



Geolocation and navigation

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

46	POI data	11
47	Beacons	11
48	Loading map tiles from a backend	11
49	SDK APIs for third-party navigation applications	12
50	Requirements	12
51	Geolocation API	12
52	Geolocation service supports signals	12
53	Geolocation implements caching	13
54	Geolocation supports backends	13
55	Navigation routing API	13
56	Navigation routing supports different navigation applications	13
57	Navigation route list API	14
58	Navigation route guidance API	14
59	Type-ahead search and address completion supports backends	14
60	Geocoding supports backends	14
61	SDK has default implementations for all backends	15
62	SDK APIs do not vary with backend	15
63	Third-party navigation applications can be used as backends	15
64	Backends operate asynchronously	15
65	2D map rendering API	15
66	2.5D or 3D map rendering API	16
67	Reverse geocoding API	16
68	Forward geocoding API	16
69	Type-ahead search and address completion API	16
70	Geofencing API	17
71	Geofencing service can wake up applications	17
72	Geofencing API signals on national borders	17
73	Geocoding API must be locale-aware	17
74	Geolocation provides error bounds	17
75	Geolocation implements dead-reckoning	18
76	Geolocation uses navigation and sensor data if available	18
77	General points of interest streams are queryable	18
78	Location information requires permissions to access	18
79	Rate limiting of general point of interest streams	19
80	Application-provided data requires permissions to create	19
81	Existing geo systems	19
82	W3C Geolocation API	19
83	Android platform location API	20
84	Google Location Services API for Android	20
85	iOS Location and Maps API	20
86	GNOME APIs	21
87	Navigation routing systems	22
88	NavServer	23
89	GENIVI	23
90	Google web APIs	24
91	Approach	26

92	Backends	26
93	Navigation application	27
94	2D map display	27
95	2.5D or 3D map display	28
96	Geolocation	28
97	Navigation routing	29
98	Navigation route list API	30
99	Navigation route guidance progress API	31
100	Navigation route guidance turn-by-turn API	32
101	Horizon API	33
102	Forward and reverse geocoding	35
103	Address completion	36
104	Geofencing	37
105	Location security	38
106	Systemic security	39
107	Testability	40
108	Requirements	41
109	Suggested roadmap	43
110	Open questions	43
111	Summary of recommendations	43
112	Appendix: Recommendations for third-party navigation applications	45
113	Appendix: place URI scheme	47
114	Examples	48
115	Appendix: nav URI scheme	49
116	Examples	50

117 This documents existing solutions for geo-related features (sometimes known
118 as location-based services, LBS) which could be integrated into Apertis for
119 providing geolocation, geofencing, geocoding and navigation routing support
120 for 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://jwd.pages.apertis.org/apertis-website/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
195 not 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
242 a 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

252 The proprietary geo-backend in a derivative of Apertis may expose functionality
253 beyond what is described in this design, which the system integrator might want
254 to use in their own 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
308 general area of the city which the tour should go through, but not necessarily
309 using specific roads or visiting specific points. These waypoints are more like
310 'way-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**

325 The user has to stop off at their house on their way to work to answer the
326 paging message. The paging application knows this, and includes the user's
327 home address as a navigation waypoint on the way to their destination. This is
328 reflected in the route chosen by the navigation 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 the
346 navigation system or sensor system, such as snapping the location to the near-
347 est 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 function-
359 ality.

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

²https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/#General_POI_providers

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.

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**

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

384 **Third-party navigation application backend**

385 If the user installs a full-featured third-party navigation application, they may
386 want to use it to provide 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
440 they may be in scope to be handled by other components of Apertis. Further
441 details 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 at
454 all, as advertising the location of a fast vehicle needs a different physical layer
455 approach than Beacons, which are designed for low-speed devices carried by
456 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.

³https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

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

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

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.

- Optionally rendering the vehicle's current position as a map layer (see [Relocating the vehicle][Relocating the vehicle]).

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

2.5D or 3D map rendering API

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

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

Reverse geocoding API

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

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

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

Forward geocoding API

Forward geocoding must be supported for querying addresses at selected coordinates on a map. Limiting the search results to a certain number of results should be supported.

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

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

Type-ahead search and address completion API

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

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

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

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,
657 for example, the user’s home locale could be used, or the locale of the country
658 the 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://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/#General_POI_providers

⁶https://jwd.pages.apertis.org/apertis-website/concepts/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
722 details of the implementations of features that seem particularly interesting
723 or relevant.

724 **W3C Geolocation API**

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

730 **Android platform location API**

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

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

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

736 Google Location Services API for Android

737 The [Google Location Services API for Android](#)⁹ is a more fully featured API
738 than the platform location API, supporting geocoding and geofencing in addi-
739 tion to geolocation. It requires the device to be connected to the internet to
740 access Google Play services. It hides the complexity of calculating and tracking
741 the device's location much more than the platform location API.

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

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

748 iOS Location and Maps API

749 The [iOS Location Services and Maps API](#) is available on both iOS and OS X. It
750 supports many features: geolocation, geofencing, forward and reverse geocoding,
751 navigation routing, and local search.

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

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

757 Geocoding uses a network service; both forward and reverse geocoding are sup-
758 ported.

759 The [MapKit API](#)¹⁰ provides an embeddable map renderer and widget, including
760 annotation and overlay support.

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

767 It also supports navigation routing using a system provider, which requires
768 a network connection. Calculated routes include metadata such as distance,

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

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

769 expected travel time, localised advisory notices; and the set of steps for the
770 navigation. It supports returning multiple route options for a given navigation.

771 The [local search API](#)¹² differs from the geocoding API in that it supports *types*
772 of locations, such as ‘coffee’ or ‘fuel’. As with geocoding, the local search API
773 requires a network connection.

774 GNOME APIs

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

777 GeoClue

778 [GeoClue](#)¹³ is a geolocation service which supports multiple input backends, such
779 as GPS, cellular network location and Wi-Fi based geolocation.

780 Wi-Fi location uses the [Mozilla Location Service](#)¹⁴ and requires network connec-
781 tivity.

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

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

789 Geocode-glib

790 [Geocode-glib](#)¹⁶ is a library for forward and reverse geocoding. It uses the Nom-
791 inatim API, and is currently [hard-coded to query nominatim.gnome.org](#)¹⁷. It
792 requires network access to perform geocoding.

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

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

¹²https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/EnablingSearch/EnablingSearch.html#//apple_ref/doc/uid/TP40009497-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://developer.gnome.org/geocode-glib/stable/>

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

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

798 **libchamplain**

799 [libchamplain](#)¹⁹ is a map rendering library, providing a map widget which sup-
800 ports annotations and overlays. It supports loading or rendering map tiles from
801 multiple sources.

802 **NavIt**

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

810 **Navigation routing systems**

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

814 **GraphHopper**

815 [GraphHopper](#)²¹ is a routing system written in Java, which is available as a server
816 or as a library for offline use. It uses OpenStreetMap data, and is licenced under
817 Apache License 2.0.

818 **OSRM**

819 [OSRM](#)²² is a BSD-licensed C++ routing system, which can be used as a server
820 or as a library for offline use. It uses OpenStreetMap data.

821 **YOURS**

822 [YOURS](#)²³ is an online routing system which provides a web API for routing
823 using OpenStreetMap data.

824 **NavServer**

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

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

²⁰<http://www.navit-project.org/>

²¹<https://graphhopper.com/>

²²<http://project-osrm.org/>

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

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

830 **GENIVI**

831 GENIVI implements its geo-features and navigation as various components un-
832 der the umbrella of the [IVI Navigation project](http://projects.genivi.org/ivi-navigation/documentation)²⁴. The core APIs it provides are
833 detailed below.

834 **Navigation**

835 The navigation application is based on NavIt, using OpenStreetMap for its
836 mapping data. It implements route calculation, turn-by-turn instructions, and
837 map rendering.

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

841 **Fuel stop advisor**

842 The fuel stop advisor is a demo application which consumes information from
843 the geolocation API and the vehicle API to get the vehicle's location and fuel
844 levels, in order to predict when (and where) would be best to refuel.

845 **POI service**

846 The points of interest (POI) service is implemented using multiple 'content
847 access module' (CAM) plugins, each providing points of interest from a different
848 provider or source. When searching for POIs, the service sends the query to all
849 CAMs, with a series of attributes and values to match against them using a set
850 of operators (equal, less than, greater than, etc.); plus a latitude and longitude
851 to base the query around. CAMs return their results to the service, which then
852 forwards them to the client which originally made the search request.

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

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

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

²⁴<http://projects.genivi.org/ivi-navigation/documentation>

860 **Positioning**

861 The positioning API provides a large amount of positioning data: time, position
862 in three dimensions, heading, rate of change of position in three dimensions, rate
863 of change of angle in three dimensions, precision of position in three dimensions,
864 and standard deviation of position in three dimensions and heading.

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

868 **Google web APIs**

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

871 **Google Maps Geocoding API**

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

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

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

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

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

887 **Google Places API**

888 The [Google Places API](#)²⁶ supports several different operations: returning a list
889 of places which match a user-provided search string, returning details about a
890 place, and auto-completing a user's search string based on places it possibly
891 matches.

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

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

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

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

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

900 Google Maps Roads API

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

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

908 Google Maps Geolocation API

909 The [Google Maps Geolocation API](#)²⁸ provides a way for a mobile device (any
910 device which can detect mobile phone towers or Wi-Fi access points) to look up
911 its likely location based on which mobile phone towers and Wi-Fi access points
912 it can currently see.

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

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

922 Approach

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

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

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

929 **Backends**

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

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

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

944 Backends may be implemented in the IVI domain (for example, the default
945 backends in the SDK must be implemented in the IVI domain, as the SDK has
946 no other domains), or in the automotive domain. If a backend is implemented
947 in the automotive domain, its functionality must be exposed as a proxy service
948 in the IVI domain, which implements the SDK API. Communications between
949 this proxy service and the backend in the automotive domain will be over the
950 inter-domain communications link.

951 See the Inter-Domain Communications design

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

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

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

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

966 Navigation application

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

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

978 2D map display

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

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

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

990 libchamplain supports rendering custom widgets provided by the application on
991 top of the map layer.

992 Route list layer on the 2D map

993 A new libchamplain layer should be provided by the Apertis SDK which renders
994 the vehicle's current route list as an overlay on the map, updating it as necessary
995 if the route is changed. If no route is set, the layer must display nothing (i.e.
996 be transparent).

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

999 In order for this to work, the application must have permission to query the
1000 vehicle's route list.

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

1001 **Vehicle location layer on the 2D map**

1002 A new libchamplain layer should be provided by the Apertis SDK which renders
1003 the vehicle's current location as a point on the map using a standard icon or
1004 indicator. The location should be updated as the vehicle moves. If the vehicle's
1005 location is not known, the layer must display nothing (i.e. be transparent).

1006 Applications can add this layer to an instance of libchamplain in order to easily
1007 get this functionality.

1008 In order for this to work, the application must have permission to query the
1009 vehicle's location.

1010 **2.5D or 3D map display**

1011 Our initial suggestion is to use [NavIt](#)³⁰ for 3D map display. However, we are
1012 currently unsure of the extent to which it can be used as a library, so we cannot
1013 yet recommend an API for 2.5D or 3D map display. Similarly, we are unsure
1014 of the extent to which route information and custom rendering can be input to
1015 the library to integrate it with other routing engines; or whether it always has
1016 to use routes calculated by other parts of NavIt.

1017 **Open question:** Is it possible to use NavIt as a stand-alone 2.5D or 3D map
1018 widget?

1019 **Geolocation**

1020 [GeoClue](#)³¹ should be used for geolocation. It supports multiple backends, so
1021 closed source as well as open source backends can be used. Some of the advanced
1022 features which do not impact on the API could be implemented in an automotive
1023 backend, although other backends would benefit if they were implemented in the
1024 core of GeoClue instead. For example, cached locations and dead-reckoning of
1025 the location based on previous velocity for when GPS signal is lost.

1026 The vehicle's velocity may be queried from the sensors; see the Sen-
1027 sors and Actuators design

1028 GeoClue supports geolocation using GPS (from a modem), 3G and Wi-Fi. It
1029 supports [accuracy bounds for locations](#)³², but does not pair that with informa-
1030 tion about the time of measurement. That would need to be added as a new
1031 feature in the API. Speed, altitude and bearing information [are supported](#)³³.
1032 The other two angles of movement, the rate of change of angle of movement in

³⁰<http://www.navit-project.org/>

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>

1033 all three dimensions, and uncertainty bounds and standard deviation for non-
1034 position measurements are not currently included in the API, and should be
1035 added.

1036 The API already supports signalling of failure to geolocate the vehicle, by setting
1037 its Location property to ‘/’ (rather than a valid org.freedesktop.GeoClue2.Location
1038 object path).

1039 If the navigation application implements a snap-to-road feature, it should be
1040 used as a further source of input to GeoClue for refining the location.

1041 **Geolocation signals**

1042 GeoClue emits a [LocationUpdated signal](#)³⁴ whenever the vehicle’s location
1043 changes more than the [DistanceThreshold](#)³⁵. GeoClue currently does not
1044 support rate limiting emission of the LocationUpdated signal for minimum and
1045 maximum times between updates. That would need to be added to the core of
1046 GeoClue.

1047 **Navigation routing**

1048 Navigation routing will be implemented internally by the OEM-provided naviga-
1049 tion application, or potentially by a third-party navigation application installed
1050 by the user. In either case, there will not be an Apertis SDK API for calculating
1051 routes.

1052 However, the SDK will provide an API to launch the navigation application and
1053 instruct it to calculate a new route. This API is the [content hand-over](#)³⁶ API,
1054 where the navigation application can be launched using a nav URI. The nav URI
1055 scheme is a custom scheme (which must not be confused with the standard [geo](#)
1056 [URI scheme](#)³⁷, which is for identifying coordinates by latitude and longitude).
1057 See [Appendix: nav URI scheme](#) for a definition of the scheme, and examples.

1058 When handling a content hand-over request, the navigation application should
1059 give the user the option of whether to replace their current route with the new
1060 route — but this behaviour is internal to the navigation application, and up to
1061 its developer and policy.

1062 The behaviour of the navigation application if it is passed an invalid URI, or
1063 one which it cannot parse (for example, due to not understanding the address
1064 format it uses) is not defined by this specification.

³⁴[http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.
Client.html#gdbus-signal-org-freedesktop-GeoClue2-Client.LocationUpdated](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](http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-property-org-freedesktop-GeoClue2-Client.DistanceThreshold)

³⁶https://jwd.pages.apertis.org/apertis-website/concepts/content_hand-over/

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

1065 As per the [content hand-over design](#)³⁸, the user may choose a third-party navigation application as the handler for nav URIs, in which case it will be launched as the default navigation application.

1068 If no navigation application is installed, or none is set as the handler for nav URIs, the error must be handled as per the content hand-over design.

1070 Navigation route list API

1071 Separately from the content hand-over API for sending a destination to the navigation application (see [Navigation routing](#)), there should be a navigation route list API to expose information about the route geometry to SDK applications.

1074 This API will be provided by the SDK, but implemented by one of many backends — the navigation application will be one such backend, but another backend will be available on the SDK to expose mock data for testing applications against. The mock data will be provided by an emulator; see [Testing applications](#) for more information.

1079 These backends will feed into a system service which provides the SDK API to applications, and exposes:

- 1081 • The planned route geometry, including destination, waypoints and way-
1082 areas.
- 1083 • Potential alternative routes.

1084 The API will not expose the vehicle's current position — that's done by the [Geolocation](#). Similarly, it will not expose points of interest or other horizon data — that's done by the [points of interest API](#)³⁹; or guidance information — that's done by the [Navigation route guidance API](#).

1088 The API should emit signals on changes of any of this information, using the standard org.freedesktop.DBus.Properties signals. We recommend the following API, exposed at the well-known name org.apertis.Navigation1:

```
1091 /* This object should implement the org.freedesktop.DBus.ObjectManager standard
1092    API to expose all existing routes to clients.
1093 */
1094 object /org/apertis/Navigation1/Routes {
1095     read-only i property CurrentRoute; /* index into routes, or negative for 'no current route' */
1096     read-only ao property Routes; /* paths of objects representing potential routes, with the most highly recommended first */
1097 }
1098
1099 /* One of these objects is created for each potential route.
1100    Each object path is suffixed by a meaningless ID to ensure its uniqueness.
1101    Route objects are immutable once published. Any change should be done by
1102    adding a new route and removing the old one.
```

³⁸https://jwd.pages.apertis.org/apertis-website/concepts/content_hand-over/

³⁹https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

```

1103 */
1104 object /org/apertis/Navigation1/Routes/$ID {
1105     read-only a(dd) property Geometry; /* array of latitude-longitude pairs in order, from the start point to the d
1106     read-only u property TotalDistance; /* in metres */
1107     read-only u property TotalTime; /* in seconds */
1108     read-only a(ss) property Title; /* mapping from language name to a human-
1109     readable title of the route in this language. Language names are using the POSIX locale format with locale identi
1110     read-only a(ua{ss}) property GeometryDescriptions; /* array of pairs of index and description, which attach a c
1111 }

```

1112 Syntax is a pseudo-IDL and types are as defined in the D-Bus speci-
1113 fication, [http://dbus.freedesktop.org/doc/dbus-specification.html#](http://dbus.freedesktop.org/doc/dbus-specification.html#type-system)
1114 [type-system](http://dbus.freedesktop.org/doc/dbus-specification.html#type-system)

1115 Navigation route list backends

1116 The backend for the route list service is provided by the navigation application
1117 bundle, which may be built-in or provided by a user-installed bundle. If no
1118 navigation bundle is installed, no backend is used, and the route list service
1119 must return an error when called.

1120 On the Apertis SDK, a navigation bundle is not available, so a mock backend
1121 should be written which presents route lists provided by the developer. This
1122 will allow applications to be tested using route lists of the developer's choice.

1123 Navigation route guidance progress API

1124 There should be a navigation route guidance API to expose information about
1125 progress on the current route.

1126 This API will be provided as interfaces by the SDK, but implemented by the
1127 navigation application. The UI responsible for displaying progress information
1128 or third party applications can use this API to fetch the info from the navigation
1129 application.

1130 The route guidance progress API should be a D-Bus API which exposes esti-
1131 mates such as time to destination and time elapsed in the journey.

1132 The API should emit signals on changes of any of this information, using the
1133 standard `org.freedesktop.DBus.Properties` signals. We recommend the following
1134 API, exposed at the well-known name `org.apertis.NavigationGuidance1.Progress`
1135 on the object `/org/apertis/NavigationGuidance1/Progress`:

```

1136 interface org.apertis.NavigationGuidance1.Progress {
1137     read-only t property StartTime; /* UNIX timestamp, to be interpreted in the local timezone */
1138     read-only t property EstimatedEndTime; /* UNIX timestamp, to be interpreted in the local timezone */
1139 }

```

1140 Additional properties may be added in future.

1141 Syntax is a pseudo-IDL and types are as defined in the
1142 D-Bus specification, [http://dbus.freedesktop.org/doc/dbus-](http://dbus.freedesktop.org/doc/dbus-specification.html)
1143 [specification.html](http://dbus.freedesktop.org/doc/dbus-specification.html)\type-system⁴⁰

1144 Navigation route guidance turn-by-turn API

1145 There should be a navigation route guidance API to allow turn-by-turn guidance
1146 notifications to be presented.

1147 This API will be provided as interfaces by the SDK, but implemented by a
1148 system component which is responsible for presenting notifications — this will
1149 most likely be the compositor. The navigation application can then call the
1150 TurnByTurn API to display new turn-by-turn driving instructions.

1151 The route guidance turn-by-turn API should be a D-Bus API which exposes a
1152 method for presenting the next turn-by-turn navigation instruction.

1153 We recommend the following API, exposed at the well-known name
1154 org.apertis.NavigationGuidance1.TurnByTurn on the object /org/apertis/NavigationGuidance1/TurnByTurn:

```
1155 interface org.apertis.NavigationGuidance1.TurnByTurn {
1156     /* See https://people.gnome.org/~mccann/docs/notification-spec/notification-
1157     spec-latest.html#command-notify */
1158     method Notify (in u replaces_id,
1159                   in s icon_name,
1160                   in s summary,
1161                   in s body,
1162                   in a{sv} hints,
1163                   in i expire_timeout,
1164                   out u id)
1165
1166     /* See https://people.gnome.org/~mccann/docs/notification-spec/notification-
1167     spec-latest.html#command-close-notification */
1168
1169     method CloseNotification (in u id)
1170 }
```

1171 Syntax is a pseudo-IDL and types are as defined in the
1172 D-Bus specification, [http://dbus.freedesktop.org/doc/dbus-](http://dbus.freedesktop.org/doc/dbus-specification.html)
1173 [specification.html](http://dbus.freedesktop.org/doc/dbus-specification.html)\type-system⁴¹

1174 The design of the turn-by-turn API is based heavily on the freedesktop.org
1175 [Notifications specification](https://people.gnome.org/~mccann/docs/notification-spec/notification-spec-latest.html)⁴², and could share significant amounts of code with
1176 the implementation of normal (non-guidance-related) notifications.

⁴⁰<http://dbus.freedesktop.org/doc/dbus-specification.html%5Ctype-system>

⁴¹<http://dbus.freedesktop.org/doc/dbus-specification.html%5Ctype-system>

⁴²<https://people.gnome.org/~mccann/docs/notification-spec/notification-spec-latest.html>

1177 **Route guidance UI**

1178 By using a combination of the navigation route guidance API and the [points of](#)
1179 [interest API](#)⁴³, it should be possible for an OEM to provide a route guidance UI
1180 which is separate from their main navigation application UI, but which provides
1181 sufficient information for route guidance and display of points of interest, as
1182 described in [Separate route guidance UI](#).

1183 There is no need for a separate API for this, and it is expected that if an OEM
1184 wishes to provide a route guidance UI, they can do so as a component in their
1185 navigation application bundle, or as part of the implementation of the system
1186 chrome (depending on their desired user experience). The only requirement is
1187 that only one component on the system implements the route guidance D-Bus
1188 interfaces described above.

1189 **Horizon API**

1190 The [Horizon API](#) is a shared library which applications are recommended to use
1191 when they want to present horizon data in their UI — a combination of the route
1192 list and upcoming points of interest. The library should query the [Navigation](#)
1193 [route list API](#) and [points of interest APIs](#)⁴⁴ and aggregate the results for display
1194 in the application. It is provided as a convenience and does not implement
1195 functionality which applications could not otherwise implement themselves. The
1196 library should be usable by applications and by system components.

1197 Any OEM-preferred policy for aggregating points of interest may be imple-
1198 mented in the horizon library in order to be easily usable by all applications;
1199 but applications may choose to query the SDK route list and points of interest
1200 APIs directly to avoid this aggregation and implement their own instead.

1201 To use the horizon API, an application must have permission to query both the
1202 route list API and the points of interest API.

1203 What applications do with the horizon data once they have received it is up to
1204 the application — they may, for example, put it in a local cache and store it
1205 to prevent re-querying for historical data in future. This is entirely up to the
1206 application developer, and is out of the scope of this design.

1207 There was a choice between implementing the horizon API as a library or as a
1208 service. Implementing it as a service would not reduce memory consumption,
1209 as all consumers would likely still have to keep all horizon data in memory for
1210 rendering in their UI. It would not reduce CPU consumption, as aggregation
1211 of horizon data is likely to be simple (merging points of interest with the same
1212 name and location). It would double the number of IPC hops each piece of
1213 information had to make (changing from producer → consumer, to producer
1214 → horizon service → consumer). As all consumers of horizon data are likely
1215 to be interested in most points of interest, it would not significantly reduce

⁴³https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

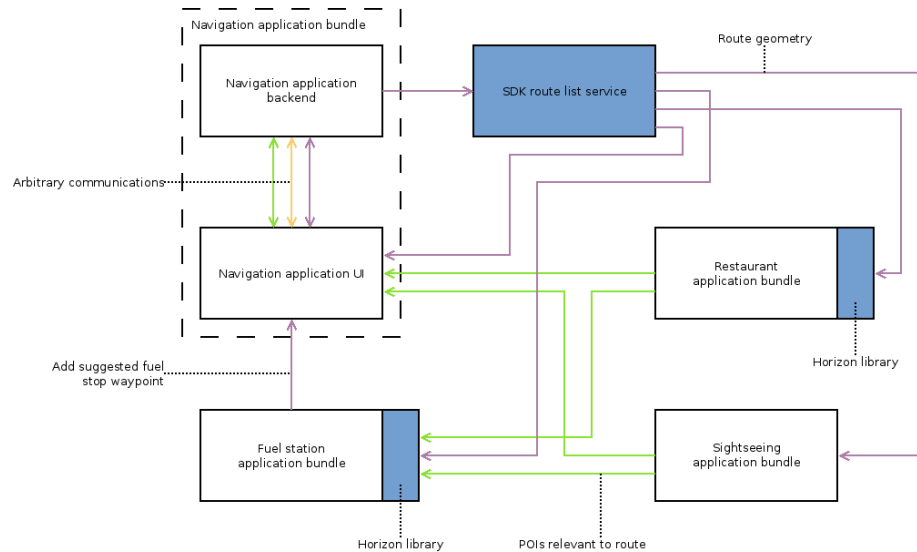
⁴⁴https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

1216 the amount of data needing to be forwarded between producers and consumers.
1217 Hence a library is a more appropriate solution.

1218 One potential downside of using a library is that it is harder to guarantee con-
1219 sistency in the horizon seen by different applications — the implementation
1220 should be careful to be deterministic so that users see the same horizon in all
1221 applications using the library.

1222 See the following figure for a diagram of the flow of points of interest and route
1223 lists around the system. Key points illustrated are:

- 1224 • Producers of points of interest choose which POIs are sent to which con-
1225 sumers (there is no central POI service), though the horizon library may
1226 perform filtering or aggregation according to system policy.
- 1227 • The route list API is provided by the SDK, but uses one backend out
1228 of zero or more provided by the navigation application bundle or other
1229 bundles.
- 1230 • The navigation UI is another application with no special powers; the arbi-
1231 trary communications between its UI and backend may carry route lists,
1232 points of interest, or other data, but does not have to use the formats or
1233 APIs defined in the SDK.
- 1234 • Applications choose which waypoints to send to via the [navigation routing
1235 API][Navigation routing] to be added into the route — this might result
1236 in the user being prompted ‘do you want to add this waypoint into your
1237 route’, or they might be added unconditionally. For example, the fuel
1238 station application bundle may add a fuel stop waypoint to the route,
1239 which is then exposed in the route list API if the navigation application
1240 accepts it.
- 1241 • The navigation UI does not necessarily use information from the route list
1242 API to build its UI (although such information may contribute).
- 1243 • If no navigation bundle is installed, the SDK route list service still exists,
1244 it just returns no data.



Forward and reverse geocoding

We recommend that [Geocode-glib](https://developer.gnome.org/geocode-glib/stable/)⁴⁵ is used as the SDK geocoding API. Geocode-glib is currently hard-coded to use GNOME's Nominatim service; it would need to be modified to support multiple backends, such as an Apertis-specific [Nominatim server](http://wiki.openstreetmap.org/wiki/Nominatim)⁴⁶, or a geocoding service from the automotive backend. It only needs to support querying one backend at a time; results do not need to be aggregated. Backends should be loadable and unloadable at runtime.

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

Geocoding backends

On an integrated system, geocoding services will be provided by the automotive domain, via inter-domain communications (See the Inter-domain Communications design). On the Apertis SDK, an internet connection is always assumed to be available, so geocoding may be performed using an online Nominatim backend. In both cases, geocode-glib would form the SDK API used by applications. Another alternative backend, which could be written and used by OEMs instead of an automotive backend, would be a [Google Maps geocoding API](https://developers.google.com/maps/documentation/geocoding/intro)⁴⁷ backend which runs in the IVI domain.

Although it is tempting to do so, points of interest should *not* be fed into geocode-glib via another new backend, as the semantics of points of interest (a large number of points spread in a radius around a focal point) do not match

⁴⁵<https://developer.gnome.org/geocode-glib/stable/>

⁴⁶<http://wiki.openstreetmap.org/wiki/Nominatim>

⁴⁷<https://developers.google.com/maps/documentation/geocoding/intro>

1267 the semantics of reverse geocoding (finding the single most relevant address to
1268 describe a given latitude and longitude). Points of interest should be queried
1269 separately using a library provided by the [Points of Interest design](#)⁴⁸.

1270 **Localisation of geocoding**

1271 Nominatim supports exporting localised place names from OpenStreetMap, but
1272 geocode-glib does not currently expose that data in its query results. It would
1273 need to be modified to explicitly expose locale data about results.

1274 It does currently support supplying the locale of input queries using the language
1275 parameter to [geocode_forward_new_for_params](#)⁴⁹.

1276 **Address completion**

1277 Address completion is a complex topic, both because it requires being able to
1278 parse partially-complete addresses, and because the datasets required to answer
1279 completion queries are large.

1280 Nominatim does not provide dedicated address completion services, but it is
1281 possible to implement them in a separate web service using a filtered version
1282 of the OpenStreetMap database data. An example is available as [Photon](#)⁵⁰.
1283 Google also provides a paid-for web service for address completion: the [Google](#)
1284 [Places Web API](#)⁵¹.

1285 As address completion is a special form of forward geocoding (i.e. forward
1286 geocoding operating on partial input), it should be provided as part of the
1287 geocoding service, and by the same backends which provide the geocoding func-
1288 tionality.

1289 If Nominatim (via geocode-glib) is found to be insufficient for address completion
1290 in the SDK, an Apertis-hosted Photon instance could be set up, and a Photon
1291 backend added to the geocoding service.

1292 In target devices, address completion should be provided by the automotive
1293 backend, or not provided at all if the backend does not implement it.

1294 The address completion API should be an extension to the existing geocode-glib
1295 API for forward geocoding. There must be a way for it to signal there are no
1296 known results. Results should be ranked by relevance or likelihood of a match,
1297 and should include information about which part of the search term caused the
1298 match (if available), to allow that to be highlighted in the widget.

1299 A separate library (which has a dependency on the SDK widget library) should
1300 provide a text entry widget which implements address completion using the

⁴⁸https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

⁴⁹<https://developer.gnome.org/geocode-glib/stable/GeocodeForward.html#geocode-forward-new-for-params>

⁵⁰<http://photon.komoot.de/>

⁵¹<https://developers.google.com/places/web-service/autocomplete>

1301 API on the geocoding service, so that application developers do not have to
1302 reimplement it themselves. This could be similar to [GtkSearchEntry](#)⁵² (but
1303 using the Apertis UI toolkit).

1304 Address completion backends

1305 As the backends for the address completion service (i.e. the geocoding backends)
1306 may access sensitive data to answer queries, they must be able to check the
1307 permissions of the application which originated the query. If the application
1308 does not have appropriate permissions, they must not return sensitive results.

1309 For example, a backend could be added which resolves ‘home’ to the user’s
1310 home address, ‘work’ to their work address, and ‘here’ to the vehicle’s current
1311 location. In order to maintain the confidentiality of this data, applications must
1312 have permission to access the system address book (containing the home and
1313 work addresses), and permission to access the vehicle’s location (see [Location security](#)). If the application does not have appropriate permissions, the backend
1314 must not return results for those queries.
1315

1316 As normal geocoding operation is not sensitive (the results do not differ depend-
1317 ing on who’s submitting a query), backends which require permissions like this
1318 must be implemented in a separate security domain, i.e. as a separate process
1319 which communicates with geocode-glib via D-Bus. They can get the request-
1320 ing application’s unforgeable identifier from D-Bus requests in order to check
1321 permissions.

1322 Geofencing

1323 We recommend that GeoClue is modified upstream to implement geofencing of
1324 arbitrary regions, meaning that geofencing becomes part of [Geolocation](#). Signals
1325 on entering or exiting a geofenced area should be emitted as a D-Bus signal
1326 which the application subscribes to. Delivery of signals to bundles which are
1327 not currently running may cause activation of that application.

1328 This is intended to be provided by a system service for activation
1329 of applications based on subscribed signals; the design is tracked in
1330 <https://phabricator.apertis.org/T640>.

1331 The geofencing service should include a database of country borders, and pro-
1332 vide a convenience API for adding a geofence for a particular country (or, for
1333 example, the current country and its neighbours). This would load the polygon
1334 for the country’s border and add it as a geofence as normal.

1335 The geofencing service must be available as the well-known name org.freedesktop.GeoClue2.Geofencing
1336 on D-Bus. It must export the following object:

```
1337 /org/freedesktop/GeoClue2/Geofencing/Manager {  
1338     /* Adds a geofence as a latitude, longitude and radius (in metres), and
```

⁵²<https://developer.gnome.org/gtk3/stable/GtkSearchEntry.html>


```

1339     * returns the ID of that geofence. The dwell\_time (in seconds) gives
1340     * how long the vehicle must dwell inside the geofence to be signalled
1341     * as such by the GeofenceActivity signal. */
1342     method AddGeofenceByRadius(in (dd) location, in u radius, in u dwell_time, out u id)
1343
1344     /* Adds a geofence by taking an ordered list of latitude and longitude
1345     * points which form a polygon for the geofence's boundary. */
1346     method AddGeofenceByPolygon(in a(dd) points, in u dwell\_time, out u id)
1347
1348     /* Remove some geofences. */
1349     method RemoveGeofences(in au ids)
1350
1351     /* Return a (potentially empty) list of the IDs of the geofences the
1352     * vehicle is currently dwelling in. */
1353     method GetDwellingGeofences(out au ids)
1354
1355     /* Signal emitted every time a geofence is entered or exited, which
1356     * lists the IDs of all geofences entered and exited since the previous
1357     * signal emission, plus all the geofences the vehicle is currently
1358     * dwelling inside. */
1359     signal GeofenceActivity(out au entered, out au dwelling, out au exited)
1360 }

```

1361 IDs are global and opaque — applications cannot find the area referenced by
1362 a particular geofence ID. A geofence may only be removed by the application
1363 which added it. Currently, the GeofenceActivity signal is received by all appli-
1364 cations, but they cannot dereference the opaque geofence identifiers for other
1365 applications. In future, if application-level containerisation is implemented, this
1366 signal will only be filtered per application.

1367 We have informally discussed the possibility of adding geofencing with the Geo-
1368 Clue developers, and they are in favour of the idea.

1369 **Location security**

1370 libchamplain is a rendering library, and does not give access to sensitive infor-
1371 mation.

1372 geocode-glib is a thin interface to a web service, and does not give access to
1373 sensitive information. All web service requests must be secured with best web
1374 security practices, such as correct use of HTTPS, and sending a minimum of
1375 identifiable information to the web service.

1376 GeoClue provides access to sensitive information about the vehicle's location.
1377 It currently allows limiting the accuracy provided to applications as [specified](#)
1378 [by the user](#)⁵³; this could be extended to implement a policy determined by the

⁵³<http://www.freedesktop.org/software/geoclue/docs/gdbus-org.freedesktop.GeoClue2.Client.html#gdbus-property-org-freedesktop-GeoClue2-Client-RequestedAccuracyLevel>

1379 capabilities requested in the application’s manifest.

1380 Similarly for GeoClue’s geofencing feature, when it is added — clients have
1381 separated access to its D-Bus API to allow them to be signalled at different
1382 accuracies and rates. This applies to navigation routing as well, as it may
1383 provide feedback to applications about progress along a route, which exposes
1384 information about the vehicle’s location.

1385 Application bundles asking for fine-grained location data should be subjected
1386 to closer review when submitted to the Apertis application store.

1387 **Systemic security**

1388 As the geo-features can source information from application bundles, they form
1389 part of the security boundary around application bundles.

1390 In order to avoid denial of service attacks from an application bundle which
1391 emits too much data as, for example, a general point of interest stream, the
1392 system should rate limit such streams in time (number of POIs per unit time)
1393 and space (number of POIs per map area).

1394 Location updates emitted by GeoClue must be rate limited between the mini-
1395 mum and maximum distance and time limits set by each client. These limits
1396 must be checked to ensure that a client is not requesting updates too frequently.

1397 For the components in the system which provide access to sensitive information
1398 (the vehicle’s location), a security boundary needs to be defined between them
1399 and application bundles. Geolocation, navigation route lists, route guidance
1400 and geofencing are the sensitive APIs — these are all implemented as services,
1401 so the D-Bus APIs for those services form the security boundary. In order to use
1402 any of these services, an application must have the appropriate permission in its
1403 manifest. Address completion as a whole is not treated as sensitive, but some
1404 of its backends may be sensitive, and perform their own checks according to
1405 other permissions (which may include those below). The following permissions
1406 are suggested:

- 1407 • *content-hand-over*: Required to use the content hand-over API for setting
1408 a new [navigation route][Navigation routing].
- 1409 • *location*: Required to access the geolocation, geofencing, or navigation
1410 route list services.
- 1411 • *navigation-route*: Required to access the navigation route list services
1412 (note that the *location* permission is also required).
- 1413 • *navigation-guidance*: Required to access the navigation guidance services
1414 (note that the *location* permission is also required).

1415 The libchamplain layers which applications can use (see [here][Route list layer
1416 on the 2D map] and [here][Vehicle location layer on the 2D map]) are treated

1417 as application code which is using the relevant services (geolocation and navi-
1418 gation route guidance), and hence the *location* or *location* and *navigation-route*
1419 permissions are required to use them. Similarly for the horizon library.

1420 Any service which accepts data from applications, such as points of interest or
1421 waypoints to add into the route list, must check that the user has not disabled
1422 that application from providing data. If the user has disabled it, the data must
1423 be ignored and an error code returned to the application if the API allows it,
1424 to indicate to the application that sharing was prevented.

1425 **Testability**

1426 There are several components to testability of geo-functionality. Each of the
1427 components of this system need to be testable as they are developed, and as new
1428 backends are developed for them (including testing the backends themselves).
1429 Separately, application developers need to be able to test their applications'
1430 behaviour in a variety of simulated geo-situations.

1431 **Testing geo-functionality**

1432 Each of the services must have unit tests implemented. If the service has back-
1433 ends, a mock backend should be written which exposes the backend API over
1434 D-Bus, and integration tests for that service can then feed mock data in via
1435 D-Bus and check that the core of the service behaves correctly.

1436 Just like how libfolks' dummy backend is used in its unit tests, [https:](https://git.gnome.org/browse/folks/tree/backends/dummy)
1437 [//git.gnome.org/browse/folks/tree/backends/dummy](https://git.gnome.org/browse/folks/tree/backends/dummy)

1438 **Testing backends**

1439 Each service which has backends must implement a basic compliance test suite
1440 for its backends, which will load a specified backend and check that its public
1441 API behaves as expected. The default backends will further be tested as part
1442 of the integration tests for the entire operating system.

1443 **Testing applications**

1444 In order to allow application developers to test their applications with mock
1445 data from the geo-services, there must be an emulator program available in the
1446 SDK which uses the mock backend for each service to feed mock data to the
1447 application for testing.

1448 For example, the emulator program could display a map and allow the developer
1449 to select the vehicle's current location and the accuracy of that location. It
1450 would then feed this data to the mock backend of the geolocation service, which
1451 would pass it to the core of the geolocation service as if it were the vehicle's
1452 real location. This would then be passed to the application under test as the
1453 vehicle's location, by the SDK geolocation API.

1454 The emulator could additionally allow drawing a route on the map, which it
1455 would then send to the mock backend for the route list API as the current route
1456 — this would then be passed to the application under test as the vehicle's current
1457 route.

1458 This means that the API of the mock backend for each service must be stable
1459 and well defined.

1460 Requirements

1461 This design fulfils the following requirements:

- 1462 • **Geolocation API** — use GeoClue
- 1463 • **Geolocation service supports signals** — use GeoClue; augment its signals
- 1464 • **Geolocation implements caching** — to be added to GeoClue
- 1465 • **Geolocation supports backends** — GeoClue supports backends
- 1466 • **Navigation routing API** — use content hand-over design, passing a nav
1467 URI to the navigation application
- 1468 • **Navigation routing supports different navigation applications** — content
1469 hand-over supports setting different applications as the handlers for the
1470 nav URI scheme
- 1471 • **Navigation route list API** — new D-Bus API which is implemented by the
1472 navigation application backend
- 1473 • **Navigation route guidance API** — new D-Bus API implemented by the
1474 system UI (i.e. the compositor) which is called by the navigation applica-
1475 tion
- 1476 • **Type-ahead search and address completion supports backends** — imple-
1477 mented as part of geocoding, to be added to geocode-glib
- 1478 • **Geocoding supports backends** — to be added to geocode-glib
- 1479 • [SDK has default implementations for all backends][SDK has default imple-
1480 mentations for all backends] — Gypsy or a mock backend for geolocation;
1481 custom online Nominatim server for geocoding; online OpenStreetMap for
1482 2D maps; libnavit for 3D maps, **subject to further evaluation**; custom
1483 mock backend for navigation route list; custom online Nominatim or Pho-
1484 ton server for address completion
- 1485 • **SDK APIs do not vary with backend** — GeoClue API for geolocation;
1486 geocode-glib for geocoding; libchamplain for 2D maps; libnavit for 3D
1487 maps, **subject to further evaluation**; content hand-over API for navi-
1488 gation routing; new API for navigation route lists and guidance; new API
1489 extensions to geocode-glib for address completion

- 1490 • **Third-party navigation applications can be used as backends** — backends
1491 are implemented as loadable libraries installed by the navigation applica-
1492 tion
- 1493 • **Backends operate asynchronously** — backends are implemented over D-
1494 Bus so are inherently asynchronous
- 1495 • **2D map rendering API** — use libchamplain with a local or remote tile
1496 store
- 1497 • **2.5D or 3D map rendering API** — use libnavit, **subject to further eval-**
1498 **uation**
- 1499 • **Reverse geocoding API** — use geocode-glib
- 1500 • **Forward geocoding API** — use geocode-glib
- 1501 • **Type-ahead search and address completion API** — to be added to geocode-
1502 glib
- 1503 • **Geofencing API** — to be implemented as a new feature in GeoClue
- 1504 • **Geofencing service can wake up applications** — to be implemented as a
1505 new feature in GeoClue
- 1506 • **Geofencing API signals on national borders** — to be added as a data set
1507 in GeoClue
- 1508 • **Geocoding API must be locale aware** — to be added to geocode-glib to
1509 expose existing OpenStreetMap localised data
- 1510 • **Geolocation provides error bounds** — GeoClue provides accuracy informa-
1511 tion, but it needs augmenting
- 1512 • **Geolocation implements dead reckoning** — to be added to GeoClue
- 1513 • **Geolocation uses navigation and sensor data if available** — to be added
1514 as another backend to GeoClue
- 1515 • **General points of interest streams are queryable** — to be designed and
1516 implemented as part of the [points of interest design](#)⁵⁴
- 1517 • **Location information requires permissions to access** — to be implemented
1518 as manifest permissions for application bundles
- 1519 • **Rate limiting of general point of interest streams** — security boundary im-
1520 plemented as D-Bus API boundary; rate limiting applied on signal emis-
1521 sion and processed general point of interest streams
- 1522 • **Application provided data requires permissions to create** — all geo-services
1523 must check settings before accepting data from applications

⁵⁴https://jwd.pages.apertis.org/apertis-website/concepts/points_of_interest/

1524 Suggested roadmap

1525 As the SDK APIs for geo-features are, for the most part, provided by FOSS
1526 components which are available already, the initial deployment of geo-features
1527 requires GeoClue, geocode-glib and libchamplain to be packaged for the distri-
1528 bution, if they are not already.

1529 The second phase would require modification of these packages to implement
1530 missing features and implement additional backends. This can happen once the
1531 initial packaging is complete, as the packages fulfil most of Apertis' requirements
1532 in their current state. This requires the address completion APIs to be added to
1533 to geocode-glib, and the geofencing APIs to be added to GeoClue, amongst other
1534 changes. These API additions should be prioritised over other work, so that
1535 application development (and documentation about application development)
1536 can begin.

1537 This second phase includes modifying the packages to be container-friendly, so
1538 that they can be used by compartmentalised apps without leaking sensitive data
1539 from one app to another. This requires further in-depth design work, but should
1540 require fairly self-contained changes.

1541 The second phase also includes writing the new services, such as the [Navigation](#)
1542 [route list API](#), the [Navigation route guidance API](#) and the [Horizon API](#).

1543 Open questions

- 1544 1. What review checks should be performed on application bundles which
1545 request permissions for location data?
- 1546 2. Is it possible to use NavIt as a stand-alone 2.5D or 3D map widget?

1547 Summary of recommendations

1548 As discussed in the above sections the recommendations are:

- 1549 • Packaging and using libchamplain for 2D map display.
- 1550 • Adding a route list layer for libchamplain.
- 1551 • Adding a vehicle location layer for libchamplain.
- 1552 • Packaging and using libnavit for 3D map display, **subject to further**
1553 **investigation.**
- 1554 • Packaging and using GeoClue for geolocation. It needs measurement
1555 times, cached location support, dead-reckoning and more measurements
1556 and uncertainty bounds to be added upstream.
- 1557 • Adding minimum and maximum update periods for clients to upstream
1558 GeoClue, alongside the existing distance threshold API for location update
1559 signals.

- 1560 • Adding a new navigation application backend to GeoClue to implement
1561 snap-to-road refinement of its location.
- 1562 • Adding a new mock backend to GeoClue for the SDK.
- 1563 • Implementing a library for parsing place and nav URIs and formalising the
1564 specifications so they may be used by third-party navigation applications.
- 1565 • Adding support for place and nav URIs to the content hand-over service
1566 (Didcot) and adding support for a default navigation application.
- 1567 • Implementing a navigation route list service with support for loadable
1568 backends, including a mock backend for the SDK.
- 1569 • Implementing a navigation route guidance API in the system compositor.
- 1570 • Implementing a horizon library for aggregating and filtering points of in-
1571 terest with the route list, suitable for use by applications.
- 1572 • Packaging and using geocode-glib for forward and reverse geocoding. It
1573 needs support for exposing localised place names to be added upstream.
- 1574 • Adding support for loadable backends to geocode-glib, including a mock
1575 backend for the SDK.
- 1576 • Auditing geocode-glib to ensure it maintains data privacy by, for example,
1577 using TLS for all requests and correctly checking certificates.
- 1578 • Auditing GeoClue to ensure it maintains data privacy by, for example,
1579 using TLS for all requests and correctly checking certificates.
- 1580 • Implementing an address completion API in geocode-glib and implement-
1581 ing it in the Nominatim and mock backends.
- 1582 • Implementing an address completion widget based on the address comple-
1583 tion API.
- 1584 • Implementing a geofencing API in upstream GeoClue.
- 1585 • Integrating the geo-services with the app service proxy to apply access
1586 control rules to whether applications can communicate with it to retrieve
1587 potentially sensitive location or navigation data. Only permit this if the
1588 appropriate permissions have been set on the application bundle's mani-
1589 fest.
- 1590 • Implementing an emulator program in the SDK for controlling mock data
1591 sent by the SDK geo-APIs to applications under test.
- 1592 • Providing integration tests for all geo-functionality.

1593 **Appendix: Recommendations for third-party navigation** 1594 **applications**

1595 While this design explicitly does not cover providing SDK APIs purely to be
1596 used in implementing navigation applications, various recommendations have
1597 been considered for what navigation applications probably should do. These
1598 are non-normative, provided as suggestions only.

- 1599 • Support different types of vehicle — the system could be deployed in a
1600 car, or a motorbike, or a HGV, for example. Different roads are available
1601 for use by these different vehicles.
- 1602 • Support calculating routes between the start, waypoints and destination
1603 using public transport, in addition to the road network. For example,
1604 this could be used to provide a comparison against the car route; or to
1605 incorporate public transport schemes such as park-and-ride into route sug-
1606 gestions.
- 1607 • Support audio turn-by-turn navigation, reading out instructions to the
1608 driver as they are approached. This allows the driver to avoid looking at
1609 the IVI screen to see the map, allowing them to focus on the road.
- 1610 • Route guidance must continue even if another application takes the fore-
1611 ground focus on the IVI system, meaning the guidance system must be
1612 implemented as an agent.
- 1613 • Support different optimisation strategies for route planning: minimal
1614 travel time, scenic route, minimal cost (for fuel), etc.
- 1615 • In order to match the driver's view out of the windscreen, the map dis-
1616 played when navigating should be a 3D projection to better emphasise
1617 navigational input which is coming up sooner (for example, the nearest
1618 junction or road signs).
- 1619 • Support changing the destination mid-journey and calculating a new
1620 route.
- 1621 • Support navigating via zero or more waypoints before reaching the final
1622 destination. Support adding new waypoints mid-journey.
- 1623 • Provide the driver with up-to-date information about the estimated travel
1624 time and distance left on their route, plus the total elapsed travel time
1625 since starting the route. This allows the driver to work out when to take
1626 rest breaks from driving.
- 1627 • Detect when the driver has taken a wrong turning while navigating, and
1628 recalculate the route to bring them back on-route to their destination,
1629 reoptimising the route for minimal travel time (or some other criterion)
1630 from their new location. The new route might be radically different from
1631 the old one if this is more optimal. The route recalculation should happen

quickly (on the order of five seconds) so that the driver gets updated routing information quickly.

- Support recalculating and potentially changing a navigation route if traffic conditions change and make the original route take significantly longer than an alternative. There must be some form of hysteresis on these recalculations so that two routes which have very similar estimated travel times, but which keep alternating as the fastest, do not continually replace each other as the suggested route. The application may ask or inform the driver about route recalculations, as the driver may be able to assess and predict the traffic conditions better than the application.
- Provide information to the driver about the roads the vehicle is travelling along or heading towards and going to turn on to in future, such as the road name and number, and major towns or cities the road leads to. This allows the driver to match up the turn-by-turn navigation directions with on-road signage. (This is often known as route guidance or driver assistance information.)
- If the driver takes a rest break during a navigation route, and turns the vehicle off, the application must give the driver the option to resume the navigation route when the vehicle is turned on again. The route must be recalculated from the vehicle's current location to ensure the resumed route is still optimal for current traffic conditions.
- Support cancelling the navigation when the vehicle is turned on again, at which point all navigation and turn-by-turn directions stop.
- If driven abroad, the navigation application should provide locale-sensitive navigation information, such as speed limits in the local units, and road descriptions which match the local signage conventions.
- Support feeding geolocation data in to the system geolocation service, if such data may be more precise than the raw GPS positions; for example, if it can be snapped to the nearest road.
- Support feeding other geo-information to the other geo-services, such as answering geocoding queries or performing geo-fencing. Support being a full replacement for the inbuilt navigation application and all the SDK services it provides.
- Query the system POI API for restaurants and toilets at times and frequencies suitable for recommending food or toilet breaks to the driver appropriately. Allow the driver to dismiss or disable these, or to change the intervals. Do not recommend a break if the journey is predicted to end soon. Launch the application which provided the POI if the user clicks on a POI in the navigation application, so that they can perform further actions on that POI (for example, if it's a restaurant, they could reserve a table). When POIs are displayed in the navigation application, they can be rendered as desired by the navigation application developer; when they

1674 are displayed in other applications, they are rendered as desired by those
1675 applications' developers.

1676 **Appendix: place URI scheme**

1677 The place URI scheme is a non-standard URI scheme suggested to be used
1678 within Apertis for identifying a particular place, including a variety of redundant
1679 forms of information about the place, rather than relying on a single piece of
1680 information such as a latitude and longitude. To refer to a location by latitude
1681 and longitude *only*, use the standard [geo URI scheme](#)⁵⁵.

1682 A place URI is a non-empty human-readable string describing the location,
1683 followed by zero or more key–value pairs providing additional information. The
1684 key–value pairs are provided as parameters as defined in [RFC 5870], i.e. each
1685 parameter is separated by a semicolon, keys and values are separated by an
1686 equals sign, and percent-encoding is used to encode reserved characters.

1687 The location string must be of the format `1*paramchar`, as defined in RFC
1688 5870. All non-ASCII characters in the string must be [percent-encoded](#)⁵⁶, and
1689 implementations must interpret the decoded string as [UTF-8](#)⁵⁷.

1690 Implementations may support the following parameters, and must ignore un-
1691 recognised parameters, as more may be added in future. All non-ASCII char-
1692 acters in parameter keys and values must be percent-encoded, and implementa-
1693 tions must interpret the decoded strings as UTF-8. The semicolon and equals
1694 sign separators must not be percent-encoded. The ordering of parameters does
1695 not matter, unless otherwise specified. Each parameter may appear zero or one
1696 times, unless otherwise specified.

- 1697 • `location`: the latitude and longitude of the place, as a geo URI (*with the*
1698 `geo:` scheme prefix)
- 1699 • `postal-code`: the postal code for the place, in the country's postal code
1700 format
- 1701 • `country`: the [ISO 3166-1 alpha-2](#)⁵⁸ country code
- 1702 • `region`: the human-readable name of a large administrative region of the
1703 country, such as a state, province or county
- 1704 • `locality`: the human-readable name of the locality, such as a town or city
- 1705 • `area`: the human-readable name of an area smaller than the locality, such
1706 as a neighbourhood, suburb or village
- 1707 • `street`: the human-readable street name for the place

⁵⁵<http://tools.ietf.org/html/rfc5870>

⁵⁶<http://tools.ietf.org/html/rfc5870#section-3.5>

⁵⁷<http://www.unicode.org/versions/Unicode6.0.0/ch03.pdf>

⁵⁸http://www.iso.org/iso/country_codes.htm

- `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`
 - Location string: Paris
 - No parameters
- `place:Paris;location=geo%3A48.8567%2C2.3508;country=FR;formatted-address=Paris%2C%20France`
 - Location string: Paris
 - Parameters:
 - * `location`: `geo:48.8567,2.3508`
 - * `country`: `FR`
 - * `formatted-address`: `Paris, France`
- `place=K%C3%B6nigsstieg%20104%2C%2037081%20G%C3%B6ttingen;location=geo%3A51.540060%2C9.911850;country=DE;postal-code=37081;street=K%C3%B6nigsstieg;building=104;formatted-address=K%C3%B6nigsstieg%20104%2C%2037081%20G%C3%B6ttingen%20Germany`
 - Location string: Königsstieg 104, 37081 Göttingen
 - Parameters:
 - * `location`: `geo:51.540060,9.911850`
 - * `country`: `DE`
 - * `locality`: `Göttingen`
 - * `postal-code`: `37081`
 - * `street`: `Königsstieg`
 - * `building`: `104`
 - * `formatted-address`: `Königsstieg 104, 37081 Göttingen, Germany`
- `place=CN Tower;location=geo%3A43.6426%2C-79.3871;formatted-address=301%20Front%20St%20W%2C%20Toronto%20Ontario%20Canada`
 - Location string: CN Tower
 - Parameters:
 - * `location`: `geo:43.6426,-79.3871`
 - * `formatted-address`: `301 Front St W, Toronto, ON M5V 2T6, Canada`

1745 Appendix: nav URI scheme

1746 The nav URI scheme is a non-standard URI scheme suggested to be used within
1747 Apertis for identifying a navigation route, including its destination, intermediate
1748 destinations (waypoints) and points or areas to route via but which are not
1749 named destinations (via-points). Each point or area may be provided as a place
1750 or a location.

1751 A nav URI is a non-empty destination place, followed by zero or more key-
1752 value pairs providing additional information. The key-value pairs are provided
1753 as parameters as defined in [RFC 5870], i.e. each parameter is separated by a
1754 semicolon, keys and values are separated by an equals sign, and percent-encoding
1755 is used to encode reserved characters.

1756 The destination place must be provided as [Appendix: place URI scheme](#) (*with*
1757 the `place:` URI prefix), or as a geo URI (*with* the `geo:` URI prefix); and must be
1758 encoded in the format `1*paramchar`, as defined in RFC 5870; i.e. all non-ASCII
1759 and reserved characters in the string must be percent-encoded.

1760 Implementations may support the following parameters, and must ignore un-
1761 recognised parameters, as more may be added in future. All non-ASCII char-
1762 acters in parameter keys and values must be percent-encoded, and implementa-
1763 tions must interpret the decoded strings as UTF-8. The semicolon and equals
1764 sign separators must not be percent-encoded. The ordering of parameters does
1765 not matter, unless otherwise specified. Each parameter may appear zero or one
1766 times, unless otherwise specified.

- 1767 • `description:` a human-readable description of the route, intended to be
1768 displayed in the UI rather than machine-parsed
- 1769 • `way:` a named intermediate destination, as a place URI (*with* the `place:`
1770 `scheme` prefix) or as a geo URI (*with* the `geo:` `scheme` prefix); these pa-
1771 rameters are order-dependent (see below)
- 1772 • `via:` a non-named intermediate routing point, as a place URI (*with* the
1773 `place:` `scheme` prefix) or as a geo URI (*with* the `geo:` `scheme` prefix); these
1774 parameters are order-dependent (see below)

1775 The `way` and `via` parameters are order-dependent: they will be added to the
1776 route in the order they appear in the nav URI. Way-places and via-places may
1777 be interleaved — they form a single route. The destination place always forms
1778 the final point in this route. The `way` and `via` parameters may each appear zero
1779 or more times.

1780 Additional routing specific parameters can be added. If those parameters are
1781 not provided, the default value is left to the routing engine. It may be different
1782 for each type of vehicle, or due to other logic in the routing engine. Many param-
1783 eters represent a single value; for example, it is not meaningful to specify both
1784 `optimize=fastest` and `optimize=shortest`. URIs with multiple values for a single-
1785 valued parameter, for example `place:Home;optimize=fastest;optimize=shortest`,

1786 should be treated as though that parameter was not present. Apertis does not
1787 currently define multi-valued preferences. All values should be specified at most
1788 once. However, OEMs may define and use their own multi-valued properties.
1789 The naming convention is defined below. Boolean values can be specified as 1
1790 and 0 (or equivalently `true` and `false` while being case insensitive). Other values
1791 are considered invalid.

- 1792 • `vehicle`: vehicle for which the route should be calculated. Apertis defines
1793 `car`, `walk`, and `bike`. `car` routes are optimized for being used by car. `walk`
1794 routes are optimized for walking. `bicycle` routes are optimized for being
1795 ridden by bicycles.
- 1796 • `optimize`: Optimizes route calculation towards a set of criteria. Apertis
1797 defines `fastest`, and `shortest` criteria. `fastest` routes are optimized to
1798 minimize travel duration. `shortest` routes are optimized to minimize travel
1799 distance.
- 1800 • `avoid-tolls`: Boolean. If true, the calculated route should avoid tolls. If
1801 usage can't be avoided, the handler application is responsible for informing
1802 the user.
- 1803 • `avoid-motorways`: Boolean. If true, the generated route should avoid mo-
1804 torways. If usage can't be avoided, the handler application is responsible
1805 for informing the user.
- 1806 • `avoid-ferries`: Boolean. If true, the generated route should avoid fer-
1807 ries. If usage can't be avoided, the handler application is responsible for
1808 informing the user.

1809 Additionally, vendor specific parameters can be provided for vendor specific
1810 features. To avoid contradictory definitions for the same parameter in different
1811 implementations, vendor-specific parameters must be named in the form `x-`
1812 `vendorname-paramname`, similar to [the convention for extending .desktop files](https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending)⁵⁹.
1813 Note that unlike keys in `.desktop` files, parameter names in `nav:` URIs are not
1814 case-sensitive: consistently using lower-case is encouraged. For instance, one
1815 of the [examples] below assumes that a manufacturer has introduced a boolean
1816 option `x-batmobile-avoid-detection`, and a string-valued parameter `x-batmobile-`
1817 `auto-pilot-mode`.

1818 Examples

1819 This section is non-normative. Each example is given as a fully encoded string,
1820 followed by it split up into its un-encoded components.

1821 Example 1

⁵⁹<https://specifications.freedesktop.org/desktop-entry-spec/desktop-entry-spec-latest.html#extending>

1822 nav:place%3AKings%2520Cross%2520station%252C%2520London%3Blocality%3DLondon%3Bpostal-
1823 code%3DN19AL

- 1824 • Destination place:
 - 1825 – Location string: Kings Cross station, London
 - 1826 – Parameters:
 - 1827 * locality: London
 - 1828 * postal-code: N19AL

1829 Example 2

1830 nav:geo%3A51.531621%2C-0.124372

- 1831 • Destination place: geo:51.531621,-0.124372

1832 Example 3

1833 nav:place%3ABullpot%2520Farm%3Blocation%3Dgeo%253A54.227602%252C-2.517940;way=place%3ABirmingham%2520New%25
1834 1.897904;via=place%3AHornby%3Blocation%3Dgeo%253A54.112245%252C-2.636527%253Bu%253D2000;way=place%3AInglesp
1835 code%3DLA63EB%3Bcountry%3DGB

- 1836 • Destination place:
 - 1837 – Location string: Bullpot Farm
 - 1838 – Parameters:
 - 1839 * location: geo:54.227602,-2.517940
- 1840 • Parameters:
 - 1841 – way:
 - 1842 * Location string: Birmingham New Street station
 - 1843 * Parameters:
 - 1844 · location: geo:52.477620,-1.897904
 - 1845 – via:
 - 1846 * Location string: Hornby
 - 1847 * Parameters:
 - 1848 · location: geo:54.112245,-2.636527;u=2000
 - 1849 – way:
 - 1850 * Location string: Inglesport, Ingleton
 - 1851 * Parameters:
 - 1852 · street: The Square
 - 1853 · building: 11
 - 1854 · locality: Ingleton
 - 1855 · postal-code: LA63EB
 - 1856 · country: GB

1857 Example 4

1858 nav:geo%3A51.531621%2C-0.124372;vehicle=walk;optimize=shortest

- 1859 • Destination place: geo:51.531621,-0.124372
- 1860 • Parameters:

```
1861         - vehicle: walk
1862         - optimize: shortest
```

1863 **Example 5**

```
1864 nav:geo%3A51.531621%2C-0.124372;vehicle=car;avoid-tolls=false;x-batmobile-
1865 auto-pilot-mode=full;x-batmobile-avoid-detection=true
```

- 1866 • Destination place: geo:51.531621,-0.124372
- 1867 • Parameters:
 - 1868 - vehicle: car
 - 1869 - avoid-tolls: false
 - 1870 - x-batmobile-avoid-detection: true
 - 1871 - x-batmobile-auto-pilot-mode: full

1872 **Example 6**

```
1873 nav:place:Cambridge;x-myvendor-avoid-road=A14;x-myvendor-avoid-road=M11
```

- 1874 • Destination place: Cambridge
- 1875 • Parameters:
 - 1876 - x-myvendor-avoid-road: multi-valued: A14, M11