recodex-worker abstract file\_manager\_base + get\_file(string src, string dst): void + put\_file(string src, string dst): void broker\_connection<proxy> config\_: worker\_config socket\_: proxy - logger\_: spdlog::logger broker\_connection\_context<proxy> - broker\_cmds\_: command\_holder<br/>broker\_connection\_contextproxy>> worker\_core command\_holder<context\_t> fallback\_file\_manager - job\_server\_cmds\_: command\_holder<br/>broker\_connection\_contextproxy>> args\_: vector<string> # functions\_: map<string, callback\_fn> - primary\_fm\_: file\_manager\_base + config: woker\_config - reconnect\_delay\_: chrono::seconds config\_filename\_: string secondary\_fm\_: file\_manager\_base - context\_: command\_context<context\_t> prefixed\_file\_manager + broker\_connection(worker\_config conf, proxy socket, spdlog::logger log) config\_: worker\_config - prefix\_: string + fallback\_file\_manager(file\_manager\_base primary, file\_manager\_base secondary) + command\_holder(context\_t context, spdlog::logger log) job\_client\_context + connect(): void working\_directory\_: path - fm\_: file\_manager\_base + call\_function(string cmd, vector<string> args): void + receive\_tasks(): void logger\_: spdlog::logger + evaluator: job\_evaluator\_base + register\_command(string cmd, callback\_fn callback): bool + prefixed\_file\_manager(file\_manager\_base fm, string prefix) - remote\_fm\_: file\_manager\_base + socket: zmq::socket\_t - cache\_fm\_: file\_manager\_base - job\_receiver\_: job\_receiver fileman\_config http\_manager connection\_proxy command\_context<context\_t> cache\_manager broker\_: broker\_connection<connection\_proxy> + remote\_url: string - configs\_: vector<fileman\_config> broker\_: zmq::context\_t + logger: spdlog::logger caching\_dir\_: path - zmq\_context\_: zmq::context\_t + username: string - logger\_: spdlog::logger - jobs\_: zmq::context\_t + command\_holder(context\_t context, spdlog::logger log) - logger\_: spdlog::logger + password: string + worker\_core(vector<string> args) progress\_: zmq::context\_t + http\_manager(spdlog::logger logger) + cache\_manager(spdlog::logger log) + run(): void - items\_: zmq::pollitem\_t[] + http\_manager(vector<fileman\_config> confs, spdlog::logger log) + cache\_manager(string caching\_dir, spdlog::logger log) - context\_: zmq::context\_t # find\_config(string url): fileman\_config + get\_caching\_dir(): void + connection\_proxy(zmq::context\_t context) + connect(string addr): void namespace broker\_commands + poll(message\_origin::set result, int timeout, bool terminate, chrono::milliseconds elapsed\_time): void + process\_eval<context\_t>(vector<string> args, + send\_broker(vector<string> msg): bool command\_context<context\_t> context): void namespace jobs\_server\_commands + send\_jobs(vector<string> msg): bool + process\_intro<context\_t>(vector<string> args, + process\_done<context\_t>(vector<string> args, command\_context<context\_t> context): void + recv\_broker(vector<string> target, bool terminate): bool command\_context<context\_t> context): void + recv\_jobs(vector<string> target, bool terminate): bool job\_evaluator\_base progress\_callback\_base + recv\_progress(vector<string> target, bool terminate): bool + evaluate(eval\_request request): eval\_response + submission\_downloaded(string job\_id): void namespace jobs\_client\_commands job\_receiver + job\_results\_uploaded(string job\_id): void + process\_eval<context\_t>(vector<string> args, command\_context<context\_t> context): void - socket\_: zmq::socket\_t + job\_started(string job\_id): void + job\_ended(string job\_id): void evaluator\_: job\_evaluator + task\_completed(string job\_id, string task\_id): void - logger\_: spdlog::logger job\_evaluator + task\_failed(string job\_id, string task\_id): void - commands\_: command\_holder<job\_client\_context> working\_directory\_: path + task\_skipped(string job\_id, string task\_id): void + job\_receiver( zmq::context\_t &context, - archive\_url\_: string job\_evaluator eval, spdlog::logger log) archive\_name\_: path + start\_receiving(): void archive\_path\_: path progress\_callback - submission\_path\_: path - socket\_: zmq::socket\_t log\_config eval\_request empty\_progress\_callback - source\_path\_: path + log\_path: string command\_: string + job\_id: string results\_path: path + log\_basename: string connected\_: bool + job\_url: string - job\_temp\_dir\_: path + log\_suffix: string - logger\_: spdlog::logger + result\_url: string result\_url\_: string + log\_level: string static helpers + progress\_callback(zmq::context\_t context, job\_id\_: string spdlog::logger logger) + log\_file\_size: int + topological\_sort(task\_base root, vector<task\_base> result): void - job\_meta\_: job\_metadata + log\_files\_count: int eval\_response + copy\_directory(path src, path dst): void - worker\_conf\_: worker\_config - result\_: size\_t + job\_id: string + build\_job\_metadata(YAML::Node conf): job\_metadata working\_directory\_: path - job\_results\_: vector<pair<string, task\_results>> + result: string + get\_bind\_dirs(YAML::Node lim): vector<tuple<string, string, dir\_perm>> worker\_config source\_path\_: path - remote\_fm\_: file\_manager\_base + create\_null\_logger(): spdlog::logger worker\_id\_: size\_t - cache\_fm\_: file\_manager\_base - result\_path\_: path + get\_log\_level(string lev): spdlog::level::level\_enum working\_directory\_: string sandbox\_config - logger\_: spdlog::logger - factory\_: task\_factory\_base job\_metadata + get\_log\_level\_number(spdlog::level::level\_enum lev): int - broker\_uri\_: string + name: string + job\_id: string progress\_callback\_: progress\_callback\_base config\_: worker\_config + compare\_log\_levels(spdlog::level::level\_enum first, spdlog::level::level\_enum second): int headers\_: multimap<string, string> + loaded\_limits: map<string, sandbox\_limits> + language: string - job\_variables\_: map<string, string> - progress\_callback\_: progress\_callback\_base + send\_through\_socket(zmq::socket\_t socket, const vector<string> msg): bool - hwgroup\_: string - root\_task\_: task\_base + fm\_url: string + job\_evaluator(spdlog::logger log, worker\_config conf, + recv\_from\_socket(zmq::socket\_t socket, vector<string> target, bool terminate): bool - max\_broker\_liveness\_: size\_t file\_manager\_base remote\_fm, file\_manager\_base cache\_fm, + log: bool - task\_queue\_: vector<task\_base> path working\_directory) - broker\_ping\_interval\_: chrono::milliseconds logger\_: spdlog::loger + tasks: vector<task\_metadata> - cache\_dir\_: string static archivator + job(job\_metadata job\_meta, worker\_config conf, path working\_dir, - log\_config\_: log\_config + compress(string dir, string dest): void path src\_path, path res\_path, file\_manager\_base fileman) sandbox\_limits - filemans\_config\_: vector<fileman\_config> + decompress(string filename, string destination): void + get\_task\_queue(): vector<task\_base> + memory\_usage: size\_t - limits\_: sandbox\_limits + run(): vector<pair<std::string, task\_results>> + cpu\_time: float + wall\_time: float + extra\_time: float + stack\_size: size\_t + files\_size:size\_t + disk\_size: size\_t + disk\_files: size\_t + stdin: string abstract task\_base + stdout: string # id\_: size\_t enum dir\_perm task\_factory\_base + stderr: string # task\_meta\_: task\_metadata + RO + create\_internal\_task(size\_t id, task\_metadata task\_meta): task\_base + chdir: string # execute\_: bool + RW + create\_sandboxed\_task(create\_params data): task\_base + processes: size\_t + NOEXEC # parents\_: vector<task\_base> + share\_net: bool # children\_: vector<task\_base> + environ\_vars: vector<pair<string, string>> + MAYBE + bound\_dirs: vector<tuple<string, string, dir\_perm>> + task\_base(size\_t id, task\_metadata task\_meta) + DEV + run(): task\_results task\_factory + add\_children(task\_base add): void - fileman\_: file\_manager\_base + get\_children(): vector<task\_base> + task\_factory(file\_manager\_base fileman) + add\_parent(task\_base add): void + get\_id(): size\_t

abstract sandbox\_base

+ run(string binary, vector<string> args): sandbox\_results

fake\_sandbox

sandbox\_results

+ exitcode: int

+ wall\_time: float

+ memory: size\_t

+ max\_rss: size\_t

+ status: isolate\_status

+ time: float

+ exitsig: int

+ killed: bool

+ message: string

task\_results

+ sandbox\_res: sandbox\_results

+ failed: bool

enum isolate\_status

+ OK

+ error\_message: string

# sandboxed\_dir\_: string

+ get\_dir(): string

isolate\_sandbox

+ isolate\_sandbox(sandbox\_limits lim, size\_t id,

string temp\_dir, spdlog::logger log)

- limits\_: sandbox\_limits

- logger\_: spdlog::logger

- isolate\_binary\_: string

- temp\_dir\_: string

- meta\_file\_: string

- max\_timeout\_: int

- id\_: size\_t

+ get\_task\_id(): string + get\_priority(): size\_t task\_metadata + get\_fatal\_failure(): bool + task\_id: string + get\_cmd(): string + priority: size\_t + get\_args(): vector<string> + fatal\_failure: bool + get\_dependencies(): vector<string> + dependencies: vector<string> + is\_executable(): bool + binary: string + set\_execution(bool set): void + cmd\_args: vector<string> + set\_children\_execution(bool set): void + sandbox: sandbox\_config create\_params + worker\_id: size\_t + id: size\_t + task\_meta: task\_metadata + limits\_: sandbox\_limits + logger\_: spdlog::logger + temp\_dir\_: string root\_task external\_task - worker\_id\_: size\_t - - sandbox\_: sandbox\_base - limits\_: sandbox\_limits - logger\_: spdlog::logger extract\_task - temp\_dir\_: string + external\_task(create\_params params) + get\_limits(): sandbox\_limits

mkdir\_task

archivate\_task

rm\_task