



Contacts

1 Contents

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

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

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

38 We propose a contact system which uses the Folks contact aggregator to retrieve
39 contacts from multiple sources and automatically match up contacts which cor-
40 respond to a single person. This will give a thorough and coherent view of one's
41 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**

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

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

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

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

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

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

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

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

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

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

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

114 From a user perspective, the configuration of chat and VoIP accounts within
115 Apertis would be simple. In most cases, just providing a username and password

116 will add that user's tens or hundreds of service contacts to the local address
117 book. For limited effort, this can significantly increase the ways the user can
118 reach their acquaintances in the future.

119 **Web services**

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

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

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

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

137 **SIM Card**

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

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

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

148 **Standard Behavior and Operations**

149 **Contact Management**

150 Our proposed system will support adding, editing, and removing contacts. New
151 contacts will be added to [Local sources](#). Though the [Components](#) which will en-
152 able contact management already support these features, [User interfaces](#) needs

153 to be implemented to present these functions to the user. Similarly, contacts
154 will need to be presented as necessary by end-user applications.

155 **Contact Aggregation and Linking**

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

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

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

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

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

177 **Local Address Book Management**

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

184 **Search**

185 This contact system will include the ability to search for contacts by text.
186 Search results will be drawn from all available contact sources and will sup-
187 port support “fuzzy” matching where appropriate. For instance, a search for

²<mailto:bob@example.com>

188 the phone number “(555) 123-4567” will return a contact with the phone num-
189 ber “+15551234567” and a search for the name “Rob” will match a contact
190 named “Robert.”

191 Each type of contact detail field supports checking for both equality (for exam-
192 ple, “Alice” “Carol”) and equivalence (for example, the phone number “(555)
193 456 7890” is equivalent to “4567890”). This allows the contact system to add or
194 change fuzzy matching for fields without needing to break API or treat certain
195 field details specially based upon their names.

196 **Sorting and Pagination**

197 As a convenience for applications and potentially an optimization, the contacts
198 system will support returning search results in sorted order (for example, by
199 first name).

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

203 **Event Logging**

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

208 Events will include at least the following fields:

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

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

221 **Out of Scope**

222 Email conversations will be out of scope due to their relatively large message
223 sizes and their common use for indirect conversations (such as mailing list mes-

³<mailto:alice@example.jabber.org>

224 sages, advertisements or promotions, social networking status updates, and so
225 on).

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

230 **Caching**

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

233 **Opportunistic Caching**

234 It may be best to defer bandwidth-intensive operations (such as full contact list
235 and avatar downloads) until the Apertis system can connect to an accessible
236 WiFi network (such as the user’s home or work network).

237 **Open Questions**

238 Will there be a general framework for libraries and applications to check whether
239 network data should be considered “cheap” or “too expensive”? And should the
240 contacts system factor that into its network operations?

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

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

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

252 **Components**

253 **Folks**

254 [Folks](http://telepathy.freedesktop.org/wiki/Folks)⁴ is a contact management library (libfolks) and set of backends for dif-
255 ferent contact sources. One of Folks’ core features is the ability to aggregate
256 meta-contacts from different contacts (which may come from multiple back-
257 ends). These meta-contacts give a high-level view of people within the address
258 book, making it easy to select the best method of communication when needed.

⁴<http://telepathy.freedesktop.org/wiki/Folks>

259 For instance, the driver could just as easily call someone by their SIP address
260 as their mobile phone if they prefer it for call quality or billing reasons.

261 The actively-maintained Folks backends include:

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

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

270 Bindings

271 The Folks libraries have native bindings for both the C/C++ and Vala program-
272 ming languages. There is also support for binding any languages supported
273 by GObject Introspection (including Python, Javascript, and other languages),
274 though this approach has less real-world testing than the C/C++ and Vala
275 bindings.

276 Required work

277 As described in [Contact aggregation and linking](#), our system will support au-
278 tomatic linking of contacts as well as anti-linking (for mismatched automatic
279 links). Folks currently supports recommending links but does not yet act upon
280 these recommendations automatically, so this would need to be implemented.

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

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

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

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

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

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

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

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

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

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

303 Abstract contact address book creation and deletion within Folks will require
304 new work.

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

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

310 **Out of scope**

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

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

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

326 Support for contact caching for abstract third-party backends certainly would
327 be possible and would likely take the form of a vCard contact store. However,
328 at this time, Collabora recommends not implementing this feature. We would
329 much prefer to delay this until there exist at least two third-party Folks backends
330 with which to test this functionality during development. This is primarily due
331 to the risks involved with committing to an API. Once officially released, this
332 API will need to be kept stable. So it is critical that the API be tested by
333 multiple independent code bases before finalization. Furthermore, at this time,

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

334 there exist no known third-party Folks backends. In the meantime, third-party
335 backends could still implement opaque contact caches suited to their own needs
336 and migrate to a centralized implementation if and when it is created.

337 **Telepathy**

338 The [Telepathy](#)⁸ communications framework, which Collabora created and main-
339 tains, retrieves contacts for many types of chat services, including Google Talk,
340 Facebook, XMPP, and most other popular chat services. It also supports sup-
341 ports audio and video calls over SIP, standard mobile phone services, and the
342 previously-mentioned chat services (depending upon provider).

343 **Evolution Data Server (EDS)**

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

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

354 **libsocialweb**

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

360 **SyncEvolution**

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

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

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

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

¹¹<http://syncevolution.org/>

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

368 **Zeitgeist**

369 [Zeitgeist](#)¹⁴ is open source event-tracking software that will serve as the [Event](#)
370 [logging](#) service for Apertis. It is a flexible event store and uses external services
371 to store their events in a central location. So, by its nature, it supports third-
372 party applications without prior knowledge of them.

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

375 **Required Work**

376 A simple service to monitor and send Telepathy chat and VoIP call events
377 to Zeitgeist is in progress, so this work will need to be finished and merged
378 upstream.

379 **Architecture**

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

384 Not pictured in is the optional linking between the application and each back-
385 end's utility library (for accessing service-specific contact features).

386 **Accessibility of Contacts By Source**

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

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

¹²<http://en.wikipedia.org/wiki/OBEX>

¹³<http://www.bluez.org/>

¹⁴<http://zeitgeist-project.com/>

399 **User interfaces**

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

- 403 • **Gnome Contacts** – an “address book” application which supports con-
404 tact management and searching
- 405 • **Empathy** – a chat application which provides a chat-style contact list
406 and both audio/video call and chat handler programs

407 Together, these components provide most contact functionality including:

- 408 • Adding new contacts
- 409 • Editing or removing contacts
- 410 • Browsing/searching through contacts
- 411 • Importing contacts from a Bluetooth-paired phone
- 412 • Initiating and accepting incoming phone calls

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

418 **Multiple Users**

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

422 **Storage considerations**

423 The storage requirements for our proposed contacts system will be very modest.
424 Storage of local address book contacts should be under a few megabytes for
425 even large sets of contacts with up to several megabytes of storage for contacts’
426 avatars.

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

428 **Abstracting Contacts Libraries**

429 In general, Collabora discourages direct, complete abstractions of libraries be-
430 cause the resulting library tends to have fewer features, more bugs, and gives
431 its users less control than the libraries it’s meant to abstract. Particularly,
432 when abstracting two similar libraries, the resultant library contains the “least
433 common denominator” of the original libraries’ features.

434 However, partial-abstraction “utility” libraries which simplify common use pat-
435 terns can prove useful for limited domains. For instance, if many applications

436 required the ability to simply play an audio file without extended multimedia
437 capabilities, a utility library could dramatically simplify the API for these ap-
438 plications.

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

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