



Contacts

Contents

2	Integrated Address Book Versus Alternative Solutions	3
3	Contact Sources	3
4	Local Sources	3
5	Bluetooth-paired phone	3
6	Chat and Voice-over-IP Services	4
7	Web services	5
8	SIM Card	5
9	Read-only Operation for External Sources	5
10	Standard Behavior and Operations	5
11	Contact Management	5
12	Contact Aggregation and Linking	6
13	Local Address Book Management	6
14	Search	6
15	Event Logging	7
16	Caching	8
17	Components	8
18	Folks	8
19	Telepathy	10
20	Evolution Data Server (EDS)	11
21	libsocialweb	11
22	SyncEvolution	11
23	Zeitgeist	11
24	Architecture	12
25	Accessibility of Contacts By Source	12
26	User interfaces	12
27	Multiple Users	13
28	Storage considerations	13
29	Abstracting Contacts Libraries	13

This document outlines our design for address book contacts within the Apertis system. It describes the many sources the system can draw upon for the user's contacts, how it will manage those contacts, which components will be necessary, and what work will be needed in the middleware space to enable these features.

Contacts are representations of people that contain some details about that person. These details are often directly actionable: phone numbers can be called, street addresses may be used as destinations for the navigation system. Other details, such as name and avatar are purely representational.

We propose a contact system which uses the Folks contact aggregator to retrieve contacts from multiple sources and automatically match up contacts which correspond to a single person. This will give a thorough and coherent view of one's contacts with minimal effort required of the user.

42 Integrated Address Book Versus Alternative Solutions

43 The following design is based around the concept of a heavily-integrated address
44 book which links together contacts from many contact sources, providing a
45 common interface for applications to access these contacts. As presented below,
46 the only available contacts which will not be fully-integrated into the common
47 contact view will be contacts available on a paired Bluetooth device.

48 The level of contact source integration is flexible. If it is preferred to limit
49 contact integration to the local address book and chat/Voice-over-IP contacts to,
50 for example, isolate Facebook or Twitter contacts in their own address book(s),
51 to be accessed by a special library, Collabora is ready and able to adjust this
52 design.

53 Contact Sources

54 There are many potential sources for contacts, as people's contact details are
55 frequently split over many services. The proposed system aggregates contacts
56 from multiple sources in a way that is seamless to the user. See the **Components**
57 section on **Folks** for more details of the components involved.

58 Local Sources

59 New contacts may be created locally by importing contacts from a **Bluetooth-**
60 **paired phone** or a contact editor dialog (see **User interfaces**).

61 These local contacts may contain a wide variety of detail types, including (but
62 not limited to):

- 63 • Full name
- 64 • Phone numbers
- 65 • Street addresses
- 66 • Email addresses
- 67 • Chat usernames on various services
- 68 • User-selected groups
- 69 • Notes

70 Bluetooth-paired phone

71 **Synchronization** Contacts may be simply synchronized to a Apertis system
72 by means of a **SyncML**¹ contact transfer from a phone paired with the Apertis
73 system over Bluetooth. This operation is designed to intelligently merge fields
74 added to contacts on the source phone to avoid creating duplicates.

75 To manage complexity, this function will only be supported from a phone to
76 the Apertis system, not the other way around. Systems which support two-way
77 contact synchronization have a number of issues to contend with, including:

¹<http://en.wikipedia.org/wiki/SyncML>

- 78 • Contacts do not contain “last modified”time stamps, so it is rarely obvious
79 how to resolve conflicts
- 80 • “Fuzzy-matching”fields for cases of equivalent names or phone numbers is
81 not consistently implemented across different systems (if it is implemented
82 at all)
- 83 • Even if equivalent fields are correctly matched, it is not clear which version
84 should be preferred
- 85 • Because conflict resolution may not be symmetrical between the two direc-
86 tions of synchronization, the contacts in the two systems may never reach
87 a stable state, potentially causing other side effects (such as duplicates on
88 the phone)

89 By limiting synchronization from the phone to the Apertis instance (with a
90 “source wins”conflict resolution policy), we can avoid the aforementioned issues
91 and more. This simpler scheme will also be easier for users to understand,
92 improving the user exeperience.

93 Synchronization will be performed automatically each connection of a phone to
94 the Apertis system.

95 Each phone device will receive its own contact address book on the Apertis
96 system which will be created upon first connection and re-used upon subsequent
97 connections. This is meant to make it trivial to remove old address books based
98 upon storage requirements.

99 **Chat and Voice-over-IP Services**

100 Most chat and some Voice-over-IP (VoIP) services maintain contact lists, so
101 these are another potential source of contacts. We recommend supporting con-
102 tacts from audio- and video-capable services, such as Session Initiation Protocol
103 (SIP), Google Talk, and XMPP. These contacts and their services provide an al-
104 ternative type of audio call which users may occasionally prefer to mobile phone
105 calls for purposes of call quality and billing.

106 Additionally, contacts on some of these services may provide extended informa-
107 tion, such as a street address, which the user might not otherwise have in their
108 address book.

109 Our system will cache these contacts and their avatars from the service contact
110 list. This will allow Apertis applications to always display these contacts. When
111 the user attempts to call a chat/VoIP contact while offline, the system may
112 prompt the user to go online and connect that account to complete the action.

113 From a user perspective, the configuration of chat and VoIP accounts within
114 Apertis would be simple. In most cases, just providing a username and password
115 will add that user’s tens or hundreds of service contacts to the local address book.
116 For limited effort, this can significantly increase the ways the user can reach their
117 acquaintances in the future.

118 **Web services**

119 The growing number of web services with social networking is yet another source
120 of contacts for many users. Some services may provide useful contact informa-
121 tion, such as postal addresses or phone numbers. In these cases, it may be
122 worthwhile to include web service contacts (since implementation for some ser-
123 vices already exist within **Folks** and **libsocialweb**).

124 In the case of multi-seat configurations, it may also be worthwhile to support
125 additional web services for entertainment purposes. Potential uses include play-
126 back of contacts'YouTube videos, reading through contacts'Facebook status up-
127 dates, Twitter tweets, and other use cases which do not apply to a driver due
128 to their attention requirements.

129 In general, web services require third parties access their content through a
130 specially-issued developer key. In many cases, this will require to secure license
131 agreements with the provider to guarantee reliable service as their terms of
132 service change frequently (usually toward less access).

133 Our system will cache these contacts and their avatars from the service contact
134 list. This will allow Apertis applications to always display these contacts, even
135 when offline.

136 **SIM Card**

137 Contacts may be retrieved from a SIM card within a vehicle's built-in mobile
138 phone stack. These contacts will be accessible from the Apertis contacts system.
139 However, any changes to these contacts will not be written back to the SIM card.
140 See **Read-only operation for external sources**.

141 **Read-only Operation for External Sources**

142 Modifications of contacts will be limited to **Local sources**. Depending upon the
143 user interfaces created, users may be able to set details upon local contacts
144 which may appear to affect external contacts such as web service contacts or
145 Bluetooth-connected phone contacts. However, these changes will not actually
146 be written to the corresponding contact on the external source.

147 **Standard Behavior and Operations**

148 **Contact Management**

149 Our proposed system will support adding, editing, and removing contacts. New
150 contacts will be added to **Local sources**. Though the **Components** which will en-
151 able contact management already support these features, **User interfaces** needs
152 to be implemented to present these functions to the user. Similarly, contacts
153 will need to be presented as necessary by end-user applications.

154 **Contact Aggregation and Linking**

155 Contacts will be automatically aggregated into “meta-contacts” which contain
156 the sum details amongst all sub-contacts. The criteria for matching up contacts
157 will be:

- 158 • **Equivalent identifier fields** –for instance, two contacts with the email
159 address bob@example.com² or phone numbers “+18001234567” and “1-800-
160 123-4567”
- 161 • **Similar name fields** –for instance, contacts with the full names “Robert
162 Doe”, “Rob Doe”, and “Bob Doe” (which all contain variations of the same
163 given name)

164 This system will be careful to avoid matching upon unverified fields which would
165 allow a remote contact to spoof their identity for the purpose of being matched
166 with another contact. In a real-world example, Facebook contacts may claim to
167 own any chat name (even those which belong to other people). If we automat-
168 ically matched upon this field, they could, theoretically, initiate a phone call
169 and appear to the user as that other person.

170 The user will also be able to manually “link” together any contacts or, similarly,
171 manually “anti-link” any contacts which are accidentally mismatched through
172 the automatic process.

173 Linking and anti-linking will be reversible operations. This will avoid a user
174 experience issue found in some contact aggregation systems, such as the one
175 used on the Nokia N900.

176 **Local Address Book Management**

177 The Apertis contacts system will support adding and removing local contact
178 stores in an abstract way that does not assume prior knowledge of the under-
179 lying address book store. In other words, to add or remove an underlying
180 Evolution Data Server contact database, a client application will be able to use
181 functionality within Folks and, indeed, not even need to know how the contacts
182 are stored.

183 **Search**

184 This contact system will include the ability to search for contacts by text. Search
185 results will be drawn from all available contact sources and will support support
186 “fuzzy” matching where appropriate. For instance, a search for the phone number
187 “(555) 123-4567” will return a contact with the phone number “+15551234567”
188 and a search for the name “Rob” will match a contact named “Robert.”

189 Each type of contact detail field supports checking for both equality (for example,
190 “Alice” “Carol”) and equivalence (for example, the phone number “(555) 456
191 7890” is equivalent to “4567890”). This allows the contact system to add or

²<mailto:bob@example.com>

192 change fuzzy matching for fields without needing to break API or treat certain
193 field details specially based upon their names.

194 **Sorting and Pagination** As a convenience for applications and potentially
195 an optimization, the contacts system will support returning search results in
196 sorted order (for example, by first name).

197 Furthermore, the search system will support returning a limited number of
198 results at a time (“paginating” the result set). This may improve performance
199 for user interfaces which only require a small number of results at once.

200 **Event Logging**

201 Related to the contacts system, Collabora will provide an event logging which
202 logs simple, direct communication between the user and their contacts. Sup-
203 ported events include VoIP and standard mobile phone calls, SMS messages,
204 and chat conversations.

205 Events will include at least the following fields:

- 206 • **User Account ID** –e.g., “+15551234567”, “alice@example.jabber.org³”
- 207 • **Contact service ID** –the unique ID of the contact involved
- 208 • **Direction** –sent or received
- 209 • **Event type** –call, text message
- 210 • **Timestamp**
- 211 • **Message content** –for text messages of any type
- 212 • **Success** –whether the call successfully connected, whether a text message
213 was successfully sent

214 The contact service ID can be used by applications to look up extended infor-
215 mation from the contacts system, such as full names and avatars. These details
216 can then be displayed within the application to provide a consistent view of
217 contacts when displaying their conversations.

218 **Out of Scope** Email conversations will be out of scope due to their relatively
219 large message sizes and their common use for indirect conversations (such as
220 mailing list messages, advertisements or promotions, social networking status
221 updates, and so on).

222 Messages exchanges with web service contacts will not be supported by default.
223 However, the event logging service will allow third-party software to add events
224 to the database. So events not logged by default by the middleware may be
225 added by entirely third-party applications.

³<mailto:alice@example.jabber.org>

226 Caching

227 In general, contact sources will be responsible for maintaining their own cache
228 in a way that is transparent to client applications.

229 **Opportunistic Caching** It may be best to defer bandwidth-intensive opera-
230 tions (such as full contact list and avatar downloads) until the Apertis system
231 can connect to an accessible WiFi network (such as the user’s home or work
232 network).

233 **Open Questions** Will there be a general framework for libraries and ap-
234 plications to check whether network data should be considered “cheap” or “too
235 expensive”? And should the contacts system factor that into its network opera-
236 tions?

237 Most bare contact lists (not including avatars) have trivial data length. For
238 example, my very large Google contacts list of 1,600 contacts only contains 171
239 kilobytes of data. Common contact lists are substantially smaller than that.

240 When factoring in avatars (for the first contact list download), contact list sizes
241 can potentially reach a few megabytes in the worst case. This could be an
242 unacceptable amount of data to transfer on a pay-as-you-go data plan. But
243 at the same time, this is a relatively small amount of data and will only get
244 relatively smaller as data service plans improve.

245 Considering the factors above, would it be worthwhile for the contacts system
246 to support opportunistically caching remote contact lists on bandwidth-limited
247 networks?

248 Components

249 Folks

250 **Folks**⁴ is a contact management library (libfolks) and set of backends for dif-
251 ferent contact sources. One of Folks’ core features is the ability to aggregate
252 meta-contacts from different contacts (which may come from multiple back-
253 ends). These meta-contacts give a high-level view of people within the address
254 book, making it easy to select the best method of communication when needed.
255 For instance, the driver could just as easily call someone by their SIP address
256 as their mobile phone if they prefer it for call quality or billing reasons.

257 The actively-maintained Folks backends include:

- 258 • – **Telepathy** – Chat and audio/video call contacts, including Google
259 Talk, Facebook, and SIP
- 260 – **Evolution Data Server (EDS)** – Local address book contacts
- 261 – **libsocialweb** – Web service contacts, including YouTube and Flickr

⁴<https://wiki.gnome.org/action/show/Projects/Folks>

262 Many of these backends have associated utility libraries which allow client soft-
263 ware to access contact features which are unique to that service. For instance,
264 the Telepathy backend library provides Telepathy contacts, which may be used
265 to initiate phone calls.

266 **Bindings** The Folks libraries have native bindings for both the C/C++ and
267 Vala programming languages. There is also support for binding any languages
268 supported by GObject Introspection (including Python, Javascript, and other
269 languages), though this approach has less real-world testing than the C/C++
270 and Vala bindings.

271 **Required work** As described in [Contact aggregation and linking](#), our system
272 will support automatic linking of contacts as well as anti-linking (for mismatched
273 automatic links). Folks currently supports recommending links but does not
274 yet act upon these recommendations automatically, so this would need to be
275 implemented.

276 Along with this, Folks will need the ability to mark contacts specifically as non-
277 matches (by anti-linking them). There is preliminary code for this feature, but
278 it will need to be completed for this functionality.

279 In order to enable display of chat/VoIP contacts while offline, we will need to
280 implement a chat/VoIP contact list cache within Folks. This will be similar to
281 existing code for caching avatars, but simpler.

282 Similarly, we will need to implement a web service contact cache to display web
283 service contacts while offline.

284 Search functionality in Folks is nearly complete but still needs to be merged to
285 [mainline](#)⁵.

286 Additionally, the ability to perform “deep”searches will require support for
287 [search-only backends](#)⁶.

288 The search functionality will also need to support sorting and pagination as
289 described in [Sorting and pagination](#) before it can be merged upstream.

290 Folks external contact sources will need the ability to be designated as
291 “synchronize-only” or “keep-remote”. Contact sources designated as synchronize-
292 only will be automatically synchronized as necessary (such as when a phone
293 is connected over Bluetooth). Keep-remote sources will not be synchronized
294 to the Apertis system and will only be accessible while the remote source is
295 available (whether over a local or Internet connection).

296 For Folks to access contacts stored on a vehicle’s built-in SIM card, we will need
297 to write an oFono backend to retrieve the contacts from that hardware.

⁵https://bugzilla.gnome.org/show_bug.cgi?id=646808

⁶https://bugzilla.gnome.org/show_bug.cgi?id=660299

298 Abstract contact address book creation and deletion within Folks will require
299 new work.

300 In case **Opportunistic caching** is required for the contacts system, this will need
301 to be added as a new feature to Folks and its Telepathy and libsocialweb back-
302 ends.

303 Support for storing arbitrary data in contacts has not yet been implemented in
304 Folks, but has already been [discussed](#)⁷ and will be implemented.

305 **Out of scope** We recommend application logic for synchronizing an entire
306 address book from a Bluetooth-paired phone be implemented in a new library
307 or application on top of SyncEvolution (which we will provide in our Reference
308 images). The contacts created in this process will automatically be stored as
309 any other local contact.

310 Speech-based search has been identified as a major use case for the address
311 book software in Apertis. The text-based search portion of this use case will be
312 supported by Folks; however, the parsing of audio data into a text for searching
313 will be the responsibility of specific software above the middleware. Global
314 search in general will be covered in the upcoming document “Apertis Global
315 Search”.

316 Collabora recommends to implement the voice search in whole or in part as a
317 service daemon started automatically upon boot. This would allow dependent
318 functionality, including Folks, to be initialized in advance of user interaction.
319 This will be necessary to minimize latency between voice search and the display
320 of results.

321 Support for contact caching for abstract third-party backends certainly would
322 be possible and would likely take the form of a vCard contact store. However,
323 at this time, Collabora recommends not implementing this feature. We would
324 much prefer to delay this until there exist at least two third-party Folks backends
325 with which to test this functionality during development. This is primarily due
326 to the risks involved with committing to an API. Once officially released, this
327 API will need to be kept stable. So it is critical that the API be tested by
328 multiple independent code bases before finalization. Furthermore, at this time,
329 there exist no known third-party Folks backends. In the meantime, third-party
330 backends could still implement opaque contact caches suited to their own needs
331 and migrate to a centralized implementation if and when it is created.

332 Telepathy

333 The **Telepathy**⁸ communications framework, which Collabora created and main-
334 tains, retrieves contacts for many types of chat services, including Google Talk,

⁷https://bugzilla.gnome.org/show_bug.cgi?id=641211

⁸<http://telepathy.freedesktop.org/wiki/>

335 Facebook, XMPP, and most other popular chat services. It also supports sup-
336 ports audio and video calls over SIP, standard mobile phone services, and the
337 previously-mentioned chat services (depending upon provider).

338 **Evolution Data Server (EDS)**

339 Evolution Data Server is a service which stores local address book contacts
340 and can retrieve contacts stored in Google accounts or remote LDAP contact
341 stores. Contacts may contain all defined and [arbitrary](#)⁹ [vCard](#)¹⁰ attributes
342 and parameters, which is a common contact exchange format in address book
343 systems. This allows Folks to store and retrieve contacts with many types of
344 details.

345 EDS is the official address book store for the Gnome Desktop and has been
346 used in Nokia's internet tablet devices and N900 mobile phone. It has been
347 the default storage backend for Folks since Gnome 3.2, which was released in
348 September, 2011.

349 **libsocialweb**

350 In the case that we support web service contacts, libsocialweb will be the compo-
351 nent that provides these contacts through its Folks backend. Note that exactly
352 which web services can be used depends upon both implementation in libso-
353 cialweb and license agreements with those services. See [Web services](#) for more
354 details.

355 **SyncEvolution**

356 [SyncEvolution](#)¹¹ is a service which supports synchronizing address books be-
357 tween two sources. While it supports many protocols and storage services, it
358 best supports synchronizing contacts from a SyncML client over Bluetooth to
359 Evolution Data Server, which will be our primary contact store. Many mobile
360 phones support the SyncML protocol as a means of contact synchronization.

361 This method requires Bluetooth [OBEX](#)¹² data transfer support, which is widely
362 supported by most Bluetooth stacks, including [BlueZ](#)¹³.

363 **Zeitgeist**

364 [Zeitgeist](#)¹⁴ is open source event-tracking software that will serve as the [Event](#)
365 [logging](#) service for Apertis. It is a flexible event store and uses external services

⁹<http://www.ietf.org/rfc/rfc2426.txt>

¹⁰<http://en.wikipedia.org/wiki/VCard>

¹¹<http://syncevolution.org/>

¹²<http://en.wikipedia.org/wiki/OBEX>

¹³<http://www.bluez.org/>

¹⁴<https://zeitgeist.freedesktop.org/>

366 to store their events in a central location. So, by its nature, it supports third-
367 party applications without prior knowledge of them.

368 Zeitgeist is committed to API stability in part because Ubuntu's Unity user
369 interface depends upon it.

370 **Required Work** A simple service to monitor and send Telepathy chat and
371 VoIP call events to Zeitgeist is in progress, so this work will need to be finished
372 and merged upstream.

373 Architecture

374 In our recommended architecture, contacts applications will use libfolks directly.
375 Libfolks, in turn, will use its Telepathy backend for chat and VoIP service con-
376 tacts; Evolution Data Server backend for local contacts, and its libsocialweb
377 backend for web service contacts.

378 Not pictured in is the optional linking between the application and each backend'
379 s utility library (for accessing service-specific contact features).

380 Accessibility of Contacts By Source

381 Contacts within this system are accessible on two levels: Meta-contacts, rep-
382 resenting an entire person, are available for all contacts in the system. Each
383 meta-contact contains at least one contact. For many use cases, applications
384 can work entirely with meta-contacts and ignore the underlying contacts. For
385 use cases requiring service-specific functionality, such as initiating an audio call
386 with a Telepathy contact, applications can iterate through a meta-contact's sub-
387 contacts.

388 Additionally, applications can access contacts for each user account. Each ac-
389 count has a corresponding contact store containing only the contacts for that
390 account. So, an application could be written to display only contacts from sin-
391 gle account or service provider at a time (ignoring any parent meta-contacts if
392 it instead wishes to work in terms of service contacts).

393 User interfaces

394 As Folks and Telepathy are a set of libraries and low-level services, they do
395 not provide user interfaces. There exist a few open source, actively-maintained
396 applications based upon Folks and Telepathy:

- 397 • **Gnome Contacts** –an “address book” application which supports contact
398 management and searching
- 399 • **Empathy** –a chat application which provides a chat-style contact list and
400 both audio/video call and chat handler programs

401 Together, these components provide most contact functionality including:

- 402 • Adding new contacts
- 403 • Editing or removing contacts
- 404 • Browsing/searching through contacts
- 405 • Importing contacts from a Bluetooth-paired phone
- 406 • Initiating and accepting incoming phone calls

407 However, these applications are designed for use on a typical desktop environ-
408 ment and do not suit the needs of an in-vehicle infotainment user experience.
409 We recommend to examine these applications as real-world examples of contact
410 applications which use the components we recommend for the Apertis contacts
411 system.

412 Multiple Users

413 Each user in the system will have their own contacts database, chat/VoIP ac-
414 counts, and web service accounts. Changes by one user will not affect the
415 contacts or accounts of another user.

416 Storage considerations

417 The storage requirements for our proposed contacts system will be very modest.
418 Storage of local address book contacts should be under a few megabytes for
419 even large sets of contacts with up to several megabytes of storage for contacts'
420 avatars.

421 These storage requirements do not factor in files received from contacts.

422 Abstracting Contacts Libraries

423 In general, Collabora discourages direct, complete abstractions of libraries be-
424 cause the resulting library tends to have fewer features, more bugs, and gives its
425 users less control than the libraries it's meant to abstract. Particularly, when ab-
426 stracting two similar libraries, the resultant library contains the “least common
427 denominator” of the original libraries' features.

428 However, partial-abstraction “utility” libraries which simplify common use pat-
429 terns can prove useful for limited domains. For instance, if many applications
430 required the ability to simply play an audio file without extended multimedia
431 capabilities, a utility library could dramatically simplify the API for these ap-
432 plications.

433 As such, Collabora recommends against abstracting Folks or Zeitgeist on a per-
434 component basis as they are designed to be relatively easy to integrate into
435 applications. But, for example, it would make sense to create a library or two
436 which provide widgets based upon these libraries. This could create a contact
437 selector widget based on top of Folks, allowing applications to prompt the user
438 to pick a contact with only a small amount of code.

439 Another recommended widget to add to such a library is a “type-ahead”contact
440 selector as is common in many email applications. As the user types into a
441 “To:”entry field, the widget would the Folks search capabilities to return a list
442 of suggestions for the user to select from.