this method is a subset of GetCameraPosition

method GetCameraHeight

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

height = height from the ground in meters

out u height

SetMapViewPerspective = This method sets the map perspective

method SetMapViewPerspective

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

perspective = enum(INVALID,2D,3D, ...)

in q perspective

GetMapViewPerspective = This method returns the current map perspective

method GetMapViewPerspective

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

perspective = enum(INVALID,2D,3D, ...)

out q perspective

MapViewPerspectiveChanged = This signal is emitted when the map perspective changes

signal MapViewPerspectiveChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

perspective = enum(INVALID,2D,3D, ...)

out q perspective

GetSupportedMapViewPerspectives = This method retrieves the supported mapview perspectives

method GetSupportedMapViewPerspectives

perspectiveList = array[perspective]

perspective = enum(INVALID,2D,3D, ...)

out aq perspectiveList

SetMapViewObjectVisibility = This method specifies the type of objects to show on the map.

method SetMapViewObjectVisibility

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

objectVisibilityList = array[objectVisibility]

objectVisibility = dictionary[key,value]

key = enum[BUILDINGS,TERRAIN, ...]

value = value of type 'b'; if true the objects are shown else they are not shown

in a{qb} objectVisibilityList

GetMapViewObjectVisibility = This method gets the type of objects shown on the map.

method GetMapViewObjectVisibility

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

objectVisibilityList = array[objectVisibility]

objectVisibility = dictionary[key,value]

key = enum[BUILDINGS,TERRAIN, ...]

value = value of type 'b'; if true the objects are shown else they are not shown

out a{qb} objectVisibilityList

MapViewObjectVisibilityChanged = This signal is emitted when the type of objects shown on the map changes

signal MapViewObjectVisibilityChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

Out **u** mapViewInstanceHandle

objectVisibilityList = array[objectVisibility]

objectVisibility = dictionary[key,value]

key = enum[BUILDINGS,TERRAIN, ...]

value = value of type 'b'; if true the objects are shown else they are not shown

Out **a{qb}** objectVisibilityList

GetSupportedMapViewObjectVisibilities = This method gets the supported object visibilities.

method GetSupportedMapViewObjectVisibilities

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in **u** mapViewInstanceHandle

objectVisibilityList = array[objectVisibility]

objectVisibility = enum[BUILDINGS,TERRAIN, ...]

Out **aq** objectVisibilityList

GetScaleList = This method returns a list of supported map scales

Note: a mapscale consists of an unique ID, a ScaleValue, a ScaleUnit and a number of MillimetresPerPixel

method GetScaleList

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in **u** mapViewInstanceHandle

scaleList = array[struct(scaleId,scaleValue,scaleUnit,millimetersPerPixel)]

scaleId = scale identifier. Range [0:256]

scaleValue = scale value. It can assume values like 100, 200, 1000, ...

scaleUnit = unit of measurement. It is an enum(INVALID,METER,KM,MILE,YARD,FOOT, ...)

millimetersPerPixel = number indicating the number of millimeters per pixel

Example: ((1,100,METER,1000),(2,200,METER,1000),(3,500,METER,1000), ...)

Out **a(qqu)** scaleList

SetMapViewScale = This method sets the map scale by specifying a ScaleID

method SetMapViewScale

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in **u** sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in **u** mapViewInstanceHandle

scaleID = scale identifier. Range[0:256]

in **q** scaleID

SetMapViewScaleByDelta = This method sets the map scale by specifying a delta value with respect to the currently set ScaleID

method SetMapViewScaleByDelta

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

scaleDelta = This parameter can have either positive or negative values. '0' means no change. Positive values indicate larger scales

in n scaleDelta

SetMapViewScaleByMetersPerPixel = This method sets the map scale by specifying the number of meters that a pixel represents

method SetMapViewScaleByMetersPerPixel

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

metersPerPixel = meters per pixel

in d metersPerPixel

GetMapViewScale = This method returns the currently used map scale

method GetMapViewScale

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

scaleID = scale identifier. Range[0:256]

out y scaleID

isMinMax = enum(INVALID,MIN,MAX,MID, ...)

MIN = scale ID minimal value

MID = scale ID intermediate value

MAX = scale ID maximal value

out q isMinMax

SetMapViewBoundingBox = This method sets the map bounding box

method SetMapViewBoundingBox

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

boundingBox = struct(top-left-corner,bottom-right-corner)

top-left-corner = struct(lat,lon)

bottom-right-corner = struct(lat,lon)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

in ((dd)(dd)) boundingBox

GetMapViewBoundingBox = This method returns the bounding box of a given map instance

method GetMapViewBoundingBox

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

boundingBox = struct(top-left-corner,bottom-right-corner)

top-left-corner = struct(lat,lon)

bottom-right-corner = struct(lat,lon)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

out ((dd)(dd)) boundingBox

MapViewBoundingBoxChanged = This signal is emitted when the bounding box of a map instance changes

signal MapViewBoundingBoxChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

boundingBox = struct(top-left-corner,bottom-right-corner)

top-left-corner = struct(lat,lon)

bottom-right-corner = struct(lat,lon)

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

out ((dd)(dd)) boundingBox

SetMapViewSaveArea = This methods defines the area that the HMI guarantees not to cover with other windows or user interface elements

method SetMapViewSaveArea

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

boundingBox = struct(rect-left,rect-right,rect-top,rect-bottom)

rect-left = covered area on the left. Range [0:1]

rect-right = covered area on the right. Range [0:1]

rect-top = covered area on top. Range [0:1]

rect-bottom = covered area at the bottom. Range [0:1]

Note: 0.0 means there is no covered area (offset) from that side

When all four parameters are 0 then the save area is equal to the viewport area (being the default)

in (dddd) saveArea

SetMapViewSaveArea = This methods defines the area that the HMI guarantees not to cover with other windows or user interface elements

method GetMapViewSaveArea

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

boundingBox = struct(rect-left,rect-right,rect-top,rect-bottom)

rect-left = covered area on the left. Range [0:1]

rect-right = covered area on the right. Range [0:1]

rect-top = covered area on top. Range [0:1]

rect-bottom = covered area at the bottom. Range [0:1]

Note: 0.0 means there is no covered area (offset) from that side

When all four parameters are 0 then the save area is equal to the viewport area (being the default)

out (dddd) saveArea

MapViewSaveAreaChanged = This signal is emitted when the area that the HMI guarantees not to cover with other windows or user interface elements changes

signal MapViewSaveAreaChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

boundingBox = struct(rect-left,rect-right,rect-top,rect-bottom)

rect-left = covered area on the left. Range [0:1]

rect-right = covered area on the right. Range [0:1]

rect-top = covered area on top. Range [0:1]

rect-bottom = covered area at the bottom. Range [0:1]

Note: 0.0 means there is no covered area (offset) from that side

When all four parameters are 0 then the save area is equal to the viewport area (being the default)

out (dddd) saveArea

SetMapViewPan = This method pans a given map instance

method SetMapViewPan

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

panningAction = enum(PAN_START,PAN_TO,PAN_END)

PAN_START, start panning the map at the specified screen coordinate identified by the arguments 'x' and 'y'

PAN_TO, pan the map to the specified screen coordinate identified by the arguments 'x' and 'y'; it has no effect before PAN_START or after PAN_END

PAN_END, pan the map to the specified screen coordinate identified by the arguments 'x' and 'y' and end panning; it has no effect before PAN_START

in q panningAction

pixelCoordinates = array[struct(x,y)]

x = x-coordinate (x=0 indicates the first left pixel of the map view)

y = y-coordinate (y=0 indicates the first top pixel of the map view)

in a(qq) pixelCoordinates

GetMapViewPan

This method is meant for debugging purposes

method GetMapViewPan

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

valueToReturn = enum(PAN_START,PAN_TO,PAN_END)

in q valueToReturn

pixelCoordinates = array[struct(x,y)]

x = x-coordinate (x=0 indicates the first left pixel of the map view)

y = y-coordinate (y=0 indicates the first top pixel of the map view)

in a(qq) pixelCoordinates

SetMapViewRotation = This method rotates the map

method SetMapViewRotation

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

rotationAngle = rotation angle in degrees measured from the North axis clockwise. Range[0:360]

in i rotationAngle

rotationAnglePerSecond = partial rotation for each second

The value can be set implement a smooth rotation

If rotationAnglePerSecond = rotationAngle it means that the rotation must be instantaneous

in i rotationAnglePerSecond

GetMapViewRotation = This method is particularly interesting for debugging purposes

method GetMapViewRotation

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

rotationAngle = rotation angle in degrees measured from the North axis clockwise. Range[0:360]

out i rotationAngle

rotationAnglePerFrame = partial rotation for each map frame in degrees

out i rotationAnglePerFrame

SetMapViewVisibilityMode = This method sets the current visibility mode

method SetMapViewVisibilityMode

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

visibilityMode = enum(INVALID,VISIBLE,INVISIBLE,FROZEN, ...)

in q visibilityMode

GetMapViewVisibilityMode = This method returns the current visibility mode

method GetMapViewVisibilityMode

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

visibilityMode = enum(INVALID,VISIBLE,INVISIBLE,FROZEN, ...)
out q visibilityMode

GetSupportedMapViewVisibilityModes = This method retrieves the supported mapview visibility modes

method GetSupportedMapViewVisibilityModes

visibilityModeList = array[visibilityMode]
visibilityMode = enum(INVALID,VISIBLE,INVISIBLE,FROZEN, ...)
out aq visibilityModeList

MapViewVisibilityChanged = This signal is emitted when the MapView visibility changes

signal MapViewVisibilityChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value
out u mapViewInstanceHandle

visibilityMode = enum(INVALID,VISIBLE,INVISIBLE,FROZEN, ...)
out q visibilityMode

SetMapViewPerformanceLevel = This method sets the performance level of a given map instance

Note: it can be used to set the MapView in application specific performance mode (e.g. low CPU-usage or low memory-usage)

method SetMapViewPerformanceLevel

sessionHandle = Session handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

performanceLevel = enum(INVALID,LEVEL1,LEVEL2,LEVEL3,LEVEL4,LEVEL5, ...)
Note: performance levels are application specific
in q performanceLevel

GetMapViewPerformanceLevel = This method returns the performance level of a given map instance

method GetMapViewPerformanceLevel

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7fffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

performanceLevel = enum(INVALID,LEVEL1,LEVEL2,LEVEL3,LEVEL4,LEVEL5, ...)
out q performanceLevel

MapViewPerformanceLevelChanged = This signal is emitted when the performance level of a map instance changes
signal MapViewPerformanceLevelChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
out u mapViewInstanceHandle

performanceLevel = enum(INVALID, LEVEL1, LEVEL2, LEVEL3, LEVEL4, LEVEL5, ...)
out q performanceLevel

GetSupportedMapViewPerformanceLevels = This method retrieves the supported performance levels
method GetSupportedMapViewPerformanceLevels

performanceLevelList = array[performanceLevel]
performanceLevel = enum(INVALID, LEVEL1, LEVEL2, LEVEL3, LEVEL4, LEVEL5, ...)
out aq performanceLevelList

DisplayRoute = This method visualizes one of the calculated routes
method DisplayRoute

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

routeHandle = Route handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u routeHandle

highlighted = flag. TRUE means highlighted, FALSE means not highlighted
Note: the highlighted route must be visualized on top of the other routes
in b highlighted

HideRoute = This method hides one of the visible routes
method HideRoute

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

routeHandle = Route handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
Note: the routeHandle has to be one of the visible routes
in u routeHandle

GetDisplayedRoutes = This method returns a list of displayed routes
method GetDisplayedRoutes

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

displayedRoutes = array[struct(routeHandle,highlighted)]
routeHandle = Route handle of a displayed route. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
highlighted = flag. TRUE means highlighted,FALSE means not highlighted
out a(ub) displayedRoutes

DisplayedRoutes = This signal is emitted when the list of displayed routes change
signal DisplayedRoutes

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
out u mapViewInstanceHandle

displayedRoutes = array[struct(routeHandle,highlighted)]
routeHandle = Route handle of a displayed route. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
highlighted = flag. TRUE means highlighted,FALSE means not highlighted
out a(ub) displayedRoutes

GetPoiCategoriesVisible = Get the set of POI categories displayed on the map.
method GetPoiCategoriesVisible

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

poiCategoryIds = array[poiCategoryId]
poiCategoryId = a POI category as defined in the 'GENIVI POIService API'.
out aq poiCategoryIds

SetPoiCategoriesVisible = Add POI categories to the set of POI categories displayed on the map. Any specified category that until now was displayed with scale limits is now displayed without limits.
method SetPoiCategoriesVisible

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

poiCategoryIds = array[poiCategoryId]
poiCategoryId = a POI category as defined in the 'GENIVI POIService API'.
in aq poiCategoryIds

SetPoiCategoriesVisible = Add POI categories to the set of POI categories displayed on the map, where the POI's are only displayed in a specific range of scales. Any specified category that until now was displayed without scale limits is now displayed with limits.
method SetPoiCategoriesVisibleWithinLimits

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

poiCategoryIds = array[poiCategoryId]

poiCategoryId = a POI category as defined in the 'GENIVI POIService API'.

in aq poiCategoryIds

minScaleID = minimum scale on which the POI categories are displayed

in y minScaleID

maxScaleID = maximum scale on which the POI categories are displayed

in y maxScaleID

SetPoiCategoriesNotVisible = Remove POI categories from the set of POI categories displayed on the map.

method SetPoiCategoriesNotVisible

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

poiCategoryIds = array[poiCategoryId]

poiCategoryId = a POI category as defined in the 'GENIVI POIService API'.

in aq poiCategoryIds

PoiCategoriesVisibilityChanged = This signal is emitted when the visibility of POI categories on a map instance changes

signal PoiCategoriesVisibilityChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

poiCategoryIds = array[poiCategoryId]

poiCategoryId = a POI category as defined in the 'GENIVI POIService API'.

out aq poiCategoryIds

visible = flag. TRUE means that the POI categories are visible

out b visible

minScaleID = minimum scale on which the POI categories are displayed

out y minScaleID

maxScaleID = maximum scale on which the POI categories are displayed

out y maxScaleID

SetTrafficIncidentsVisibility = Set the visibility of Traffic Incidents on the map.

method SetTrafficIncidentsVisibility

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

visible = If true, Traffic Incidents are shown on the map, else they are not shown.

in b visible

SetMapViewTheme = This method configures the theme of a given map view instance

method SetMapViewTheme

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

mapViewTheme = enum(INVALID,THEME_1,THEME_2,THEME_3, ...)

Note: Themes are implementation specific. Example: THEME_1 = day color, THEME_2 = night color

in q mapViewTheme

GetMapViewTheme = This method returns the current theme of a given map view instance

method GetMapViewTheme

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

mapViewTheme = enum(INVALID,THEME_1,THEME_2,THEME_3, ...)

Note: Themes are implementation specific. Example: THEME_1 = day color, THEME_2 = night color

out q mapViewTheme

MapViewThemeChanged = This signal is emitted when the theme of a map view instance changes

signal MapViewThemeChanged

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u mapViewInstanceHandle

mapViewTheme = enum(INVALID,THEME_1,THEME_2,THEME_3, ...)

Note: Themes are implementation specific. Example: THEME_1 = day color, THEME_2 = night color

out q mapViewTheme

GetSupportedMapViewThemes = This method retrieves the supported mapview themes

method GetSupportedMapViewThemes

mapViewThemeList = array[mapViewTheme]

mapViewTheme = enum(INVALID,THEME_1,THEME_2,THEME_3, ...)

Note: Themes are implementation specific. Example: THEME_1 = day color, THEME_2 = night color

out aq mapViewThemeList

ConvertPixelCoordsToGeoCoords = This method converts pixel coordinates to geographical coordinates

method ConvertPixelCoordsToGeoCoords

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

pixelCoordinates = array[struct(x,y)]

x = x-coordinate (x=0 indicates the first left pixel of the map view)

y = y-coordinate (y=0 indicates the first top pixel of the map view)

in a(qq) pixelCoordinates

geoCoordinates = array[struct(lat,lon)]

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

out a(dd) geoCoordinates

ConvertGeoCoordsToPixelCoords = This method converts geographical coordinates into pixel coordinates

method ConvertGeoCoordsToPixelCoords

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

geoCoordinates = array[struct(lat,lon)]

lat = latitude in format %3.6f. Range[-90:+90]

lon = longitude in format %3.6f. Range[-180:+180]

in a(dd) geoCoordinates

pixelCoordinates = array[struct(x,y)]

x = x-coordinate (x=0 indicates the first left pixel of the map view)

y = y-coordinate (y=0 indicates the first top pixel of the map view)

out a(qq) pixelCoordinates

DisplayCustomElements = This method visualizes a set of custom elements on the map

method DisplayCustomElements

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u mapViewInstanceHandle

customElements = array[struct(name,iconUri,coordinate,anchorPoint)]

name = name of the custom element

iconUri = uri to the icon of the custome element

coordinate = struct(lat,lon)

lat = latitude in format %3.6f. Range[-90:+90]. Example: 48.053250

lon = longitude in format %3.6f. Range[-180:+180]. Example: 8.321000

anchorPoint=struct(anchorX,anchorY)

anchorPoint defines which point on the icon is used as the reference for associating the icon to the map

coordinate

(0,0) is the center of the icon

(-1,-1) is the top left corner of the icon
(1,1) is the bottom right corner of the icon
anchorX = anchor x value
anchorY = anchor y value
in a(ss(dd)(nn)) customElements

customElementHandles = handles to the custom elements displayed on the map. The order of the handles is the same as the order of custom elements specified in the argument 'customElements'. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

Out au customElementHandles

HideCustomElements = This method hides a set of custom elements which were visualized by DisplayCustomElements
method HideCustomElements

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u sessionHandle

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

customElementHandles = Custom element handles. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in au customElementHandles

GetDisplayedCustomElements = This method retrieves the visualized custom elements on the map
method GetDisplayedCustomElements

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

customElements = array[customElement]
customElement = tuple[customElementHandle,struct(name,iconUri,coordinate,anchorPoint)]
customElementHandle = Custom element handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
name = name of the custom element
iconUri = uri to the icon of the custome element
coordinate = struct(lat,lon)
lat = latitude in format %3.6f. Range[-90:+90]. Example: 48.053250
lon = longitude in format %3.6f. Range[-180:+180]. Example: 8.321000
anchorPoint=struct(anchorX,anchorY)
anchorPoint defines which point on the icon is used as the reference for associating the icon to the map
coordinate
(0,0) is the center of the icon
(-1,-1) is the top left corner of the icon
(1,1) is the bottom right corner of the icon
anchorX = anchor x value
anchorY = anchor y value
Out a{u(ss(dd)(nn))} customElements

SelectElementsOnMap = This method selects elements on the map view which are at the position specified by user input
method SelectElementsOnMap

mapViewInstanceHandle = Map instance handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value
in u mapViewInstanceHandle

pixelCoordinates = struct(x,y)

x = x-coordinate of the position on the map view specified by user input (x=0 indicates the first left pixel of the map view)

y = y-coordinate of the position on the map view specified by user input (y=0 indicates the first top pixel of the map view)

in (qq) pixelCoordinate

selectableTypes = array[selectableType]

selectableType =

enum(INVALID,CUSTOM_ELEMENT,CURRENT_POSITION,WAYPOINT,POI,TRAFFIC_INCIDENT,ROUTE,GEOCOORDINATES)

Note: The order of priority by which the elements are selected is implementation dependent

in aq selectableTypes

maxNumberOfSelectedElements = maximum number of selected elements to return. If 0, all possible elements which can be selected will be returned

in q maxNumberOfSelectedElements

selectedElements = array[selectableType,struct(lat,lon),value]

selectableType =

enum(INVALID,CUSTOM_ELEMENT,CURRENT_POSITION,WAYPOINT,POI,TRAFFIC_INCIDENT,ROUTE,GEOCOORDINATES)

lat = latitude of the selected element in format %3.6f. Range[-90:+90]

lon = longitude of the selected element in format %3.6f. Range[-180:+180]

selectableType = CUSTOM_ELEMENT, value = value of type '(uss(nn))' that expresses the extra data for a custom element

Note: the extra data for a custom element is expressed as a

struct(customElementHandle,name,iconUri,struct(anchorX,anchorY))

selectableType = CURRENT_POSITION, value = null

selectableType = WAYPOINT, value = value of type '(uq)' that expresses the extra data for a waypoint

Note: the extra data for a waypoint is expressed as a struct(routeHandle,waypointIndex) where waypointIndex is the index of the waypoint on the route (the first waypoint is index 0)

selectableType = POI, value = value of type 'u' that expresses a POI handle

selectableType = TRAFFIC_INCIDENT, value = value of type 'i' that expresses a traffic incident identifier

selectableType = ROUTE, value = value of type 'u' that expresses a route handle

selectableType = GEOCOORDINATES, value = null

out a(q(dd)v) selectedElements

interface **org.genivi.mapviewer.Session**

version 3.0.0 (21-01-2014)

Session = This interface offers functions to create and delete sessions

GetVersion = This method returns the API version implemented by the server application

method GetVersion

version = struct(major,minor,micro,date)

major = when the major changes, then backward compatibility with previous releases is not granted

minor = when the minor changes, then backward compatibility with previous releases is granted, but something changed in the implementation of the API (e.g. new methods may have been added)

micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or documentation modifications)

date = release date (e.g. 21-06-2011)

out (qqqs) version

CreateSession = This method creates a new session

method CreateSession

client = name or identifier of the client application that requests a new session

The navigation core must internally associate this name to the returned session handle

This parameter can be used to identify the client application and determine if a given feature is enabled for it
in s client

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out u sessionHandle

This error is generated if no more session handles are available

error org.genivi.mapviewer.Session.Error.NoMoreSessionHandles

DeleteSession = This method deletes a session and its associated resources

method DeleteSession

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in u sessionHandle

This error is generated if an application tries to delete a session handle that is not available

error org.genivi.mapviewer.Session.Error.SessionNotAvailable

GetSessionStatus = This method returns whether a given session handle is available or not (for example because it was deleted)

method GetSessionStatus

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

in **u** sessionHandle

sessionStatus = enum(INVALID,AVAILABLE,NOT_AVAILABLE)

out **q** sessionStatus

GetAllSessions = This method returns a list of all available sessions

method GetAllSessions

sessionsList = array[struct(sessionHandle,client)]

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

client = name or identifier of the client application that requested the sessionHandle

out **a(us)** sessionsList

SessionDeleted = This signal is emitted when a session is deleted

signal SessionDeleted

sessionHandle = Session handle. Range[0x0:0x7ffffff]. 0x0 is reserved as an invalid handle value

out **u** sessionHandle

interface

org.genivi.mapviewer.Configuration

version 3.0.0 (21-01-2014)

Configuration = This interface offers functions to set and retrieve configuration parameters

GetVersion = This method returns the API version implemented by the server application

method GetVersion

version = struct(major,minor,micro,date)

major = when the major changes, then backward compatibility with previous releases is not granted

minor = when the minor changes, then backward compatibility with previous releases is granted, but something changed in the implementation of the API (e.g. new methods may have been added)

micro = when the micro changes, then backward compatibility with previous releases is granted (bug fixes or documentation modifications)

date = release date (e.g. 21-06-2011)

out (qqqs) version

SetUnitsOfMeasurement = This method sets the units of measurement

method SetUnitsOfMeasurement

unitsOfMeasurementList = array[unitsOfMeasurement]

unitsOfMeasurement = tuple (key,value)

key = enum(INVALID,LENGTH, ...)

key = LENGTH, value = value of type 'q', that represents an enum(INVALID,METER,MILE, ...)

in a{qv} unitsOfMeasurementList

GetUnitsOfMeasurement = This method retrieves the units of measurement

method GetUnitsOfMeasurement

unitsOfMeasurementList = array[unitsOfMeasurement]

unitsOfMeasurement = tuple (key,value)

key = enum(INVALID,LENGTH, ...)

key = LENGTH, value = value of type 'q', that represents an enum(INVALID,METER,MILE, ...)

out a{qv} unitsOfMeasurementList

GetSupportedUnitsOfMeasurement = This method retrieves the supported units of measurement

method GetSupportedUnitsOfMeasurement

unitsOfMeasurementList = array[unitsOfMeasurement]

unitsOfMeasurement = dictionary[key,value]

dictionary = array of tuples (key,value)

key = enum(INVALID,LENGTH, ...)

key = LENGTH, value = value of type 'aq'; 'q' is an enum(INVALID,METER,MILE, ...)

out a{qv} unitsOfMeasurementList

SetTimeFormat = This method sets the time format

method SetTimeFormat

timeFormat = enum(INVALID,12H,24H, ...)
in q timeFormat

GetTimeFormat = This method retrieves the time format

method GetTimeFormat

timeFormat = enum(INVALID,12H,24H, ...)
out q timeFormat

GetSupportedTimeFormats = This method retrieves the supported time formats

method GetSupportedTimeFormats

timeFormatList = array[timeFormat]
timeFormat = enum(INVALID,12H,24H, ...)
out aq timeFormatList

SetCoordinatesFormat = This method sets the coordinates format

method SetCoordinatesFormat

coordinatesFormat = enum(INVALID,DEGREES,MINUTES,SECONDS, ...)
DEGREES format = d.d°
MINUTES format = d°m.m'
SECONDS format = d°m's"
in q coordinatesFormat

GetCoordinatesFormat = This method retrieves the coordinates format

method GetCoordinatesFormat

coordinatesFormat = enum(INVALID,DEGREES,MINUTES,SECONDS, ...)
DEGREES format = d.d°
MINUTES format = d°m.m'
SECONDS format = d°m's"
out q coordinatesFormat

GetSupportedCoordinatesFormats = This method retrieves the supported coordinates formats

method GetSupportedCoordinatesFormats

coordinatesFormatList = array[coordinatesFormat]

coordinatesFormat = enum(INVALID,DEGREES,MINUTES,SECONDS, ...)
DEGREES format = d.d°
MINUTES format = d°m.m'
SECONDS format = d°m's"
out aq coordinatesFormatList

SetLocale = This method sets the current language and country

method SetLocale

language = ISO 639-3 language code (lower case)
in s language

country = ISO 3166-1 alpha 3 country code (upper case)
in s country

GetLocale = This method retrieves the current language and country

method GetLocale

language = ISO 639-3 language code (lower case)
out s language

country = ISO 3166-1 alpha 3 country code (upper case)
out s country

GetSupportedLocales = This method retrieves the supported languages and countries

method GetSupportedLocales

localeList = array[struct(language,country)]
language = ISO 639-3 language code (lower case)
country = ISO 3166-1 alpha 3 country code (upper case)
out a(ss) localeList

ConfigurationChanged = This signal is sent to the clients when one or more configuration settings changes

signal ConfigurationChanged

changedSettings = array[setting]
setting = enum(INVALID,UNITS_OF_MEASUREMENT,LOCALE,TIME_FORMAT,COORDINATES_FORMAT, ...)
out aq changedSettings

constants *MapView* version 3.0.0 (21-01-2014)

- *This document defines the constants used in the MapViewer APIs*

- *INVALID = 0x0000*

- *ALL = 0xffff*

- *AVAILABLE = 0x0001*

- *NOT_AVAILABLE = 0x0002*

- *TIME_FORMAT = 0x0003*

- *12H = 0x0004*

- *24H = 0x0005*

- *COORDINATES_FORMAT = 0x0006*

- *DEGREES = 0x0007*

- *MINUTES = 0x0008*

- *SECONDS = 0x0009*

- *MAIN_MAP = 0x0010*

- *SPLIT_SCREEN = 0x0011*

- *2D = 0x0020*

- *3D = 0x0021*

- *LOCALE = 0x0025*

- *UNITS_OF_MEASUREMENT = 0x0030*

- *LENGTH = 0x0031*

- *METER = 0x0032*

- *MILE = 0x0033*

- *KM = 0x0034*

- *YARD = 0x0035*

- *FOOT = 0x0036*

- *MIN = 0x0040*

- MAX = 0x0041

- MID = 0x0042

- VISIBLE = 0x0043

- INVISIBLE = 0x0044

- FROZEN = 0x0045

- LEVEL1 = 0x0050

- LEVEL2 = 0x0051

- LEVEL3 = 0x0052

- LEVEL4 = 0x0053

- LEVEL5 = 0x0054

- THEME_1 = 0x0060

- THEME_2 = 0x0061

- THEME_3 = 0x0062

- CONSTANT_ANGLE = 0x0070

- TRACK_UP = 0x0071

- TOWARDS_TARGET = 0x0072

- PAN_START = 0x0100

- PAN_TO = 0x0101

- PAN_END = 0x0102

- BUILDINGS = 0x0080

- TERRAIN = 0x0081

- AUTOMATIC = 0x0110

- MANUAL = 0x0111

- HYBRID = 0x0112

- CUSTOM_ELEMENT = 0x0120

- CURRENT_POSITION = 0x0121

- WAYPOINT = 0x0122

- POI = 0x0123

- *TRAFFIC_INCIDENT* = 0x0124
-

- *ROUTE* = 0x0125
-

- *GEOCOORDINATES* = 0x0126