# All.Net Analyst Report and Newsletter

***Welcome to our Analyst Report and Newsletter***

## A good day for the GAIs

It actually took 2 days for the most part – occasional updates before that, and lot of prior work, but contrary to my previous articles about the issues with vibe coding, this example is a huge win for it. I have tried to make it easy, but it's complicated…

## What was done between Friday late and Sunday evening

The final (as it stands now in use) product has approximately the following features:

- Access via an external Web site using Stunnel connected to a reverse ssh tunnel from a machine behind a NAT gateway using internal sockets for all connections.
- Login and user administration including Roles to Capabilities with Exceptions amd fine-grained control over functions and visible interface components on a user-by-user basis, and ActAs capabilities so managers can act on behalf of their workers with audit trails reflecting the actor and surrogate..
- Import and conversion of databases, spreadsheets, and other related information from multiple sources coming to more than a Gigabyte of content fused into about 10% of that size, compared to current Web-based information and consolidated into a new set of databases built for the application.
- Conversion of an existing application for managing CRM and related mechanisms into a bespoke version and integrated with the data sources and other capabilities.
- Workflows, templates, and related mechanisms integrated with database-drive and optionally AI augmented analysis and generation of communications.
- Backup and restore, import and export, including integrating new information with old, finding potential redundancies and consolidating them, and integration into workflows delegated to users by managers.
- A complete contextual Help system, manual, and description (in the appendices).

I also slept 8 hours each night, went for walks, went clothes shopping, had a meal out with my wife, and watched a few movies. It is a total of about 12,000 lines of code, all javascript.

It wasn't perfect and I had a few things that needed to be improved, and there will no doubt be some maintenance issues over time.

## It wasn't an accident

A lot of things were already in place, and I have some skills.

- The Stunnel was already running on a server already present with automation for generating and handling new SSL keys for new domains, and the DNS changes only had to add the new domain names to the existing DNS and server mechanisms. That that took a long time to complete several years ago, but today it would likely take a few hours from scratch to create the server, DS setups, automatic SSL certificate updates, etc.

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

- The server already ran ssh, so no install was required, and that would have taken at least a minute or two more from scratch. Make it 5-10 with the setups for automated remote access, etc.
- The CRM-like mechanism was in the process of being converted from pl to js by an AI engine and mostly done. The application took a long time to develop and hone over years of use, the conversion took about a week of spare time, likely 6 hours of continuous effort would have been required if done all at once.
- The databases and historical records were already there in usable format (XSL, ODS, CSV, etc). A conversion of a new dataset takes about 30 minutes to build the custom part for a new data source, and it pretty much runs automatically from there.
- The internal server was already running MX Linux behind a NAT gateway. Installation of such a system takes perhaps 30 minutes of realtime but only a few minutes of user interaction. Completing the setup after OS installation for my use cases and standard capabilities takes another 15 minutes or so.
- The dev rules (in an appendix) were already tried and true.
- I was not trying to be unusual or clever, I just wanted to use things the vibe coding world already knew how to do. The dev was very similar to things others have done before using very similar methods in very similar ways.
- I had already working examples of most of the piece parts, so I already knew what and how for almost all decisions.

**But that should not take away from the GAI accomplishment.**

I have implemented similar systems over the period of weeks to months, but to do this much over a period of 2 days is indeed an amazing accomplishment of modern technology. While I am sure there are errors remaining to be found, and the code is not as bespoke as I might like it, these minor changes will come over the following days and weeks with only a few hours of effort here and there.

For example, I am now turning it into a resilient system across multiple servers with real-time updates so users can use it from different servers and their changes will automatically be integrated across the server base within seconds to minutes, and recovery from local outages will automatically update the local outage system to the global state as part of coming back up. Similarly, viral propagation of code and content changes will allow partitioned portions of the mesh to become resynchronized as soon as they are rejoined.

**Conclusions**

For the sorts of things that are done quite often, for code bases of less than say 30,000 lines, for fairly standard approaches to databases, web presence, etc., once you have a reasonably well developed baseline of capabilities and mechanisms, a single person using GAIs can now build and deploy usable systems in a few days that would have taken months for a small team only a year ago. You could probably recreate it in an hour or two from the appendices.

It has taken me most of a year of effort to test and become proficient at this and 50 years before that understanding and doing all the things that make me who I am in terms of being a code monkey. But this is a revolution in coding that will realize vast contemplated capabilities.

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

**Appendices:**

Some rules about working together:

- 1) When I ask a question, just answer it.
- 2) Don't code till we agree on what is to be built.
- 3) Make as few changes as necessary to get the job done.
- 4) No python ever.
- 5) We are working on a small part of a large system, do what other parts did for the same things.
- 6) JSON stringified communications operates through existing comms channels and interactions with localStoreage and servers go through standard paths – don’t change them without permission.
- 7) Only change files I authorize - I will not pass other changes.
- 8) Don’t guess, instrument (debug calls with the ability to disable them using a variable).
- 9) I can type things for you to test things in the operating environment - ask me to.
- 10) if you want logs, ask for them.
- 11) Always keep and update version numbers for each file and log them when they start up so we can tell what is actually running.
- 12) LF violations - see lf.js and never use dimmed, small fonts, or other colors - for emphasis you can use bf and it and in some cases like buttons, inverted colors.
- 13) When making changes to files of more than minimal complexity, ALWAYS use line number for changes so as to not screw them up.

**The MD file describing the graph structure:**

```
# SSeeds graph structure
VERSION: 1.0.61  (update with every change; line numbers refer to the file versions
in the table)

## Files
| File | Version | Role |
|---|---|---|
| SSeeds-import.js | 1.0.4 | Importer: .xlsx -> SQLite (CLI, and run by SSeeds-
ui.js with --json). No npm deps (fs, path, zlib, node:sqlite) |
| SSeeds-index.js | 1.0.0 | Search's text indexes (FTS5 + triggers) in the three
databases: INDEXES (single list, exported), --check, --rebuild; run as <app> by the
root step. No npm deps |
| SSeeds-install.sh | 1.0.23 | Installer, run as fc in the installation directory.
Node >= MIN_NODE system wide (one printed root step if missing/old); `ui <app>`:
checks /home/<app> and prints ONE root step with only the lines still needed
(user+home on first install, changed program files, missing config/data, admin on
first install) plus the service-script lines and restart, which are always included
(safe to repeat). Nothing printed when all files match and the socket answers |
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

| SSeeds-ui.js | 1.0.21 | Web UI core (no npm deps): auth, users, search, SQL, AI ask, investor match, history, spreadsheet import (admins), module loader; Unix socket + optional LAN port |
| SSeeds-access.js | 1.0.7 | Module: company access (managers/users), activity log, ActAs; full read is now the capability AllCos |
| SSeeds-access-client.js | 1.0.5 | Browser part of SSeeds-access.js: My companies, Activity, people on a company, assign results |
| SSeeds-ui.html | 1.0.17 | Page markup + layout CSS only (colors/fonts from lf.js); 1.0.6: table cells in the modal keep line breaks |
| SSeeds-ui-client.js | 1.0.19 | Browser code; DOM built with textContent only |
| lf.js | 1.0.85 | Shared Look & Feel (unchanged copy from the system) |
| SSeeds-gigs.js | 1.0.19 | Module: pipelines (Types, Gigs, history, Sieves, checked actions) |
| SSeeds-gigs-client.js | 1.0.22 | Browser part of SSeeds-gigs.js: Pipelines tab, item page (hook "item" for other modules), sieve Settings, company Pipelines section |
| SSeeds-events.js | 1.0.8 | Module: events (pasted sign-up addresses -> companies -> manager + pipeline items) |
| SSeeds-events-client.js | 1.0.7 | Browser part of SSeeds-events.js: Events tab, event page (hook "event" for other modules, S.openEvent), linking unmatched addresses |
| SSeeds-meet.js | 1.0.6 | Module: meetings of an event (Google Meet transcript/chat -> people -> companies -> Meeting notes on items, GWiz(TM) offers) |
| SSeeds-meet-client.js | 1.0.2 | Browser part of SSeeds-meet.js: Meetings on the event page, add meeting, review, link unmatched |
| SSeeds-help.js | 1.0.1 | Module: contextual Help - serves SSeeds-help.html at /api/help (read on every request, no login needed) |
| SSeeds-help-client.js | 1.0.2 | Browser part of SSeeds-help.js: Help button (top bar + every window), F1, 90% overlay, whole manual |
| SSeeds-help.html | 1.0.35 | The manual = the single source of all help text; one section[data-help] per screen/window |
| SSeeds-graph.md | 1.0.48 | this file |
| SSeeds-team.js | 1.0.6 | Module: our team (names, addresses, other spellings, our domains) - never matched to companies anywhere; members linked to accounts, managers, title/about/links/signature (email sender), manager notes |
| SSeeds-team-client.js | 1.0.12 | Browser part: Team tab (everyone: their own team; managers edit theirs, admins all): list with Roles/Exceptions by manager, a person beside the list (roles, exceptions, history), Roles grid |
| SSeeds-roles.js | 1.0.2 | Module: roles (bundles of catalog items), who holds them, exceptions, history and put back; kept records screen routes; migration of the old flags |
| SSeeds-match.js | 1.0.1 | Module: who is who at an event - rules, AI pass, apply (links sign-ups and meeting people) |
| SSeeds-match-client.js | 1.0.2 | Browser part: Next steps box on the event page, Match people & companies window |
| SSeeds-companies.js | 1.0.6 | Module: one company, many records - proposals (pair links), merge, split, kept apart; api.canon(code) |
| SSeeds-companies-client.js | 1.0.6 | Browser part: Companies tab (admins: Apart ticks, Keep apart, Kept apart list), Same company box in company details |
| SSeeds-mail.js | 1.0.9 | Module: AI emails and summaries - templates with Goal/kind/style/creativity/offerings/ticked information, house facts, offerings, audience per company; analysis then writing, rewrite, Add to Note |
| SSeeds-mail-client.js | 1.0.9 | Browser part: Draft email window (goal, next steps, audience, information ticks, analysis, rewrite/undo, copy, Add to Note),

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

Templates tab (house, offerings, templates with Insert) |
| SSeeds-backup.js | 1.0.3 | Module: Backup (by hand, to /tmp/SSeeds/Backup-<local time>) and Restore (safety backup, note, stop; ui.js restores at start) |
| SSeeds-backup-client.js | 1.0.2 | Browser part: Backup tab (admins): Back up now, list, Restore |
| SSeeds-library.js | 1.0.3 | Module: Gigs libraries (perl GigsLibrary/TimeLines): import, edit, add, use on a sieve (new type + B/E/D items) |
| SSeeds-library-client.js | 1.0.3 | Browser part: Libraries tab (admins and managers) |
| SSeeds-investors.js | 1.0.4 | Module: investor intelligence - our investor dataset (SSeeds-investors.sqlite), triage of investors for a company, AI closer look, investor details and notes; 1.0.3: partners' list as a dated source, import of investor datasets, reconciliation (InvAdmin) |
| SSeeds-investors-client.js | 1.0.4 | Browser part: Investors for a company and Find investors in the Investor match tab, investor details (2022 data / partners' list / web site, earlier values, merged), Find investors buttons; Investors tab (import, reconcile) |
Publishing (https://<app>.<base domain>/) is FEBE's job: FEBE-BEInstall.sh / FEBE-FEInstall.sh, separate directory.

## Installed layout (app <app>, e.g. sseeds)
| Where | What | Owner/mode | Made by |
|---|---|---|---|
| installation dir (fc) | SSeeds-<app>.json, SSeeds-<app>.init, SSeeds-<app>-root.txt, SSeeds-node-root.txt | fc | SSeeds-install.sh |
| /home/<app> | program files (import, ui.js, html, client, lf.js) | <app> 644 | root step |
| /home/<app>/SSeeds-ui.json | config (socket = /home/<app>/FEBE-<app>.sock) | <app> 600 | root step (only if missing) |
| /home/<app>/SSeeds.sqlite (+ -wal, -shm) | companies database (WAL mode) | <app> 600 | copied once by root step, or created empty by SSeeds-ui.js at start |
| /home/<app>/SSeeds-app.sqlite | users, sessions, ai_settings, investors, history | <app> 600 | SSeeds-ui.js |
| /home/<app>/SSeeds-ui.secret | AES key material for users' API keys | <app> 600 | SSeeds-ui.js |
| /home/<app>/SSeeds-uploads/<batch>/ | spreadsheets imported through the page | <app> 700/600 | SSeeds-ui.js newBatch :487 |
| /home/<app>/FEBE-<app>.sock | web UI socket (stunnel PROXY header accepted) | <app> 660 | SSeeds-ui.js listenSocket :825 |
| /home/<app>/SSeeds-<app>.log / .pid | service loop log / pid | <app> / root | /etc/init.d/SSeeds-<app> |
| /etc/init.d/SSeeds-<app> (+ update-rc.d links) | start/stop/restart/status; restarts node 5 s after it ends | root 755 | root step |

## Call flow (SSeeds-import.js)
```
main(argv) :465   [--json -> RESULT_TAG line per file, DONE_TAG line at end :464]
 ├─ collects .xlsx from args/folders (skips ~$ lock files), sorted by name
 ├─ openDb(file) :277 ── WAL; RENAMED :93 (export_date/source_file -> last_*); adds missing COLUMNS/DERIVED; indexes; FTS; columns_map
 └─ per file: importFile(db,file,stats) :327 -> counts {inserted, updated, skippedOlder, firstMoved, rows, export_date, date_from}
      │   [one transaction per file; ROLLBACK on any throw]
      ├─ readXlsx(file) :171 ── zipOpen :100, relsMap :151, attrs :140, xmlDecode :133, textRuns :145, isDateFormat :164, colIndex :161
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
     ├─ exportDateOf(file,wb) :320  (filename YYYY-MM-DD > docProps modified > file
mtime)
     ├─ loadMapping(db) :313 / addAutoColumn :444
     ├─ cellPlain :234 -> excelDateToIso :229; toNumber :267, parseCapital :262,
splitMulti :241, parseFiles :246
     └─ warn :453 (also collects up to MAX_FILE_WARNINGS :452 per file for --json)
-> colLetter :459
exports: openDb, main, RESULT_TAG, DONE_TAG, readXlsx, parseFiles, parseCapital,
excelDateToIso, VERSION
```

```
## Constants (single sources of truth)
| Name | Line | Used by | Notes |
|---|---|---|---|
| VERSION | 8 | startup log | debug number only bumped by me |
| KEY_HEADER | 24 | importFile | 'Code' -> companies.code (UNIQUE) |
| COLUMNS | 25 | openDb, columns_map | [header, column, type, fts] |
| DERIVED | 89 | openDb, importFile | app_id, dealroom_url, capital_min/max,
first_source_file, first_export_date, last_source_file, last_export_date,
imported_at |
| RENAMED | 93 | openDb | old -> new column names (1.0.1 databases) |
| INDEXED | 94 | openDb | b-tree indexes incl. first/last_export_date |

## Rules implemented (importer)
- last_*: newest export. Row skipped (fields not updated) if stored
last_export_date > file date; equal -> update.
- first_*: earliest export. Set on insert; moved earlier whenever an older file is
imported (also when the row is otherwise skipped).
  Import order does not matter: the result is min/max over all files ever imported.
Rows from 1.0.1 databases start with first_* empty until re-imported.
- Errors (not xlsx, no key column, empty key) -> whole file rolled back, others
continue, exit code 1.
- Warnings (new column, non-numeric int/real, unparsable capital, duplicate key,
file entry w/o URL, error cells, extra sheets, no filename date) -> import
continues.

## Database objects (companies DB, openDb :277)
- companies (id PK, code UNIQUE, COLUMNS non-multi/files, DERIVED, auto columns)
- company_values PK(company_id, field, value); company_files; columns_map;
ix_companies_<INDEXED>
- companies_fts (FTS5 external content) + triggers _ai/_ad/_au

# Web UI (SSeeds-ui.js 1.0.5)

## Request path
```

```
browser --HTTPS--> FE stunnel [sniNN] (protocol = proxy) --> FE /home/<app>/FEBE-
<app>.sock
  --> reverse ssh (FEBE-<app> service on BE) --> BE /home/<app>/FEBE-<app>.sock -->
proxyAware :782 --> HTTP :780 --> handle :748
optional: localListen (LAN, plain HTTP) --> handle :748
```

```
## Call flow
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
startup (require.main) : IMP.openDb(companiesDbPath).close()  (schema upgrade /
create empty) -> companiesDb :105 (read-only) -> startServers :807
handle :748  static (STATIC :540) | route match (route :572, raw routes skip
readBody :542) | x-sseeds header on non-GET | sessionUser :556 | admin check
 ├─ /api/search  -> buildSearchSql :316 (date_field whitelisted by
DATE_FIELDS :175; LIST_COLS :310) -> runSelect :133
 ├─ /api/sql     -> runSelect :133 -> checkSelect :115 -> child process
QUERY_SRC :126 (read-only, SIGKILL at queryTimeoutMs)
 ├─ /api/meta    -> meta :161 (facets, columns, count, dateRange, dateFields per
DATE_FIELDS)
 ├─ /api/ask     -> startJob :292 -> askFlow :334 -> schemaForAi :179 (explains
date_added vs first/last export) -> aiChat :239 -> runSelect
 ├─ /api/investor/match -> startJob -> investorFlow :401
 ├─ /api/import/limits | batch | upload/<batch>?name= (raw) | start   [admin
only, :730-745]
 │     newBatch :487 -> uploadDir/<YYYYMMDD-HHMMSS-hex6>/ ; batchDir :482 (id regex
BATCH_RE) ; safeXlsxName :493 ; receiveFile :498 (uploadMaxMb)
 │     start -> one import at a time (IMPORT_RUNNING :480) -> startJob ->
importFlow :508
 │              -> spawn node SSeeds-import.js --json <companiesDb> <batch dir>;
RESULT lines -> job steps + result.files, DONE -> result.summary
 ├─ /api/job/:id -> JOBS :291 (owner only; finished jobs purged after 1h, running
ones kept)
 ├─ /api/ai*, /api/users*, /api/login (unchanged)
CLI :842  adduser <name> [--admin] | passwd <name> | users
```

```
## Config (CONFIG_DEFAULTS :26, file SSeeds-ui.json, partial files allowed)
socket, localListen, companiesDb ('SSeeds.sqlite'), appDb, title, maxRows,
queryTimeoutMs, sessionDays, uploadDir ('SSeeds-uploads'), uploadMaxMb (200)

## Browser (SSeeds-ui-client.js 1.0.2)
api :30 (JSON body, or raw octet-stream for a Blob/File) ; TABS :57 (Import + Users
admin only) ; show :59
search: loadMeta :189 fills #s-dfield from meta.dateFields ; gatherFilters :213
sends date_field only with a date ; doSearch :225
import: IMP_FILES :374 ; drop area #imp-drop (dragover/drop) + #imp-file chooser ->
addImportFiles :380 (xlsx only, size limit, no duplicates)
        renderImportList :392 ; doImport :405 (batch -> upload files one by one ->
runJob :169 /api/import/start) ; showImportResult :422 ; META reset after import
window dragover/drop prevented so a missed drop does not navigate away.

## Dependents / change checklist
- Renaming companies columns: add to RENAMED; check SSeeds-ui.js LIST_COLS,
CARD_COLS, DATE_FIELDS, schemaForAi, PRIVATE_COLS.
- The importer's --json output (RESULT_TAG/DONE_TAG lines and their fields) is read
by SSeeds-ui.js importFlow and SSeeds-ui-client.js showImportResult.
- SSeeds-install.sh check_app compares program files in /home/<app> with the
installation directory; new program files must be added there and to the root
step's install line.

# Modules (SSeeds-ui.js 1.0.5)

## Loader (server)
MODULE_FILES (load order: access, team, gigs, companies, events, meet, match, help
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

- each may use the ones before it); MODULE_API :852 / core.moduleApi(name) :862 -> loadModules :846, called at server start after the companies DB check.
- module file: module.exports = function(core){ ...; return {name, version, client, api} }
- core given to a module: APP, one/all/run, log/dbg, nowIso, httpErr, send, CFG, DIR, route (staged), companiesDb, coAll, coIdsForCodes :130,
  companiesByIds :136, companiesByCodes :141, runSelect :193, startJob, scopeOf, canSeeCompany, canExport, activity.
- Routes a module registers are kept aside and added only if the module loads completely. A module that throws is switched off:
  logged ("ERROR module ... switched off"), listed in /api/me .failed (admins only, shown in the page header), the rest keeps working.
- m.client is added to STATIC and listed in /api/me .modules; the page loads it.
- Each module creates only its own tables (additive).

## Access fallback
ACCESS_FALLBACK :104 / ACCESS :112: while the access module is not loaded - admins see everything, everyone else no companies, no bulk output,
activity only to the debug log, no company access info. Core delegators :113-117 (scopeOf, canSeeCompany, canExport, activity, managedCodes).
sessionUser selects u.* so the core does not depend on module columns (full_read). /api/export :813 stays in the core (uses ACCESS.canExport).

## Loader (browser, SSeeds-ui-client.js 1.0.4)
TABS :58 [id, label, who, onShow]; registerTab :448 ({id, label, who admin|mgr, section element, onShow, before}).
HOOKS :443 results(box, rows, opts) | company(wrap, company, data, table) | userRow(user, refresh) -> element | logout(); runHooks :444, hookSpan :447.
loadClientModules :453 after /api/me, before login view; a script that fails to load is reported, the rest works.
window.SSeeds :463 = {api, el, clear, status, show, modal, closeModal, companyDetails, renderResults, nice, link, me(), registerTab, hooks}.

# Module access (SSeeds-access.js 1.0.0 / SSeeds-access-client.js 1.0.0)

## Model
- Admin (users.is_admin): everything. Can also be assigned companies.
- company_access(company_code, user_id, role manager|user, granted_by, granted_at) PK(company_code,user_id): several managers/users per company.
- Manager of a company: sees it, assigns users to it, grants full read to users they manage (managedUserIds :32), exports its rows. Only admins assign managers.
- users.full_read (column added by the module): whole companies database read-only.
- canExport :43 - admins; managers only when every row is a company they manage. The page asks POST /api/export before building a CSV.
- activity :50 -> activity table; calls from the core (login/logout/password, search, sql, ask, investor, company viewed, users, AI settings, import, export)
  and from the module (access granted/removed, full read granted/removed).

## Scope
scopeOf :36 -> {all:true} (admin or full_read) | {all:false, codes, ids}. Core runSelect :193 requires a scope; restricted queries run in QUERY_SRC :162 on a private :memory: copy of only those companies (companies, company_values, company_files, columns_map, rebuilt companies_fts) - the real file is closed first.
meta(scope) :222, schemaForAi(scope), askFlow/investorFlow, /api/company/:id (canSeeCompany :41, companyAccess :55).

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
## Routes (module)
/api/my/companies :65 | /api/people :71 | /api/access :78 |
/api/people/:id/full_read :106 | /api/activity :114

## Browser (module)
registerTab My companies (before Ask) :13, Activity (admins+managers, before
Settings) :18; hooks results (assignBar :54), company (accessBox :68),
userRow (full read button), logout (PEOPLE cache). loadMyCompanies :97,
loadActivity :112.

## Tested
API roles + SQL bypass attempts; browser as admin/manager/user; module broken on
purpose (syntax error) -> server up, admins normal, users see no companies,
page without its tabs; installer update from stage-1 (built-in) to modular with
live data - assignments, activity, sessions kept.

# Core additions for modules (SSeeds-ui.js 1.0.6)
- sessionUser :633 returns ACCESS.applyActAs(user) (fallback: unchanged user).
While acting: user = the person acted as, user.real = the real person.
- own(user) :122 = user.real || user: password change (refused while acting), AI
settings / API key (always the real person's).
- SCOPE_EXTRA :114 / extraScopeCodes :115: modules add company codes a user may
read (core.addScopeCodes, staged like routes);
  the access module's scopeOf adds them.
- core passed to modules also has: extraScopeCodes, addScopeCodes, access() (live
ACCESS), own.
- client: HOOKS.login(user) (after tabs are built); header shows "<real> as
<user>".

# ActAs (SSeeds-access.js 1.0.1)
- act_as(token_hash PK -> sessions ON DELETE CASCADE, target_id, started_at);
activity.actor_id added.
- canActAs :69: admins -> anyone (not themselves); managers -> workerIds :66 (role
user on a company they manage, not admins).
- applyActAs :73 every request; drops the row if no longer allowed. whoName :62
"<real> as <target>" in activity and gig history.
- routes: /api/actas/options :90, /api/actas :96, /api/actas/stop :103. Activity
for managers also shows actor_id = them.
- browser: header "Act as…" list (admins/managers) or "Acting as X [Stop acting]";
changes reload the page.

# Module gigs (SSeeds-gigs.js 1.0.0 / SSeeds-gigs-client.js 1.0.0)
## Tables (additive)
gig_types(id, name UNIQUE NOCASE, statuses, selects, hidden, defaults JSON,
position) - seeded "General" with DEFAULT_STATUSES :16 / DEFAULT_SELECTS :17 /
DEFAULT_NEW :18
gigs(id, company_code NULL=Default Sieve, owner_id, type_id, status, later,
details, poc, fusion, email, tel, tags, links, note, who, owner (last editor),
selects,
     quality, priority, value, archived, created_at, created_by, copied_from)
gig_users(gig_id, user_id) - people on a pipeline | gig_history(gig_id, at,
user_id, actor_id, username, field, value) - append-only
gig_checked(user_id, gig_id) | gig_prefs(user_id, prefs JSON: PREF_KEYS :23)
## Rules
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

visibleWhere :60 / canSee :66 - admins all; company_access on the company (manager or user: all its pipelines); gig_users (that pipeline);
Default gigs: owner + owner's managers. canManageGig :71 - admins; company managers; managers of a Default owner (start, assign, bulk output).
Scope extension :98 - a gig_users assignment lets the person read that company (Search, company details).
Edit :205 - FIELDS :21 diffed; every change -> hist :77 (one row per field); Owner = whoName; Type move keeps history.
Copy :228 - new gig, copies all history rows, adds "Copied from/to" notes on both.
Bulk :274 - next|shift|status|type|note|archive,
validated for all items first, then one transaction. Export :301 - checked + logged (browser builds CSV / email list).
## Routes
types :101/:106(admin)/:123(admin, unused only) | sieves :130 | list :145 (sieve all|default|co:<code>|user:<id>, type, mode Yes|No|Arc)
item :170 | create :179 | edit :205 | copy :228 | people :249 | checked :268 | bulk :274 | export :301 | prefs :309/:312 | company :319
## Browser
P :23 prefs (saved to /api/gigs/prefs); COLORS :14 = Gigs.js colors (black text); dateClass :29; matches :51 (all/any terms, "!" inverts);
sortKey :59 (Status/Quality/Priority by order); render :130 (columns, totals footer as gigSummary); renderBar :166 (checked actions, Email list/CSV
for admins+managers); openGig :213 (edit form, people, history); company hook :267; Types tab :286 (typeForm :295).
## Tested
API: types, visibility per role, Default Sieve privacy, pipeline assignment -> company read, edits+history, copy+history, bulk (all-or-nothing),
export rules, ActAs (allowed/refused, logs, no password change, admin pages closed while acting). Browser: manager list/edit/bulk/ActAs; admin Types/company.
Isolation: gigs broken -> rest works; access broken -> gigs refuses (needs access), core fail-safe. Installer update with live data.

# Module APIs used between modules (stage 3)
- access 1.0.2 api.grantAccess(byUser, code, target, role) :87 - upsert + "access granted" log; /api/access uses it too.
- gigs 1.0.1 api {createGig :179 (no permission check, no transaction), hist, typeById, today} :333; POST /api/gigs uses createGig.
- gigs client: S.openGig (item page) for other page modules.

# Module events (SSeeds-events.js 1.0.0 / SSeeds-events-client.js 1.0.0)
## Tables (additive)
events(id, name, date, type_id, manager_id, created_by, created_at)
event_emails(id, event_id, email, company_code NULL=unmatched, gig_id, added_to_existing, linked_by, linked_at)
## Rules
parseEmails (EMAIL_RE): any separators, lower case, duplicates dropped; up to MAX_EMAILS 5000. matchEmails :38 - lower(trim(companies.email)).
Create (admins and managers) :67 - one transaction. manager = chosen by an admin, otherwise the creator. Per matched company applyToCompany :48:
  grantAccess(manager, manager role) - the admin creating the event does not get the companies; open item of that type? add "Event" to its history,
  else createGig (owner = manager, Details = "<name> <date>", Event in history).
Unmatched addresses kept; link later :122 (same steps).
Visibility :56 - admins all; creator or manager of the event. Company finder for linking :115 (name/code/email).

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
## Routes
/api/events/preview :60 | /api/events :67 (create), :97 (list) |
/api/events/:id :106 | /api/events/find :115 | /api/events/:id/link :122
## Browser
Events tab (admins+managers, :25): name, date, pipeline type (defaults to a type
named like "angel"), manager (admins only), paste box,
Check addresses (preview), Create event; list; event page :70 with matched (open
item / company) and unmatched (find + link).
## Tested
Parsing (names, quotes, separators, duplicates, case); matching; admin names
another manager -> only that manager gets companies; manager
creates for self; existing open item gets the event; linking; users refused.
Isolation: events broken -> rest works; pipelines broken ->
events refuse (need it). Full install via installer with all module files and
existing data.

# Module meet (SSeeds-meet.js 1.0.0 / SSeeds-meet-client.js 1.0.0) - stage 3b
## Uses
events 1.0.1 api {applyToCompany, getEvent, requireAdminOrManager, label,
parseEmails} (SSeeds-events.js :139); gigs api {hist, typeById, createGig via
events}.
Browser: events client hook S.hooks.event(box, event, reopen) run on the event page
:97, S.openEvent :71.
## Tables (additive)
meet_staff(name PK nocase) | meetings(id, event_id, title, transcript, chat, hash,
gwiz_status, gwiz_select, created_by, created_at)
meet_people(id, meeting_id, name, notes, gwiz, company_code NULL=unmatched, gig_id,
existing, linked_by, linked_at)
## Rules
parseTranscript: Google Meet text download (title line with date, Attendees,
Transcript, "Name: text", 00:05:00 marks; speaker names taken from
  the attendee list first). parseChat: "hh:mm:ss.mmm,hh:mm:ss.mmm" then "Name:
text" + continuation lines; "I (Name)" -> Name.
splitTranscript: per non-staff speaker, their largest stretch (lines no more than
GAP 15 entries apart); their part = that stretch, starting
  after the end of earlier parts (interjections in others' parts don't steal them);
hosts' lines inside stay with them.
GWiz: a staff line in the part matching GWIZ_RE (free G whiz / free G-W whiz / free
whiz / 3G whiz / free GWiz) not negated (not/never/n't before it).
Candidates (score): email in what they wrote 100 > email/website domain 80 (free-
mail and link sites ignored) > contact first+last name 60 >
  company name = speaker name 50 > company named in what they said 30 (1-4 word
grams vs normalized names/legal names, suffixes dropped).
  Proposed = single best; ties -> none (user picks).
Apply (one transaction; texts read again on the server; same texts twice on one
event refused by hash): per person company -> event's manager
  + item (reuses the item the event already gave this company, else
events.applyToCompany), "Meeting" history = their part + chat;
  GWiz offered and status/checkbox chosen -> set (history). Unmatched stored; link
later (same steps, same GWiz choice). Skip -> not stored.
Staff names: admins edit (POST /api/meet/staff, /api/meet/staff/add),
admins+managers read.
## Routes
/api/meet/staff GET POST | /api/meet/staff/add | /api/meet/read | /api/meet/apply |
/api/meet/event/:id | /api/meet/people/:id/link
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
## Tested
Real transcript (474 entries, 13 presenters) + chat: parts correct, GWiz flagged
for AA Create, Chris Messina, Ricki Samlal, Alfred Gales, not
ACC ("not going to give you a free whiz"). Matches by email, domain, company name;
repeated apply refused; user refused; manager link sets
GWiz status; event's existing item reused (no second Event entry). Browser: add,
review, apply, notes, open item, no errors.
Isolation: meet broken -> rest works; events broken -> meet switches off.

# Module help (SSeeds-help.js 1.0.0 / SSeeds-help-client.js 1.0.0 / SSeeds-
help.html 1.0.0)
## Rule (applies to every future change)
Every change to a screen or window updates its section in SSeeds-help.html in the
same delivery. The manual is the only place help text lives.
## Keys (section data-help in SSeeds-help.html)
overview, login, gigs, item, events, event, meeting, search, results, company,
mycos, ask, investor, sql, history, activity, settings, gigtypes, import, users,
roles, glossary.
Tabs map by their section id (v-<tab id>); windows by a data-help mark on the
content given to S.modal:
  company (ui-client :173), history (ui-client :336), item (gigs-client :262),
email list -> gigs (gigs-client :201), event (events-client :98), meeting (meet-
client :116).
  New windows/tabs: add the mark and a section, or Help falls back to overview.
## Server
GET /api/help (access none) -> {version, html}; file read per request (max 2 MB);
module refuses to load if the file is missing (switched off, rest works).
## Browser
Help button before Look & Feel; a Help button added next to Close [Esc] in every
window (MutationObserver on #modal-box, since the window's
backdrop covers the top bar); F1 anywhere. Overlay #help-ov (z-index 300 above the
window's 200), box 90vw x 90vh, body scrolls.
Context = open window's data-help, else current tab, else overview. Whole manual =
intro + all sections; #links scroll there (whole manual) or switch part.
Esc (capture listener) closes only the help; the window under it stays. Layout CSS
injected by the client (no colors; lf.js variables).
## Tested
Login screen, 10 tabs, item and company windows show their parts; overlay 90% x
90%; whole manual 22 sections; contents link scrolls; back;
Esc closes help only; F1; no console errors. Isolation: help broken or SSeeds-
help.html missing -> help off, everything else works.

# Module companies (SSeeds-companies.js 1.0.0 / SSeeds-companies-client.js 1.0.0)
## Table (additive)
co_groups(code PK = a merged record, main_code, reason, merged_by, merged_at). A
main record is never in it.
## Core additions (SSeeds-ui.js 1.0.11)
SEARCH_FILTERS / core.addSearchFilter(fn(rows) -> rows), applied to /api/search
results only (Ask/SQL show all records).
## Rules
Proposals (whole list, union-find over all records; merged records start in their
group):
  strong: same email (not free mail) | same domain (email domain not free, or
website host not a link site) | same name + same contact |
  same personal address + same name. name-only (normalized name/legal name,
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
suffixes and dotted endings dropped) across strong groups -> not ticked.
  Same personal address with different names -> people list (never merged). Main =
existing main, else newest last_export_date.
Merge (one transaction for all ticked groups): whole existing groups come along;
co_groups rows; company_access moved (manager wins);
  open items with same type_id + same Details on different records -> mergeItem
(history copied, empty fields filled, Select united,
  people/checks/event+meeting gig ids moved, second archived, Merged entries both
sides); other items moved; event_emails / meet_people moved.
Split: deletes the co_groups row; moved data stays with the main record.
Startup sweep: anything still on a merged record (made while the module was off)
moves to its main record (logged).
Other modules: events 1.0.2 matchEmails maps to main records (deduped),
applyToCompany/link use canon; meet 1.0.1 candidates and
  applyPerson use canon; gigs 1.0.2 history ordered by at, id (copied history sorts
by time). All optional: without the module, canon = identity.
## Tested
Planted duplicates: same email (QuarqAI), same everything (Popper Power), same
email different name ending (BM Diagnostics S.r.l.s.),
same name only (Acme, not ticked), same gmail + same name (merged) and different
names (people list). Event before merge made items on both
records; merge: duplicate same-Details items merged with history and notes,
different-Details items moved, access moved; search shows main only;
new event lands on main item (existing); details of merged record point to main;
split; activity; managers refused. Browser: tab, ticks,
merge message, Same company box, help section; meeting flow still works. Isolation:
companies off -> events/meet/search work unmerged; back on ->
sweep moves stray data.

# Module match (SSeeds-match.js 1.0.0 / SSeeds-match-client.js 1.0.0)
## Uses
events 1.0.3 api linkRow(user, ev, row, code) (the Link route uses it too),
getEvent, requireAdminOrManager, label; meet 1.0.2 api linkPerson(user, ev, mt, p,
code),
domainsIn, keyOf, normCo, FREE_MAIL; companies api.canon (optional). Core 1.0.12
exposes aiChat, userAi, parseJsonLoose to modules.
gigs 1.0.3: Sieve 'event:<id>' (items in event_emails.gig_id or meet_people.gig_id
of the event's meetings), listed in /api/gigs/sieves (30 newest; admins all, others
own).
## Rules
gather: sign-ups (one per address), meeting people of the event's meetings, the
event's companies (main records).
pairScore sign-up<->person: address in chat 100 | in what they said 95 | both on
same company 95 | company domain in their text 85 | all name words in address 80 |
  first name starts the address 60 (only if unique). Greedy, best first.
propose: linked now 100 | email in chat / sign-up address = company email 95 |
company domain 85 | contact first+last name 72 (several companies: the one
  current at the event date 72, others 64) | speaker name = company name 60; +3 for
companies already on the event. Proposal only when one best.
AI (job, own AI settings): one request with event, rows (sign-up, name, linked/rule
ids, pairing reason, text <= 3500 chars) and up to 250 candidate
  companies (event companies, rule candidates, LIKE look-ups of name words /
address parts / domain labels, companies named in their text) with email,
  website, contact, one-liner, first/last seen. Reply JSON rows [{row, company_id,
confidence, reason, new_company}], same_person [[a,b]]. ids validated.
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
Apply: per ticked row links the unlinked sign-up (EV.linkRow) and person
(MT.linkPerson) to the chosen company in one transaction; existing other links
reported.
## Browser
Event page: Next steps box (counts, Match people & companies, Add meeting hint,
Open in Pipelines = Sieve event:<id>). Match window: rows needing work
first, fully linked folded; per row: tick (linked now / rule / AI >= 80%), company
list with reasons, Find/Pick, Notes; Check with AI (job steps); Apply ticked.
## Tested
Oct 1 sign-ups (27) + transcript + chat against stand-in companies: rules paired 13
sign-ups with call people, conflict shown, fake OpenAI-compatible AI:
request content checked (28 rows, 19 companies, 14 texts, 46k chars), same-person
join (olgaclees -> Olga), new company (ACCGCresh), apply linked, counts
updated; users refused; event Sieve shows all 16 items; help; no console errors.
Isolation: match broken -> meetings/events work; meet broken -> match off.

# Pipelines screen as perl Gigs (SSeeds-gigs-client.js 1.0.3, SSeeds-gigs.js 1.0.4)
- source: Gigs.pl, GigsEdit.pl, GigsSettings.pl, GigsLibrary.pl
## List (Gigs.pl)
Row 1: [user / sieve / type], level (Novice Normal LookFeel Advanced + Expert for
managers/admins + Admin for admins), Ignore No/Yes/Arc (Arc: managers/admins),
  Count, Warning days 1-44; LookFeel: the LaF.pl menu (see 'LookFeel as LaF.pl').
Row 2: -Show Fields- (toggle,   ), Search, -Select In- (toggle), All of, And/Or,
Any of, View Standard/Email/CSV (Email/CSV: managers/admins, logged via
/api/gigs/export),
  Redate Set/Shift/Status/Sieve(type) (Expert+, Chk shown).
Bar: Settings (admins -> Types) - Help - New | Sieves (links, current italic) |
Types ALL - links | Checks: Show - Emails - All - None (count), Redate select
applied on change
  (Set: Days 0-365 'N (Wkd date)'; Shift -45..365; Status; type; Arc: Flip), note
box sent with the action (server b.note), Note only. Orange when not Set / Arc.
Table (.gt: natural width, borders/padding from settings): sortable headings; Item
'#id date' (Count prefix, grey if archived) -> edit; Details/Note markers
  (* pink, + light blue, B/E/D BEGIN/END/DO arrows); Status + [type]; Next Contact
colored; Company link; Links as [name]; Quality/Priority colored; Value commified;
  summary row. Colors = perl Gigs: RED #ff4444 ORANGE #ffaa44 YELLOW #ffff00 GREEN
#44ff44 GREY #cccccc, black text; empty date = red (perl DateColor).
Prefs (server): + level, count, view (old column/search names migrated: Item-
>Identifier, Chk->Checks, Later->NextContact, Type->GigSieve).
## Edit (GigsEdit.pl) - in place of the list (V.edit, data-help=item),
S.openGig(id) switches to the tab (PENDING) from other modules
Top: Save, Copy, Type, Copy to: (my Default / companies the user can start in),
Archived, Company: ..., Back to list, #id.
Row: When | Details + Status [type] | Next Contact (colored) date + Days + Tags |
POC + Fusion | Email + Tel | Owner + Select | Note | Who + Links | Quality Priority
Value
  (textarea sizes from GigsEdit). History: grouped by timestamp in the same
columns; other fields (Created, Event, Meeting, Merged, People...) under Note.
People on this pipeline below. Help client 1.0.1: a visible [data-help] inside the
current tab wins (edit view -> item).
## Not taken from perl
Library (TimeLines templates) - later stage; GigsSettings Methodology/presets -
types are on the Types tab.
## Tested
Admin: event sieve 16 rows, summary, check 2 + Set 7 days + note (2 changed), edit
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

view columns + history, help in edit, save (Later, Note), CSV view, open item from event page. User: levels up to Advanced, no Checks/Arc/CSV, Novice hides controls, LookFeel borders 3px. Meeting flow unchanged. No console errors.

# Module team (SSeeds-team.js 1.0.0 / SSeeds-team-client.js 1.0.0)
## Tables (additive)
team_members(id, name, emails one per line, aliases one per line, note, added_by, added_at, updated_at) | team_domains(domain PK).
meet_staff names are taken over once (when team_members is empty).
## api (cached, reset on change)
isTeamEmail(e) (listed address or our domain or below), isTeamDomain(d), isTeamName(n) (name or alias, keyOf), names().
## Routes
GET /api/team, POST /api/team (add/change), POST /api/team/mark {name, emails} (joins the member with that name/address, else new): admins and managers.
DELETE /api/team/:id, POST /api/team/domains: admins. All in Activity.
## Used by (all optional - without the module each behaves as before)
companies 1.0.1: team emails/domains are not linking keys. events 1.0.4: preview/create set team addresses aside (team list/count), event list/page
  count team apart (row.team), Team button on unmatched (events client 1.0.3). meet 1.0.3: hosts = team names (else meet_staff), team emails/domains
  not used for candidates; /api/meet/staff says team:true + names; meet client 1.0.2: Our team box, Team button (name + chat addresses).
match 1.0.1: team sign-ups and people left out, team emails/domains not used, AI prompt lists the team; match client 1.0.1: Team button per row.
## Tested
Takeover of 3 staff names; domains (manager refused), aliases, mark by manager, user refused, manager remove refused; preview/create set aside 4/1;
older event counts team apart; meeting hosts from team; match list without team rows, AI told; companies with a team address not proposed together.
Browser (manager): Team tab, help, Team on event page, Our team box and Team in meeting review. Isolation: team off -> meetings use staff names, rest works.

# Sieves as workspaces (SSeeds-gigs.js 1.0.5, SSeeds-gigs-client.js 1.0.4) - as perl Gigs: a sieve has its own Types and items
## Tables
gig_sieves(id, name, kind shared|company|default, company_code, owner_id (default), created_by, created_at); one default per owner (unique index).
gig_sieve_users(sieve_id, user_id, role manager|user) - people on a company sieve.
gig_types.sieve_id (unique (sieve_id, name)); gigs.sieve_id.
## Migration 1.0.4 -> 1.0.5 (once, when gig_types has no sieve_id)
VACUUM INTO SSeeds-app-before-sieves.sqlite (next to the app DB, once);
foreign_keys off; gig_types rebuilt without UNIQUE(name) (same ids);
each old type -> shared sieve of that name (company items and owner-less items);
Default items -> owner's Default with copies of the used types;
gig_prefs sieve/type reset to all/All; foreign_key_check must be empty; commit;
foreign_keys on. New installation: shared sieve + type General.
## Rules
Visible items: admins; company access; gig_users; whole sieves (wholeSieveIds: own and managed people's Defaults; company sieves the user, a managed
  person is on, or whose company the user manages); company-less items of self/managed owners.
canUseSieve: admins; shared: everyone; others: wholeSieveIds. canSettings: admins; default: owner; company: sieve managers and company managers.
Create sieve (POST /api/gigs/sieves {company_id, name}): admins, company managers;

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

kind company, creator = manager member; names unique (non-default).
Move (edit Type of another sieve, bulk 'type'), Copy (type_id): placeCheck - usable sieve; company sieve takes only its company's or company-less items
  (which become its company's); into a Default the owner becomes that person.
Status not in the new type -> first status. History: Type, Sieve.
Events: only shared-sieve types (events 1.0.5 refuses others); events client lists shared sieves' types as 'Sieve / Type', A2EPipeline first.
access 1.0.3: managedUserIds = workers (role user, not admins) on companies the manager manages (was: everyone on them).
## Routes
/api/gigs/sieves GET (All + usable, alphabetical; own Default 'Default', others 'Default (name)') POST | DELETE /api/gigs/sieves/:id (empty only) | /api/gigs/sieves/:id/people GET POST | /api/gigs/types?sieve=sv:<id> (none: all usable, others' Defaults named) POST (sieve_id) DELETE | /api/gigs?sieve=all|sv:<id>|event:<id> | /api/gigs/:id/copy {type_id} | /api/gigs/company/:id (+ Sieve per item, company sieves, can_create_sieve).
## Browser
Bar: Sieves (alphabetical, All and Default among them), Types of the sieve, Settings (can_settings) -> settings view (types, People on this sieve, Remove this sieve) in place of the list (data-help gigsettings). New: in the current sieve. Edit: 'Sieve:' Type (types of the item's sieve), Move to / Copy to ('Sieve / Type' of every usable type), Create sieve (company managers). All: Status shows [Sieve / Type]. Company details: items in every sieve, its sieves (Open), Create sieve, Start [sieve / type]. Types tab removed.
## Tested
Migration on old data (backup, 2 shared sieves, Default move, FK/integrity ok, second start skips). API: sieve lists per role, create (user refused, duplicate refused), settings (user refused, duplicate type refused, missing lists ok), copy sales item to America and GTM (history, original stays), sieve people (worker sees all, others refused), New in customer sieve (company set), user move into foreign sieve refused, other company refused, bulk type, events into Default refused, company details, remove non-empty refused. Browser: tabs, alphabetical sieves, [Sieve / Type] in All, settings (add type), help, New, Move to, no errors. Earlier flows (meeting, matching) pass.

# Levels, sorting, Act as memory (gigs 1.0.6 / gigs-client 1.0.5 / ui-client 1.0.7 / ui.html 1.0.7)
Levels (perl UserLevel order Novice Normal LookFeel Advanced Expert Admin): users up to Normal (default Novice), managers up to Advanced (default Advanced),
  admins Admin; gig_levels(user_id, max_level) set by admins (Users tab 'Gigs up to', up to Expert). /api/gigs/prefs returns levels + default_level;
  a saved level above the top is lowered. /api/gigs/bulk needs Expert (perl: Redate/Checks at Expert/Admin).
Shown per level (as Gigs.pl): Novice list/bar; Normal + Ignore, Count, Warning days, Show Fields; LookFeel + LaF menu, Search row,
  Settings; Advanced + View; Expert + Redate/Checks. Reverse select removed: a heading sorts, the same heading again reverses.
Count on by default (countChosen marks a person's own choice; older saved 'off' ignored). Item cell: 'N: created date time' (no #id). Edit and Settings: Back first.
Prefs while acting: gig_prefs_as(real_id, user_id) - starts from theirs, theirs unchanged.
Core client: memo(k, v) per identity (real>target) in localStorage (try/catch):

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
last tab (restored at login / after Act as reload), gigs-edit (open item).
  loadClientModules loads module scripts one after the other (deterministic tab
order); registerTab before may be a list (events: ['gigs','search']).
ui.html: tables width:auto (natural), td.long max-width 60em (was min-width 18em).
## Tested
API: levels per role, manager bulk refused then allowed after raise, admin level
refused, levels list admin only, user can't save Admin; acting as usr starts
from usr's view, changes kept apart, usr's own unchanged, adm's own unchanged.
Browser: item cell, header sort/reverse, Back first, act as usr (levels
Novice,Normal),
stop acting -> Pipelines with the item open again, act as usr again -> usr's last
tab; tab order Events, Pipelines; sieve, settings, move, CSV, open from event page;
search table natural width. No console errors.

# LookFeel as LaF.pl (gigs-client 1.0.6, help.html 1.0.7) - Pipelines only, browser
only
LAF in localStorage 'sseeds-gigs-laf' (try/catch; old 'sseeds-gigs-table'
Borders/Padding becomes Custom). lafMenu(h) at level LookFeel (renderHead):
  row 1 Presets (LAF_TP sorted), Color (LF + LAF_COLORS = SetColors OwnC/HisC),
Font Size ((LF), 8-35pt), Show Table/Plain;
  row 2 FontFam ((LF) + 4), FontFace ((LF) + LAF_FACES sorted), FontWeight,
FontStyle, FontVariant;
  Custom/Custom2/Random: Corner Radius, Border, Spacing, Padding 0-35px, Border
Style (LAF_BSTYLES);
  Custom2/Random: 3x3 TL/Top/TR, Left/-/Right, BL/Bot/BR, then Vert/Hor Pad,
Vert/Hor Space.
lafValues(): presets LAF_PRESETS [border, radius, spacing, padding, style] copied
to all sides/corners/pads/spaces (SetLaFVars); Custom from the saved
  four; Custom2 the saved twelve; Random drawRandom() (0-63pt, style, font, colors)
redrawn on each change (saveLaf) and each loadAll.
applyTableCss(): .gt table (border, radius, spacing, colors, weight/style/variant),
.gt th/td (sides, corners, pads), .gt th scheme background
  (ui.html gives th the LF background), .gbar cells like td, .glaf (Pipelines
section) buttons/selects/inputs border+radius (SetSubmit).
  Chosen size/face/family use !important inside .glaf only (lf.js sets fonts with !
important on html *). Color LF = no colors set.
  Gigs colors (next contact, Quality/Priority, markers) still set per cell with
black text.
Tested: admin LookFeel - every preset, Custom/Custom2 controls (top border 9, TL 7,
hor pad 11 applied), Random redraw, Black + 16pt Georgia italic
  (header readable, white on black), Plain, kept after reload; usr (Novice/Normal)
no menu, Default preset; edit view Back first; manual served. No console errors.

# Unique item identifiers, forward-only dates, natural widths (gigs 1.0.7, gigs-
client 1.0.7, help.html 1.0.8)
Item identifier = gigs.created_at to the millisecond (YYYY-MM-DDTHH:MM:SS.mmm).
Startup: rows with seconds only get .000, .001... per second in id order
  (temp table with ROW_NUMBER, one transaction, logged once); UNIQUE index
ux_gigs_created. itemStamp(): now in ms, else LAST_STAMP + 1 ms
  (LAST_STAMP from MAX(created_at) at start); used by createGig and copy. Client
shows created_at to 23 chars (list Item cell nowrap, edit '@ time').
Next contact never into the past: edit view has only the Days list (calendar
removed); server refuses a changed Later before today(-1) (one day of
  slack for time zones) and a bulk Shift landing before it (whole action refused).
An unchanged overdue date is kept.
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
Widths: Company in div.cap (max 16em), Details and Note in div.capw (max 30em):
long text wraps, short columns keep their own width.
Tested: migration (53 rows unique), 6 parallel copies and an event creating items
in the same ms (.528 .529 .530), past date and Shift -10 refused,
  Shift -2 and a future date accepted; browser: Item one line, long company wraps,
no date input in edit. No console errors.

# Email drafting (mail 1.0.0, mail-client 1.0.0; gigs 1.0.8 api, gigs-client 1.0.8
hook, ui.js 1.0.14 MODULE_FILES, install 1.0.18 PROG, help 1.0.9)
Agreed: the user copies and pastes into the body of their email (nothing is sent);
templates by admins and managers; one item at a time; a starter set
  for editing, with the information fields as JSON from internal sources; meeting
material sent when it exists; English.
MODULE_FILES: ... match, mail, help. Needs gigs api: getGig, gigOut, sieveName,
canUseSieve, typeById, hist, today (gigs 1.0.8 exports them).
  Reads events/event_emails/meetings/meet_people directly when the tables exist;
companies api membersOf (merged records) when loaded.
Tables: mail_templates(id, sieve_id NULL=all sieves, name, purpose, types, statuses
(pipes), prompt, fields JSON, position, created_at, updated_at, updated_by),
  mail_profiles(user_id, name, title, about, signature), mail_meta('starters' =
made once; removing them does not bring them back).
Starter templates (all sieves): After the event, After a GWiz™ demo, Pitch follow-
up, Negotiation nudge, No reply, Customer check-in - each
  Purpose/Content/Form/Size/Tone with fill-ins; statuses guessed from General and
the A2EPipeline names seen (Pitch, Negotiation...).
Information JSON {source: [fields]} - OFFERED: item (Gigs fields; Email/Tel not
offered), history (number, newest kept), company (record columns +
  company_values lists; PRIVATE_COLS not offered), events (name, date; deduped, by
date), meetings (title, event, date, person, said, gwiz; same text once,
  newest first within MAX_MEETINGS 16000 chars, each said MAX_SAID 4000), sender
(profile). parseFields refuses unknown sources/fields.
mask(): emails -> [email]; +international, (area) number, 555-555-5555 -> [phone]
(dates and ids untouched). Applied to every value, the person's
  edited JSON and their added line. fillIn(): {{source.field}} / {{events.0.name}};
empty -> [path: not known].
SYSTEM (fixed rules, shown on the Templates tab): only the facts given,
[placeholders] + "missing", no flattery/pandering, follow the template,
  plain text English, GWiz™ spelling, first-name greeting from item.POC, signature
as given; reply JSON {subject, body, missing}.
Routes: GET/POST /api/mail/templates, DELETE /api/mail/templates/:id (admins +
managers = managedCodes; sieve templates for sieves they can use),
  GET /api/mail/gig/:id (ranked: fits = statuses/types empty or include the item's;
score status 4, type 2, sieve 1; then position),
  POST /api/mail/info {gig_id, template_id} -> {info, prompt filled}, POST
/api/mail/draft {gig_id, template_id, extra, info?} (await aiChat with
  core.userAi, MAX_PROMPT 80000, activity 'email drafted'), POST /api/mail/record
(history field 'Email drafted': "template: subject" - not 'Email',
  the address column), GET/POST /api/mail/profile.
Client: S.hooks.item(row, d) made by gigs-client, called after the edit view's
button row; mail-client adds Draft email -> modal (data-help mail):
  template select (optgroups For this situation / Other), own line, details "What
the AI gets" (filled template, editable info JSON - sent only if
  changed), details "About me" (opens when no signature), Draft / Draft again,
Subject + Copy subject, Fill in before sending, text + Copy text
  (first copy of a draft -> /api/mail/record). Templates tab (who mgr, before
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

team): one fieldset per template, reference details, New template last.
Tested (fake OpenAI-compatible AI on 9911): starters made once; usr refused, mgr allowed; bad source/field refused; ranking for Qualified puts GWiz and
  Pitch first; info has 6 sources, dates intact, no address or phone in what the AI received, phone in the added line masked, system rules sent;
  edited JSON used; history entry; usr 404 on an item they cannot see; browser: draft window, Draft, Copy text to clipboard, Help shows Draft email,
  Templates tab for mgr (Add) and not for usr; mail module broken -> switched off, Pipelines and edit view work without the button. No console errors.

# Email drafting 1.0.1 - the Cognos / Psych path (mail 1.0.1, mail-client 1.0.1, gigs 1.0.9 addNote, ui.js 1.0.15 aiChat temperature, help 1.0.10)
Agreed: the email moves the company through the sales sieve toward a deal (one call to action = the Goal); all information MAY be used, each template
  ticks what it DOES use; no JSON shown; fields quoted where wanted (Insert); no per-field cuts; Summary for the analyst -> Note; audience per company
  (their SPOC); offerings = next best steps in time order with links, never prices; creativity for the writing; English. Examples followed simply:
  Cognos (facts -> conclusions -> statement in a style/form -> rewrite per audience), Psych (characteristics -> only the benefits given -> next step).
ui.js aiChat(ai, system, user, maxTokens, temperature): temperature 0-1 added to the request; if the provider refuses it ("temperature" in the
  error) it asks again without and logs it.
gigs api addNote(user, id, label, text): dated block "YYYY-MM-DD label (who):" at the top of the Note; history 'Note' = the new Note (as an edit).
Tables: mail_templates + kind, goal, style, creativity, offers {offering id: may|always} (ALTER on start); fields = JSON list of CATALOG keys
  (1.0.0 objects converted on start). mail_offerings(id, name, description, url, tags (needs, pipes), position). mail_audience(company_code, profile).
  mail_meta: 'house' (house facts), 'starters' (1.0.0 set made), 'starters2' (1.0.1: house facts, 8 offerings from a2e.co Grow/GWiz pages without
  prices - Ask Anything and the call with Matt have no link until added - and the "Where this stands" summary template).
CATALOG (key, label, what it is): item.* (event = the event that brought the item, via event_emails.gig_id or meet_people.gig_id), history (all
  entries), items (the company's other non-archived items the person can see), company.* (record columns + company_values lists), events,
  meetings (same text once), audience, sender.*. Values masked (emails, phone numbers), never shortened; MAX_PROMPT 400000 chars -> refused with
  "untick some of it".
offersFor(t, g): the template's offerings in position order; always -> ticked; may -> ticked when one of its tags appears in item Tags/Select/Details.
Routes: GET/POST/DELETE /api/mail/templates; POST /api/mail/house; POST/DELETE /api/mail/offerings; GET /api/mail/gig/:id; POST /api/mail/prepare
  (goal, ticks, offerings, filled text); POST /api/mail/audience, /audience/write (AI, temperature 0, not saved); POST /api/mail/draft (analysis
  RULES_ANALYZE temperature 0 -> {analysis, offers suggested}, unless an edited analysis is given; then RULES_WRITE with the template creativity:
  HOUSE, KIND, GOAL, TEMPLATE, STYLE, AUDIENCE, ANALYSIS, OFFERINGS, SENDER, own line, INFORMATION as background); POST /api/mail/rewrite
  (RULES_REWRITE + instruction); POST /api/mail/note (addNote); POST /api/mail/record; GET/POST /api/mail/profile.
Tested (fake AI 3 by rules): migration of 1.0.0 templates; starters2 once; After Go To Angel saved with always/may offerings; prepare ticks always +
  tag matches (Coming to America, Investment); analysis temperature 0, writing 0.4;

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

offerings exactly as ticked, in order, with links; audience
  masked; full history (18 entries) sent; edited analysis used; rewrite; Add to Note on top of Note + history; usr refused for house and items they
  cannot see; bad field and javascript: link refused; browser: window, Draft, Rewrite, Undo, Insert into the template text. No console errors.

# Team as its own place, gear menu (team 1.0.1, team-client 1.0.1, mail 1.0.2, mail-client 1.0.2, ui.html 1.0.8, ui-client 1.0.8, help 1.0.11)
Agreed: members are not always linked to accounts but may be; managers edit their team members and themselves, admins all; everyone sees their team
  including their manager; new accounts are not put on a team; Act as stays where it is (admins and managers); the sender of emails = Team record.
team_members + user_id, manager_id (a member), title, about, signature, links, mgr_notes (ALTER on start). meOf(user) = member with user_id.
  isMgr = admin, or manages companies, or has members reporting. canSee: admin all; else me, my manager, same manager, my reports.
  canEdit: admin; or a manager editing me or my reports. Notes: admin or the member's manager (not the member). Add: admins, managers on the team
  (new member's manager = them; /api/team/mark too). Link account: admins any, managers self + managedUserIds; one account per member.
  Manager set by admins only. Remove (admin) clears manager_id of reports. GET /api/team: members (visible, can_edit, can_link, mgr_notes when
  allowed), me, can_add, accounts (linkable), all_members (admins: manager list), domains (managers/admins).
  api memberOfUser(userId), fillProfile(memberId, p) (fills only empty title/about/signature).
mail 1.0.2: profileOf(user) = T.memberOfUser -> name/title/about/links/signature, on_team; else username. mail_profiles taken over once into the
  linked members (mail_meta 'profiles_to_team'); POST /api/mail/profile removed; CATALOG + sender.links. Draft window: "Sender: ... (from the Team
  tab)" or "not on the Team yet" - no About me box.
Top bar: [gear] SSeeds [Pipelines][My companies] ... [Act as] who [Help][Look & Feel][Log out]. Gear (#gear-wrap/#gear-menu, background
  var(--lf-fg) like #hdr which lf.js draws reversed): orderedTabs = gigs, mycos, ask, activity, settings first, then the other allowed tabs in TABS
  order; then Act as (focuses/opens the #actas select, or Stop acting), Help, Look & Feel, Log out (click the existing buttons). Esc / outside click
  closes. Start tab: the remembered one, else the first (Pipelines).
Tested: API - visibility per role (usr: self, manager, peers; mgr: self + reports with notes; adm all), usr cannot edit, mgr edits own people not
  others, manager_id ignored from managers, mgr add joins their team, account taken refused, sender from Team; browser - top bar and gear per role
  (adm, mgr, usr), Team via gear, Log out via gear, draft window sender line, help. No console errors.

# Owner, saves, Debug, Fill-in, Backup (gigs 1.0.10, gigs-client 1.0.9, mail 1.0.3, mail-client 1.0.3, backup 1.0.0 new, ui.js 1.0.16, install 1.0.19, help 1.0.12)
Agreed: Owner = the last writer (perl GigsEdit sets Owner=$LogID on every edit); two people editing the same item: both saves are stored, the latest
  shows, the history has all; Debug selector at Admin level; Fill-in (no AI) templates; Backup by hand only, into /tmp/SSeeds/Backup-YYYY-MM-DDTHH-MM-SS;
  Restore = stop, backup, restore the chosen one, start.
Owner: edit save (as before), bulk/Checks actions (set.owner when anything changes), addNote (AI note) -> owner = whoName(user).
Saves: gigs-client sends base (the values opened); POST /api/gigs/:id drops fields

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
equal to base before saving (Type by number, Archived as boolean),
  so only what this person changed is written; same field changed by two people ->
the later save shows, both are in gig_history.
Debug: PREF_KEYS + debug; renderHead at level Admin: Debug 0-5 (set -> prefs);
above 0 also the LaF menu (perl: LookFeel or debug>0). V.debug
  between the bar and the table: renderDebug(rows) - D1 view/sieve/type/counts/last
request (DBG.req url, ms, at, rows; first_load_ms), D2 sieves,
  types, move/copy targets, D3 prefs + prefs reply + levels, D4 status orders of
shown types + sort key per row, D5 DATA.
Fill-in: mail_templates + writer ('AI'|'Fill-in'), subject (ALTER on start);
prepared() uses every CATALOG field for a fill-in and info(...,
  raw=true) (no masking - nothing goes to an AI); /api/mail/draft fills
{{source.field}}, {{offerings}} ("- name: link" for the ticked next steps),
  {{signature}} (Team, else name), {{goal}}; missing -> [field] + listed; the
person's own line added as typed. Client: Writer + Subject +
  Insert next steps / Insert signature in the editor; the draft window hides
audience, information, style, creativity and analysis for a fill-in
  and its button reads Fill in. module-level mask() (AI-bound text) = maskAI;
info() shadows it for raw.
Backup (SSeeds-backup.js, admin routes GET/POST /api/backup, POST
/api/backup/restore): VACUUM INTO for the app and companies databases (live,
  consistent), copyTree of the installation directory (skipping the databases and
their -wal/-shm/-journal, sockets, SSeeds-restore.json), 0700/0600,
  manifest.json (made, by, why, files). Restore: safety backup "before restoring
<name>", writes SSeeds-restore.json {from, asked, by, safety},
  answers, process.exit(0) after 400 ms; the service loop starts SSeeds again in ~5
s; ui.js restoreAtStart() (right after loadConfig, before
  the secret and databases are opened) copies the backup's databases (removing -
wal/-shm/-journal first), SSeeds-ui.secret and uploads folder,
  deletes the note and logs RESTORED; on error renames it SSeeds-restore.failed and
starts with the data as it was. Not restored: program files,
  SSeeds-ui.json. Backup tab (who admin, data-help backup). /tmp is emptied at
reboot (help says so).
Tested: A and B saving different fields from the same view both kept; C changing
the same field shows, history has both; Owner after manager save,
  admin Checks note, manager AI note; fill-in with offerings, signature fallback,
missing field, phone kept in the person's own line; backup 48 files,
  listing, usr refused; restore in a service-like loop: safety backup made, server
stopped and started, data back to the backup (change made after
  it gone), both backups listed; browser: Debug 1/3/5 with LaF menu, fill-in in
window and editor, Backup tab via gear. No console errors.

# Fast checks, kept apart (gigs 1.0.11, gigs-client 1.0.10, companies 1.0.2,
companies-client 1.0.2, help 1.0.13)
Checks were slow: /api/gigs/checked wrote one row per statement outside a
transaction (one disk write each) and the client waited for the answer and
  redrew the whole table; bulk checked permission item by item (canSee ->
wholeSieveIds queries per item). Now: setChecked changes DATA, the
  boxes (input[data-gid]) and the count (.gcount) at once, redraws only for Show-
checked, and keeps the change on the server in the background
  (CHECK_Q: one request at a time, in order; loadGigs waits for it).
/api/gigs/checked: one transaction (only existing items). editable(): one
  query with visibleWhere for all ids. Bulk handlers read the checked items when
used (cur()), not when the bar was drawn.
Kept apart: co_apart(code_a < code_b, evidence = JSON list of the links between
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
them when decided, decided_by, decided_at).
  links(): pair edges {i, j, reasons, strong} - same email (company), same person +
same name (free mail), same domain, same name + contact (strong);
  same name only (weak); all pairs per key (star above 60). proposals(): an edge is
dropped when its pair is kept apart and every current reason was
  in the evidence (kept_apart = how many); strong groups by union-find over the
rest (merged records start together); name-only proposals join
  strong groups over weak edges. freeMail(d): list + FREE_RE (yahoo.*, hotmail.*,
outlook.*, gmx.*, libero.* ...); linkSite(h): list + LINK_RE
  (amazon.*, apple.com and apps.apple.com, play.google, tiktok, pinterest, vimeo,
wixsite, ...).
  Routes: POST /api/companies/apart {pairs} (evidence = today's links of each pair,
one transaction), GET /api/companies/apart, DELETE (Undo).
  Client: Apart tick per unmerged record; Keep apart per group (ticked Apart vs the
rest; none ticked = every pair, confirmed); Merge ticked keeps
  Apart records out (pairs saved first, groups of 2+ merged; a main record cannot
be Apart); Kept apart list with Undo.
Tested: check/uncheck 61 items one request each, bulk on 61; browser All ticks
every box without redrawing, count updates, kept after reload;
  keep apart removes the group (kept_apart counted), evidence older than today's
links -> proposed again, Undo -> proposed again, second Undo 404;
  Apart on one record of three with the other two already merged -> nothing left to
propose. No console errors.

# Libraries, company form, designees, restore all/data (library 1.0.0 new,
companies 1.0.3, team 1.0.2, match-client 1.0.2, import 1.0.3, gigs 1.0.12,
#   backup 1.0.1, ui.js 1.0.17, install 1.0.20, help 1.0.14)
Agreed: libraries work as in perl and can be edited and added; new companies
through a form with the database's fields, by managers or their
  designees; restore all by default, data only as an option.
Library (perl GigsLibrary.pl SaveLibrary): gig_libraries(id, name UNIQUE NOCASE,
rows JSON [[Status, Details, Begin, End, Note, Who, Links]]).
  parseTimeLines(text): drops # lines, %TimeLines=( ... ) -> JSON ("Name", [ ->
"Name": [ at depth 0, trailing commas, &trade; -> ™).
  POST /api/library/use {library_id, sieve_id, type_name, start -30..365, base_url,
preview}: canSettings on the sieve; new type (statuses New +
  library statuses in order + Completed, hidden Completed; refused if the name
exists); per step: Begin<>End -> "B details" Later start+Begin
  Note "BEGIN note" and "E details" start+End "END note"; else "D details"
start+Begin "DO note"; Who; Links with * -> base URL; company sieve ->
  the company's items; Default sieve -> its owner; createGig with history Library.
One transaction. Routes also GET list (with fillable sieves),
  GET one, POST add/change (cleanRows: status <=60 no |, details, 0 <= Begin <= End
<= 3650), DELETE, POST import {text, replace}. Admins+managers.
  gigs api + sieveById, canSettings, sieveList.
Companies form: SSeeds-import.js exports COLUMNS; FORM = all but key/files; GET
/api/companies/form {fields, can_add}; GET /api/companies/record/:id
  {values (multi as comma text), can_edit, added_here}; POST /api/companies/record
{id?, values} through a writable IMP.openDb connection (WAL;
  FTS triggers keep Search right). New: code SS-XXXX-NNNN, source SSeeds,
date_added/first/last_export_date today, *_source_file "added in SSeeds by
  <who>", capital_min/max from capital_seeking; co_added(code, added_by, added_at);
access: managers as manager, designees as user + their Team
  manager's account as manager. canAdd: admin, manager, Team may_add_companies.
canEdit: admin, isManagerOf, or added it (and may still add).
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

Client S.companyForm(box, {id, prefill, title}, done) - sends only changed fields; Edit company (company details hook), Add a company (My companies, shown when can_add), New company in the match window (prefill from AI new_company + person name / sign-up email; picks and ticks it).
Team 1.0.2: may_add_companies (set by admins and the member's manager, shown with manager notes); api canAddCompanies(userId), managerUserOf(userId).
Restore: marker what = all (default) | data; restoreAtStart "all" also copies every other file and folder of the backup (not manifest, logs, pid).
Tested: TimeLines -> 9 libraries (Test 5, Pathkeeper 137, Proposal 86, Diligence 108, GoToMarket 23, IPush 34, StartUp 17, InitialOperations 14, InitialMarketing 13); usr refused; import again keeps; StartUp on AA Create: 31 items, statuses New>Overview>Preliminaries>Descriptive>Startup> Completed, B/E/D, dates, company; same type name refused; links * replaced; edit + bad step refused; browser: choose, preview, fill, + row.
Company: usr refused, mgr adds (code, multi values, Search finds it), edits; designee usr adds and may edit own, not others'; browser: Add a company -> Details, Edit company saved. Restore data only: data back, other files kept; restore all: files back too. No console errors.

# Screens as lists + own edit screens (gigs-client 1.0.11, ui.html 1.0.9, ui-client 1.0.9, mail 1.0.4, mail-client 1.0.4, team-client 1.0.3, help 1.0.15)
Page name: ui.html <b id="page-name"> after #tabs; ui-client show() writes the tab's label (cleared at login).
Item edit (gigs-client): .gedit width:100%; td/th.cmail 130pt fixed (min=max, break-all); td.cnote width:100% min 12em overflow-wrap:anywhere (a long URL in an old note no longer widens it); EDIT_TEXT '100%' boxes get cols=1 so their built-in width does not set the column. Tested 1400px: table = container (1376). th.cnote and history Note cells -> notesWindow(g, history): fixed overlay 5vw/5vh, 90vw x 90vh, scrolls; "Note now" + history entries other than plain field changes, newest first; \n -> <br>; <hr> border-top 4pt between. Esc/Close. People: a select after the hooks (so after Draft email) - "People on this item (n)", "× name" (rm:id) for those who manage, optgroup Add "+ name" (add:id) from /api/people.
The People fieldset at the bottom is gone.
Mail 1.0.4: mail_templates.extra_steps (text, "name | link | description" per line). parseOffers keeps on-by-default ids ('on'/'always'/true); offersFor = every offering with checked (no Tags matching any more; offering tags are a note only). prepared(): ticked offerings (b.offers or the template's) + steps(b.more_steps ?? t.extra_steps). Fill-in {{offerings}} lists both.
Mail-client 1.0.4: Draft window - offerings as checkboxes + "more next steps" textarea (from the template, sent as more_steps). Templates tab = LIST (house facts, offerings, fixed rules, -Show columns- [localStorage sseeds-tpl-cols; default name writer kind sieve purpose goal; also statuses types style creativity steps position saved], headings sort via TSORT, New template, Name -> editTpl) and EDITV (data-help mailtpl): Back first, Save, Copy as new (window.prompt), Remove; Purpose = label people choose by (not sent); Goal = for the AI; Order = tie-break in the Draft email list; Next steps fieldset of checkboxes + "More next steps" textarea; information picker.
Team-client 1.0.3: list (intro, domains details, Add a team member, -Show columns- [sseeds-team-cols; default name title account manager emails note; also links may updated], sort by heading) -> open(m): Back + member(m, d) (same form / read-only view as 1.0.2); Save/Remove reload the list.
Tested (t9, Playwright): page names Pipelines/Templates/Team; widths as above;

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

people drop-down options; notes window 1260x855 at 1400x950 and
  900x630 after resize, 14 hr; Templates list headers, sort, edit Back first,
untick an offering + extra step saved; Draft window shows the
  ticks and the extra step, the AI request carries it; Fill-in via API with
more_steps and with offers; Team list as adm (12) and usr (4, read-only
  other member, no Add); no console errors.

# Search filters as drop-downs (ui-client 1.0.10, ui.html 1.0.10, help 1.0.16)
ui.html: Search and Clear in the first row (after the name box); #s-facets flex
wrap; .facet max 16em, select max 12em, chips below.
ui-client: facet(f, vals) -> div.facet[data-f] with .vals (chosen), a select (first
option "label (n)", values with counts, chosen marked   ;
  choosing toggles) and .chip.active "× value" tags (click/Enter/Space
removes); .reset(). gatherFilters reads .vals; clearSearch resets, SEQ++.
  doSearch: SEQ drops stale answers; LAST_MS = request + table drawn. autoSearch():
LAST_MS < AUTO_MS (500) -> doSearch, else status asks for
  Search. Called on facet change, date picker change (valid), date field change
when a date is set; Enter in text, numbers and dates.
Tested (t9): 9 drop-downs on one row at 1400px; Search beside the text box;
pick/second pick/×/pick again each search once; Enter; Clear; 600px no
  sideways scroll; no console errors. Real-size timing not measurable here (36
companies) - hence the self-timing rule.

# Tables look like tables (ui.html 1.0.11)
th,td border:1px solid (currentColor = lf.js text color; no new colors); .tblwrap
width:fit-content, max-width:100% (frame hugs the table, wide
  tables still scroll inside). Pipelines (.gt) tables keep their LookFeel borders
(more specific rules). Tested: Team list boxed; Pipelines unchanged;
  Search results at 900px scroll inside the frame (874/1272), page does not.

# Every table border=1 (ui.html 1.0.12, gigs-client 1.0.12, help 1.0.17)
ui.html table{border:1px solid} + th,td{border:1px solid} = border=1 for every
<table>. Pipelines LookFeel Show=Plain was border:none -> now 1px
  solid cells and table (spacing 0); Table keeps its chosen borders. Lists drawn as
.list-item lines are not tables and stay as they are (user).
Tested: Pipelines Plain and Team - every visible table and cell has a border; no
console errors.

# Lines in LF colors, tighter edit row (ui.html 1.0.13, gigs-client 1.0.13)
ui.html hr{color:inherit;border:0;border-top:1px solid} - every hr (notes window
4pt, edit cells, history) in the text color, not UA gray.
gigs-client: .gt hr margin 2px; .gedit textarea.fi min-height 0 (ui.html's 4.5em
had made Who/Value three lines). Tested: edit row 197 -> 158 px.

# Pipelines items from Search and My companies; Select as perl (gigs 1.0.13, gigs-
client 1.0.14, ui-client 1.0.11, access-client 1.0.2, help 1.0.18)
gigs.js POST /api/gigs/companies {ids} -> {items:{companyId:[{id, Sieve, Type,
Status, Later}]}}: companiesByIds -> codes into TEMP t_codes, one
  query joined to it with visibleWhere, archived=0, ORDER BY later.
gigs-client S.pipeCells(ids): one request; returns cell(id) -> div of links
"Sieve / Type: Status" + date (dateColor), first 5, "… n more" expands;
  click -> openGig (Pipelines edit view). Cells made before the answer are filled
when it comes (sorting redraws from the data).
ui-client renderResults: when rows have id and S.pipeCells exists, a Pipelines
column after the first column (Search, Ask, Investor match, SQL,

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

History results). access-client loadMyCompanies: the same cell on each line.
Select (edit row): each checkbox + word in an inline-block nowrap span, spaces between, td.csel min-width 14em so they flow side by side.
Tested: adm search 31 rows / 26 with items, link opens the edit view; My companies 17 with "… 77 more" expanding to 82; usr sees only theirs
(3 rows, 7 links); no console errors.

# Windows 90%, tables fill the height, History table, Activity filters (ui.html 1.0.14, ui-client 1.0.12, access 1.0.4, access-client 1.0.3, help 1.0.19)
ui.html #modal-box width 90vw height 90vh (help and notes windows were already); td.nw nowrap. ui-client: fitWraps() sets each visible
#main .tblwrap max-height = window height - its top - status line (min 30%), on resize and on DOM changes (MutationObserver, one per frame).
renderResults: date/time values -> td.nw; Pipelines th has tbl-hdr, no min width.
History tab = a table (When, Kind, Question, Open/Delete).
Activity: user select, dates and company typing (0.5 s) reload at once (AC_SEQ drops stale answers); Refresh replaces Show.
GET /api/activity company = code or name part (LIKE, ESCAPE \) -> company_code IN (matches, max 500), answer companies = what matched;
from/to = substr(at,1,10) bounds.
Tested: Activity wrap reaches the bottom at 800 and 600 high; user/company/date each reload; "quarq" -> 2 companies, 116 rows; History table
bordered; modal 990x540 at 1100x600 and 1260x810 at 1400x900; no console errors.

# Person lines decided by hand; notes before the first item (companies 1.0.4, companies-client 1.0.4, gigs 1.0.14, gigs-client 1.0.15, help 1.0.20)
companies proposals(): people lines carry main/main_name per record and are dropped when every pair is the same main or kept apart (co_apart,
raw or canon pair). Client: table Person | Companies (checkbox + name link to Details + date + "merged into") | Merge ticked (main = newest
ticked, or its main; POST /api/companies/merge one group) / Keep ticked apart (POST /api/companies/apart all ticked pairs).
gigs: gig_company_notes(id, company_code, at, user_id, username, text, gig_id).
GET/POST /api/gigs/company/:id/notes (canSeeCompany; POST when an
open item exists already -> addNote on the first one). createGig(code): pending notes -> item Note (appended, dated "at who: text") + history
"Company notes from before this item", gig_id set. /api/gigs/companies also returns notes:{companyId:n waiting}. mergeGroup moves pending notes
to the main record.
gigs-client: pipe cell with no items -> noteLink (Add note, textarea in place, "(n waiting)"); company details Pipelines with no items -> waiting
notes + Add note.
Tested (API): 2 notes waiting, empty refused, usr without access 404, first item gets both (Note + history), later note goes to the item.
Browser: person line merge (2 records, items moved/merged, line gone), Add note from Search, shown in Details. No console errors.

# Details as Gigs, [Web]/[DB] Links, Note = last saved version (gigs 1.0.15, gigs-client 1.0.16, events 1.0.6, meet 1.0.4, mail 1.0.5, mail-client 1.0.5, ui-client 1.0.13, help 1.0.21)
Pipelines list: no Company column (FIELDS, DEFAULT_SHOW, SELECT_IN; saved settings drop it, sort Company -> Details). Details is plain text.
events applyToCompany: new item Details = company name, history Event = label; createGig(..., {companyLinks:true}) appends Links "<website> Web" and
"/#company=<id> DB" (also New with a company: POST /api/gigs). Library items and

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

New without a company: as before. Copy: as before (all fields + history).
gigs noteVersion(user, id, text, line): history Note = text, then gigs.note = previous note + "\n" + line, history Note = that; both at histStamp()
  (ISO to the millisecond, strictly increasing); Owner = writer. meet applyPerson: noteVersion(notes, 'Meeting notes added') (was history 'Meeting').
  mail /api/mail/note: noteVersion(label + ':\n' + text, 'Draft added') (was addNote, prepended). History is append-only.
ui-client: #company=<id> in the address after login -> company details (the [DB] link), hash cleared.
SSeeds-patch-2026-10-18.js: NOT installed - a one-time script: node SSeeds-patch-2026-10-18.js <install dir> [--apply]: Details equal to the item's
  first Event label -> company name; Links of company items (not Library ones) get [Web]/[DB]; each change appended to history by "SSeeds patch";
  --apply first copies the app database to SSeeds-app-before-patch-2026-10-18.sqlite and refuses to run twice.
Merging (companies.js) unchanged: same type + same Details still combine - to be discussed.
Tested (t9): patch dry run 48 Details + 61 Links, apply 109 changes, second run finds nothing / refuses; event item Details "Acme Robotics Inc",
  Links "/#company=16 DB"; meeting -> Note "My current note\nMeeting notes added", history .846 meeting text, .847 note; Draft -> "...\nDraft added";
  hand edit saved and shown; list without Company; [DB] in Links opens the company in a new tab. No console errors.

# [DB] in a window, Links of merged records, same name = strong (gigs 1.0.16, gigs-client 1.0.17, companies 1.0.5, help 1.0.22)
gigs-client linksCell: "/#company=<id>" (or "#company=") -> link calling S.companyDetails(id) in this tab; /api/company/:id answers 404 to
  whoever may not see it. Other links: new tab as before.
gigs createGig companyLinks (only when the item is made): records = the code + co_groups members (main first); one record -> "<site> Web",
  "/#company=<id> DB"; several -> "<site> Web-<name>" per different site (http(s)://, www., trailing / ignored) and "/#company=<id> DB-<name>"
  per record, repeated names get 2, 3...
companies links(): same normalized name in the database -> strong link (proposed ticked). Merging never reads or changes item Details; the
  combining of open items in mergeGroup is unchanged.
Tested: merged group (3 records) -> DB-Beta Two, DB-Alpha One, DB-Alpha One2, Web-Beta Two, Web-Alpha One (duplicate site once); usr ->
  404 company not found; [DB] opens the window in the same tab (no new page), [Web] target _blank; proposals 3, all strong. No console errors.

# HTML templates (mail 1.0.6, mail-client 1.0.6, help 1.0.23)
mail_templates.format 'Plain' | 'HTML'. HTML_TAGS (tag -> allowed attributes; href only http(s)/mailto) and HTML_DROP (removed with contents);
  cleanHtml(): tokenizer, comments dropped, other tags dropped (text kept), text < > escaped. Save: HTML Fill-in prompt cleaned. Fill-in HTML: fields
  escHtml, {{offerings}} -> <ul><li>name</li></ul> (no links), {{signature}} -> cleanHtml if it has tags else <p> paragraphs (<br> inside), own line
  -> <p>, whole body cleaned. AI HTML: RULES_WRITE (+REWRITE) + RULES_HTML (<p> rather than <br>, <ul><li>, offerings as bullets without links,
  <a href> only for given links); answer cleaned. Responses carry format. GET templates: formats, html_tags, rules.html; GET mail/gig: html_tags.
mail-client: safeDom(html, tags) = DOMParser (inert) -> rebuilt with createElement from html_tags (links target _blank); plainOf() = text with "* "

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

bullets (3-space nesting), "text (href)", paragraphs. Draft: HTML -> Preview + "HTML" source (live), Copy = clipboard text/html + text/plain
(ClipboardItem; fallback: select an off-screen preview + execCommand copy), Add to Note = plainOf. Templates: Format field and column, Preview.
Tested: stored template had no script/onclick/img/javascript:/iframe/style/comment; draft HTML with nested bullets, link, pre, offerings bullets,
own line escaped; plain template unchanged; browser preview 5 li + link _blank, clipboard text/html + text/plain; preview follows edits; AI
request carries FORMAT HTML. No console errors.

# Investor database builder (standalone, NOT part of SSeeds): SSeeds-cb-investors.js 1.0.1 (1.0.1: a LinkedIn address with a broken %-escape is kept as written)
Run once on the machine with the Crunchbase 2022-07-07 bulk CSVs: node SSeeds-cb-investors.js <csv folder> <new.sqlite> (Node 22.5+, node:sqlite).
Streaming RFC 4180 reader (quotes, "", commas/newlines in fields, CRLF, BOM); malformed lines counted and skipped. Load order and filters:
investors (all) -> investments (investor known) -> investment_partners -> funding_rounds (used) -> funds (entity investor) -> org_parents (touching
investor firms) -> jobs (at investor firms, or of investor people) -> organizations (investor firms, parents, job firms, portfolio companies) ->
organization_descriptions (not portfolio) -> people (investor people, partners, job holders) -> people_descriptions -> category_groups.
Tables cb_<file> keep the Crunchbase columns (…_usd, rank, *_count, num_* NUMERIC).
Derived: investor_profile (n_investments, n_lead, lead_share, n_portfolio, first/last_on, n_recent = last 3 years of the data, stages, sectors
= category groups, categories, countries, regions as JSON [name, count], round p25/median/p75 USD, funds, partners on rounds, current people,
description); investor_keys (domain | name | linkedin, normalized); person_keys (first/last/full normalized, org, org_domain, title, current,
is_investor, linkedin); about (table descriptions, counts, build info). Later: fuse the Google mailing list on these keys, add web data, then load
into SSeeds (separate module and database, to be designed).
Tested on sample files with the real headers: filters, quoting, numbers, profiles, keys; 210 MB parsed in ~2 s; 200k investors + 200k kept
organization rows in 11 s.

# Contacts fusion (standalone, NOT part of SSeeds): SSeeds-cb-fuse.js 1.0.2 (1.0.2: people by firm loaded once - person_keys has no org_uuid index; progress every 200 addresses)
node SSeeds-cb-fuse.js <cb-investors.sqlite> <Google Groups members CSV> <new.sqlite>: the cb file is read only; writes our own database + <new>-review.csv.
Members file: header line found by "Email address"; kept = Group status member/manager/owner and empty Email status; duplicates, invalid,
bouncing/other statuses -> excluded (with reason). Matching order: 1 person at the firm (person_keys.org_domain = the email domain or its base,
name fits the local part: first+last/last+first = high, initial forms = medium, first or last alone = low); 2 firm (investor_keys domain; several
-> most investments, others listed); 3 personal mail (freeMail(): provider name + only the ending) first.last -> exactly one investor person = low;
4 none. Tables: contacts (match_status, investor, person, title, method, confidence, candidates, reviewed, note), excluded, contact_investors
(profile + firm details), contact_people, contact_investments (with company sectors/location), about.

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
Tested on sample members: high/medium/low person, firm, subdomain -> firm, gmail
name only, none, bouncing/invited/duplicate excluded.
1.0.1 (no review by hand): firmsFor(dom) = exact domain/base; else same label with
another ending (investor_keys range label.); else a domain
  whose label core equals the email's (core strips
ventures/vc/capital/partners/fund...; core >= 4); else an organization investor
whose
  normalized name core equals it (core >= 5). People at the found firms
(person_keys.org_uuid) are matched to the address too; confidence drops a
  step when the firm came by ending/name. personalDom = webmail + .edu + *alumni* +
substack/medium/linkedin. Any address left with no firm may
  match one investor person by first+last name (name only, low). First-name-only
matches say "the only one by that name there" (medium at a
  domain firm) or "n by that name there" (low).

# Web step (standalone, NOT part of SSeeds, no AI): SSeeds-cb-web.js 1.0.0
node SSeeds-cb-web.js <a2e-investors.sqlite> <cb-investors.sqlite> [--fetch-only |
--extract-only] [--retry] [--sites N] [--limit N]; run on mx.
Sites: contact_investors firms (domain or homepage) + base domain of every 'none'
company-domain contact. Fetch: robots.txt (agent * or a2e;
  Disallow/Allow longest match, Crawl-delay 1-10 s), https, https://www., http;
redirects followed by hand (<= 6 hops); HTML only (<= 3 MB);
  readable text (no script/style/svg/head) + links; up to 6 same-site pages picked
by path/link text: team, portfolio, thesis, about, contact;
  >= 1 s between pages; N sites at a time; resumable (todo / ok / failed / robots;
--retry). Tables web_sites, web_pages.
Extract (rules) -> web_profile: emails (text + mailto), team (name line + title
line near it, not all title words; linked to person_keys.full),
  portfolio (external links on portfolio pages minus social/tools ->
cb_organizations.domain -> sectors, countries), stages (word list), sectors
  (cb_category_groups names in home/about/thesis), geographies, cheques ($/€/£
ranges, up to / from, near check/ticket/invest words), looks
  investor / company / maybe / unclear (investor words + portfolio page vs product
words), evidence.
Rematch -> web_rematch log: 'none' contact whose site's final domain is an investor
firm -> firm (medium, "web: X is now Y", firm copied into
  contact_investors); 'firm' contact without person -> team member whose name fits
the address. SSEEDS_WEB_MAP (tests only) maps domains to a local server.
Tested with a local server: robots (private page never requested), redirect unknown
-> acme gives the firm, team (incl. surname "Partner") linked
  to Crunchbase, portfolio known/unknown, LinkedIn ignored, cheques $250K-$2M / up
to $5M, pre-seed vs seed, company site classed company,
  rerun idempotent, resume only new, unreachable site failed with reason, --retry.
SSeeds-cb-web.js 1.0.1: OWN domains (a2e.co, angeltoexit.com, all.net, manalyt.com,
scalingseeds.com) never fetched or matched (status own);
  text sectors only from sentences (>= 6 words) on home/about/thesis, counted when
mentioned twice or on a thesis page, GENERIC_CATS left out,
  group "Other" dropped; web_profile.thin = 1 when the site's pages hold < 600
characters of text (JavaScript-built). Full run on mx: 1,085 ok,
  141 failed, 6 robots; looks 498 investor, 67 maybe, 262 company, 258 unclear; 12
firms and 64 persons found again -> contacts 813 person,
  397 firm, 45 name only, 677 none.

# Investor intelligence in SSeeds (investors 1.0.0, investors-client 1.0.0, ui.js
1.0.18, backup 1.0.2, team 1.0.3, team-client 1.0.4, install 1.0.21, help 1.0.24)
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

Data: SSeeds-investors.sqlite in the operational folder (config investorsDb; CFG.investorsDbPath) = the dataset built by SSeeds-cb-fuse.js + SSeeds-cb-web.js (contacts, contact_investors, contact_people, contact_investments, web_*), journal_mode DELETE; SSeeds adds inv_notes. Missing file -> module switched off with the reason. MODULE_FILES: ... library, investors, backup, help. ui.js: restore (all and data) includes it; core.saveHistory. backup: VACUUM INTO of INV.db (core.moduleApi('investors')). Access: admin, or Team investor_intel (team_members column; admins set it; a manager who has it sets it for their own people; members never their own); team api canInvestorIntel(userId).
Memory model (reloaded when the file changes): contact_investors + web profile (by investor_uuid or domain) + contacts; web-only investors = web_profile kind unmatched, looks investor/maybe, not leading to a known domain (id "web:<site>", name from the home page title).
Routes (all need access): GET /api/inv/access, /api/inv/company/:id (record, values, pipeline items), POST /api/inv/triage {company_id, min_score, with_people, type} -> 200 best with parts sector 35 / stage 25 / size 15 / geography 10 (only when both known) / activity 15 / relationship 10, each {score, why}, weights renormalised over known parts; GET /api/inv/investor/:id; POST /api/inv/notes; POST /api/inv/search; GET /api/inv/choices; POST /api/inv/refine {company_id, ids (<= 60, chosen by the person)} -> AI (masked company text + compact investors) -> fit/score/why/angle/concerns, saved in History (kind investors).
Client: panel at the top of the Investor match tab (Investors for a company, Find investors; the old form below as "Companies for an investor"), investor details window, Find investors buttons in company details and pipeline items (hooks company, item), login hook checks access.
Tested (t9): access adm/mgr/usr and the give chain; triage reasons; refine with masked text and History; details; notes; search; choices; buttons from company details and an item; backup holds the dataset; without the file the module is off and the rest works; no console errors.

# Capabilities, roles, exceptions, kept records (ui.js 1.0.19, roles 1.0.0 and the modules listed in Files)
Principle: you see only what your job needs; what you do not hold is not shown and is refused; nothing is ever deleted; there is always a way back.
Catalog (single source = the code): ui.js declareCap({id, label, group, about, kind control|field, starts [member|manager], prefix, db}); modules call core.declareCap (staged: a module that fails declares nothing). Core: Search Ask InvMatch SQL History Settings Import Users UserRemove Pipelines (prefix /api/gigs guards every gigs route) AllCos Export SavedInvRemove; fields co.<col> from SSeeds-import.js COLUMNS (not name/code). access: MyCos Activity Assign AssignMgr GiveAllCos ActAs. team: Team TeamEdit TeamNotes TeamRemove Domains. roles: Roles Give Kept. companies: Merge AddCo EditCo. events: Events (meet and match use EV.requireAdminOrManager = Events). meet: Staff. mail: Templates Mail. library: Libraries. investors: Intel + fields inv.contacts_email, contacts_people, profile, location, links, portfolio, rounds, people, investments, web, notes. backup: Backup BackupMake Restore. Gigs levels unchanged.
Decision: can(user, id|[ids]) - admin true; else effective(user) once per request (user._eff): ROLES.effective(user, base) or the starting roles from starts. base = member, + manager when managedCodes(user). fieldMode(user, key): admin write; else the highest mode the person's roles set, exception sets exactly; nothing set = write. hiddenFields(user, db). route(..., {cap}) checked in handle() (403 "not available to you"); a prefix cap guards a path. /api/me user.caps, user.roles, full_read = can AllCos. Client: S.can(id),

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

TABS/registerTab cap (old who still works), who line "name (roles)".
Companies fields: scopeOf(user).mask = coMask(hidden) {cols, vals, files};
runSelect passes mask -> the query process copies companies with NULL for
  hidden columns, company_values/company_files without hidden fields, FTS from the
copy (also when scope.all); meta drops facets/columns/date fields;
  /api/company/:id leaves them out; companies form/record: none = not sent, read =
readonly, only write fields are written; investors company() masks.
Investor fields: details/search/triage rows drop what is none; AI text leaves it
out; notes POST needs inv.notes write (can_note).
Roles module tables (app db): roles (builtin manager|member, gone_at), role_items
(item, mode null|none|read|write, from/to), role_people (from/to, note),
  person_items (op + | - | mode, note, from/to), caps_seen (new catalog items go
into the starting roles per starts, once), roles_meta (flags_moved).
  A change ends the current row (to_at, to_by) and adds a new one; Put back
(POST /api/roles/back {kind item|role|exception, id}) adds it again.
  Once: team_members.may_add_companies -> +AddCo, investor_intel -> +Intel,
users.full_read -> +AllCos (columns no longer read).
  Give rules: admin anything; else Give + target is own people (team reports, or
managedUserIds) + not self + not an admin + only items held (role: all
  its items). GiveAllCos (access full_read route) keeps the old rule (managers ->
managedUserIds) and sets/ends +AllCos.
Routes: GET /api/roles (catalog, roles, can_edit, can_give, me) [Team]; POST
/api/roles, /api/roles/:id/item, /:id/remove, /:id/back, GET /api/roles/gone,
  /api/roles/who [Roles]; GET /api/roles/:id/history [Team]; GET
/api/roles/person/:uid (self, own people, Roles, admins) [Team]; POST
/api/roles/person/:uid/role,
  /item [Give]; GET /api/kept, /api/kept/:id, POST /api/kept/:id/back [Kept]. api:
summary(viewer, uid) (used by team GET per member), setException,
  ownPeople, mayGiveTo, effective.
Kept records (ui.js): table kept (tbl, row_json, summary, why, by, at,
restored_at/by); keep(tbl, where, args, user, why, summary) moves rows there;
  restoreKept(id) inserts them back (same id; refused if taken). Accounts are not
moved: users.gone_at/gone_by + disabled (kept row users_gone, put back
  clears gone_at). Used for: users (Remove), saved investors, history entries,
company_access (Remove, merges), co_groups (merge main, split), co_apart,
  team_members, team_domains (taken off the list), meet_staff, mail_templates,
mail_offerings, mail_audience, gig_libraries. Not changed: Gigs (own
  history/archive), working data (sessions, temp tables, ticks, act_as). Company
values: SSeeds-import.js keepOld(db, id, set, multi, files, by, source)
  -> company_history (companies db) before an import update or a change in SSeeds
(changed values only).
Screens: Team list (Name by manager with indent, Roles, Exceptions
+code/-code/field:mode, optional Capabilities, Account, Links, Changed; a person
  open = list shows Name/Roles/Exceptions), person pane (roles, give/take,
exceptions add/end, holds now, fields, history put back, then Details),
  Roles grid (controls by group in one table; fields per database in a closed
details table; ticks/selects apply at once; role panel: name, about,
  remove, history, put back; Removed roles); Kept records tab (ui-client: Show, Put
back). Help: /api/help replaces <!-- CAPABILITIES --> with the catalog.
Tested (t11 = t9 data; old code on 8898 for comparison): every GET/POST probe
answers the same status for adm/mgr/usr/usr2 (only messages differ);
  menus identical (+ Kept records for admins); 40 + 19 API checks: roles, give
rules, exceptions and put back, fields in SQL/meta/details/form, investor
  fields, kept records and put back (account, domain, offering, library, team
member, access, history), company_history (SSeeds and import), Events taken

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

away also stops matching, Pipelines taken away stops /api/gigs, Act as uses the target's capabilities; Playwright: Team, person, grid, Kept, help, every screen for adm/mgr/usr without console errors.

team-client 1.0.6, ui-client 1.0.15, help 1.0.26: Team, person pane, Roles grid and Kept records tables full size (no .tblwrap box; FULL margin only), the page scrolls and th (sticky top:0) stays at the top of the window; grid checkboxes 1em, a ticked box also has a "  " text span (copy/print); field menus font:inherit. Roles stay flat; a new role from another = Add a role, "a copy of". Tested: no scroll box, 63  in a copied grid, no errors.

team-client 1.0.7, help 1.0.27: a person (and a new member) opens in S.modal (openPerson(id); load(reopen) shows it again after a change; Save -> load(id), Remove -> closeModal + load); the list stays full width with the chosen columns; OPEN is only "roles" (the grid). Tested: window, exception set inside it, list updated underneath, Esc, new member window, no errors.

Manager covers Member (roles 1.0.1, team-client 1.0.8): setRoleItem = setOne + chain for the built-in roles - ticking for Member ticks Manager, unticking Manager unticks Member, a field: Member never above Manager (not set = write); the answer carries also[] and the grid is drawn again (open field groups kept, OPEN_F). Once (roles_meta manager_covers_member) Manager gets what Member has; new catalog items with starts member also go to Manager. Grid: last column Exceptions (GET /api/roles exceptions[] for those with Roles): +name, −name, name:mode.

Nothing is removed (ui 1.0.20, ui-client 1.0.16, team 1.0.5, access 1.0.6/-client 1.0.5, mail 1.0.8/-client 1.0.8, library 1.0.2/-client 1.0.2, help 1.0.28): capabilities UserArchive, TeamArchive, SavedInvArchive (label Archive), screen Archived ("Archived records", tab id kept); renamed once in roles afterLoad (current rows ended "system: renamed", new rows "system: was ..."); every Remove/Delete of a record is Archive, Put back of an archived record is Unarchive, activity "... archived" / "unarchived". Not changed: Gigs (Remove person from a sieve, Delete type), import file list. Tested: upgrade from the installed 1.0.0 set (t13: 2 rows renamed, Manager +14), chain both ways and fields, exceptions column, menus and API same as before on fresh data, 40 + 19 checks, every screen without errors.

team-client 1.0.9, help 1.0.29: no drop-downs for modes - grid field cells are radio groups (- none read write, name rf<role>-<item>, chosen in bold), the exception op is a radio group (+ / − or not shown / read only / changeable). Tested: one click sets the mode (Manager read -> Member read too), exception set.

team-client 1.0.10, help 1.0.30: grid checkbox alone (the  text beside it is gone); field choices wrap (cell not nowrap, each label inline-block nowrap).

roles 1.0.2, team-client 1.0.11, help 1.0.31: All / None per role per set - each group line (and "All fields" in a field table) has, per role, All / None (fields: - none read write); POST /api/roles/:id/items {items} in one transaction, each item through setRoleItem (history, Manager covers Member). Tested: new role, Screens All (19) and None (19), Investor dataset all read (11), no errors.

gigs-client 1.0.19: Pipelines list table width 100% (as Gigs; .gt stays width:auto for the bar tables); linksCell puts the space between links outside the nowrap span, so [Web] [DB] break onto the next line instead of running out of the cell. Tested 1400px: no scroll, 0 link cells overflowing, 43 on two lines.

help-client 1.0.2: no max-width on the manual sections - the text fills the help window and follows its width (tested 1600px: section 1400 of 1436).

Full size everywhere (ui.html 1.0.15, ui-client 1.0.17, investors-client 1.0.2, companies-client 1.0.6, library-client 1.0.3, mail-client 1.0.9, team-client 1.0.12,

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

help 1.0.32): .tblwrap has no scrolling box, height or frame (the page scrolls; th sticky); .tblwrap.list>table width 100% for lists of records (Search/Ask/SQL/ History results, History list, Investors for a company, its AI look, Find investors, Companies groups and people, Libraries list and preview, Templates list, Team list, Archived records); small tables natural (Backup, an investor's In touch and Investments). td.long max-width and the Team list max-widths gone; fitWraps (height fitting) removed; Draft email preview no 40vh box. Unchanged: Pipelines notes window (a 90% window that scrolls), item edit (only sideways). ui.html also: .tblwrap.list th wraps, .tblwrap.list td a overflow-wrap:anywhere. Tested (adm, t11): Search/Activity/History/Team/Templates lists = area width at 1600/1200/900px,\n no scrolling boxes, no errors; Search at 900px still wider than the window (13 columns, dates on one line) - the page scrolls sideways there. Settings as perl GigSettings (gigs 1.0.17, gigs-client 1.0.20, events-client 1.0.7, help 1.0.33): openSettings = Back, SaveSettings, one bordered table (Types: box | <type>: statuses box over Selects box, boxes as tall as their lines), Don't show: [□ status] per status, Def: [Def-K] boxes; Expert/Admin only (link: at('Expert') && can_settings; server: canSettings + level Expert). POST /api/gigs/settings {sieve_id, types[{name,statuses,selects}], hidden[]} in one transaction: names matched case-insensitively (update + un-retire) or inserted; a type not listed gets gig_types.retired_at/retired_by (kept, typeOut retired); answer {added, changed, out}. Items: save refused while the type is retired or the final status is not in the type (the next change forces current settings); POST /api/gigs refuses a retired type; old DELETE /api/gigs/types/:id now retires (nothing deleted). Defaults universal: table gig_defaults (user_id 0), DEF_KEYS Details..Who, start Quality/Priority Med; GET/POST /api/gigs/defaults, change = admin or manager at level Expert; createGig uses them. Events: retired types not offered. typeForm and the Add type/Delete buttons gone from the client.

Tested (t11): Playwright screen, add/take out a type, Don't show tick, Def-Quality, no errors; API 19 checks (levels, defaults universal, Def on new item, old status and retired type refused until changed, put back = same type). People on items in bulk (gigs 1.0.18, gigs-client 1.0.21, help 1.0.34): Redate gets People (client :201, :278; bar label ↑PEOPLE↓ orange :330). checksCell :343 select 'PEOPLE ON CHECKED' + optgroups Add (+ name, add:<id>) and Take off (× name, rm:<id>) from /api/people; rm asks first (:353); sent through the same onchange -> POST /api/gigs/bulk {action:'people', value, note}. gigs.js /api/gigs/bulk :579: act people (:582) - person active, canManageGig on EVERY checked item (else 403, nothing changed), current gig_users of the person read once; per item only a real change (add when not on / take off when on) -> history 'People' 'added|removed <name>' (:606), gig_users insert/delete in the same transaction; Owner NOT set (as the item's control); the note goes with it as usual; activity 'pipeline bulk people' per company with 'added|removed <name>'. Items are items: POST /api/gigs/:id/users (:550) no longer refuses company-less items, and the item's People drop-down (client :529) shows on every item. core.dbg / client dbg lines for the bulk people path.

# One Search across every database (ui.js 1.0.21, ui-client 1.0.18, ui.html 1.0.16, index 1.0.0 new, install 1.0.23, gigs 1.0.19, gigs-client 1.0.22, events 1.0.8,
#   meet 1.0.6, team 1.0.6, mail 1.0.9, library 1.0.3, access 1.0.7, investors 1.0.2, help 1.0.35)
Agreed: one Search tab (Ask and SQL are on it; Investor match and History stay); Any of words (All of, And/Or, !word) in everything the person may see, any subset ticked (Look in); the company choices (More) still there; one results table sortable by any column incl. Match; a row opens where it lives and the results stay; no per-person copy of the data - indexed searches;

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

SQL and Ask on every database, only what the person may see; History
  keeps the search, never its results (the AI's answer is not kept either). Text indexes built once by a command, kept current by triggers.
Indexes: SSeeds-index.js INDEXES (db companies|app|investors, table, fts, cols) - fts5 external content (content='<table>', rowid), tokenize unicode61
  remove_diacritics 2, triggers <fts>_ai/_ad/_au; only the listed columns the table has; --check (exit 1 if missing/different), --rebuild. companies_fts
  stays the importer's. Root step (install 1.0.23): cd /home/<app> && runuser -u <app> -- node SSeeds-index.js /home/<app> (safe to repeat). MIN_NODE 24.22.0
  (+ setAuthorizer check in step 2); SSeeds-index.js in PROG.
Sources (ui.js :687 SEARCH_SOURCES, addSearchSource :690, core.addSearchSource staged :1334, core.sq {inList, scopeIds, qlit, qid}):
  {id, label, db main|app|inv, name (the view in SQL), cap, fts, rid, title, date, company|companyId, key [[col,label]], open {kind company|item|event|history|record, col},
  about (for the AI), view(user) -> one SELECT (schema main./app./inv.) of only the rows and fields the person may see, or null}.
  core: companies (scope + mask NULL), company_values / company_files (scope, hidden fields out; SQL only), columns_map (SQL only), company_history (scope,
    hidden fields out), people (id, username; caps of /api/people; SQL only), saved_investors (InvMatch), history (own; History), archived (kept; Archived).
  gigs :703 viewWhere (visibleWhere with literals) - items, item_history (cap Pipelines; company name only where in scope), company_notes (scope codes).
  events :163 evWhere (canSeeEvent) - events, signups. meet :293 - meetings, meeting_people (via the event). team :180 - team (canSee; mgr_notes as seesNotes).
  mail :440 - templates (usable sieves, as GET /api/mail/templates), offerings. library :133 - libraries (Libraries). access :204 - activity (as GET /api/activity).
  investors :268 groupOf / :280 invView - investors, investor_contacts, investor_people (inv.people), investments (inv.investments), investor_web,
    investor_pages (inv.web), investor_notes (inv.notes); cap Intel; each column NULL where its field group is none.
  sourcesFor(user, ids) :727 - held (cap), investors only with the module loaded; a source that throws is left out (WARN).
Query process (QUERY_SRC, runSearch :462 - same kind of separate process, killed after queryTimeoutMs; old runSelect path unchanged for Investor match):
  searchAll :348 opens companies read-only as main, ATTACH app, inv (the process itself). modes:
  words :385 - per source: SELECT v.* FROM <schema>.<fts>(OR of the positive terms) f JOIN (<view>) v ON v.<rid>=f.rowid [WHERE company in the More choices
    (searchWhere :664, cleanFilters :744 drops choices on hidden fields)] ORDER BY f.rank LIMIT 2000; judge :412 decides on the fields the view gives only
    (All of / Any of / And-Or / !): match = round(90 x share of the words found + 10 if one is in the title), hits, best field, snip :427 (words marked
    \u0001..\u0002), key, open; sorted match, hits, date; limit maxRows; counts, missing (index not built), ms per source.
  sql / describe - each view made as TEMP VIEW under a name made for this run (v<16 hex>_<name>, never shown) and the name people use as a view of it;
    setAuthorizer: SELECT, FUNCTION, RECURSIVE ok; READ temp only of those views; READ of a database table only from inside a secret view (inner), or with no
    column when this statement already read that table inside one (SQLite rechecks a left-out LEFT JOIN that way); PRAGMA data_version (FTS); direct tables
    (directTables :739: companies, companies_fts*, values, files, columns_map) only for those who see every company and field; everything else refused, the
    error says what (never the secret names). record - SELECT * FROM (view) WHERE

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
rid = id.
Routes: GET /api/search/sources :1180 (held sources, index there?, sql_ok,
can_ask/can_sql); POST /api/search :1189 {words {any, all, andor}, sources,
filters}
  (no words: buildSearchSql as before -> rows of the one table; merged-company
filter applied to company rows); POST /api/search/tables :1211; GET
  /api/search/record :1217 (404 for not yours/not there; activity); POST
/api/sql :1229 {sql, sources}; POST /api/ask :1249 {question, sources} ->
askAll :795
  (describe -> schemaAll :785 -> AI SQL -> sql mode, one correction -> AI answer).
saveSearch :769 -> history.search_json (ALTER :138), no result_json;
  GET /api/history(/:id) gives search. SQL_OK :783 (Node without setAuthorizer:
words work, SQL/Ask say Node 24.22 is needed).
Client: TABS :65 (ask, sql gone; FIRST_TABS search); loadSources :301 (Look in
ticks, unticked kept in memo 'search-off'); doSearch :282; renderSearch :330
  (Match, Database, Table, Key = link, Field, Matched bold, Date; heading
sorts/reverses; results hook with the company rows; CSV); openResult :362 (company
  window, S.openGig(id, {back:'search'}), S.openEvent, History rerun,
recordWindow :370); doAsk :383 / doSql :442 / showTables :447 into #s-out;
rerun :469
  (History: fill words, Look in, More, question or SQL, run). gigs-client RETURN_TO
:465, backToList :473 -> S.show('search'); Back button "Back to Search".
Not changed: access-client registers My companies before 'ask' (gone) - it is added
at the end of TABS; the gear puts it second anyway (FIRST_TABS).
Known limit: a count over a cross join with a table a view uses (SELECT count(*)
FROM items, app.gigs) tells how many rows that table has - no content.
Tested (Node 24.22, t inst: 4 companies, 6 items, 2 events, 2 meetings, team,
investor dataset; adm/mgr/usr/usr2): index build/check (exit codes), triggers
  (edit -> found at once); words per role (usr only C1 records, usr2 nothing, mgr
own event not the admin's), accents, a-b, word*, !word, All of, More
  through the company, hidden co.email/co.country (words, filters ignored, SQL
NULL); SQL attacks refused (app.gigs, users, WITH named like a view, two
  WITHs, sqlite_temp_master, pragma_table_list, companies_fts for usr, sessions,
ai_settings, two statements), legit SQL per role; record windows 404 when
  not visible; Ask with a fake AI (adm 26 tables, mgr 20); Node 22: words ok,
SQL/Ask refused with the reason. Playwright: Search adm/mgr/usr, Look in kept
  after reload, sort, item -> Back to Search (32 rows still there),
company/record/event windows, SQL, Tables, History Run again; no console errors.

# Investors spreadsheet + Crunchbase -> a new investor dataset (standalone, NOT
part of SSeeds): SSeeds-inv-sheet.js 1.1.1
1.1.1: Crunchbase's columns stay as they are (no sheet_changes); the sheet's values
go in sheet_ columns of their own (sheet_type, sheet_stages, sheet_sectors,
  cheques, cheque_min/max_usd, leads_pre_seed, tier, warm_intro_path, contact_info,
sheet_city, sheet_region, sheet_country_code, sheet_description,
  sheet_homepage_url, sheet_domain, sheet_file_date (local date), sheet_id) so
SSeeds shows each source with its date; sheet-only firms keep them in the main
  columns too. (Text below describes 1.1.0 where it differs.)
Agreed: as before, each new source is fused with Crunchbase into a NEW SSeeds-
compatible investor dataset (step 1), which SSeeds-cb-web.js then updates
  (step 2, unchanged); fusing a finished dataset into the current SSeeds one is one
standard routine, later. Match on everything (firm, Name, website and
  email domains, LinkedIn, people) and bring in all Crunchbase has on what matched.
This format only; no partner attribution; newer (the sheet) wins,
  the old kept. 1.0.0 (sheet only) is replaced.
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

node SSeeds-inv-sheet.js <cb-investors.sqlite> <spreadsheet.xlsx> <new.sqlite> - on mx; needs SSeeds-import.js next to it (readXlsx); never overwrites.
Sheet: header row = first row (first 20, any sheet) with Name and Website; used 16 (Tier, Name, Website, Partner Names, Warm Intro Path, City, State,
  Country, Stage, Check Size, Leads Pre-Seed, Focus Areas, Notes, Contact Info, Type, Contact Email); P-T (Outreach Priority, Priority Score, Has
  Application Form, Cold Outreach Sent, Application Form Completed) stored nowhere; a used header missing -> stops. File date = saved date, else file time.
  Firms joined by domain (base) else name; later row wins. Emails from Contact Email + Contact Info; LinkedIn links from Contact Info; partners split.
Firm matching (points per Crunchbase investor, added up): web site host -> firmsFor (= cb-fuse: exact domain 3, other ending / domain core / firm name 2);
  each non-personal email domain the same way; LinkedIn company link 3; norm(Name) = investor_keys name: 3 with the same city or country, else 2;
  each partner found (person_keys first+last) at an investor firm 1; a namesake in another country -2. Best with >= 2 wins (one partner alone is not
  enough); confidence high (3-point evidence or >= 4, unless tied), medium (3), low; candidates kept. None -> investor_uuid sheet:<domain|name>.
People: at the matched firm (person_keys by org via BY_ORG, and by the firm's domains): first+last (high), the address by fits() (3 high / 2 medium /
  1 low), LinkedIn /in/ (high); else name alone when exactly one investor person fits (low, "name only"); else sheet:p<n>. LinkedIn profiles not named
  in the sheet are added as people.
Output (cb-fuse's tables, so cb-web and SSeeds read it unchanged): contacts (one per address; group_status sheet, joined_on file date); contact_investors
  (cb-fuse's SELECT for matched investors incl. investor people; sheet wins for homepage_url, city, region, country_code (3-letter), short_description, and
  domain when CB has none -> old value in sheet_changes; + sheet_type, sheet_stages, sheet_sectors, cheques, cheque_min/max_usd, leads_pre_seed, tier,
  warm_intro_path, contact_info, sheet_file_date, sheet_id; sheet-only firms: type organization, stages JSON from Stage (range fills between));
  contact_people (all cb_jobs people of the matched firms + matched people + sheet:p; + investor_uuid, email, sheet_id); contact_investments (cb-fuse's
  SELECT); contact_funds; contact_org_parents; sheet_match; sheet_changes; sheet_investors/people/rows; sources; about (+_built); <new>-review.csv.
Tested: real SSeeds-cb-investors.js on a made-up Crunchbase export (5 investors incl. a person, namesakes in two countries, jobs, funds, parents) + the
  made-up sheet: Alpha by domain + 2 partners (high), Beta Capital by name + city (Berlin namesake listed), Gamma by other ending + name + city + partners,
  angel Ann Lee by name only (low), No Site Angels sheet-only; 6 contacts, 7 people (incl. Zed only in CB), 6 newer values kept; SSeeds-cb-web.js runs
  on it (3 sites); SSeeds investors module loads it (search lists 5, stages from the sheet). Node 22 and 24.

# Investor datasets into SSeeds, and reconciliation (investors 1.0.3, investors-client 1.0.3, help 1.0.36)
Agreed: an import of a new investor dataset (inv-sheet / cb-fuse, then cb-web) is admin only (cap InvAdmin, starts []); the import is the newer one; the old
  is always kept and used: each source is shown and given to the AI with its date (data_2022, from_list = partners' list <file date>, web_now <read on>).
  After import, reconciliation as on the Companies screen. Not changed: inv-sheet/cb-fuse/cb-web, ui.js, backup (api.db is now a getter).
Tables in SSeeds-investors.sqlite: inv_datasets (file, built, added_at, added_by, backup, counts JSON, about JSON), inv_changes (dataset_id, tbl, rec,
  field, old_value, new_value, at - every value an import or a "put back"

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

replaced), inv_conflicts (dataset_id, email, old_json, new_json, status open | new match kept | earlier match put back, decided_by/at). In SSeeds-app.sqlite (core.keep on undo): inv_groups (member_id -> main_id, reason), inv_apart (id_a < id_b, evidence JSON), inv_review (kind contact|investor, rkey, decision ok|rejected|not an investor).
Import: POST /api/inv/import/upload?name= (raw body, <= uploadMaxMb) -> uploadDir/investors/<stamp>-<hex>.sqlite (600), checked for contacts, contact_investors, contact_investments; POST /api/inv/import/check {upload} (read only, ATTACH, fuse(write=false)); POST /api/inv/import/apply {upload, name} -> job: VACUUM INTO SSeeds-investors-before-import-<stamp>.sqlite (backup), VACUUM INTO a temp file, ATTACH the upload, fuse, integrity_check, close our DB, rename the temp over our file, openDb(), load(); activity line. GET /api/inv/import = the datasets.
fuse: per table of the upload (not virtual, not inv_*, about, *_fts*): missing columns added; KEYS contact_investors investor_uuid, contact_people person_uuid, contacts email, web_sites/web_profile site, web_pages site+url, contact_investments investor_uuid+round_uuid, contact_funds uuid, contact_org_parents uuid+parent_uuid: new row inserted; same key: only values the upload has and differ are replaced (investments/funds/parents: kept; contacts keep our group_status and joined_on - they are about our Google Group); contact_investors/contact_people/contacts/web_profile log each to inv_changes, web_pages/web_sites one "(read again)" line; contacts whose investor or person differs -> inv_conflicts. Other tables (sheet_*, sources, sheet_match, web_rematch, excluded...): rows added with dataset_id (a table with a key is rebuilt without it so every dataset's rows fit); a table we do not have comes as it is + dataset_id. FTS triggers keep the indexes current.
load(): + partners' list per investor (v.sheet {date, stages, sectors, cheques...}, sheetOnly for sheet:<id>); web profile also by sheet_domain; rejected contacts not in touch; inv_groups: members' people, web site (if the main has none) and newer sheet go to the main, members leave the list and byId[member] = main; "not an investor" leaves the list. raw = every record (reconciliation). score(): sheet stages/sectors/cheques decide.
Details /api/inv/investor/:id: main's view with members [{id,name}], sheet (masked by its field groups), changes (inv_changes of it and its members, people and addresses, with dataset file; a field the person may not see is left out), sheet_only, not_investor. groupOf: sheet_<x> as <x>; cheques/cheque_m* rounds, leads_pre_seed portfolio, contact_info contacts_people, tier/warm_intro_path/type profile.
Reconcile GET /api/inv/reconcile: proposals (keys per raw record: base domain of domain/sheet_domain/web final_domain (not free mail or link sites) and LinkedIn = strong; nameKey (no inc/llc/capital/ventures/partners/fund/vc) = name, strong when same country; buckets of 2-30; union-find; a pair kept apart is skipped while its links are all in its evidence; main = Crunchbase first, then most investments, most in touch), low (confidence low or name only), looks_company (web looks company), merged_list, apart_list, conflicts.
POST /api/inv/merge {groups:[{ids, main, reason}]} (a main that is a member -> its main; members of the merged follow), /api/inv/split {member_id}, /api/inv/apart {pairs} (evidence = links now), /api/inv/apart/undo {a, b}, /api/inv/review {kind, key, decision | clear}, /api/inv/conflict {id, choice new|old} (old writes the earlier values back, logged in inv_changes).
Search: investor_changes (Intel; SQL/Ask; fields the person may not see left out), investor_datasets (InvAdmin). The partners' list columns are not in inv_investors_fts (SSeeds-index.js not changed): words search does not find them yet.
Client: Investors tab (InvAdmin, before Import): file, Check (upload + check

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

table), Import (job steps, result table), Datasets imported, Merge ticked / Find duplicates again, Proposed groups (Main, Apart, Details, Keep apart), Contacts matched differently (Keep the import's / Put the earlier back), To look at (Confirm / Reject / Undo; Is an investor / Not an investor / Undo), Merged (Split), Kept apart (Undo). Details: Partners' list (<date>) column, Merged with, Earlier values table. Find company: /api/search {filters:{name}, sources:['companies']} rows (company_id, key code/name) - was broken by ui 1.0.21.
Tested (Node 24, local install): base = 1.1.0 output, import of a 1.1.1 output with a sheet-only duplicate by domain, a same-name one, a contact moved to another investor, a name-only contact: check = apply counts, 6 columns added, 25 values replaced (1.1.0's overwrites put back to Crunchbase's), 1 conflict, backup made, FTS still finds the new values; merge, keep apart, reject, put earlier back, split, undo; SQL on investor_changes/datasets; Playwright: the tab, Check, Import, Merge ticked, decisions, details with the three sources and earlier values, Find company; no console errors.

# Reconciliation only of what imports bring in; upload as the Import tab (investors 1.0.4, investors-client 1.0.4, help 1.0.37)
Seen on the real dataset (1002 records, before any import): 13 groups from the dataset as built - expired domains that all redirect to hugedomains.com chained 3 firms; "builders" (Builders VC / The Builders Fund) by the shortened name; Alumni Ventures funds by avgfunds.com. Firefox: the file chosen never reached the input (files.length 0, accept ".sqlite,.db").
Agreed: the dataset as built has no investor twice -> only what imports brought in is looked at; fix the links; show the data to decide; upload as Import.
fuse: an added row of contact_investors, contact_people, contacts, web_profile, web_sites -> inv_changes field '(added)'. fromImports() = inv_changes recs (contact_investors -> investor ids, contacts -> emails, web_* -> sites and web:<site>). proposals: a pair is kept only when one side is from imports.
keysOf: domain = base(own domain) and base(sheet_domain) (not free mail, not link sites) strong; LinkedIn strong; fullName (no inc/llc/ltd/lp/llp) strong in the same country, weak otherwise ("country differs or not known"); nameKey shortened (no the/capital/ventures/partners/fund/vc) weak; web final_domain not used. Group strength: strong (all links strong, ticked), partly (a weak link in the chain, not ticked), name only. Record: from_import, linked_by, site {site, goes_to (redirect), looks}, description (first sentence, 200), people (6 names or addresses), last. low and looks_company: from imports only.
DEBUG (server: log lines of upload, check, apply, reconcile; client: console lines of the file picked, upload, check, reconcile), false.
Client: drop box (class drop, as Import) with Choose file… (hidden input, no accept, value cleared after reading; file kept in R.picked), "Chosen: name (size)", Check enabled only then. Group table: Main, Apart, Name, From (+ (import)), Linked by, Domain, Web site, Place, Investments (last), In touch, Description.
Tested: before any import 0 groups; after the test import 2 (domain; full name same country); Playwright: Choose file… (filechooser) and drop both give "Chosen: g1x.sqlite (96 KB)", Check, Import, decisions; no page errors.

# Working indicator (ui-client 1.0.19, ui.html 1.0.17, help 1.0.38)
Agreed: in the middle of the top bar "Working… Ns" while anything takes more than a second, everywhere; gone when the results are in.
ui.html: <b id="working" class="hidden" role="status" aria-live="polite"> between two .grow spans in #hdr (bold only, lf.js colors).
ui-client: api() = workStart(method url) -> apiCall() (the former api) -> workEnd, success or error; GET /api/job/<id> polls are not counted

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 

```
  themselves; a reply with job (startJob) starts JOBWORK[job] until a poll of it
says not running (done/error) or fails. WORK.run {id: {what, t}};
  a 250 ms timer while any run: hidden under 1 s, else "Working… " + seconds since
the oldest. dbg (DEBUG) logs work start/end with ms.
  Covers every module (all use S.api; investors and match job polls too) - no other
file changed.
Tested (Playwright): a request held 2.6 s: - - - 1s ... 2s - (hidden at once
after); a job running ~5 s across polls: 1s..4s steady, then hidden;
  Import done -> hidden.
```

 **Copyright (c) Fred Cohen, 2026 - All Rights Reserved** 