



Geolocation and navigation

Contents

2	Terminology and concepts	5
3	Coordinates	5
4	Geolocation	5
5	Forward geocoding	5
6	Reverse geocoding	5
7	Geofencing	5
8	Route planning	6
9	Route replanning	6
10	Route cancellation	6
11	Point of interest	6
12	Route list	6
13	Horizon	6
14	Route guidance	6
15	Text-to-speech (TTS)	7
16	Location-based services (LBS)	7
17	Navigation application	7
18	Routing request	7
19	Use cases	8
20	Relocating the vehicle	8
21	Automotive backend	8
22	SDK backend	8
23	Viewing an address on the map	8
24	Adding custom widgets on the map	9
25	Finding the address of a location on the map	9
26	Type-ahead search and completion for addresses	9
27	Navigating to a location	9
28	Navigating a tour	10
29	Navigating to an entire city	10
30	Changing destination during navigation	10
31	Tasks nearby	10
32	Turning on house lights	10
33	Temporary loss of GPS signal	11
34	Navigation- and sensor-driven refinement of geolocation	11
35	Application-driven refinement of geocoding	11
36	Malware application bundle	11
37	Traffic rule application	11
38	Installing a navigation application	12
39	No navigation application	12
40	Web-based backend	12
41	Navigation route guidance information	12
42	2.5D or 3D map widget	13
43	Separate route guidance UI	13
44	User control over applications	13
45	Non-use-cases	13

46	POI data	14
47	Beacons	14
48	Loading map tiles from a backend	14
49	SDK APIs for third-party navigation applications	14
50	Requirements	15
51	Geolocation API	15
52	Geolocation service supports signals	15
53	Geolocation implements caching	15
54	Geolocation supports backends	15
55	Navigation routing API	16
56	Navigation routing supports different navigation applications	16
57	Navigation route list API	16
58	Navigation route guidance API	17
59	Type-ahead search and address completion supports backends	17
60	Geocoding supports backends	17
61	SDK has default implementations for all backends	17
62	SDK APIs do not vary with backend	17
63	Third-party navigation applications can be used as backends	18
64	Backends operate asynchronously	18
65	2D map rendering API	18
66	2.5D or 3D map rendering API	19
67	Reverse geocoding API	19
68	Forward geocoding API	19
69	Type-ahead search and address completion API	19
70	Geofencing API	19
71	Geofencing service can wake up applications	20
72	Geofencing API signals on national borders	20
73	Geocoding API must be locale-aware	20
74	Geolocation provides error bounds	20
75	Geolocation implements dead-reckoning	20
76	Geolocation uses navigation and sensor data if available	20
77	General points of interest streams are queryable	21
78	Location information requires permissions to access	21
79	Rate limiting of general point of interest streams	22
80	Application-provided data requires permissions to create	22
81	Existing geo systems	22
82	W3C Geolocation API	22
83	Android platform location API	22
84	Google Location Services API for Android	22
85	iOS Location and Maps API	23
86	GNOME APIs	24
87	Navigation routing systems	25
88	NavServer	25
89	GENIVI	25
90	Google web APIs	26
91	Approach	28

92	Backends	28
93	Navigation application	29
94	2D map display	29
95	2.5D or 3D map display	30
96	Geolocation	31
97	Navigation routing	31
98	Navigation route list API	32
99	Navigation route guidance progress API	33
100	Navigation route guidance turn-by-turn API	34
101	Horizon API	35
102	Forward and reverse geocoding	37
103	Address completion	38
104	Geofencing	39
105	Location security	40
106	Systemic security	41
107	Testability	42
108	Requirements	43
109	Suggested roadmap	44
110	Open questions	45
111	Summary of recommendations	45
112	Appendix: Recommendations for third-party navigation applications .	46
113	Appendix: place URI scheme	48
114	Examples	49
115	Appendix: nav URI scheme	50
116	Examples	52

117 This documents existing solutions for geo-related features (sometimes known as
118 location-based services, LBS) which could be integrated into Apertis for pro-
119 viding geolocation, geofencing, geocoding and navigation routing support for
120 application bundles.

121 As of version 0.3.0, the recommended solutions for most of the geo-requirements
122 for Apertis are already implemented as open source libraries which can be in-
123 tegrated into Apertis. Some of them require upstream work to add smaller
124 missing features.

125 Larger pieces of work need to be done to add address completion and geofencing
126 features to existing open source projects.

127 The major considerations with all of these features are:

- 128 • Whether the feature needs to work offline or can require the vehicle to
129 have an internet connection.
- 130 • Privacy sensitivity of the data used or made available by the feature —
131 for example, access to geolocation or navigation routing data is privacy
132 sensitive as it gives the user's location.

- All features must support pluggable backends to allow proprietary solutions to be used if provided by the automotive domain.

The scope of this design is restricted to providing services to applications which need to handle locations or location-based data. This design does not aim to provide APIs suitable for implementing a full vehicle navigation routing system —this is assumed to be provided by the [automotive domain](#)¹, and may even provide some of the implementations of geo-related features used by other applications. This means that the navigation routing API suggested by this design is limited to allowing applications to interact with an external navigation routing system, rather than implement or embed one themselves. [Appendix: Recommendations for third-party navigation applications](#) when implementing a navigation application are given.

Terminology and concepts

Coordinates

Throughout this document, *coordinates* (or *a coordinate pair*) are taken to mean a latitude and longitude describing a single point in some well-defined coordinate system (typically WGS84).

Geolocation

Geolocation is the resolution of the vehicle's current location to a coordinate pair. It might not be possible to geolocate at any given time, due to unavailability of sensor input such as a GPS lock.

Forward geocoding

Forward geocoding is the parsing of an address or textual description of a location, and returning zero or more coordinates which match it.

Reverse geocoding

Reverse geocoding is the lookup of the zero or one addresses which correspond to a coordinate pair.

Geofencing

Geofencing is a system for notifying application bundles when the vehicle enters a pre-defined 'fenced' area. For example, this can be used for notifying about jobs to do in a particular area the vehicle is passing through, or for detecting the end of a navigation route.

¹<https://www.apertis.org/glossary/#automotive-domain>

165 **Route planning**

166 *Route planning* is where a start, destination and zero or more via-points are
167 specified by the user, and the system plans a road navigation route between
168 them, potentially optimising for traffic conditions or route length.

169 **Route replanning**

170 *Route replanning* is where a route is recalculated to follow different roads, with-
171 out changing the start, destination or any via-points along the way. This could
172 happen if the driver took a wrong turn, or if traffic conditions change, for ex-
173 ample.

174 **Route cancellation**

175 *Route cancellation* is when a route in progress has its destination or via-points
176 changed or removed. This does not necessarily happen when the vehicle is
177 stopped or the ignition turned off, as route navigation could continue after an
178 over-night stop, for example.

179 **Point of interest**

180 A *point of interest* (*POI*) is a specific location which someone (a driver or
181 passenger) might find interesting, such as a hotel, restaurant, fuel station or
182 tourist attraction.

183 **Route list**

184 A *route list* is the geometry of a navigation route, including the start point, all
185 destinations and all vertices and edges which unambiguously describe the set of
186 roads the route should use. Note that it is different from *route guidance*, which
187 is the set of instructions to follow for the route.

188 **Horizon**

189 The *horizon* is the collection of all interesting objects which are ahead of the
190 driver on their route ('on the horizon'). Practically, this is a combination of
191 upcoming *points of interest*, and the remaining *route list*.

192 **Route guidance**

193 *Route guidance* is the set of turn-by-turn instructions for following a navigation
194 route, such as 'take the next left' or 'continue along the A14 for 57km'. It is not
195 the *route list*, which is the geometry of the route, but it may be possible to
196 derive it from the route list.

197 **Text-to-speech (TTS)**

198 *Text-to-speech (TTS)* is a user interface technology for outputting a user interface
199 as computer generated speech.

200 **Location-based services (LBS)**

201 *Location-based services (LBS)* is another name for the collection of geo-related
202 features provided to applications: geolocation, geofencing, geocoding and navi-
203 gation routing.

204 **Navigation application**

205 A *navigation application* is assumed (for the purposes of this document) to be
206 an application bundle which contains

- 207 • a *navigation UI* for choosing a destination and planning a route;
- 208 • a *guidance UI* for providing guidance for that route while driving, poten-
209 tially also showing points of interest along the way;
- 210 • a *navigation service* which provides the (non-SDK) APIs used by the two
211 UIs, and can act as a backend for the various SDK geo-APIs; and
- 212 • a *routing engine* which calculates the recommendation for a route to a
213 particular destination with particular parameters, and might be imple-
214 mented in either the IVI or automotive domain. It is conceptually part of
215 the *navigation service*.

216 These two UIs may be part of the same application, or may be separate appli-
217 cations (for example with the guidance UI as part of the system chrome). The
218 navigation service may be a separate process, or may be combined with one or
219 both of the UI processes.

220 The navigation service might communicate with systems in the automotive do-
221 main to provide its functionality.

222 Essentially, the ‘navigation application’ is a black box which provides UIs and
223 (non-SDK) services related to navigation. For this reason, the rest of the docu-
224 ment does not distinguish between ‘navigation UI’, ‘guidance UI’ and ‘navigation
225 service’.

226 **Routing request**

227 A *routing request* is a destination and set of optional parameters (waypoints,
228 preferred options, etc.) from which the [routing engine][Navigation application]
229 can calculate a specific route.

230 Use cases

231 A variety of use cases for application bundle usage of geo-features are given
232 below. Particularly important discussion points are highlighted at the bottom
233 of each use case.

234 In all of these use cases, unless otherwise specified, the functionality must work
235 regardless of whether the vehicle has an internet connection. i.e. They must
236 work offline. For most APIs, this is the responsibility of the automotive backend;
237 **SDK backend** can assume an internet connection is always available.

238 Relocating the vehicle

239 If the driver is driving in an unfamiliar area and thinks they know where they
240 are going, then realises they are lost, they must be able to turn on geolocation
241 and it should pinpoint the vehicle's location on a map if it's possible to attain a
242 GPS lock or get the vehicle's location through other means.

243 Automotive backend

244 A derivative of Apertis may wish to integrate its own geo-backend, running in
245 the automotive domain, and providing all geo-functionality through proprietary
246 interfaces. The system integrators may wish to use some functionality from this
247 backend and other functionality from a different backend. They may wish to
248 ignore some functionality from this backend (for example, if its implementation
249 is too slow or is missing) and not expose that functionality to the app bundle
250 APIs at all (if no other implementation is available).

251 **Custom automotive backend functionality** The proprietary geo-backend
252 in a derivative of Apertis may expose functionality beyond what is described
253 in this design, which the system integrator might want to use in their own
254 application bundles.

255 If this functionality is found to be common between multiple variants, the official
256 Apertis SDK APIs may be extended in future to cover it.

257 SDK backend

258 Developers using the Apertis SDK to develop applications must have access
259 to geo-functionality during development. All geo-functionality must be imple-
260 mented in the SDK.

261 The SDK can be assumed to have internet access, so these implementations may
262 rely on the internet for their functionality.

263 Viewing an address on the map

264 The user receives an e-mail containing a postal address, and they want to view
265 that address on a map. The e-mail client recognises the format of the address,

266 and adds a map widget to show the location, which it needs to convert to latitude
267 and longitude in order to pinpoint.

268 In order to see the surrounding area, the map should be a 2D top-down atlas-
269 style view.

270 In order for the user to identify the map area in relation to their current journey,
271 if they have a route set in the navigation application, it should be displayed as
272 a layer in the map, including the destination and any waypoints. The vehicle's
273 current position should be shown as another layer.

274 **Adding custom widgets on the map**

275 A restaurant application wants to display a map of all the restaurants in their
276 chain, and wants to customise the appearance of the marker for each restaurant,
277 including adding an introductory animation for the markers. They want to ani-
278 mate between the map and a widget showing further details for each restaurant,
279 by flipping the map over to reveal the details widget.

280 **Finding the address of a location on the map**

281 The user is browsing a tourist map application and has found an interesting-
282 looking place to visit with their friends, but they do not know its address. They
283 want to call their friends and tell them where to meet up, and need to find out
284 the address of the place they found on the map.

285 **Type-ahead search and completion for addresses**

286 A calendar application allows the user to create events, and each event has an
287 address/location field. In order to ease entering of locations, the application
288 wishes to provide completion options to the user as they type, if any potential
289 completion addresses are known to the system's map provider. This allows the
290 user to speed up entry of addresses, and reduce the frequency of typos on longer
291 addresses while typing when the vehicle is moving.

292 If this functionality cannot be implemented, the system should provide a normal
293 text entry box to the user.

294 **Navigating to a location**

295 A calendar application reminds the user of an event they are supposed to attend.
296 The event has an address set on it, and the application allows the user to select
297 that address and set it as the destination in their navigation application, to
298 start a new navigation route.

299 Once the navigation application is started, it applies the user's normal prefer-
300 ences for navigation, and does not refer back to the calendar application unless
301 the user explicitly switches applications.

302 **Navigating a tour**

303 A city tour guide application comes with a set of pre-planned driving tour routes
304 around various cities. The driver chooses one, and it opens in the navigation
305 application with the vehicle's current position, a series of waypoints to set the
306 route, and a destination at the end of the tour.

307 At some of the waypoints, there is no specific attraction to see —merely a general
308 area of the city which the tour should go through, but not necessarily using
309 specific roads or visiting specific points. These waypoints are more like ‘way-
310 areas’.

311 **Navigating to an entire city**

312 The driver wants to navigate to a general area of the country, and refine their
313 destination later or on-route. They want to set their destination as an entire
314 city (for example, Paris; rather than 1 Rue de Verneuil, Paris) and have this
315 information exposed to applications.

316 **Changing destination during navigation**

317 Part-way through navigating to one calendar appointment, the user gets a pag-
318 ing message from their workplace requiring them to divert to work immediately.
319 The user is in an unfamiliar place, so needs to change the destination on their
320 navigation route to take them to work —the paging application knows where
321 they are being paged to, and has a button to set that as the new navigation
322 destination. Clicking the button updates the navigation application to set the
323 new destination and start routing there.

324 **Navigating via waypoints** The user has to stop off at their house on their
325 way to work to answer the paging message. The paging application knows
326 this, and includes the user's home address as a navigation waypoint on the way
327 to their destination. This is reflected in the route chosen by the navigation
328 application.

329 **Tasks nearby**

330 A to-do list application may allow the user to associate a location with a to-do
331 list item, and should display a notification if the vehicle is driven near that
332 location, reminding the driver that they should pop by and do the task. Once
333 the task is completed or removed, the geo-fenced notification should be removed.

334 **Turning on house lights**

335 A ‘smart home’ application may be able to control the user's house lights over
336 the internet. If the vehicle is heading towards the user's house, the app should
337 be able to detect this and set turn the lights on over the internet to greet the
338 user when they get home.

339 **Temporary loss of GPS signal**

340 When going through a tunnel, for example, the vehicle may lose sight of GPS
341 satellites and no longer have a GPS fix. The system must continue to provide
342 an estimated vehicle location to apps, with suitably increasing error bounds, if
343 that is possible without reference to mapping data.

344 **Navigation- and sensor-driven refinement of geolocation**

345 The location reported by the geolocation APIs may be refined by input from
346 the navigation system or sensor system, such as snapping the location to the
347 nearest road, or supplementing it with dead-reckoning data based on the vehicle's
348 velocity history.

349 **Application-driven refinement of geocoding**

350 If the user installs an application bundle from a new restaurant chain ('Ham-
351 burger Co', who are new enough that their restaurants are not in commercial
352 mapping datasets yet), and wants to search for such a restaurant in a particular
353 place (London), they may enter 'Hamburger Co, London'. The application bun-
354 dle should expose its restaurant locations as a general [point of interest stream](#)²,
355 and the geocoding system should query that in addition to its other sources.

356 The user might find the results from a particular application consistently ir-
357 relevant or uninteresting, so might want to disable querying that particular
358 application —but still keep the application installed to use its other functional-
359 ity.

360 **Excessive results from application-driven refinement of geocoding**

361 A badly written application bundle which exposes a general point of interest
362 stream might return an excessive number of results for a query —either results
363 which are not relevant to the current geographic area, or too many results to
364 reasonably display on the current map.

365 **Malware application bundle**

366 A malicious developer might produce a malware application bundle which, when
367 installed, tracks the user's vehicle to work out opportune times to steal it. This
368 should not be possible.

369 **Traffic rule application**

370 The user is travelling across several countries in Europe, and finds it difficult
371 to remember all the road signs, national speed limits and traffic rules in use in
372 the countries. They have installed an application which reminds them of these
373 whenever they cross a national border.

²https://www.apertis.org/concepts/archive/application/points_of_interest/#General_POI_providers

374 **Installing a navigation application**

375 The user wishes to try out a third-party navigation application from the Apertis
376 store, which is different to the system-integrator-provided navigation application
377 which came with their vehicle. They install the application, and want it to be
378 used as the default handler for navigation requests from now on.

379 **Third-party navigation-driven refinement of geolocation** The third-
380 party application has some advanced dead-reckoning technology for estimating
381 the vehicle's position, which was what motivated the user to install it. The user
382 wants this refinement to feed into the geolocation information available to all
383 the applications which are installed.

384 **Third-party navigation application backend** If the user installs a full-
385 featured third-party navigation application, they may want to use it to provide
386 all geo-functionality in the system.

387 **No navigation application**

388 A driver prefers to use paper-based maps to navigate, and has purchased a non-
389 premium vehicle which comes without a built-in navigation application bundle,
390 and has not purchased any navigation bundles subsequently (i.e. the system
391 has no navigation application bundle installed).

392 The rest of the system must still work, and any APIs which handle route lists
393 should return an error code—but any APIs which handle the horizon should
394 still include all other useful horizon data.

395 **Web-based backend**

396 An OEM may wish to use (for example) Google's web APIs for geo-services in
397 their implementation of the system, rather than using services provided by a
398 commercial navigation application. This introduces latency into a lot of the
399 geo-service requests.

400 **Navigation route guidance information**

401 A restaurant application is running on one screen while the driver follows a route
402 in their navigation application on another screen. The passenger is using the
403 restaurant application to find and book a place to eat later on in the journey,
404 and wants to see a map of all the restaurants nearby to the vehicle's planned
405 route, plus the vehicle's current position, route estimates (such as time to the
406 destination and time elapsed), and the vehicle's current position so they can
407 work out the best restaurant to choose. (This is often known as route guidance
408 or driver assistance information.)

409 While the passenger is choosing a restaurant, the driver decides to change their
410 destination, or chooses an alternative route to avoid bad traffic; the passenger
411 wants the restaurant application to update to show the new route.

412 **2.5D or 3D map widget**

413 A weather application would like to give a perspective view over a large area of
414 the country which the vehicle's route will take it through, showing the predicted
415 weather in that area for the next few hours. It would like to give more emphasis
416 to the weather nearby rather than further away, hence the need for perspective
417 (i.e. a 2.5D or 3D view).

418 **Separate route guidance UI**

419 An OEM wishes to split their navigation application in two parts: the navigation
420 application core, which is used to find destinations and waypoints and to plan
421 a route (including implementation of calculating the route, tracking progress
422 through the journey, and recalculating in case of bad traffic, for example); and
423 a guidance UI, which is always visible, and is potentially rendered as part of the
424 system UI. The guidance UI needs to display the route, plus points of interest
425 provided by other applications, such as restaurants nearby. It also needs to
426 display status information about the vehicle, such as the amount of fuel left,
427 the elapsed journey time, and route guidance.

428 Explicitly, the OEM does *not* want the navigation application core to display
429 points of interest while the user is planning their journey, as that would be
430 distracting.

431 **User control over applications**

432 The user has installed a variety of applications which expose data to the geo-
433 services on the system, including points of interest and waypoint recommenda-
434 tions for routes. After a while, the user starts to find the behaviour of a fuel
435 station application annoying, and while they want to continue to use it to find
436 fuel stations, they do not want it to be able to add waypoints into their routes
437 for fuel station stops.

438 **Non-use-cases**

439 The following use cases are not in scope to be handled by this design —but they
440 may be in scope to be handled by other components of Apertis. Further details
441 are given in each subsection below.

442 **POI data**

443 Use cases around handling of points of interest is covered by the [Points of interest](#)
444 [design](#)³, which is orthogonal to the geo-APIs described here. This includes
445 searching for points of interest nearby, displaying points of interest while driving
446 past them, adding points of interest into a navigation route, and looking up
447 information about points of interest. It includes requests from the navigation
448 application or guidance UI to the points of interest service, and the permissions
449 system for the user to decide which points of interest should be allowed to appear
450 in the navigation application (or in other applications).

451 **Beacons**

452 The iOS Location and Maps API supports advertising a device's [location](#)⁴ using
453 a low-power beacon, such as Bluetooth. This is not a design goal for Apertis
454 at all, as advertising the location of a fast vehicle needs a different physical
455 layer approach than Beacons, which are designed for low-speed devices carried
456 by people.

457 **Loading map tiles from a backend**

458 There is no use case for implementing 2D map rendering via backends and (for
459 example) loading map tiles *from a backend in the automotive domain*. 2D map
460 rendering can be done entirely in the IVI domain using a single libchamplain
461 tile source. At this time, the automotive domain will not carry 2D map tile
462 data.

463 This may change in future iterations of this document to, for example, allow
464 loading pre-rendered map tiles or satellite imagery from the automotive domain.

465 **SDK APIs for third-party navigation applications**

466 Implementing a navigation application is complex, and there are many ap-
467 proaches to it in terms of the algorithms used. In order to avoid implement-
468 ing a lot of this complexity, and maintaining it as a stable API, the Apertis
469 platform explicitly does not want to provide geo-APIs which are only useful for
470 implementing third-party navigation applications.

471 Third parties may write navigation applications, but the majority of their im-
472 plementation should be internal; Apertis will not provide SDK APIs for routing,
473 for example.

³https://www.apertis.org/concepts/archive/application/points_of_interest/

⁴https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html#//apple_ref/doc/uid/TP40009497-CH9-SW1

474 Requirements

475 Geolocation API

476 Geolocation using GPS must be supported. Uncertainty bounds must be pro-
477 vided for the returned location, including the time at which the location was
478 measured (in order to support cached locations). The API must gracefully
479 handle failure to geolocate the vehicle (for example, if no GPS satellites are
480 available).

481 Locations must be provided with position in all three dimensions, speed and
482 heading information if known. Locations should be provided with the other two
483 angles of movement, the rate of change of angle of movement in all three dimen-
484 sions, and uncertainty bounds and standard deviation for all measurements if
485 known.

486 See [Relocating the vehicle](#).

487 Geolocation service supports signals

488 Application bundles must be able to register to receive a signal whenever the
489 vehicle's location changes significantly. The bundle should be able to specify
490 a maximum time between updates and a maximum distance between updates,
491 either of which may be zero. The bundle should also be able to specify a
492 *minimum* time between updates, in order to prevent being overwhelmed by
493 updates.

494 See [Navigating to a location](#).

495 Geolocation implements caching

496 If an up-to-date location is not known, the geolocation API may return a cached
497 location with an appropriate time of measurement.

498 See [Relocating the vehicle](#), [Tasks nearby](#).

499 Geolocation supports backends

500 The geolocation implementation must support multiple backend implementa-
501 tions, with the selection of backend or backends to be used in a particular
502 distribution of Apertis being a runtime decision.

503 The default navigation application (requirement 5.6) must be able to feed ge-
504 olocation information into the service as an additional backend.

505 See [Automotive backend](#), [Third-party navigation-driven refinement of geoloca-](#)
506 [tion](#)

507 **Navigation routing API**

508 Launching the navigation application with zero or more waypoints and a des-
509 tination must be supported. The navigation application can internally decide
510 how to handle the new coordinates —whether to replace the current route, or
511 supplement it. The application may query the user about this.

512 The interface for launching the navigation application must also support ‘way-
513 areas’, or be extensible to support them in future. It must support setting some
514 waypoints as to be announced, and some as to be used for routing but not as
515 announced intermediate destinations.

516 It must support setting a destination (or any of the waypoints) as an address
517 or as a city.

518 It must support systems where no navigation application is installed, returning
519 an error code to the caller.

520 It must provide a way to request navigation routing suitable for walking or
521 cycling, so that the driver knows how to reach a point of interest after they
522 leave the vehicle.

523 See [Navigating to a location](#), [Changing destination during navigation](#), [Navigat-](#)
524 [ing via waypoints](#), [Navigating a tour](#), [Navigating to an entire city](#), [No navigation](#)
525 [application](#).

526 **Navigation routing supports different navigation applications**

527 The mechanism for launching the navigation application (requirement 5.5) must
528 allow a third-party navigation application to be set as the default, handling all
529 requests.

530 See [Installing a navigation application](#), [Third-party navigation-driven refine-](#)
531 [ment of geolocation](#).

532 **Navigation route list API**

533 A navigation route list API must be supported, which exposes information from
534 the navigation application about the current route: the planned route, including
535 destination, waypoints and way-areas.

536 The API must support signalling applications of changes in this information, for
537 example when the destination is changed or a new route is calculated to avoid
538 bad traffic.

539 If no navigation application is installed, the route list API must continue to be
540 usable, and must return an error code to callers.

541 See [Navigation route guidance information](#), [No navigation application](#).

542 **Navigation route guidance API**

543 A route guidance API must be supported, which allows the navigation applica-
544 tion to expose information about the directions to take next, and the vehicle's
545 progress through the current route. It must include:

- 546 • Estimates such as time to destination and time elapsed in the journey.
547 Equivalently, the journey departure and estimated arrival times at each
548 destination.
- 549 • Turn-by-turn navigation instructions for the route.

550 The API must support data being produced by the navigation application and
551 consumed by a single system-provided assistance or guidance UI, which is part
552 of the system UI. It must support being called by the navigation application
553 when the next turn-by-turn instruction needs to be presented, or the estimated
554 journey end time changes.

555 See [Navigation route guidance information](#).

556 **Type-ahead search and address completion supports backends**

557 The address completion implementation must support multiple backend imple-
558 mentations, with the selection of backend or backends to be used in a particular
559 distribution of Apertis being a runtime decision.

560 See [Automotive backend](#), [Type-ahead search and completion for addresses](#).

561 **Geocoding supports backends**

562 The geocoding implementation must support multiple backend implementations,
563 with the selection of backend or backends to be used in a particular distribution
564 of Apertis being a runtime decision.

565 See [Automotive backend](#).

566 **SDK has default implementations for all backends**

567 A free software, default implementation of all geo-functionality must be provided
568 in the SDK, for use by developers. It may rely on an internet connection for its
569 functionality.

570 The SDK implementation must support all functionality of the geo-APIs in order
571 to allow app developers to test all functionality used by their applications.

572 See [SDK backend](#).

573 **SDK APIs do not vary with backend**

574 App bundles must not have to be modified in order to switch backends: the
575 choice of backend should not affect implementation of the APIs exposed in the

576 SDK to app bundles.

577 If a navigation application has been developed by a vendor to use vendor-specific
578 proprietary APIs to communicate with the automotive domain, that must be
579 possible; but other applications must not use these APIs.

580 See [Automotive backend](#), [Custom automotive backend functionality](#).

581 **Third-party navigation applications can be used as backends**

582 A third-party or OEM-provided navigation application must, if it imple-
583 ments the correct interfaces, be able to act as a backend for some or all
584 geo-functionality.

585 See [Third-party navigation application backend](#).

586 **Backends operate asynchronously**

587 As backends for geo-functionality may end up making inter-domain requests,
588 or may query web services, the interfaces between applications, the SDK APIs,
589 and the backends must all be asynchronous and tolerant of latency.

590 See the Inter-Domain Communications design.

591 See [Web-based backend](#).

592 **2D map rendering API**

593 Map display has the following requirements:

- 594 • Rendering the map (see [Relocating the vehicle](#)).
- 595 • Rendering points of interest, including start and destination points for
596 navigation.
- 597 • Rendering a path or route.
- 598 • Rendering a polygon or region highlight.
- 599 • The map display must support loading client side map tiles, or server-
600 provided ones.
- 601 • The map rendering may be done client-side (vector maps) or pre-computed
602 (raster maps).
- 603 • Rendering custom widgets provided by the application.
- 604 • Optionally rendering the current route list as a map layer.
- 605 • Optionally rendering the vehicle's current position as a map layer (see
606 [\[Relocating the vehicle\]](#)[\[Relocating the vehicle\]](#)).

607 See [Viewing an address on the map](#), [Adding custom widgets on the map](#).

608 **2.5D or 3D map rendering API**

609 For applications which wish to present a perspective view of a map, a 2.5D or
610 3D map widget should be provided with all the same features as the 2D map
611 rendering API.

612 See [2.5D or 3D map widget](#).

613 **Reverse geocoding API**

614 Reverse geocoding must be supported, converting an address into zero or more
615 coordinates. Limiting the search results to coordinates in a radius around a
616 given reference coordinate pair must be supported.

617 Reverse geocoding may work when the vehicle has no internet connection, but
618 only if that is easy to implement.

619 See [Viewing an address on the map](#).

620 **Forward geocoding API**

621 Forward geocoding must be supported for querying addresses at selected coor-
622 dinates on a map. Limiting the search results to a certain number of results
623 should be supported.

624 Forward geocoding may work when the vehicle has no internet connection, but
625 only if that is easy to implement.

626 See [Finding the address of a location on the map](#).

627 **Type-ahead search and address completion API**

628 Suggesting and ranking potential completions to a partially entered address
629 must be supported by the system, with latency suitable for use in a type-ahead
630 completion system. This should be integrated into a widget for ease of use by
631 application developers.

632 Address completion may work when the vehicle has no internet connection, but
633 only if that is easy to implement.

634 This may need to be integrated with other keyboard usability systems, such as
635 typing suggestions and keyboard history. If the functionality cannot be imple-
636 mented or the service for it is not available, the system should provide a normal
637 text entry box to the user.

638 See [Type-ahead search and completion for addresses](#).

639 **Geofencing API**

640 Application bundles must be able to define arbitrary regions –either arbitrary
641 polygons, or points with radii –and request a signal when entering, exiting, or

642 dwelling in a region. The vehicle is dwelling in a region if it has been in there
643 for a specified amount of time without exiting.

644 See [Tasks nearby](#), [Turning on house lights](#).

645 **Geofencing service can wake up applications**

646 It must be possible for geofencing signals to be delivered even if the application
647 bundle which registered to receive them is not currently running.

648 See [Tasks nearby](#), [Turning on house lights](#).

649 **Geofencing API signals on national borders**

650 The geofencing API should provide a built-in geofence for the national borders
651 of the current country, which applications may subscribe to signals about, and
652 be woken up for as normal, if the vehicle crosses the country's border.

653 See [Traffic rule application](#).

654 **Geocoding API must be locale-aware**

655 The geocoding API must support returning results or taking input, such as
656 addresses, in a localised form. The localisation must be configurable so that, for
657 example, the user's home locale could be used, or the locale of the country the
658 vehicle is currently in.

659 See [Traffic rule application](#).

660 **Geolocation provides error bounds**

661 The geolocation API must provide an error bound for each location measurement
662 it returns, so calling code knows how accurate that data is likely to be.

663 See [Temporary loss of GPS signal](#).

664 **Geolocation implements dead-reckoning**

665 The geolocation API must implement dead reckoning based on the vehicle's
666 previous velocity, to allow a location to be returned even if GPS signal is lost.
667 This must update the error bounds appropriately ([Geolocation provides error](#)
668 [bounds](#)).

669 See [Temporary loss of GPS signal](#).

670 **Geolocation uses navigation and sensor data if available**

671 If such data is available, the geolocation API may use navigation and sensor data
672 to improve the accuracy of the location it reports, for example by snapping the

673 GPS location to the nearest road on the map using information provided by the
674 navigation application.

675 See [Navigation- and sensor-driven refinement of geolocation](#).

676 **General points of interest streams are queryable**

677 The general [point of interest streams](#)⁵ exposed by applications must be queryable
678 using a library API.

679 The approach of exposing points of interest via the geocoding system, as results
680 for reverse geocoding requests, was considered and decided against. Reverse
681 geocoding is designed to turn a location (latitude and longitude) into informa-
682 tion describing the nearest address or place —not to a series of results describing
683 every point of interest within a certain radius. Doing so introduces problems
684 with defining the search radius, determining which of the results is the geocoding
685 result, and eliminating duplicate points of interest.

686 See the [Points of interest design](#)⁶.

687 Further requirements and designs specific to how applications expose such gen-
688 eral points of interest streams are covered in the Points of Interest design.

689 See [Application-driven refinement of geocoding](#).

690 **Location information requires permissions to access**

691 There are privacy concerns with allowing bundles access to location data. The
692 system must be able to restrict access to any data which identifies the vehicle's
693 current, past or planned location, unless the user has explicitly granted a bundle
694 access to it. The system may differentiate access into coarse-grained and fine-
695 grained, for example allowing application bundles to request access to location
696 data at the resolution of a city block, or at the resolution of tens of centimetres.
697 Note that fine-grained data access must be allowed for geofencing support, as
698 that essentially allows bundles to evaluate the vehicle's current location against
699 arbitrary location queries.

700 Application bundles asking for fine-grained location data must be subjected to
701 closer review when submitted to the Apertis application store.

702 See [Malware application bundle](#).

703 **Open question:** What review checks should be performed on application bun-
704 dles which request permissions for location data?

⁵https://www.apertis.org/concepts/archive/application/points_of_interest/#General_POI_providers

⁶https://www.apertis.org/concepts/archive/application/points_of_interest/

705 **Rate limiting of general point of interest streams**

706 When handling general point of interest streams generated by applications, the
707 system must prevent denial of service attacks from the applications by limiting
708 the number of points of interest they can feed to the geolocation and other
709 services, both in the rate at which they are transferred, and the number present
710 in the system at any time.

711 See [Excessive results from application-driven refinement of geocoding](#).

712 **Application-provided data requires permissions to create**

713 The user must be able to enable or disable each application from providing data
714 to the system geo-services, such as route recommendations or points of interest,
715 without needing to uninstall that application (i.e. so they can continue to use
716 other functionality from the application).

717 See [User control over applications](#).

718 **Existing geo systems**

719 This chapter describes the approaches taken by various existing systems for
720 exposing sensor information to application bundles, because it might be useful
721 input for Apertis' decision making. Where available, it also provides some details
722 of the implementations of features that seem particularly interesting or relevant.

723 **W3C Geolocation API**

724 The [W3C Geolocation API](#)⁷ is a JavaScript API for exposing the user's location
725 to web apps. The API allows apps to query the current location, and to register
726 for signals of position changes. Information about the age of location data (to
727 allow for cached locations) is returned. Information is also provided about the
728 location's accuracy, heading and speed.

729 **Android platform location API**

730 The [Android platform location API](#)⁸ is a low-level API for performing geoloca-
731 tion based on GPS or visible Wi-Fi and cellular networks, and does not provide
732 geofencing or geocoding features. It allows geolocation and cached geolocation
733 queries, as well as signals of changes in location. Its design is highly biased
734 towards making apps energy efficient so as to maintain mobile battery life.

735 **Google Location Services API for Android**

736 The [Google Location Services API for Android](#)⁹ is a more fully featured API
737 than the platform location API, supporting geocoding and geofencing in addi-

⁷<http://www.w3.org/TR/geolocation-API/>

⁸<http://developer.android.com/guide/topics/location/strategies.html>

⁹<http://developer.android.com/training/location/index.html>

tion to geolocation. It requires the device to be connected to the internet to access Google Play services. It hides the complexity of calculating and tracking the device's location much more than the platform location API.

It allows apps to specify upper and lower bounds on the frequency at which they want to receive location updates. The location service then calculates updates at the maximum of the frequencies requested by all apps, and emits signals at the minimum of this and the app's requested upper frequency bound.

It also defines the permissions required for accessing location data more stringently, allowing coarse- and fine-grained access.

iOS Location and Maps API

The **iOS Location Services and Maps API** is available on both iOS and OS X. It supports many features: geolocation, geofencing, forward and reverse geocoding, navigation routing, and local search.

For geolocation, it supports querying the location and location change signals, including signals to apps which are running in the background.

Its geofencing support is for points and radii, and supports entry and exit signals but not dwell signals. Instead, it supports hysteresis based on distance from the region boundary.

Geocoding uses a network service; both forward and reverse geocoding are supported.

The **MapKit API**¹⁰ provides an embeddable map renderer and widget, including annotation and overlay support.

iOS (but not OS X) supports using arbitrary apps as routing providers for rendering **turn-by-turn navigation instructions**¹¹. An app which supports this must declare which geographic regions it supports routing within (for example, a subway navigation app for New York would declare that region only), and must accept routing requests as a URI handler. The URIs specify the start and destination points of the navigation request.

It also supports navigation routing using a system provider, which requires a network connection. Calculated routes include metadata such as distance, expected travel time, localised advisory notices; and the set of steps for the navigation. It supports returning multiple route options for a given navigation.

The **local search API**¹² differs from the geocoding API in that it supports *types*

¹⁰https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/MapKit/MapKit.html#//apple_ref/doc/uid/TP40009497-CH3-SW1

¹¹https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/ProvidingDirections/ProvidingDirections.html#//apple_ref/doc/uid/TP40009497-CH8-SW5

¹²https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/EnablingSearch/EnablingSearch.html#//apple_ref/doc/uid/TP40009

771 of locations, such as ‘coffee’ or ‘fuel’. As with geocoding, the local search API
772 requires a network connection.

773 **GNOME APIs**

774 GNOME uses several libraries to provide different geo-features. It does not have
775 a library for navigation routing.

776 **GeoClue** [GeoClue](http://freedesktop.org/wiki/Software/GeoClue/)¹³ is a geolocation service which supports multiple input
777 backends, such as GPS, cellular network location and Wi-Fi based geolocation.

778 Wi-Fi location uses the [Mozilla Location Service](https://wiki.mozilla.org/CloudServices/Location)¹⁴ and requires network connec-
779 tivity.

780 It supports [geolocation signals](#)¹⁵ with a minimum distance between signals, but
781 no time-based limiting. It does not support geofencing, but the developers are
782 interested in implementing it.

783 GeoClue’s security model allows permissions to be applied to individual apps,
784 and location accuracy to be restricted on a per-app basis. However, this model
785 is currently incomplete and does not query the system’s trusted computing base
786 (TCB) (see the Security design for definitions of the TCB and trust).

787 **Geocode-glib** [Geocode-glib](https://gitlab.gnome.org/GNOME/geocode-glib)¹⁶ is a library for forward and reverse geocod-
788 ing. It uses the Nominatim API, and is currently [hard-coded to query nomi-](#)
789 [natin.gnome.org](http://wiki.openstreetmap.org/wiki/Nominatim)¹⁷. It requires network access to perform geocoding.

790 The [Nominatim API](#)¹⁸ does not require an API key (though it does require
791 a contact e-mail address), but it is highly recommended that anyone using it
792 commercially runs their own Nominatim server.

793 geocode-glib is tied to a single Nominatim server, and does not support multiple
794 backends.

795 **libchamplain** [libchamplain](https://wiki.gnome.org/Projects/libchamplain)¹⁹ is a map rendering library, providing a map
796 widget which supports annotations and overlays. It supports loading or render-
797 ing map tiles from multiple sources.

497-CH10-SW1

¹³<http://freedesktop.org/wiki/Software/GeoClue/>

¹⁴<https://wiki.mozilla.org/CloudServices/Location>

¹⁵<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-signal-org-freedesktop-GeoClue2-Client.LocationUpdated>

¹⁶<https://gitlab.gnome.org/GNOME/geocode-glib>

¹⁷https://bugzilla.gnome.org/show_bug.cgi?id=756311

¹⁸<http://wiki.openstreetmap.org/wiki/Nominatim>

¹⁹<https://wiki.gnome.org/Projects/libchamplain>

798 **NavIt** [NavIt](https://www.navit-project.org/)²⁰ is a 3D turn-by-turn navigation system designed for cars. It
799 provides a GTK+ or SDL interface, audio output using espeak, GPS input
800 using gpsd, and multiple map rendering backends. It seems to expose some of
801 its functionality as a shared library (libnavit), but it is unclear to what extent it
802 could be re-used as a component in an application, without restructuring work.
803 It may be possible to package it, with modifications, as a third-party navigation
804 application, or as the basis of one.

805 **Navigation routing systems**

806 Three alternative routing systems are described briefly below; a full analysis
807 based on running many trial start and destination routing problems against
808 them is yet to be done.

809 **GraphHopper** [GraphHopper](https://graphhopper.com/)²¹ is a routing system written in Java, which is
810 available as a server or as a library for offline use. It uses OpenStreetMap data,
811 and is licenced under Apache License 2.0.

812 **OSRM** [OSRM](http://project-osrm.org/)²² is a BSD-licensed C++ routing system, which can be used
813 as a server or as a library for offline use. It uses OpenStreetMap data.

814 **YOURS** [YOURS](https://wiki.openstreetmap.org/wiki/YOURS)²³ is an online routing system which provides a web API for
815 routing using OpenStreetMap data.

816 **NavServer**

817 NavServer is a proprietary middleware navigation solution, which accesses a core
818 navigation system over D-Bus. It is designed to be used as a built-in navigation
819 application.

820 It is currently unclear as to the extent which NavServer could be used to feed
821 data in to the SDK APIs (such as geolocation refinements).

822 **GENIVI**

823 GENIVI implements its geo-features and navigation as various components under
824 the umbrella of the [IVI Navigation project](https://at.projects.genivi.org/wiki/display/NAV/IVI+Navigation+Home)²⁴. The core APIs it provides are
825 detailed below.

²⁰<https://www.navit-project.org/>

²¹<https://graphhopper.com/>

²²<http://project-osrm.org/>

²³<https://wiki.openstreetmap.org/wiki/YOURS>

²⁴<https://at.projects.genivi.org/wiki/display/NAV/IVI+Navigation+Home>

826 **Navigation** The navigation application is based on NavIt, using Open-
827 StreetMap for its mapping data. It implements route calculation, turn-by-turn
828 instructions, and map rendering.

829 It is implemented in two parts: a navigation service, which uses libnavit to
830 expose routing APIs over D-Bus; and the navigation UI, which uses these APIs.
831 The navigation service is implemented as a set of plugins for NavIt.

832 **Fuel stop advisor** The fuel stop advisor is a demo application which con-
833 sumes information from the geolocation API and the vehicle API to get the
834 vehicle's location and fuel levels, in order to predict when (and where) would be
835 best to refuel.

836 **POI service** The points of interest (POI) service is implemented using mul-
837 tiple 'content access module'(CAM) plugins, each providing points of interest
838 from a different provider or source. When searching for POIs, the service sends
839 the query to all CAMs, with a series of attributes and values to match against
840 them using a set of operators (equal, less than, greater than, etc.); plus a lat-
841 itude and longitude to base the query around. CAMs return their results to
842 the service, which then forwards them to the client which originally made the
843 search request.

844 CAMs may also register categories of POIs which they can provide. These
845 categories are arranged in a tree, so the user may limit their queries to certain
846 categories.

847 Additionally, POI searches may be conducted against a given route list, finding
848 points of interest along the route line, instead of around a single centre point.

849 Finally, it supports a 'proximity alert' feature, where the POI system will signal
850 a client if the vehicle moves within a given distance of any matching POI.

851 **Positioning** The positioning API provides a large amount of positioning data:
852 time, position in three dimensions, heading, rate of change of position in three
853 dimensions, rate of change of angle in three dimensions, precision of position in
854 three dimensions, and standard deviation of position in three dimensions and
855 heading.

856 The service emits signals about position changes at arbitrary times, up to a
857 maximum frequency of 10Hz. It exposes information about the number of GPS
858 satellites currently visible, and signals when this changes.

859 **Google web APIs**

860 Google provides various APIs for geo-functionality. They are available to use
861 subject to a billing scale which varies based on the number of requests made.

862 **Google Maps Geocoding API** The [Google Maps geocoding API](#)²⁵ is a
863 HTTPS service which provides forward and reverse geocoding services, and
864 provides results in JSON or XML format.

865 Forward geocoding supports address formats from multiple locales, and supports
866 filtering results by county, country, administrative region or postcode. It also
867 supports artificially boosting the importance of results within a certain bounds
868 box or region, and returning results in multiple languages.

869 Reverse geocoding takes a latitude and longitude, and optionally a result type
870 which allows the result to be limited to an address, a street, or country (for
871 example).

872 The array of potential results returned by both forward and reverse geocoding
873 requests include the location's latitude and longitude, its address as a formatted
874 string and as components, details about the address (if it is a postal address)
875 and the type of map feature identified at those coordinates.

876 The service is designed for geocoding complete queries, rather than partially-
877 entered queries as part of a type-ahead completion system.

878 **Google Places API** The [Google Places API](#)²⁶ supports several different op-
879 erations: returning a list of places which match a user-provided search string,
880 returning details about a place, and auto-completing a user's search string based
881 on places it possibly matches.

882 The search API supports returning a list of points of interest, and metadata
883 about them, which are within a given radius of a given latitude and longitude.

884 The details API takes an opaque identifier for a place (which is understood by
885 Google and by no other services) and returns metadata about that place.

886 The autocomplete API takes a partial search string and returns a list of poten-
887 tial completions, including the full place name as a string and as components,
888 the portion which matched the input, and the type of place it is (such as a road,
889 locality, or political area).

890 **Google Maps Roads API** The [Google Maps Roads API](#)²⁷ provides a snap
891 to roads API, which takes a list of latitude and longitude points, and which
892 returns the same list of points, but with each one snapped to the nearest road
893 to form a likely route which a vehicle might have taken.

894 The service can optionally interpolate the result so that it contains more points
895 to smoothly track the potential route, adding points where necessary to disam-
896 biguate between different options.

²⁵<https://developers.google.com/maps/documentation/geocoding/intro>

²⁶<https://developers.google.com/places/web-service/>

²⁷<https://developers.google.com/maps/documentation/roads/intro>

897 **Google Maps Geolocation API** The [Google Maps Geolocation API](#)²⁸ pro-
898 vides a way for a mobile device (any device which can detect mobile phone
899 towers or Wi-Fi access points) to look up its likely location based on which
900 mobile phone towers and Wi-Fi access points it can currently see.

901 The API takes some details about the device's mobile phone network and carrier,
902 plus a list of identifiers for nearby mobile phone towers and Wi-Fi access points,
903 and the signal strength the device sees for each of them. It returns a best guess
904 at the device's location, as a latitude, longitude and accuracy radius around
905 that point (in metres).

906 If the service cannot work out a location for the device, it tries to geolocate based
907 on the device's IP address; this will always return a result, but the accuracy will
908 be very low. This option may be disabled, in which case the service will return
909 an error on failure to work out the device's location.

910 Approach

911 Based on the [Existing geo systems](#)) and [Requirements](#), we recommend the fol-
912 lowing approach for integrating geo-features into Apertis. The overall summary
913 is to use existing freedesktop.org and GNOME components for all geo-features,
914 adding features to them where necessary, and adding support for multiple back-
915 ends to support implementations in the automotive domain or provided by a
916 navigation application.

917 Backends

918 Each of the geo-APIs described in the following sections will support multi-
919 ple backends. These backends must be choosable at runtime so that a newly
920 installed navigation application can be used to provide functionality for a back-
921 end. Switching backends may require the vehicle to be restarted, so that the
922 system can avoid the complexities of transferring state between the old backend
923 and the new one, such as route information or GPS location history.

924 Applications must not be able to choose which backend is being used for a
925 particular geo-function —that is set as a system preference, either chosen by the
926 user or fixed by the system integrator.

927 If there are particular situations where it is felt that the application developer
928 knows better than the system integrator about the backend to use, that signals
929 a use case which has not been considered, and might be best handled by a
930 revision of this design and potentially introducing a new SDK API to expose
931 the backend functionality desired by the application developer.

932 Backends may be implemented in the IVI domain (for example, the default
933 backends in the SDK must be implemented in the IVI domain, as the SDK has
934 no other domains), or in the automotive domain. If a backend is implemented

²⁸<https://developers.google.com/maps/documentation/geolocation/intro>

935 in the automotive domain, its functionality must be exposed as a proxy service
936 in the IVI domain, which implements the SDK API. Communications between
937 this proxy service and the backend in the automotive domain will be over the
938 inter-domain communications link.

939 See the Inter-Domain Communications design

940 This IPC interface serves as a security boundary for the backend.

941 Third-party applications (such as navigation applications) may provide back-
942 ends for geo-services as dynamically loaded libraries which are installed as part
943 of their application bundle. As this allows arbitrary code to be run in the
944 context of the geo-services (which form security boundaries for applications, see
945 **Systemic security**), the code for these third-party backends must be audited and
946 tested (**Testing backends**) carefully as part of the app store validation process.

947 Due to the potential for inter-domain communications, or for backends which
948 access web services to provide functionality, the backend and SDK APIs must
949 be asynchronous and tolerant of latency.

950 Backends may expose functionality which is not abstracted by the SDK APIs.
951 This functionality may be used by applications directly, if they wish to be tied
952 to that specific backend. As noted above, this may signal an area where the
953 SDK API could be improved or expanded in future.

954 **Navigation application**

955 Throughout this design, the phrase *navigation application* should be taken to
956 mean the navigation application bundle, including its UI, a potentially separate
957 **Route guidance ui** and any agents or backends for **Backends**. While the naviga-
958 tion application bundle may provide backends which feed data to a lot of the
959 geo-APIs in the SDK, it may additionally use a private connection and arbitrary
960 protocol to communicate between its backends and its UI. Data being presented
961 in the navigation application UI does not necessarily come from SDK APIs. See
962 [this non-use-case][SDK APIs for third-party navigation applications].

963 A system might not have a navigation application installed (see **Navigation**
964 **routing API**), in which case all APIs which depend on it must return suitable
965 error codes.

966 **2D map display**

967 **libchamplain**²⁹ should be used for top-down map display. It supports map
968 rendering, adding markers, points of interest (with explanatory labels), and
969 start and destination points. Paths, routes, polygons and region highlights can
970 be rendered as using Clutter API on custom map layers.

²⁹<https://wiki.gnome.org/Projects/libchamplain>

971 libchamplain supports pre-rendered tiles from online (ChamplainNetworkTile-
972 Source) or offline (ChamplainFileTileSource) sources. It supports rendering tiles
973 locally using libmemphis, if compiled with that support enabled.

974 On an integrated system, map data will be available offline on the vehicle's file
975 system. On the Apertis SDK, an internet connection is always assumed to be
976 available, so map tiles may be used from online sources. libchamplain supports
977 both.

978 libchamplain supports rendering custom widgets provided by the application on
979 top of the map layer.

980 **Route list layer on the 2D map** A new libchamplain layer should be pro-
981 vided by the Apertis SDK which renders the vehicle's current route list as an
982 overlay on the map, updating it as necessary if the route is changed. If no route
983 is set, the layer must display nothing (i.e. be transparent).

984 Applications can add this layer to an instance of libchamplain in order to easily
985 get this functionality.

986 In order for this to work, the application must have permission to query the
987 vehicle's route list.

988 **Vehicle location layer on the 2D map** A new libchamplain layer should
989 be provided by the Apertis SDK which renders the vehicle's current location as
990 a point on the map using a standard icon or indicator. The location should be
991 updated as the vehicle moves. If the vehicle's location is not known, the layer
992 must display nothing (i.e. be transparent).

993 Applications can add this layer to an instance of libchamplain in order to easily
994 get this functionality.

995 In order for this to work, the application must have permission to query the
996 vehicle's location.

997 **2.5D or 3D map display**

998 Our initial suggestion is to use [NavIt](https://www.navit-project.org/)³⁰ for 3D map display. However, we are
999 currently unsure of the extent to which it can be used as a library, so we cannot
1000 yet recommend an API for 2.5D or 3D map display. Similarly, we are unsure
1001 of the extent to which route information and custom rendering can be input to
1002 the library to integrate it with other routing engines; or whether it always has
1003 to use routes calculated by other parts of NavIt.

1004 **Open question:** Is it possible to use NavIt as a stand-alone 2.5D or 3D map
1005 widget?

³⁰<https://www.navit-project.org/>

1006 **Geolocation**

1007 [GeoClue](#)³¹ should be used for geolocation. It supports multiple backends, so
1008 closed source as well as open source backends can be used. Some of the advanced
1009 features which do not impact on the API could be implemented in an automotive
1010 backend, although other backends would benefit if they were implemented in the
1011 core of GeoClue instead. For example, cached locations and dead-reckoning of
1012 the location based on previous velocity for when GPS signal is lost.

1013 The vehicle's velocity may be queried from the sensors; see the Sen-
1014 sors and Actuators design

1015 GeoClue supports geolocation using GPS (from a modem), 3G and Wi-Fi. It
1016 supports [accuracy bounds for locations](#)³², but does not pair that with informa-
1017 tion about the time of measurement. That would need to be added as a new
1018 feature in the API. Speed, altitude and bearing information [are supported](#)³³.
1019 The other two angles of movement, the rate of change of angle of movement in
1020 all three dimensions, and uncertainty bounds and standard deviation for non-
1021 position measurements are not currently included in the API, and should be
1022 added.

1023 The API already supports signalling of failure to geolocate the vehicle, by setting
1024 its Location property to '/' (rather than a valid org.freedesktop.GeoClue2.Location
1025 object path).

1026 If the navigation application implements a snap-to-road feature, it should be
1027 used as a further source of input to GeoClue for refining the location.

1028 **Geolocation signals** GeoClue emits a [LocationUpdated signal](#)³⁴ whenever
1029 the vehicle's location changes more than the [DistanceThreshold](#)³⁵. GeoClue
1030 currently does not support rate limiting emission of the LocationUpdated signal
1031 for minimum and maximum times between updates. That would need to be
1032 added to the core of GeoClue.

1033 **Navigation routing**

1034 Navigation routing will be implemented internally by the OEM-provided naviga-
1035 tion application, or potentially by a third-party navigation application installed
1036 by the user. In either case, there will not be an Apertis SDK API for calculating
1037 routes.

³¹<http://freedesktop.org/wiki/Software/GeoClue/>

³²[http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Location.html#gdbus-property-org-freedesktop-GeoClue2-Location.Accuracy](http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Location.html#gdbus-property-org-freedesktop-GeoClue2-Location-Accuracy)

³³<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Location.html>

³⁴<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-signal-org-freedesktop-GeoClue2-Client.LocationUpdated>

³⁵<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-property-org-freedesktop-GeoClue2-Client.DistanceThreshold>

1038 However, the SDK will provide an API to launch the navigation application and
1039 instruct it to calculate a new route. This API is the [content hand-over](#)³⁶ API,
1040 where the navigation application can be launched using a nav URI. The nav URI
1041 scheme is a custom scheme (which must not be confused with the standard [geo](#)
1042 [URI scheme](#)³⁷, which is for identifying coordinates by latitude and longitude).
1043 See [Appendix: nav URI scheme](#) for a definition of the scheme, and examples.

1044 When handling a content hand-over request, the navigation application should
1045 give the user the option of whether to replace their current route with the new
1046 route—but this behaviour is internal to the navigation application, and up to
1047 its developer and policy.

1048 The behaviour of the navigation application if it is passed an invalid URI, or
1049 one which it cannot parse (for example, due to not understanding the address
1050 format it uses) is not defined by this specification.

1051 As per the [content hand-over design](#)³⁸, the user may choose a third-party navi-
1052 gation application as the handler for nav URIs, in which case it will be launched
1053 as the default navigation application.

1054 If no navigation application is installed, or none is set as the handler for nav
1055 URIs, the error must be handled as per the content hand-over design.

1056 **Navigation route list API**

1057 Separately from the content hand-over API for sending a destination to the nav-
1058 igation application (see [Navigation routing](#)), there should be a navigation route
1059 list API to expose information about the route geometry to SDK applications.

1060 This API will be provided by the SDK, but implemented by one of many back-
1061 ends—the navigation application will be one such backend, but another backend
1062 will be available on the SDK to expose mock data for testing applications against.
1063 The mock data will be provided by an emulator; see [Testing applications](#) for
1064 more information.

1065 These backends will feed into a system service which provides the SDK API to
1066 applications, and exposes:

- 1067 • The planned route geometry, including destination, waypoints and way-
1068 areas.
- 1069 • Potential alternative routes.

1070 The API will not expose the vehicle's current position—that's done by the
1071 [Geolocation](#). Similarly, it will not expose points of interest or other horizon
1072 data—that's done by the [points of interest API](#)³⁹; or guidance information—
1073 that's done by the [Navigation route guidance API](#).

³⁶https://www.apertis.org/concepts/archive/application_framework/content_hand-over/

³⁷<http://tools.ietf.org/html/rfc5870>

³⁸https://www.apertis.org/concepts/archive/application_framework/content_hand-over/

³⁹https://www.apertis.org/concepts/archive/application/points_of_interest/

1074 The API should emit signals on changes of any of this information, using the
1075 standard org.freedesktop.DBus.Properties signals. We recommend the following
1076 API, exposed at the well-known name org.apertis.Navigation1:

```
1077 /* This object should implement the org.freedesktop.DBus.ObjectManager standard
1078     API to expose all existing routes to clients.
1079 */
1080 object /org/apertis/Navigation1/Routes {
1081     read-only i property CurrentRoute; /* index into routes, or negative for 'no current route' */
1082     read-only ao property Routes; /* paths of objects representing potential routes, with the most highly recommended first */
1083 }
1084
1085 /* One of these objects is created for each potential route.
1086     Each object path is suffixed by a meaningless ID to ensure its uniqueness.
1087     Route objects are immutable once published. Any change should be done by
1088     adding a new route and removing the old one.
1089 */
1090 object /org/apertis/Navigation1/Routes/$ID {
1091     read-only a(dd) property Geometry; /* array of latitude-longitude pairs in order, from the start point to the end point */
1092     read-only u property TotalDistance; /* in metres */
1093     read-only u property TotalTime; /* in seconds */
1094     read-only a(ss) property Title; /* mapping from language name to a human-
1095 readable title of the route in this language. Language names are using the POSIX locale format with locale identifier */
1096     read-only a{ua{ss}} property GeometryDescriptions; /* array of pairs of index and description, which attach a description to each geometry point */
1097 }
```

1098 Syntax is a pseudo-IDL and types are as defined in the D-Bus specification, <http://dbus.freedesktop.org/doc/dbus-specification.html>
1099 [#type-system](#)
1100

1101 **Navigation route list backends** The backend for the route list service is
1102 provided by the navigation application bundle, which may be built-in or provided
1103 by a user-installed bundle. If no navigation bundle is installed, no backend
1104 is used, and the route list service must return an error when called.

1105 On the Apertis SDK, a navigation bundle is not available, so a mock backend
1106 should be written which presents route lists provided by the developer. This
1107 will allow applications to be tested using route lists of the developer's choice.

1108 Navigation route guidance progress API

1109 There should be a navigation route guidance API to expose information about
1110 progress on the current route.

1111 This API will be provided as interfaces by the SDK, but implemented by the
1112 navigation application. The UI responsible for displaying progress information
1113 or third party applications can use this API to fetch the info from the navigation
1114 application.

1115 The route guidance progress API should be a D-Bus API which exposes esti-
1116 mates such as time to destination and time elapsed in the journey.

1117 The API should emit signals on changes of any of this information, using the
1118 standard `org.freedesktop.DBus.Properties` signals. We recommend the following
1119 API, exposed at the well-known name `org.apertis.NavigationGuidance1.Progress`
1120 on the object `/org/apertis/NavigationGuidance1/Progress`:

```
1121 interface org.apertis.NavigationGuidance1.Progress {  
1122     read-only t property StartTime; /* UNIX timestamp, to be interpreted in the local timezone */  
1123     read-only t property EstimatedEndTime; /* UNIX timestamp, to be interpreted in the local timezone */  
1124 }
```

1125 Additional properties may be added in future.

1126 Syntax is a pseudo-IDL and types are as defined in the D-Bus spec-
1127 ification, <http://dbus.freedesktop.org/doc/dbus-specification.html>
1128 [#type-system](#)

1129 Navigation route guidance turn-by-turn API

1130 There should be a navigation route guidance API to allow turn-by-turn guidance
1131 notifications to be presented.

1132 This API will be provided as interfaces by the SDK, but implemented by a
1133 system component which is responsible for presenting notifications —this will
1134 most likely be the compositor. The navigation application can then call the
1135 TurnByTurn API to display new turn-by-turn driving instructions.

1136 The route guidance turn-by-turn API should be a D-Bus API which exposes a
1137 method for presenting the next turn-by-turn navigation instruction.

1138 We recommend the following API, exposed at the well-known name
1139 `org.apertis.NavigationGuidance1.TurnByTurn` on the object `/org/apertis/NavigationGuidance1/TurnByTurn`:

```
1140 interface org.apertis.NavigationGuidance1.TurnByTurn {  
1141     /* See https://people.gnome.org/~mccann/docs/notification-spec/notification-  
1142     spec-latest.html#command-notify */  
1143     method Notify (in u replaces_id,  
1144                   in s icon_name,  
1145                   in s summary,  
1146                   in s body,  
1147                   in a{sv} hints,  
1148                   in i expire_timeout,  
1149                   out u id)  
1150  
1151     /* See https://people.gnome.org/~mccann/docs/notification-spec/notification-  
1152     spec-latest.html#command-close-notification */  
1153  
1154     method CloseNotification (in u id)
```

1155 }

1156 Syntax is a pseudo-IDL and types are as defined in the D-Bus spec-
1157 ification, <http://dbus.freedesktop.org/doc/dbus-specification.html>
1158 [#type-system](#)

1159 The design of the turn-by-turn API is based heavily on the freedesktop.org
1160 [Notifications specification](#)⁴⁰, and could share significant amounts of code with
1161 the implementation of normal (non-guidance-related) notifications.

1162 **Route guidance UI** By using a combination of the navigation route guidance
1163 API and the [points of interest API](#)⁴¹, it should be possible for an OEM to provide
1164 a route guidance UI which is separate from their main navigation application
1165 UI, but which provides sufficient information for route guidance and display of
1166 points of interest, as described in [Separate route guidance UI](#).

1167 There is no need for a separate API for this, and it is expected that if an OEM
1168 wishes to provide a route guidance UI, they can do so as a component in their
1169 navigation application bundle, or as part of the implementation of the system
1170 chrome (depending on their desired user experience). The only requirement is
1171 that only one component on the system implements the route guidance D-Bus
1172 interfaces described above.

1173 **Horizon API**

1174 The [Horizon API](#) is a shared library which applications are recommended to use
1175 when they want to present horizon data in their UI—a combination of the route
1176 list and upcoming points of interest. The library should query the [Navigation](#)
1177 [route list API](#) and [points of interest APIs](#)⁴² and aggregate the results for display
1178 in the application. It is provided as a convenience and does not implement
1179 functionality which applications could not otherwise implement themselves. The
1180 library should be usable by applications and by system components.

1181 Any OEM-preferred policy for aggregating points of interest may be imple-
1182 mented in the horizon library in order to be easily usable by all applications;
1183 but applications may choose to query the SDK route list and points of interest
1184 APIs directly to avoid this aggregation and implement their own instead.

1185 To use the horizon API, an application must have permission to query both the
1186 route list API and the points of interest API.

1187 What applications do with the horizon data once they have received it is up
1188 to the application—they may, for example, put it in a local cache and store it
1189 to prevent re-querying for historical data in future. This is entirely up to the
1190 application developer, and is out of the scope of this design.

⁴⁰<https://specifications.freedesktop.org/notification-spec/notification-spec-latest.html>

⁴¹https://www.apertis.org/concepts/archive/application/points_of_interest/

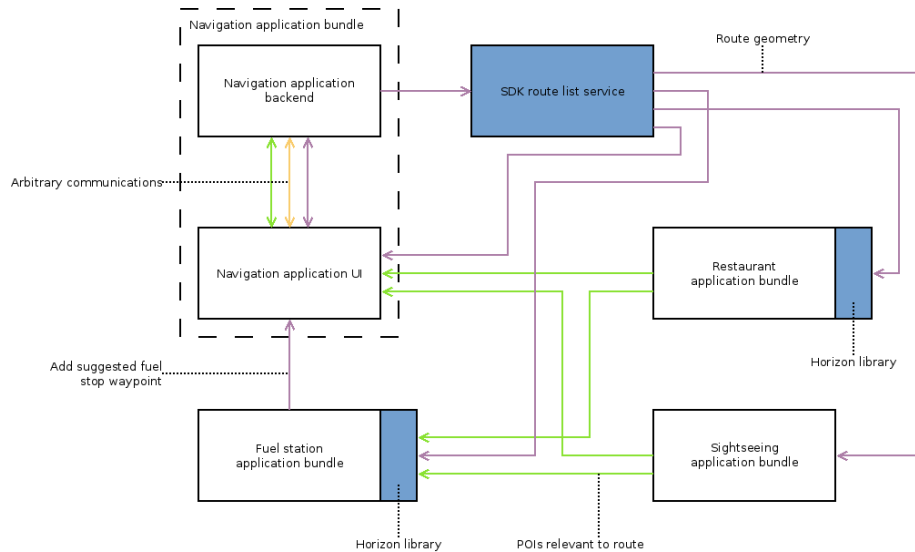
⁴²https://www.apertis.org/concepts/archive/application/points_of_interest/

1191 There was a choice between implementing the horizon API as a library or as a
1192 service. Implementing it as a service would not reduce memory consumption,
1193 as all consumers would likely still have to keep all horizon data in memory for
1194 rendering in their UI. It would not reduce CPU consumption, as aggregation
1195 of horizon data is likely to be simple (merging points of interest with the same
1196 name and location). It would double the number of IPC hops each piece of
1197 information had to make (changing from producer $\rightarrow$ consumer, to producer
1198 $\rightarrow$ horizon service $\rightarrow$ consumer). As all consumers of horizon data are likely
1199 to be interested in most points of interest, it would not significantly reduce
1200 the amount of data needing to be forwarded between producers and consumers.
1201 Hence a library is a more appropriate solution.

1202 One potential downside of using a library is that it is harder to guarantee
1203 consistency in the horizon seen by different applications —the implementation
1204 should be careful to be deterministic so that users see the same horizon in all
1205 applications using the library.

1206 See the following figure for a diagram of the flow of points of interest and route
1207 lists around the system. Key points illustrated are:

- 1208 • Producers of points of interest choose which POIs are sent to which con-
1209 sumers (there is no central POI service), though the horizon library may
1210 perform filtering or aggregation according to system policy.
- 1211 • The route list API is provided by the SDK, but uses one backend out
1212 of zero or more provided by the navigation application bundle or other
1213 bundles.
- 1214 • The navigation UI is another application with no special powers; the arbi-
1215 trary communications between its UI and backend may carry route lists,
1216 points of interest, or other data, but does not have to use the formats or
1217 APIs defined in the SDK.
- 1218 • Applications choose which waypoints to send to via the [navigation routing
1219 API][Navigation routing] to be added into the route —this might result in
1220 the user being prompted ‘do you want to add this waypoint into your route’
1221 , or they might be added unconditionally. For example, the fuel station
1222 application bundle may add a fuel stop waypoint to the route, which is
1223 then exposed in the route list API if the navigation application accepts it.
- 1224 • The navigation UI does not necessarily use information from the route list
1225 API to build its UI (although such information may contribute).
- 1226 • If no navigation bundle is installed, the SDK route list service still exists,
1227 it just returns no data.



1228

1229 Forward and reverse geocoding

1230 We recommend that [Geocode-glib](https://gitlab.gnome.org/GNOME/geocode-glib)⁴³ is used as the SDK geocoding API. Geocode-glib
 1231 is currently hard-coded to use GNOME's Nominatim service; it would need
 1232 to be modified to support multiple backends, such as an Apertis-specific [Nominatim](http://wiki.openstreetmap.org/wiki/Nominatim)
 1233 [server](http://wiki.openstreetmap.org/wiki/Nominatim)⁴⁴, or a geocoding service from the automotive backend. It only
 1234 needs to support querying one backend at a time; results do not need to be
 1235 aggregated. Backends should be loadable and unloadable at runtime.

1236 The geocode-glib API supports forward and reverse geocoding, and supports
 1237 limiting search results to a given bounding box.

1238 **Geocoding backends** On an integrated system, geocoding services will be
 1239 provided by the automotive domain, via inter-domain communications (See the
 1240 Inter-domain Communications design). On the Apertis SDK, an internet con-
 1241 nection is always assumed to be available, so geocoding may be performed using
 1242 an online Nominatim backend. In both cases, geocode-glib would form the SDK
 1243 API used by applications. Another alternative backend, which could be written
 1244 and used by OEMs instead of an automotive backend, would be a [Google Maps](https://developers.google.com/maps/documentation/geocoding/intro)
 1245 [geocoding API](https://developers.google.com/maps/documentation/geocoding/intro)⁴⁵ backend which runs in the IVI domain.

1246 Although it is tempting to do so, points of interest should *not* be fed into
 1247 geocode-glib via another new backend, as the semantics of points of interest (a
 1248 large number of points spread in a radius around a focal point) do not match
 1249 the semantics of reverse geocoding (finding the single most relevant address to

⁴³<https://gitlab.gnome.org/GNOME/geocode-glib>

⁴⁴<http://wiki.openstreetmap.org/wiki/Nominatim>

⁴⁵<https://developers.google.com/maps/documentation/geocoding/intro>

1250 describe a given latitude and longitude). Points of interest should be queried
1251 separately using a library provided by the [Points of Interest design](#)⁴⁶.

1252 **Localisation of geocoding** Nominatim supports exporting localised place
1253 names from OpenStreetMap, but geocode-glib does not currently expose that
1254 data in its query results. It would need to be modified to explicitly expose locale
1255 data about results.

1256 It does currently support supplying the locale of input queries using the language
1257 parameter to [geocode_forward_new_for_params](#)⁴⁷.

1258 **Address completion**

1259 Address completion is a complex topic, both because it requires being able to
1260 parse partially-complete addresses, and because the datasets required to answer
1261 completion queries are large.

1262 Nominatim does not provide dedicated address completion services, but it is
1263 possible to implement them in a separate web service using a filtered version
1264 of the OpenStreetMap database data. An example is available as [Photon](#)⁴⁸.
1265 Google also provides a paid-for web service for address completion: the [Google
1266 Places Web API](#)⁴⁹.

1267 As address completion is a special form of forward geocoding (i.e. forward
1268 geocoding operating on partial input), it should be provided as part of the
1269 geocoding service, and by the same backends which provide the geocoding func-
1270 tionality.

1271 If Nominatim (via geocode-glib) is found to be insufficient for address completion
1272 in the SDK, an Apertis-hosted Photon instance could be set up, and a Photon
1273 backend added to the geocoding service.

1274 In target devices, address completion should be provided by the automotive
1275 backend, or not provided at all if the backend does not implement it.

1276 The address completion API should be an extension to the existing geocode-glib
1277 API for forward geocoding. There must be a way for it to signal there are no
1278 known results. Results should be ranked by relevance or likelihood of a match,
1279 and should include information about which part of the search term caused the
1280 match (if available), to allow that to be highlighted in the widget.

1281 A separate library (which has a dependency on the SDK widget library) should
1282 provide a text entry widget which implements address completion using the
1283 API on the geocoding service, so that application developers do not have to
1284 reimplement it themselves. This could be similar to [GtkSearchEntry](#)⁵⁰ (but

⁴⁶https://www.apertis.org/concepts/archive/application/points_of_interest/

⁴⁷<https://gitlab.gnome.org/GNOME/geocode-glib>

⁴⁸<https://photon.komoot.io/>

⁴⁹<https://developers.google.com/places/web-service/autocomplete>

⁵⁰<https://developer.gnome.org/gtk3/stable/GtkSearchEntry.html>

1285 using the Apertis UI toolkit).

1286 **Address completion backends** As the backends for the address completion
1287 service (i.e. the geocoding backends) may access sensitive data to answer queries,
1288 they must be able to check the permissions of the application which originated
1289 the query. If the application does not have appropriate permissions, they must
1290 not return sensitive results.

1291 For example, a backend could be added which resolves ‘home’ to the user’s home
1292 address, ‘work’ to their work address, and ‘here’ to the vehicle’s current location.
1293 In order to maintain the confidentiality of this data, applications must have
1294 permission to access the system address book (containing the home and work
1295 addresses), and permission to access the vehicle’s location (see **Location secu-**
1296 **rity**). If the application does not have appropriate permissions, the backend
1297 must not return results for those queries.

1298 As normal geocoding operation is not sensitive (the results do not differ depend-
1299 ing on who’s submitting a query), backends which require permissions like this
1300 must be implemented in a separate security domain, i.e. as a separate process
1301 which communicates with geocode-glib via D-Bus. They can get the request-
1302 ing application’s unforgeable identifier from D-Bus requests in order to check
1303 permissions.

1304 **Geofencing**

1305 We recommend that GeoClue is modified upstream to implement geofencing of
1306 arbitrary regions, meaning that geofencing becomes part of **Geolocation**. Signals
1307 on entering or exiting a geofenced area should be emitted as a D-Bus signal
1308 which the application subscribes to. Delivery of signals to bundles which are
1309 not currently running may cause activation of that application.

1310 This is intended to be provided by a system service for activation
1311 of applications based on subscribed signals; the design is tracked in
1312 <https://phabricator.apertis.org/T640>.

1313 The geofencing service should include a database of country borders, and provide
1314 a convenience API for adding a geofence for a particular country (or, for example,
1315 the current country and its neighbours). This would load the polygon for the
1316 country’s border and add it as a geofence as normal.

1317 The geofencing service must be available as the well-known name org.freedesktop.GeoClue2.Geofencing
1318 on D-Bus. It must export the following object:

```
1319 /org/freedesktop/GeoClue2/Geofencing/Manager {  
1320     /* Adds a geofence as a latitude, longitude and radius (in metres), and  
1321      * returns the ID of that geofence. The dwell_time (in seconds) gives  
1322      * how long the vehicle must dwell inside the geofence to be signalled  
1323      * as such by the GeofenceActivity signal. */  
1324     method AddGeofenceByRadius(in (dd) location, in u radius, in u dwell_time, out u id)
```

```

1325
1326 /* Adds a geofence by taking an ordered list of latitude and longitude
1327  * points which form a polygon for the geofence's boundary. */
1328 method AddGeofenceByPolygon(in a(dd) points, in u dwell\_time, out u id)
1329
1330 /* Remove some geofences. */
1331 method RemoveGeofences(in au ids)
1332
1333 /* Return a (potentially empty) list of the IDs of the geofences the
1334  * vehicle is currently dwelling in. */
1335 method GetDwellingGeofences(out au ids)
1336
1337 /* Signal emitted every time a geofence is entered or exited, which
1338  * lists the IDs of all geofences entered and exited since the previous
1339  * signal emission, plus all the geofences the vehicle is currently
1340  * dwelling inside. */
1341 signal GeofenceActivity(out au entered, out au dwelling, out au exited)
1342 }

```

1343 IDs are global and opaque —applications cannot find the area referenced by
1344 a particular geofence ID. A geofence may only be removed by the application
1345 which added it. Currently, the GeofenceActivity signal is received by all appli-
1346 cations, but they cannot dereference the opaque geofence identifiers for other
1347 applications. In future, if application-level containerisation is implemented, this
1348 signal will only be filtered per application.

1349 We have informally discussed the possibility of adding geofencing with the Geo-
1350 Clue developers, and they are in favour of the idea.

1351 **Location security**

1352 libchamplain is a rendering library, and does not give access to sensitive infor-
1353 mation.

1354 geocode-glib is a thin interface to a web service, and does not give access to
1355 sensitive information. All web service requests must be secured with best web
1356 security practices, such as correct use of HTTPS, and sending a minimum of
1357 identifiable information to the web service.

1358 GeoClue provides access to sensitive information about the vehicle's location.
1359 It currently allows limiting the accuracy provided to applications as [specified
1360 by the user](#)⁵¹; this could be extended to implement a policy determined by the
1361 capabilities requested in the application's manifest.

1362 Similarly for GeoClue's geofencing feature, when it is added —clients have
1363 separated access to its D-Bus API to allow them to be signalled at different

⁵¹<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-property-org-freedesktop-GeoClue2-Client-RequestedAccuracyLevel>

1364 accuracies and rates. This applies to navigation routing as well, as it may
1365 provide feedback to applications about progress along a route, which exposes
1366 information about the vehicle's location.

1367 Application bundles asking for fine-grained location data should be subjected
1368 to closer review when submitted to the Apertis application store.

1369 **Systemic security**

1370 As the geo-features can source information from application bundles, they form
1371 part of the security boundary around application bundles.

1372 In order to avoid denial of service attacks from an application bundle which
1373 emits too much data as, for example, a general point of interest stream, the
1374 system should rate limit such streams in time (number of POIs per unit time)
1375 and space (number of POIs per map area).

1376 Location updates emitted by GeoClue must be rate limited between the mini-
1377 mum and maximum distance and time limits set by each client. These limits
1378 must be checked to ensure that a client is not requesting updates too frequently.

1379 For the components in the system which provide access to sensitive information
1380 (the vehicle's location), a security boundary needs to be defined between them
1381 and application bundles. Geolocation, navigation route lists, route guidance
1382 and geofencing are the sensitive APIs —these are all implemented as services, so
1383 the D-Bus APIs for those services form the security boundary. In order to use
1384 any of these services, an application must have the appropriate permission in its
1385 manifest. Address completion as a whole is not treated as sensitive, but some
1386 of its backends may be sensitive, and perform their own checks according to
1387 other permissions (which may include those below). The following permissions
1388 are suggested:

- 1389 • *content-hand-over*: Required to use the content hand-over API for setting
1390 a new [navigation route][Navigation routing].
- 1391 • *location*: Required to access the geolocation, geofencing, or navigation
1392 route list services.
- 1393 • *navigation-route*: Required to access the navigation route list services
1394 (note that the *location* permission is also required).
- 1395 • *navigation-guidance*: Required to access the navigation guidance services
1396 (note that the *location* permission is also required).

1397 The libchamplain layers which applications can use (see [here][Route list layer
1398 on the 2D map] and [here][Vehicle location layer on the 2D map]) are treated
1399 as application code which is using the relevant services (geolocation and navi-
1400 gation route guidance), and hence the *location* or *location* and *navigation-route*
1401 permissions are required to use them. Similarly for the horizon library.

Any service which accepts data from applications, such as points of interest or waypoints to add into the route list, must check that the user has not disabled that application from providing data. If the user has disabled it, the data must be ignored and an error code returned to the application if the API allows it, to indicate to the application that sharing was prevented.

Testability

There are several components to testability of geo-functionality. Each of the components of this system need to be testable as they are developed, and as new backends are developed for them (including testing the backends themselves). Separately, application developers need to be able to test their applications' behaviour in a variety of simulated geo-situations.

Testing geo-functionality Each of the services must have unit tests implemented. If the service has backends, a mock backend should be written which exposes the backend API over D-Bus, and integration tests for that service can then feed mock data in via D-Bus and check that the core of the service behaves correctly.

Just like how libfolks'dummy backend is used in its unit tests, <https://git.gnome.org/browse/folks/tree/backends/dummy>

Testing backends Each service which has backends must implement a basic compliance test suite for its backends, which will load a specified backend and check that its public API behaves as expected. The default backends will further be tested as part of the integration tests for the entire operating system.

Testing applications In order to allow application developers to test their applications with mock data from the geo-services, there must be an emulator program available in the SDK which uses the mock backend for each service to feed mock data to the application for testing.

For example, the emulator program could display a map and allow the developer to select the vehicle's current location and the accuracy of that location. It would then feed this data to the mock backend of the geolocation service, which would pass it to the core of the geolocation service as if it were the vehicle's real location. This would then be passed to the application under test as the vehicle's location, by the SDK geolocation API.

The emulator could additionally allow drawing a route on the map, which it would then send to the mock backend for the route list API as the current route —this would then be passed to the application under test as the vehicle's current route.

This means that the API of the mock backend for each service must be stable and well defined.

1440 Requirements

1441 This design fulfils the following requirements:

- 1442 • **Geolocation API** —use GeoClue
- 1443 • **Geolocation service supports signals** —use GeoClue; augment its signals
- 1444 • **Geolocation implements caching** —to be added to GeoClue
- 1445 • **Geolocation supports backends** —GeoClue supports backends
- 1446 • **Navigation routing API** —use content hand-over design, passing a nav URI
- 1447 to the navigation application
- 1448 • **Navigation routing supports different navigation applications** —content
- 1449 hand-over supports setting different applications as the handlers for the
- 1450 nav URI scheme
- 1451 • **Navigation route list API** —new D-Bus API which is implemented by the
- 1452 navigation application backend
- 1453 • **Navigation route guidance API** —new D-Bus API implemented by the sys-
- 1454 tem UI (i.e. the compositor) which is called by the navigation application
- 1455 • **Type-ahead search and address completion supports backends** —imple-
- 1456 mented as part of geocoding, to be added to geocode-glib
- 1457 • **Geocoding supports backends** —to be added to geocode-glib
- 1458 • [SDK has default implementations for all backends][SDK has default im-
- 1459 plementations for all backends] —Gypsy or a mock backend for geolocation;
- 1460 custom online Nominatim server for geocoding; online OpenStreetMap for
- 1461 2D maps; libnavit for 3D maps, **subject to further evaluation**; cus-
- 1462 tom mock backend for navigation route list; custom online Nominatim or
- 1463 Photon server for address completion
- 1464 • **SDK APIs do not vary with backend** —GeoClue API for geolocation;
- 1465 geocode-glib for geocoding; libchamplain for 2D maps; libnavit for 3D
- 1466 maps, **subject to further evaluation**; content hand-over API for navi-
- 1467 gation routing; new API for navigation route lists and guidance; new API
- 1468 extensions to geocode-glib for address completion
- 1469 • **Third-party navigation applications can be used as backends** —backends
- 1470 are implemented as loadable libraries installed by the navigation applica-
- 1471 tion
- 1472 • **Backends operate asynchronously** —backends are implemented over D-Bus
- 1473 so are inherently asynchronous
- 1474 • **2D map rendering API** —use libchamplain with a local or remote tile store
- 1475 • **2.5D or 3D map rendering API** —use libnavit, **subject to further eval-**
- 1476 **uation**

- 1477 • **Reverse geocoding API** —use geocode-glib
- 1478 • **Forward geocoding API** —use geocode-glib
- 1479 • **Type-ahead search and address completion API** —to be added to geocode-
- 1480 glib
- 1481 • **Geofencing API** —to be implemented as a new feature in GeoClue
- 1482 • **Geofencing service can wake up applications** —to be implemented as a new
- 1483 feature in GeoClue
- 1484 • **Geofencing API signals on national borders** —to be added as a data set in
- 1485 GeoClue
- 1486 • **Geocoding API must be locale aware** —to be added to geocode-glib to
- 1487 expose existing OpenStreetMap localised data
- 1488 • **Geolocation provides error bounds** —GeoClue provides accuracy informa-
- 1489 tion, but it needs augmenting
- 1490 • **Geolocation implements dead reckoning** —to be added to GeoClue
- 1491 • **Geolocation uses navigation and sensor data if available** —to be added as
- 1492 another backend to GeoClue
- 1493 • **General points of interest streams are queryable** —to be designed and
- 1494 implemented as part of the [points of interest design](#)⁵²
- 1495 • **Location information requires permissions to access** —to be implemented
- 1496 as manifest permissions for application bundles
- 1497 • **Rate limiting of general point of interest streams** —security boundary im-
- 1498 plemented as D-Bus API boundary; rate limiting applied on signal emis-
- 1499 sion and processed general point of interest streams
- 1500 • **Application provided data requires permissions to create** —all geo-services
- 1501 must check settings before accepting data from applications

1502 Suggested roadmap

1503 As the SDK APIs for geo-features are, for the most part, provided by FOSS
 1504 components which are available already, the initial deployment of geo-features
 1505 requires GeoClue, geocode-glib and libchamplain to be packaged for the distri-
 1506 bution, if they are not already.

1507 The second phase would require modification of these packages to implement
 1508 missing features and implement additional backends. This can happen once the
 1509 initial packaging is complete, as the packages fulfil most of Apertis' requirements
 1510 in their current state. This requires the address completion APIs to be added to
 1511 to geocode-glib, and the geofencing APIs to be added to GeoClue, amongst other

⁵²https://www.apertis.org/concepts/archive/application/points_of_interest/

1512 changes. These API additions should be prioritised over other work, so that
1513 application development (and documentation about application development)
1514 can begin.

1515 This second phase includes modifying the packages to be container-friendly, so
1516 that they can be used by compartmentalised apps without leaking sensitive data
1517 from one app to another. This requires further in-depth design work, but should
1518 require fairly self-contained changes.

1519 The second phase also includes writing the new services, such as the **Navigation**
1520 **route list API**, the **Navigation route guidance API** and the **Horizon API**.

1521 Open questions

- 1522 1. What review checks should be performed on application bundles which
1523 request permissions for location data?
- 1524 2. Is it possible to use NavIt as a stand-alone 2.5D or 3D map widget?

1525 Summary of recommendations

1526 As discussed in the above sections the recommendations are:

- 1527 • Packaging and using libchamplain for 2D map display.
- 1528 • Adding a route list layer for libchamplain.
- 1529 • Adding a vehicle location layer for libchamplain.
- 1530 • Packaging and using libnavit for 3D map display, **subject to further**
1531 **investigation**.
- 1532 • Packaging and using GeoClue for geolocation. It needs measurement times,
1533 cached location support, dead-reckoning and more measurements and un-
1534 certainty bounds to be added upstream.
- 1535 • Adding minimum and maximum update periods for clients to upstream
1536 GeoClue, alongside the existing distance threshold API for location update
1537 signals.
- 1538 • Adding a new navigation application backend to GeoClue to implement
1539 snap-to-road refinement of its location.
- 1540 • Adding a new mock backend to GeoClue for the SDK.
- 1541 • Implementing a library for parsing place and nav URIs and formalising the
1542 specifications so they may be used by third-party navigation applications.
- 1543 • Adding support for place and nav URIs to the content hand-over service
1544 (Didcot) and adding support for a default navigation application.
- 1545 • Implementing a navigation route list service with support for loadable
1546 backends, including a mock backend for the SDK.

- 1547 • Implementing a navigation route guidance API in the system compositor.
- 1548 • Implementing a horizon library for aggregating and filtering points of in-
- 1549 interest with the route list, suitable for use by applications.
- 1550 • Packaging and using geocode-glib for forward and reverse geocoding. It
- 1551 needs support for exposing localised place names to be added upstream.
- 1552 • Adding support for loadable backends to geocode-glib, including a mock
- 1553 backend for the SDK.
- 1554 • Auditing geocode-glib to ensure it maintains data privacy by, for example,
- 1555 using TLS for all requests and correctly checking certificates.
- 1556 • Auditing GeoClue to ensure it maintains data privacy by, for example,
- 1557 using TLS for all requests and correctly checking certificates.
- 1558 • Implementing an address completion API in geocode-glib and implement-
- 1559 ing it in the Nominatim and mock backends.
- 1560 • Implementing an address completion widget based on the address comple-
- 1561 tion API.
- 1562 • Implementing a geofencing API in upstream GeoClue.
- 1563 • Integrating the geo-services with the app service proxy to apply access
- 1564 control rules to whether applications can communicate with it to retrieve
- 1565 potentially sensitive location or navigation data. Only permit this if the
- 1566 appropriate permissions have been set on the application bundle's mani-
- 1567 fest.
- 1568 • Implementing an emulator program in the SDK for controlling mock data
- 1569 sent by the SDK geo-APIs to applications under test.
- 1570 • Providing integration tests for all geo-functionality.

1571 **Appendix: Recommendations for third-party navigation**

1572 **applications**

1573 While this design explicitly does not cover providing SDK APIs purely to be
 1574 used in implementing navigation applications, various recommendations have
 1575 been considered for what navigation applications probably should do. These
 1576 are non-normative, provided as suggestions only.

- 1577 • Support different types of vehicle —the system could be deployed in a car,
- 1578 or a motorbike, or a HGV, for example. Different roads are available for
- 1579 use by these different vehicles.
- 1580 • Support calculating routes between the start, waypoints and destination
- 1581 using public transport, in addition to the road network. For example,
- 1582 this could be used to provide a comparison against the car route; or to

1583 incorporate public transport schemes such as park-and-ride into route sug-
1584 gestions.

- 1585 • Support audio turn-by-turn navigation, reading out instructions to the
1586 driver as they are approached. This allows the driver to avoid looking at
1587 the IVI screen to see the map, allowing them to focus on the road.
- 1588 • Route guidance must continue even if another application takes the fore-
1589 ground focus on the IVI system, meaning the guidance system must be
1590 implemented as an agent.
- 1591 • Support different optimisation strategies for route planning: minimal
1592 travel time, scenic route, minimal cost (for fuel), etc.
- 1593 • In order to match the driver's view out of the windscreen, the map dis-
1594 played when navigating should be a 3D projection to better emphasise
1595 navigational input which is coming up sooner (for example, the nearest
1596 junction or road signs).
- 1597 • Support changing the destination mid-journey and calculating a new
1598 route.
- 1599 • Support navigating via zero or more waypoints before reaching the final
1600 destination. Support adding new waypoints mid-journey.
- 1601 • Provide the driver with up-to-date information about the estimated travel
1602 time and distance left on their route, plus the total elapsed travel time
1603 since starting the route. This allows the driver to work out when to take
1604 rest breaks from driving.
- 1605 • Detect when the driver has taken a wrong turning while navigating, and
1606 recalculate the route to bring them back on-route to their destination,
1607 reoptimising the route for minimal travel time (or some other criterion)
1608 from their new location. The new route might be radically different from
1609 the old one if this is more optimal. The route recalculation should happen
1610 quickly (on the order of five seconds) so that the driver gets updated
1611 routing information quickly.
- 1612 • Support recalculating and potentially changing a navigation route if traffic
1613 conditions change and make the original route take significantly longer
1614 than an alternative. There must be some form of hysteresis on these
1615 recalculations so that two routes which have very similar estimated travel
1616 times, but which keep alternating as the fastest, do not continually replace
1617 each other as the suggested route. The application may ask or inform the
1618 driver about route recalculations, as the driver may be able to assess and
1619 predict the traffic conditions better than the application.
- 1620 • Provide information to the driver about the roads the vehicle is travelling
1621 along or heading towards and going to turn on to in future, such as the road
1622 name and number, and major towns or cities the road leads to. This allows

1623 the driver to match up the turn-by-turn navigation directions with on-
 1624 road signage. (This is often known as route guidance or driver assistance
 1625 information.)

- 1626 • If the driver takes a rest break during a navigation route, and turns the
 1627 vehicle off, the application must give the driver the option to resume the
 1628 navigation route when the vehicle is turned on again. The route must
 1629 be recalculated from the vehicle's current location to ensure the resumed
 1630 route is still optimal for current traffic conditions.
- 1631 • Support cancelling the navigation when the vehicle is turned on again, at
 1632 which point all navigation and turn-by-turn directions stop.
- 1633 • If driven abroad, the navigation application should provide locale-sensitive
 1634 navigation information, such as speed limits in the local units, and road
 1635 descriptions which match the local signage conventions.
- 1636 • Support feeding geolocation data in to the system geolocation service, if
 1637 such data may be more precise than the raw GPS positions; for example,
 1638 if it can be snapped to the nearest road.
- 1639 • Support feeding other geo-information to the other geo-services, such as
 1640 answering geocoding queries or performing geo-fencing. Support being a
 1641 full replacement for the inbuilt navigation application and all the SDK
 1642 services it provides.
- 1643 • Query the system POI API for restaurants and toilets at times and fre-
 1644 quencies suitable for recommending food or toilet breaks to the driver
 1645 appropriately. Allow the driver to dismiss or disable these, or to change
 1646 the intervals. Do not recommend a break if the journey is predicted to end
 1647 soon. Launch the application which provided the POI if the user clicks
 1648 on a POI in the navigation application, so that they can perform further
 1649 actions on that POI (for example, if it's a restaurant, they could reserve a
 1650 table). When POIs are displayed in the navigation application, they can
 1651 be rendered as desired by the navigation application developer; when they
 1652 are displayed in other applications, they are rendered as desired by those
 1653 applications' developers.

1654 **Appendix: place URI scheme**

1655 The place URI scheme is a non-standard URI scheme suggested to be used
 1656 within Apertis for identifying a particular place, including a variety of redundant
 1657 forms of information about the place, rather than relying on a single piece of
 1658 information such as a latitude and longitude. To refer to a location by latitude
 1659 and longitude *only*, use the standard [geo URI scheme](http://tools.ietf.org/html/rfc5870)⁵³.

1660 A place URI is a non-empty human-readable string describing the location,
 1661 followed by zero or more key-value pairs providing additional information. The

⁵³<http://tools.ietf.org/html/rfc5870>

key-value pairs are provided as parameters as defined in [RFC 5870], i.e. each parameter is separated by a semicolon, keys and values are separated by an equals sign, and percent-encoding is used to encode reserved characters.

The location string must be of the format `1\*paramchar`, as defined in RFC 5870. All non-ASCII characters in the string must be [percent-encoded](#)⁵⁴, and implementations must interpret the decoded string as [UTF-8](#)⁵⁵.

Implementations may support the following parameters, and must ignore unrecognised parameters, as more may be added in future. All non-ASCII characters in parameter keys and values must be percent-encoded, and implementations must interpret the decoded strings as UTF-8. The semicolon and equals sign separators must not be percent-encoded. The ordering of parameters does not matter, unless otherwise specified. Each parameter may appear zero or one times, unless otherwise specified.

- `location`: the latitude and longitude of the place, as a geo URI (*with the geo: scheme prefix*)
- `postal-code`: the postal code for the place, in the country's postal code format
- `country`: the [ISO 3166-1 alpha-2](#)⁵⁶ country code
- `region`: the human-readable name of a large administrative region of the country, such as a state, province or county
- `locality`: the human-readable name of the locality, such as a town or city
- `area`: the human-readable name of an area smaller than the locality, such as a neighbourhood, suburb or village
- `street`: the human-readable street name for the place
- `building`: the human-readable name or number of a house or building within the street
- `formatted-address`: a human-readable formatted version of the entire address, intended to be displayed in the UI rather than machine-parsed; implementations may omit this if it is identical to the location string, but it will often be longer to represent the location unambiguously (the location string may be ambiguous or incomplete as it is typically user input)

Examples

This section is non-normative. Each example is given as a fully encoded string, followed by it split up into its un-encoded components.

- `place:Paris`

⁵⁴<http://tools.ietf.org/html/rfc5870#section-3.5>

⁵⁵<http://www.unicode.org/versions/Unicode6.0.0/ch03.pdf>

⁵⁶http://www.iso.org/iso/country_codes.htm

```

1697         – Location string: Paris
1698         – No parameters
1699     • place:Paris;location=geo%3A48.8567%2C2.3508;country=FR;formatted-
1700       address=Paris%2C%20France
1701         – Location string: Paris
1702         – Parameters:
1703             * location: geo:48.8567,2.3508
1704             * country: FR
1705             * formatted-address: Paris, France
1706     • place:K%C3%B6nigsstieg%20104%2C%2037081%20G%C3%B6ttingen;location=geo%3A51.540060%2C9.911850;country=
1707       code=37081;street=K%C3%B6nigsstieg;building=104;formatted-address=K%C3%B6nigsstieg%20104%2C%2037081%2
1708         – Location string: Königsstieg 104, 37081 Göttingen
1709         – Parameters:
1710             * location: geo:51.540060,9.911850
1711             * country: DE
1712             * locality: Göttingen
1713             * postal-code: 37081
1714             * street: Königsstieg
1715             * building: 104
1716             * formatted-address: Königsstieg 104, 37081 Göttingen, Germany
1717     • place:CN Tower;location=geo%3A43.6426%2C-79.3871;formatted-address=301%20Front%20St%20W%2C%20Toronto%2
1718         – Location string: CN Tower
1719         – Parameters:
1720             * location: geo:43.6426,-79.3871
1721             * formatted-address: 301 Front St W, Toronto, ON M5V 2T6,
1722               Canada

```

1723 Appendix: nav URI scheme

1724 The nav URI scheme is a non-standard URI scheme suggested to be used within
 1725 Apertis for identifying a navigation route, including its destination, intermediate
 1726 destinations (waypoints) and points or areas to route via but which are not
 1727 named destinations (via-points). Each point or area may be provided as a place
 1728 or a location.

1729 A nav URI is a non-empty destination place, followed by zero or more key-
 1730 value pairs providing additional information. The key-value pairs are provided
 1731 as parameters as defined in [RFC 5870], i.e. each parameter is separated by a
 1732 semicolon, keys and values are separated by an equals sign, and percent-encoding
 1733 is used to encode reserved characters.

1734 The destination place must be provided as [Appendix: place URI scheme](#) (*with*
 1735 the `place:` URI prefix), or as a geo URI (*with* the `geo:` URI prefix); and must be

1736 encoded in the format `1\*paramchar`, as defined in RFC 5870; i.e. all non-ASCII
1737 and reserved characters in the string must be percent-encoded.

1738 Implementations may support the following parameters, and must ignore un-
1739 recognised parameters, as more may be added in future. All non-ASCII char-
1740 acters in parameter keys and values must be percent-encoded, and implementa-
1741 tions must interpret the decoded strings as UTF-8. The semicolon and equals
1742 sign separators must not be percent-encoded. The ordering of parameters does
1743 not matter, unless otherwise specified. Each parameter may appear zero or one
1744 times, unless otherwise specified.

- 1745 • `description`: a human-readable description of the route, intended to be
1746 displayed in the UI rather than machine-parsed
- 1747 • `way`: a named intermediate destination, as a place URI (*with* the `place`:
1748 scheme prefix) or as a geo URI (*with* the `geo`: scheme prefix); these pa-
1749 rameters are order-dependent (see below)
- 1750 • `via`: a non-named intermediate routing point, as a place URI (*with* the
1751 `place`: scheme prefix) or as a geo URI (*with* the `geo`: scheme prefix); these
1752 parameters are order-dependent (see below)

1753 The `way` and `via` parameters are order-dependent: they will be added to the
1754 route in the order they appear in the nav URI. Way-places and via-places may
1755 be interleaved —they form a single route. The destination place always forms
1756 the final point in this route. The `way` and `via` parameters may each appear zero
1757 or more times.

1758 Additional routing specific parameters can be added. If those parameters are
1759 not provided, the default value is left to the routing engine. It may be different
1760 for each type of vehicle, or due to other logic in the routing engine. Many param-
1761 eters represent a single value; for example, it is not meaningful to specify both
1762 `optimize=fastest` and `optimize=shortest`. URIs with multiple values for a single-
1763 valued parameter, for example `place=Home;optimize=fastest;optimize=shortest`,
1764 should be treated as though that parameter was not present. Apertis does not
1765 currently define multi-valued preferences. All values should be specified at most
1766 once. However, OEMs may define and use their own multi-valued properties.
1767 The naming convention is defined below. Boolean values can be specified as `1`
1768 and `0` (or equivalently `true` and `false` while being case insensitive). Other values
1769 are considered invalid.

- 1770 • `vehicle`: vehicle for which the route should be calculated. Apertis defines
1771 `car`, `walk`, and `bike`. `car` routes are optimized for being used by car. `walk`
1772 routes are optimized for walking. `bicycle` routes are optimized for being
1773 ridden by bicycles.
- 1774 • `optimize`: Optimizes route calculation towards a set of criteria. Apertis
1775 defines `fastest`, and `shortest` criteria. `fastest` routes are optimized to
1776 minimize travel duration. `shortest` routes are optimized to minimize travel
1777 distance.

- 1778 • `avoid-tolls`: Boolean. If true, the calculated route should avoid tolls. If
1779 usage can't be avoided, the handler application is responsible for informing
1780 the user.
- 1781 • `avoid-motorways`: Boolean. If true, the generated route should avoid mo-
1782 torways. If usage can't be avoided, the handler application is responsible
1783 for informing the user.
- 1784 • `avoid-ferries`: Boolean. If true, the generated route should avoid ferries.
1785 If usage can't be avoided, the handler application is responsible for inform-
1786 ing the user.

1787 Additionally, vendor specific parameters can be provided for vendor specific
1788 features. To avoid contradictory definitions for the same parameter in different
1789 implementations, vendor-specific parameters must be named in the form `x-`
1790 `vendorname-paramname`, similar to [the convention for extending .desktop files](https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending)⁵⁷.
1791 Note that unlike keys in `.desktop` files, parameter names in `nav:` URIs are not
1792 case-sensitive: consistently using lower-case is encouraged. For instance, one
1793 of the [examples] below assumes that a manufacturer has introduced a boolean
1794 option `x-batmobile-avoid-detection`, and a string-valued parameter `x-batmobile-`
1795 `auto-pilot-mode`.

1796 Examples

1797 This section is non-normative. Each example is given as a fully encoded string,
1798 followed by it split up into its un-encoded components.

1799 **Example 1** `nav:place%3AKings%2520Cross%2520station%252C%2520London%3Blocality%3DLondon%3Bpostal-`
1800 `code%3DN19AL`

- 1801 • Destination place:
 - 1802 – Location string: Kings Cross station, London
 - 1803 – Parameters:
 - 1804 * `locality`: London
 - 1805 * `postal-code`: N19AL

1806 **Example 2** `nav:geo%3A51.531621%2C-0.124372`

- 1807 • Destination place: `geo:51.531621,-0.124372`

1808 **Example 3** `nav:place%3ABullpot%2520Farm%3Blocation%3Dgeo%253A54.227602%252C-`
1809 `2.517940;way=place%3ABirmingham%2520New%2520Street%2520station%3Blocation%3Dgeo%253A52.477620%252C-`
1810 `1.897904;via=place%3AHornby%3Blocation%3Dgeo%253A54.112245%252C-2.636527%253Bu%253D2000;way=place%3AInglesp`
1811 `code%3DLA63EB%3Bcountry%3DGB`

- 1812 • Destination place:

⁵⁷<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending>

```

1813     - Location string: Bullpot Farm
1814     - Parameters:
1815         * location: geo:54.227602,-2.517940
1816     • Parameters:
1817         - way:
1818             * Location string: Birmingham New Street station
1819             * Parameters:
1820                 · location: geo:52.477620,-1.897904
1821         - via:
1822             * Location string: Hornby
1823             * Parameters:
1824                 · location: geo:54.112245,-2.636527;u=2000
1825         - way:
1826             * Location string: Inglesport, Ingleton
1827             * Parameters:
1828                 · street: The Square
1829                 · building: 11
1830                 · locality: Ingleton
1831                 · postal-code: LA63EB
1832                 · country: GB

```

1833 **Example 4** nav:geo%3A51.531621%2C-0.124372;vehicle=walk;optimize=shortest

```

1834     • Destination place: geo:51.531621,-0.124372
1835     • Parameters:
1836         - vehicle: walk
1837         - optimize: shortest

```

1838 **Example 5** nav:geo%3A51.531621%2C-0.124372;vehicle=car;avoid-tolls=false;x-
1839 batmobile-auto-pilot-mode=full;x-batmobile-avoid-detection=true

```

1840     • Destination place: geo:51.531621,-0.124372
1841     • Parameters:
1842         - vehicle: car
1843         - avoid-tolls: false
1844         - x-batmobile-avoid-detection: true
1845         - x-batmobile-auto-pilot-mode: full

```

1846 **Example 6** nav:place:Cambridge;x-myvendor-avoid-road=A14;x-myvendor-
1847 avoid-road=M11

```

1848     • Destination place: Cambridge
1849     • Parameters:
1850         - x-myvendor-avoid-road: multi-valued: A14, M11

```