International Journal of Management and Data Analytics
IJMADA

Vol 6 (1), 449-470
ISSN: 2816-9395
DOI: https://doi.org/10.68485/ijmada.20266168
Journal website: http://ijmada.com

When Chats Split Apart: Formalizing Temporal Communication Fragmentation and the CARTA Reconstruction Framework

Author(s): Maitreya Sapariya

Department of Information Technology, A. D. Patel Institute of Technology (ADIT), Charutar Vidya Mandal (CVM) University, New Vallabh Vidyanagar, Anand, Gujarat, India.
ORCID: 0009-0003-9346-3775

Corresponding Author: maitreyasapariya@gmail.com

Received: September, 2026, Published: November, 2026

ARTICLE INFO
Keywords:
Chat archive reconstruction;
Data governance; Data provenance;
Data quality; Temporal communication fragmentation

© 2026 by the Author(s). This open-access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license, making research freely available to the public and supporting a greater global exchange of knowledge and human experiments.

ABSTRACT

When a phone fails and its owner keeps the messaging account running on a second device, the account's history can end up in two archives, each intact and each missing what the other holds. We call this Temporal Communication Fragmentation (TCF), a loss by non-union: no message is deleted, but the platform offers no operation that joins the two halves. WhatsApp's own documentation states that old and new chats cannot be merged. We define TCF by three jointly necessary and sufficient conditions and specify CARTA 1.1, a framework for reconstructing one archive from several. CARTA is an output contract owed to a reader who cannot check the result. Records that cannot be read unambiguously are quarantined with a typed reason, and a report gives each record's provenance, the conflicts and deduplication ambiguities found, and every setting used. We evaluated a reference implementation, CHAMP-Pipeline, on synthetic fragment sets with known ground truth. With its declared date inference it reproduced the ground truth in all 4,500 configurations of a parameter sweep; the invariants and the predicted outcome held in all 5,000 random configurations of two to four fragments; and reconstruction stayed exact at one million messages. A zone-offset advisory, frozen before testing, flagged all 2,800 misdeclared zones in a designed test of overlapping fragment pairs and 1,470 of 1,542 in random configurations, and raised false alarms in 2 of 600 correctly declared runs. A systematic search found 60 chat-merging tools; none fully provides more than 2 of the five output properties CARTA specifies. With a fragment's time zone left undeclared, every benchmarked tool and CHAMP-Pipeline produce the same duplicated and misdated records, and only CHAMP-Pipeline's report says so. All data are synthetic; no real archive has been reconstructed.

1.0 Introduction

A phone stops working and goes in for repair. Its owner keeps the messaging account running on a spare handset, and the conversations continue there for as long as the repair takes. When the original phone comes back, its owner holds two archives of one account: the repaired phone has everything up to the failure, the spare has everything since, and no operation on either device produces their union. Restoring a backup onto the repaired phone does not append the repair window to it; it substitutes one state for the other.

Nothing was deleted. Both archives are intact, and each holds messages the other lacks. What is absent is not data but an operation, and the vendor says so directly: chats on a new phone cannot be merged with chats on the old one, and whichever set is transferred replaces the other (WhatsApp, 2026a, 2026b). The owner has lost the history without losing a message. It is a loss by non-union: both halves survive, and what is missing is the join.

We call this state Temporal Communication Fragmentation (TCF). The opening scenario is the author's own and prompted the research questions below. It is an anecdote, not evidence, and no claim in this paper rests on it.

1.1 Temporal Communication Fragmentation

TCF is a state in which a user's communication history for a single logical account is split across two or more device-bound application instances that persist independently, and for which the platform provides no operation yielding their union. TCF holds exactly when three conditions hold together (Fig. 1). Each is necessary, and together they are sufficient.

Condition 1, Bifurcation. At failure time tf, one authoritative timeline splits into two independently updating branches. The primary is paused; the secondary is active.

Condition 2, Independent persistence. Both branches survive and accumulate new history with zero synchronisation between them. This separates TCF from linked-device configurations, in which a companion receives its history from the primary (Section 1.2).

Condition 3, Absent reconciliation. No platform-native operation yields the union of the two archives. This is a property of the platform's reunification semantics, not an event: it holds from the moment the branches exist, whether or not the user attempts any reunification. It separates TCF from optimistic replication, which attempts reconciliation rather than declining to (Section 2.3). A platform that shipped a merge operation would stop producing TCF, as a fix removes instances of a defect class without refuting its description.

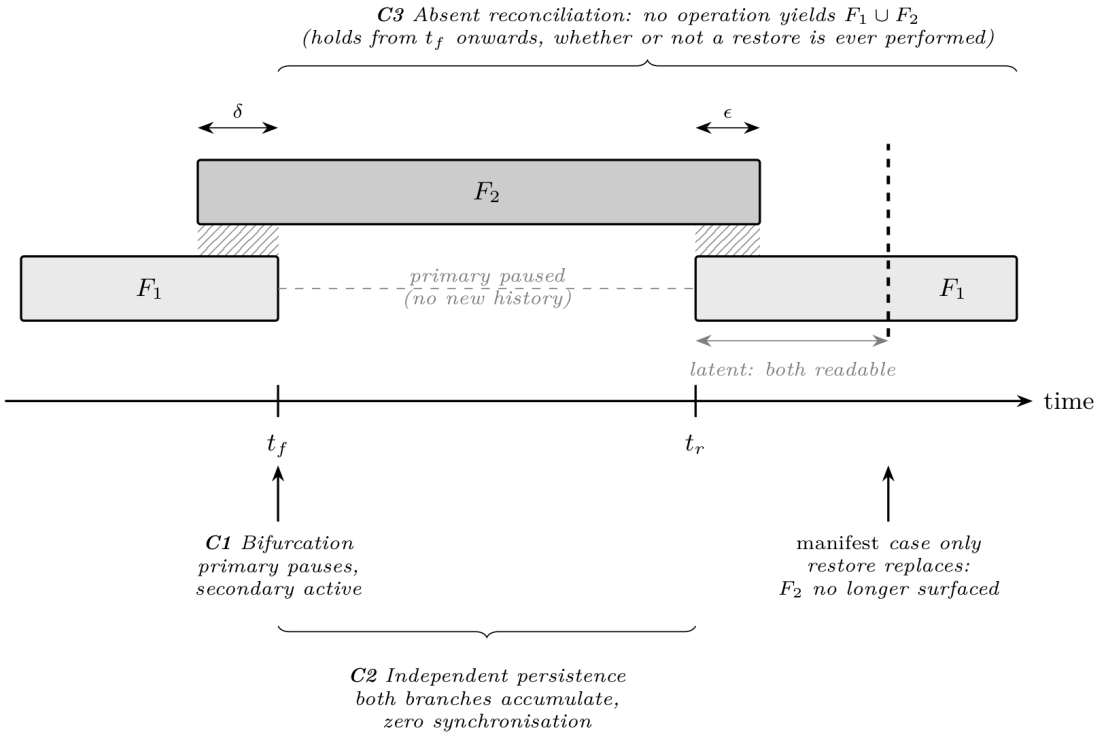


Fig. 1 TCF on a timeline. C1 and the restore are points; C2 and C3 are spans; hatching marks the overlap regions.

The conditions hold while both archives persist on their own devices: the phone the account has left is logged out, but its chats remain and reappear if the account is registered there again (WhatsApp, 2026c, 2026d). We call this the latent case, and the motivating incident is one. A restore makes the fragmentation manifest: a cloud backup holds one history per phone number, and a new backup permanently replaces the previous one (WhatsApp,

2026e), so the replaced archive survives only locally. Both cases present the same input to a reconstruction.

Formally, device failure at t_f and recovery at t_r split a ground-truth history T into $F_1 = \{m : t(m) < t_f \text{ or } t(m) \geq t_r\}$ and $F_2 = \{m : t_f - \delta \leq t(m) < t_r + \varepsilon\}$. The intervals are half-open, so with $\delta = \varepsilon = 0$ every message falls in exactly one fragment. $\delta \geq 0$ is history the secondary already holds from before the failure, and $\varepsilon \geq 0$ the time the secondary keeps receiving after the primary resumes. When either is positive the fragments overlap, so reconstruction needs deduplication, not concatenation.

The motivating incident is the boundary case $\delta = \varepsilon = 0$, where reconstruction reduces to ordered concatenation, but it is not the general case. A secondary device is usually set up from whatever backup exists, which carries pre-failure history the primary also holds; assuming disjointness would report that overlap twice. We know of no documented WhatsApp path that produces $\varepsilon > 0$, since registering the account on one phone logs it out of the other (Table 2); the model keeps ε so that a reconstruction does not depend on its absence.

A fragment is internally complete when it is well formed and gap-free over its own accumulation window; neither fragment covers T. This is an idealisation, and Section 4.6 treats its failure.

TABLE 1: TAXONOMY OF NEIGHBOURING DATA-LOSS AND CONSISTENCY PHENOMENA

Dimension	Backup staleness	Sync conflict	Git 3-way merge	TCF (this paper)
Underlying cause	Infrequent snapshotting	Concurrent multi-client writes	Divergent commit histories	Device switch and independent persistence; no merge operation
Number of timelines	Exactly 1	2+ concurrent	2 branches plus common ancestor	2 independently persistent, potentially overlapping
Platform behaviour	Restores stale snapshot	Rejection or branch fork	Automatic merge or conflict markers	No operation yields the union; every documented reunification path replaces, copies or leaves both apart (Table 2)
Physical data state	Missing (never recorded)	Present on server	Merged in working tree	Intact on both devices; no combined view
Resolution target	Data unrecoverable from the snapshot	Operational state restored	Codebase reconciliation	Reconstruction (archive plus report)

Table 1 contrasts TCF with backup staleness, sync conflicts and three-way merges. Optimistic replication also lets replicas diverge, but reconciles them on reunification (Saito & Shapiro, 2005). Multi-device use keeps one timeline but strands information on the device the user is not holding (Dearman & Pierce, 2008; Tseng et al., 2023). Information fragmentation concerns items bearing on one task, scattered across stores but reachable (Karger & Jones, 2006), and temporal fragmentation in time geography describes activities performed in pieces (Hubers et al., 2008), not a data archive.

1.2 What the platform documents

Condition 3 is a claim about the platform, so we tested it against the vendor's own documentation. On 15 October 2026 we collected the WhatsApp Help Center (locale en_US, retrieved from India), starting from its navigation and following links inside articles until no new article appeared, plus two known articles neither route reached. Of 624 articles screened, the 105 on backup, restore, transfer, registration, linked devices or export were included, with 16 blog posts. Every quotation was checked by script against the saved text, and those cited here were checked again against same-day Internet Archive snapshots. The index, screening decisions and verified quotations are in the repository.

Table 2 lists every documented path that brings two histories, or two devices, together. None yields the union, and five pages state in nearly the same words that new and old chats cannot be merged.

Four uncertainties remain, and the evidence covers one platform on one date. The documentation describes the device, not what the servers keep. Wording varies by region: our copies give 120 days of inactivity before an account is deleted where the Internet Archive's give 365, and one page states the one-backup limit both per phone number and per account. Pages change, so we cite dated snapshots. And the export pages do not say which time zone a timestamp is written in, which is why CARTA requires a declared zone for each fragment (Section 3.3).

TABLE 2: DOCUMENTED WHATSAPP PATHS AND THEIR EFFECT ON TWO HISTORIES (HELP CENTER, RETRIEVED 15 OCT 2026)

Path	Documented effect	Union?
Restore from cloud backup	"only one backup per phone number, so a new backup permanently replaces" the previous one; restore happens during setup (WhatsApp, 2026e)	No
Chat transfer, same OS	"You can't merge new chats from your new phone and old chats from your old phone" (WhatsApp, 2026a)	No
Move between Android and iPhone	"Whichever set you transfer replaces the other" (WhatsApp, 2026b)	No
Re-registration on another phone	the old phone is logged out; "your chats will remain on the device" (WhatsApp, 2026c)	No (both persist)
Linked or companion device	up to four; the primary sends a copy of recent history at link time; companions are logged out if the primary is unused for more than 14 days; they cannot be primary (WhatsApp, 2026f, 2026g)	No (synchronised from one primary)
Export chat	a text or ZIP file that "can't be re-imported because it's a text file and not a backup file" (WhatsApp, 2026h)	No (it is CARTA's input)

1.3 Reconstruction as a data-quality and governance problem

A pipeline that ingests heterogeneous records, canonicalises them, sets aside what it cannot interpret, deduplicates by a similarity predicate and reports what it did is, in structure, an extract-transform-load pipeline with a data-quality regime attached. Data governance assigns decision rights and accountability for data (Abraham et al., 2019), and fitness for use depends on who encounters the data, data subjects included (Priestley et al., 2023). In enterprise integration the data has an owner accountable for it to somebody else, and quarantine and lineage exist to discharge that accountability.

A person reconstructing their own chat history is the owner, the auditor and the only accountable reader at once, with no second copy against which to check the result. If a reconstruction resolves an unclear record by guessing, or silently merges or drops a message, nothing in its output reveals it, and a damaged archive still looks complete. A quarantine turns that invisible error into a visible list of what the procedure could not settle,

and the report is the audit trail of where each record came from, what was merged and what was set aside. Table 3 maps each CARTA output to the ISO/IEC 25012 data-quality characteristic it serves (ISO/IEC, 2008). Ehrlinger and Wöß (2022) found that quality concepts accepted in research are rarely implemented in tools; Section 2.1 finds the same gap among consumer merge tools.

TABLE 3: CARTA OUTPUTS AND THE DATA-QUALITY CHARACTERISTICS THEY SERVE

CARTA output	ISO/IEC 25012 characteristic	Governance function
Quarantine set	completeness, accuracy	nothing is lost silently; uncertainty is explicit
Typed reason codes	accuracy, consistency	routes each record to a remedy (parser fix or evidence)
Provenance tuple	traceability	each record audits back to its source bytes
Conflict log	consistency	contradictions are reported, not resolved
Deduplication ambiguities	accuracy	uncertain merges stay visible
Declared settings	traceability, compliance	the run can be audited and repeated
Determinism	credibility, recoverability	the same input gives the same archive and report

1.4 Research questions and contributions

The paper asks three questions.

> RQ1. What conditions must hold for a single account's message history to end up distributed across two or more independently preserved archives, each internally complete, that the platform provides no operation to reconcile?
>
> RQ2. Given two or more such archives, what must a reconstruction procedure guarantee for its output to be trustworthy as a record, rather than merely plausible?
>
> RQ3. Where do those guarantees sit relative to what optimistic replication, digital forensics and existing chat-merging tools already provide?

Sections 1.1 and 1.2 answer RQ1, Section 3 answers RQ2 with CARTA (Chat Archive Reconstruction and Temporal Access), and Sections 2 and 4.4 answer RQ3.

The contribution is an output contract for reconstructing personal chat history, owed to a reader who cannot check the result. It has five properties: input that cannot be interpreted is retained as output (a quarantine set with typed reason codes); every record carries per-record provenance; contradictions are reported; determinism is stated; and a structured report accounts for the run. No consumer merge tool found by a systematic search fully provides more than 2 of them, and none combines the first with the second or third (Section 2.1). We claim priority over none of the underlying mechanisms, which data integration and digital forensics have used for years; the contribution is their transfer to a setting where none of the surveyed tools provides them together. A reference implementation, CHAMP-Pipeline, is evaluated against synthetic fragment sets with known ground truth and against existing tools (Section 4).

2.0 Literature review

CARTA sits where five bodies of work meet. Section 2.1 surveys the tools that already merge chat archives. The other four supply the machinery CARTA reuses, and none addresses the reconstruction of independently preserved consumer chat archives.

2.1 Existing merge tools: a systematic survey

The submitted version described seven merge tools found by an informal search. This section replaces that incomplete account with a systematic survey, following Ehrlinger and Wöß (2022), who searched systematically for data-quality tools and applied predefined criteria. Queries, raw responses, screening decisions and coding are in the repository.

Search. On 15 October 2026 we ran 15 queries, such as "whatsapp merge" and "chat history merge", against GitHub's repository search API, which matches names, descriptions and READMEs. Fourteen queries reached the API's cap of 1,000 results, so recall is bounded by the interface. The queries returned 14,054 hits from 10,708 repositories.

Screening. A metadata screen kept a repository that is not a fork, names a messaging platform, and either names a merge operation (rule A) or ranks in the top 50 of any query (rule B). Rule B was added after a known-item check showed that rule A alone discarded two tools from the submitted version whose merge function is described only in their READMEs. With rule B, 400 repositories passed, against 194 under rule A alone. Each was read at README and source level against three inclusion criteria: working code (I1), combining two or more archives of the same account (I2), and a consumer messaging platform (I3). With one level of snowballing from included tools' READMEs, 436 repositories were screened; 60 were included, and 7 single-acquisition forensic tools were kept as comparators. Six of the seven tools in the submitted version are included and the seventh is a comparator. Every included tool was coded at a pinned commit, with file and line evidence for each y or partial code.

Coding. Each tool was coded y, partial or n on the five properties of Section 1.4 under a written codebook (Table 4). Some single properties recur; what is missing is the combination. No tool holds more than 2 of the five properties, or 3 when partial implementations count, and none combines retaining unparseable input with per-record provenance or with conflict reporting.

Sensitivity and scope. Three WhatsApp tools that merge exports from different accounts were excluded under I2; one, wa-rescue (sebnembasak, 2026), approaches the gap from the other side by merging the other party's export into the user's own backup. Including the three gives 63 tools and leaves both maxima unchanged. Settling two borderline answers in a tool's favour raises one tool to 4 of the five, and whether it retains unparseable input could not be determined. The survey shows that none of the tools a person is likely to find on GitHub provides the five properties together, not that no such tool exists.

TABLE 4: SURVEY CODEBOOK (CONDENSED) AND COUNTS OVER 60 TOOLS

Property	Coded y when the tool...	y	y + partial
P1 retains unparseable input	includes input it cannot parse, marked, in its output	1	1
P2 per-record provenance	gives every output record its source and a record-level locator (line, row, source uid or content hash)	3	8
P3 conflict reporting	detects that sources disagree about the same message or item and tells the user	4	5
P4 stated determinism	states that the same inputs give the same output	0	7
P5 structured report	writes a JSON, CSV or table record of what the merge did, or returns one to the user or a client	6	8
Most properties held by any one tool		2	3

Note. Partial is defined per property in the codebook (P1 has no partial code); y + partial counts both, and "not determined" counts as n.

2.2 Data integration, data quality and entity resolution

This is the nearest neighbour to CARTA, and it supplies each mechanism used here. Routing records that fail validation into a separate store under a recorded reason, rather than discarding them, is established practice; Souibgui et al. (2019) survey where quality defects arise along the ETL process. Moreau et al. (2015) set out the rationale for PROV, the W3C provenance interchange model, and Chapman et al. (2024) capture fine-grained provenance through data-science pipelines. The clustering question of Section 3.7 is likewise studied: Baas et al. (2021) show that taking the transitive closure of a similarity relation can decrease precision substantially, and Saeedi et al. (2018) evaluate matching and clustering schemes for multi-source entity resolution, structurally the configuration Synthesis faces.

2.3 Optimistic replication and replicated data types

Parker et al. (1983) formalised the detection of mutual inconsistency between copies, and Coda reintegrates on reconnection and names the conflicts it cannot resolve (Kistler & Satyanarayanan, 1992). Coda attempts reintegration; Condition 3 describes a platform that does not. Conflict-free replicated data types give conditions under which divergent replicas provably converge (Shapiro et al., 2011; Almeida, 2025), but convergence assumes that replicas eventually exchange their updates. TCF's branches exchange nothing while diverging, so no convergence guarantee applies. Retrofitting, as in Yu and Ignat (2020), adds causal metadata before divergence begins.

2.4 Timeline reconstruction and tooling in digital forensics

This is the closest neighbour by problem shape. The field's own systematisation handles multi-source combination informally, and provenance, quarantine, conflict detection, determinism and invariants do not appear in it as formally defined terms (Breitinger et al., 2025). Chabot et al. (2014a) do formalise provenance, as a relation of support linking a footprint to the entity deduced from it; that is an identifier-to-identifier edge, whereas CARTA binds a line range, a file hash and a record hash to each record. Chabot et al. (2014b) define a four-layer pipeline from extraction to an investigator-facing interface, but it types no inter-stage format, filters unusable records out (the opposite of quarantine), and outputs a triple store and a visualisation rather than a report. CARTA's sort key and ordering conflicts use the ordering evidence that line and sequence numbers carry beyond timestamps (Dreier et al., 2024), and its declared zones respond to the finding that timestamp normalisation does not guarantee comparability (Hornung et al., 2026).

In Plaso, the reference open-source super-timeline engine (log2timeline/plaso, 2026), the text-log parsers keep a warning for each line they fail to parse, with at most an 80-character excerpt rather than the line itself, and stop once consecutive failures exceed 20; warnings are left out of the exported timeline. Its duplicate suppression keys on event content and source path and compares only consecutive same-timestamp events, so one event seen in two sources is not collapsed, conflicts go undetected, and suppressed events appear only as a count. Chat-specific forensic work is single-acquisition: Anglano (2014) correlates WhatsApp artefacts within one Android device, and a systematic review of secure instant-messaging forensics has no category for reconciling multiple archives (Onik et al., 2025).

2.5 Personal data loss and recovery

People report attachment to their personal digital data and distress at its potential loss (Sillence et al., 2026). In a three-country survey of 1,423 participants, 46% had experienced at least one data-loss incident, and over half of those who recovered using backups held backups that were outdated or incomplete (Wunder et al., 2025). This supports the motivation without evidencing the phenomenon: we have found no study that describes two independently persistent archives of one account.

3.0 Methodology

The method is specification followed by reference implementation. The specification does not depend on the implementation, which exists to put CARTA in front of something that must execute rather than only be read. Section 4.5 reports what that produced.

3.1 The reconstruction contract

Write the raw fragment exports as $\mathcal{F}$ = {F1, ..., Fn}. CARTA composes three stages (Fig. 2):

$$\mathcal{F} \to \mathcal{R} \to (\mathcal{R}', \mathcal{Q}) \to (\mathcal{U}, \Omega) \quad (1)$$

The stages are Ingestion $\mathcal{F} \to \mathcal{R}$, Normalization $\mathcal{R} \to (\mathcal{R}', \mathcal{Q})$, and Synthesis $(\mathcal{R}', \mathcal{Q}) \to (\mathcal{U}, \Omega)$. Here $\mathcal{R}$ is the set of structured but uninterpreted records, $\mathcal{R}'$ the canonical records, $\mathcal{Q}$ the quarantine set, $\mathcal{U}$ the unified archive and Ω the reconstruction report. The quarantine set appears in the top-level contract by design, so a conforming implementation cannot discard it quietly: Synthesis does not reconcile $\mathcal{Q}$, but it must carry it into Ω. A single device failure gives n = 2; the contract is stated for n ≥ 2, since repeated device failures can accumulate more fragments.

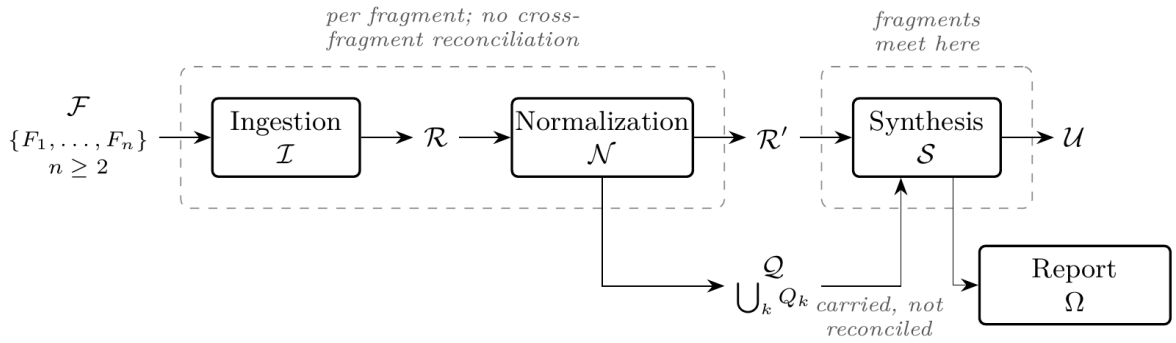


Fig. 2 The CARTA reconstruction contract.

3.2 Ingestion

Ingestion maps each fragment independently to a sequence of structured records. Each record keeps the raw text slice, the raw timestamp string, the raw sender token, a message category and a provenance tuple:

$$\text{prov}(r) = \langle\ \text{source_id}, \text{line_range: } [\ell_{\text{start}}, \ell_{\text{end}}], \text{file_sha256: } h(F_k), \text{record_sha256: } h(\text{bytes}(r))\ \rangle \quad (2)$$

The source identifier says which fragment the record came from: its file name, or an identifier declared for it when two exports share a name. The line range makes the original text re-locatable. The file hash binds the claim to a file state, and the record hash binds the record's own bytes, so a later change elsewhere in the fragment does not invalidate it. Ingestion writes nothing back, filters nothing and reconciles nothing; a stage that both parsed and reconciled would hide whether a discrepancy came from parsing or from merging.

3.3 Normalization

Normalization converts each fragment's records into a shared canonical schema with UTC instants, resolved sender identity and a fixed message-type taxonomy. An export records wall-clock times without an offset, so CARTA requires each fragment's time zone to be declared (a named zone or a fixed offset), records it in the report, and marks an undeclared zone as a default rather than a fact. Fragments may carry different zones, since a spare handset may be set to another zone or travel with its owner.

Each record meets one of three fates. A record that conforms to the canonical grammar unambiguously becomes a canonical record. A record that fails to parse moves to the quarantine set, and so does a record that parses but admits more than one reading. The date string 03/04/24 is the standard case: it is valid under both day-month-year and month-day-year conventions, and nothing inside the record decides between them. Picking the more likely convention would turn an uncertainty into an assertion that later stages, and eventually the reader, could not distinguish from an observation. Daylight-saving time adds a second kind of ambiguity: a fall-back hour names two instants, and a skipped hour names none.

Every quarantined record carries one of six typed reason codes: MALFORMED_TIMESTAMP, UNSUPPORTED_FORMAT, AMBIGUOUS_DATE_FORMAT, AMBIGUOUS_LOCAL_TIME, UNRESOLVED_SENDER and INVALID_ENCODING. The taxonomy separates a record that could not be read from one readable in several ways, because they call for different remedies: a parser fix, or evidence from outside the record, such as a date order established elsewhere in the fragment. Normalization judges each record on its own contents. An implementation may offer fragment-scope inference as a separate, declared step that reports what it inferred and on what evidence; the specification rules out only silent inference. Provenance travels with every record into whichever set it lands in.

3.4 Synthesis

Synthesis is the only stage that sees fragments together. It orders the union of the canonical record sets, groups records that describe the same message into reconciliation clusters, synthesises one canonical record per cluster, and detects observations that cannot be reconciled. Because δ and ε may both be positive, a message sent during an overlap arrives twice, in two separately parsed forms that need not agree byte for byte. Where the fragments are disjoint, every cluster is a singleton and Synthesis reduces to ordered concatenation without special-casing. A conflict is any set of observations that cannot be reconciled deterministically: contradictory sender attribution, orderings that cannot both hold, or disagreeing attachment forms.

3.5 The reconstruction report

The report Ω is a first-class output with these components:

- the provenance map, from every record in the unified archive to the provenance tuples of the observations it was synthesised from;
- the quarantine carryover, holding every quarantined record with its reason code;
- the conflict log;
- the deduplication ambiguities (Section 3.7);
- report-only advisories (Section 3.7);
- statistics;
- the declarations an audit needs to repeat the run: the specification and implementation versions, the runtime and time-zone database, each fragment's hash and declared zone, the normalisation policy and reason-code taxonomy, the evidence for any inference, and every Synthesis parameter.

The specification defines no confidence score, which would compress into one number exactly the uncertainty that quarantine and the conflict log keep explicit. A reconstruction that silently omits what it could not handle looks identical, to its reader, to one that had nothing to omit; carrying the quarantine set into the report makes the two cases distinguishable.

3.6 Invariants and conformance

Three invariants constrain any conforming implementation. They are not correctness properties, since none asserts that the unified archive is the right reconstruction. They are reporting obligations: things a reconstruction must be able to tell its reader about itself.

- Invariant 1, Zero unaccounted loss. Every ingested record either contributes provenance to a record in the unified archive or appears in the quarantine carryover.
- Invariant 2, Strict provenance traceability. Every record in the unified archive and in the quarantine carries at least one provenance entry, and every entry references an ingested fragment by identifier and file hash.
- Invariant 3, Determinism. For a fixed set of fragments and a fixed runtime, two runs yield the same archive and the same report, whatever the order in which the fragments are supplied.

These are requirements on implementations, not theorems. Because Section 3.7 leaves the clustering procedure and the field precedence open, two conforming implementations may return different archives for the same fragments. Under a correctness framing that would be fatal; under a reporting framing it is the design. An implementation conforms when it satisfies Invariants 1 to 3 and declares, in the report, every parameter of Section 3.5.

3.7 Ordering, similarity and the synthesis procedure

Synthesis uses two timestamp tolerances. The deduplication window $\Delta t_{\text{granularity}}$ governs whether two records may describe the same message. The conflict bucket $\Delta t_{\text{conflict}}$ decides which pairs of different messages are close enough for a contradiction to be worth reporting. A conforming implementation must satisfy

$$\Delta t_{\text{granularity}} \leq \Delta t_{\text{conflict}} \tag{3}$$

otherwise the same pair could be treated as one message and as two conflicting ones at once.

Records are sorted by canonical instant, then by fragment hash, then by source sequence; because the hash and not the file name breaks ties, renaming a file cannot change the result. Two records are similar when they agree on sender, body and attachment and their instants fall within the deduplication window:

$$r_{\text{a}} \sim_{\text{dedup}} r_{\text{b}} \Leftrightarrow |t(r_{\text{a}}) - t(r_{\text{b}})| \leq \Delta t_{\text{granularity}} \wedge s(r_{\text{a}}) = s(r_{\text{b}}) \wedge b(r_{\text{a}}) = b(r_{\text{b}}) \wedge a(r_{\text{a}}) \cong a(r_{\text{b}}) \tag{4}$$

where two attachments are congruent (≅) when both are absent, their file names are equal, or their stems are equal when one rendering has lost its extension.

This is a similarity relation, not an equivalence, since proximity is not transitive. CARTA 1.1 constrains clustering by a property of the input: a fragment is internally complete, so two similar records from the same fragment are two messages, not one seen twice, and a cluster holds at most one observation per fragment. Within a run of mutually similar observations, the k-th observation from each fragment, in source order, joins the k-th cluster. Where the fragments hold different numbers of copies, the procedure records a deduplication ambiguity naming the run instead of resolving it silently. Other clustering procedures remain admissible if declared; Section 4.3 compares them.

Each cluster yields one canonical record under a declared precedence rule; where observations disagree on a field the criterion does not constrain, a conflict descriptor reports the choice. The canonical record's provenance is the union of its observations' provenance, which preserves Invariant 2 across deduplication. Algorithm 1 gives the whole procedure.

Require: Normalized fragment record sets $\mathcal{R}' = \{R'_1, R'_2, \ldots, R'_n\}$, quarantine sets $\mathcal{Q} = \{Q_1, \ldots, Q_n\}$, deduplication window $\Delta t_{\text{granularity}}$, conflict bucket $\Delta t_{\text{conflict}}$ with $\Delta t_{\text{granularity}} \leq \Delta t_{\text{conflict}}$
Ensure: Unified archive $\mathcal{U}$, reconstruction report Ω

$\mathcal{U} \leftarrow \emptyset$, $\mathcal{C}_{\text{conflicts}} \leftarrow \emptyset$, $\mathcal{P}_{\text{map}} \leftarrow \emptyset$
$\mathcal{R}'_{\text{all}} \leftarrow \bigcup_{k=1}^{n} R'_k$
Sort $\mathcal{R}'_{\text{all}}$ by (t_{utc}, fragment hash, source sequence)
$(\mathcal{B}_{\text{clusters}}, \mathcal{A}_{\text{dedup}}) \leftarrow \textsc{GroupOneToOne}(\mathcal{R}'_{\text{all}}, \sim_{\text{dedup}}, \Delta t_{\text{granularity}})$ ▷ ≤ 1 observation per fragment
for each cluster $C \in \mathcal{B}_{\text{clusters}}$ **do**
    $u \leftarrow \textsc{SynthesizeCanonicalMessage}(C)$
    $u.\text{provenance} \leftarrow \bigcup_{r' \in C} r'.\text{provenance}$
    $\mathcal{P}_{\text{map}}[u.\text{id}] \leftarrow u.\text{provenance}$
    if $\textsc{HasUnresolvedConflict}(C)$ **then**
        $\mathcal{C}_{\text{conflicts}}.\text{append}(\textsc{CreateConflictDescriptor}(C))$
    end if
    $\mathcal{U}.\text{append}(u)$
end for
$\mathcal{C}_{\text{conflicts}}.\text{extend}(\textsc{DetectCrossClusterConflicts}(\mathcal{U}, \Delta t_{\text{conflict}}))$
$\mathcal{A}_{\text{zone}} \leftarrow \textsc{ZoneOffsetAdvisories}(R'_1, \ldots, R'_n)$ ▷ report-only; $\mathcal{U}$ unchanged
$\Omega \leftarrow \langle \text{provenance_map} : \mathcal{P}_{\text{map}},\ \text{quarantine} : \bigcup \mathcal{Q},\ \text{conflicts} : \mathcal{C}_{\text{conflicts}},$
$\text{dedup_ambiguities} : \mathcal{A}_{\text{dedup}},\ \text{advisories} : \mathcal{A}_{\text{zone}},\ \text{stats} : \textsc{ComputeStats}(\mathcal{U}, \mathcal{R}'_{\text{all}}, \mathcal{Q}),$
$\text{declarations} : \textsc{DeclaredSettings}() \rangle$
return $(\mathcal{U}, \Omega)$

Algorithm 1. The CARTA Synthesis and report construction procedure.

Across clusters, two records may describe different messages and still contradict each other:

$$\text{Conflict}(u_i, u_j) \Leftrightarrow \lfloor t(u_i) \rfloor = \lfloor t(u_j) \rfloor \wedge u_i \nsim_{\text{dedup}} u_j$$
$$\wedge \left([b(u_i) = b(u_j) \wedge s(u_i) \neq s(u_j)] \vee \text{seq}(u_i, u_j) \text{ is unsatisfiable} \right) \quad (5)$$

The floor brackets truncate an instant to its conflict bucket. The sender clause applies only to records with a sender and identical bodies, so ordinary same-minute conversation is not reported as contradiction, and the sequence term is unsatisfiable when the fragments order the pair oppositely. The second conjunct keeps this pass disjoint from clustering: a pair already grouped is one message seen twice. Flagged records are neither dropped nor rewritten: Equation (5) detects and reports, but resolves nothing.

Finally, a wrongly declared zone shifts every instant of a fragment by the same amount, so the overlap does not deduplicate. CARTA 1.1 adds a report-only **zone-offset advisory**. It counts one-to-one matches on (sender, body, attachment stem) between two fragments at each candidate offset, a multiple of 15 minutes within ±26 hours, and reports an offset only when the support is at least 3 across at least 3 distinct (sender, body, stem) keys and exceeds both the support at zero and the best support at any non-15-minute control offset. The advisory never changes the archive, and its rule was committed before it was tested (Section 4.1).

3.8 Reference implementation

CHAMP-Pipeline is the WhatsApp-specific reference implementation of CARTA 1.1 (Sapariya, 2026). It implements the three stages and the declared fragment-scope date step in 1,481 lines of JavaScript across four modules, with 1,258 lines of self-checks run as five checks. It has no runtime dependencies beyond Node.js, targets one export format, and has been run on synthetic input only. The absence of dependencies serves the specification: a

reconstruction a reader cannot re-run in ten years is not auditable, and every dependency is a surface through which behaviour can change underneath Invariant 3.

3.9 Synthetic data and evaluation design

WhatsApp text exports vary in date order (with no locale marker), 12-hour or 24-hour clock, multi-line and empty bodies, lines with no sender, attachment-reference forms, and system lines dated far earlier than the surrounding conversation. Five fixture files span 19 such input-form categories, recorded in a manifest. All data are generated: real personal messages cannot be used, and synthetic data with known ground truth is the established remedy in forensic tool testing (Du et al., 2021).

The generator writes a ground-truth conversation and two to four fragments of it, each with its own date order, clock and time zone, and records which line of which fragment holds each message. Eight knobs inject the situations Sections 3.3 to 3.7 are written for: (a) identical repeated messages; (b) one body attributed to two senders; (c) two messages ordered oppositely by two fragments; (d) attachment forms that differ between fragments; (e) a body that differs between copies; (f) a fragment rendered in one zone and declared in another; (g) a one-minute clock disagreement between copies; and (h) a date that is ambiguous in one zone but not in another. For each knob the generator states the outcome a conforming run should produce, computed from what it wrote rather than by parsing the files. Knobs (e) and (g) are constructions, not observed behaviour: (e) appends a marker to a body and is not WhatsApp's rendering of an edited message, and (g) is hypothetical: we have no evidence that two exports of one message ever differ in the minute.

Runs use $\Delta t_{\text{granularity}} = 0$, on the assumption that both export formats record time to the minute and that the copies of one message carry the same minute, with $\Delta t_{\text{conflict}} = 60$ s and one-to-one clustering. Where $\Delta t_{\text{granularity}}$ is varied, $\Delta t_{\text{conflict}}$ is the larger of 60 s and $\Delta t_{\text{granularity}}$, so Equation (3) holds.

E1 sweeps 100 seeds, $\delta \in \{0, 1, 7, 14, 30\}$ days, $\varepsilon \in \{0, 1, 7\}$ days and three sizes (170, 1,000 and 10,000 messages), each with and without the declared date inference. E2 draws random configurations of two to four fragments with mixed zones and knobs switched on at random. E3 tests feasibility at 100,000 and 1,000,000 messages. E4 places each knob alone in a scenario of about 120 messages with $\delta = \varepsilon = 7$ days, or 0 where the fragments are disjoint; E5 varies the deduplication settings (Section 4.3), E7 produces one small report for inspection, and E8 tests the zone-offset advisory with a rule and grid committed before any run. A benchmark, E6, runs the included tools that can read the same input (Section 4.4).

One command regenerates the results of E1 to E5, E7 and E8; the benchmark and the survey have their own scripts. On a fresh clone of a pre-release commit it reproduced 13 deterministic result files byte for byte; the scale run (E3) was not repeated there. Between that commit and release v0.2.0 only the survey coding and documentation changed; because the public history is squashed, that commit and the commit freezing the advisory rule are available on request. Byte identity is claimed within one runtime, whose Node.js, ICU and time-zone database versions every report records.

4.0 Results and Discussions

4.1 Reconstruction against known ground truth

TABLE 5: RECONSTRUCTION RESULTS

Run	Records in	Quarantined	Collapsed	Archive	Conflicts
Committed pair, with inference	192	0	20	172	0
Committed pair, strict	192	81	8	103	0
Disjoint pair ($\delta = \varepsilon = 0$), with inference	172	0	0	172	0
E1 sweep: 4,500 configurations	exact ground truth in 4,500/4,500; deterministic and order-independent in 4,500/4,500				
E2 properties: 5,000 random configurations (n = 2-4)	Invariants 1-3, order independence and expected outcome in 5,000/5,000				
E3 scale: 100k / 1M messages	exact; median 3.3 s / 27.2 s; peak 347 / 1,915 MiB				

Table 5 begins with the committed fragment pair (seed 1, δ = 14 days, ε = 7 days). With the declared date inference, CHAMP-Pipeline collapses the 20 overlap messages and returns 170 messages and 2 system lines, which is the ground truth. In strict mode it quarantines 81 records with ambiguous dates, collapses the 8 overlap messages dated unambiguously on both sides, and returns 103 records. The strict archive is smaller but not wrong: every record it lacks is in the quarantine set with its reason and provenance. The disjoint pair is concatenated in order.

In E1 the inference archive equals the ground truth in every configuration, with observations grouped exactly as the ground truth groups them. Strict mode never reproduces the ground truth, by design, since every configuration contains ambiguous dates, yet the records that reach Synthesis are always grouped correctly: strict mode loses records to quarantine, never to a wrong merge. Both modes are deterministic and order-independent throughout, in E1 and E2. E3 ran on a 2015 laptop processor, with identical output across three repetitions.

E8 tests the zone advisory. Over 600 runs with correctly declared zones it raised an advisory in 2, both from one seed on disjoint fragments of 50,000 messages. Over 2,800 runs in which one fragment's zone was misdeclared, at 46 offsets between −26 and +26 hours, it named the true offset every time and never named another.

These results establish less than their uniformity suggests. Ground truth, expected outcomes and implementation come from one project, and the expected outcome is a second encoding of the specification, not an independent oracle. Agreement shows that the implementation does what CARTA says, not that CARTA is right about real exports. The advisory works only when fragments share enough history: in E2 it stayed silent for 72 of 1,542 misdeclared zones, 51 of them because the fragments shared fewer than three messages, as its rule requires; in the other 21, quarantined records were at least as many as shared messages, a knob altered or duplicated the shared copies, or three shared messages were spread over several fragments. E3 shows feasibility on one machine and one configuration; it is not a performance measurement.

4.2 Conflicts, deduplication errors and the report

TABLE 6: SCENARIO OUTCOMES (E4)

Scenario	Injected	Archive vs ground truth	What Ω reports
(a) one-to-one	identical repeats, inside and outside the overlap	exact	nothing needed
(a) complete linkage (CARTA 1.0 implementation)	same	3 messages lost by merging	nothing
(a) boundary	3 copies straddling an overlap boundary	correct count, one copy paired with the wrong one	deduplication ambiguity
(a) boundary, disjoint	1 copy in each fragment, same minute	1 message lost (silent merge)	nothing
(b)	one body, two senders	exact	sender disagreement
(c)	two messages ordered oppositely	exact	ordering unsatisfiable
(d)	attachment forms differ	exact	2 attachment-form conflicts
(e)	edited-like body variant (synthetic)	1 extra record	nothing
(f)	fragment rendered in +05:30, undeclared	shared messages duplicated	zone-offset advisory
(g)	±1 min jitter (hypothetical)	jittered copies duplicated	nothing
(h)	ambiguous in one zone only	exact	mixed dating 1

TABLE 7: EXCERPT OF THE REPORT FROM THE REPRESENTATIVE RUN (E7)

Component	Entry
Quarantine	1 record, AMBIGUOUS_DATE_FORMAT, "01/02/24, 12:32 am", provenance F1 line 9 with file and record hashes
Conflicts	sender disagreement (u000003, u000004); attachment-form disagreement (double-filename vs standard)
Deduplication ambiguity	2 copies in F1 vs 1 in F2, resolved k-th with k-th
Declarations	CARTA 1.1; zones +00:00 (default); $\Delta t_{\text{granularity}}$ 0; $\Delta t_{\text{conflict}}$ 60 s; one-to-one; tie-break by fragment hash

Table 6 places one injected situation in each run, and its rows fall into three groups. In the first, the archive is exact: identical repeats survive as separate records under one-to-one clustering (row a), and in rows b, c, d and h the report adds what the archive alone cannot show, a sender disagreement, an ordering conflict, an attachment-form disagreement or a mixed-dating count. Under the complete linkage of the CARTA 1.0 implementation, the input of row a loses three messages and the report is silent (Section 4.3).

In the second group the archive is wrong and the report says so. With three identical copies at an overlap boundary the record count is right, but k-th-with-k-th pairing matches one copy with the wrong partner, and the deduplication ambiguity tells the reader that the pairing was a choice. An undeclared +05:30 fragment (f) shifts every instant, so the overlap

does not deduplicate and the archive gains eight duplicates; the advisory reports the suspected offset and leaves the archive unchanged.

The third group is what the report cannot see: identical messages either side of a disjoint switchover, which are indistinguishable from one message seen twice; a body that differs between copies (e); and copies a minute apart (g). Each is a limit of what the input can tell any procedure (Section 4.6).

Table 7 shows what a reader receives from a small strict-mode run (E7). The date 01/02/24 is valid under both conventions, so the record is set aside with its reason, its line and hashes binding it to the file and to its own bytes. Each conflict and the ambiguity name the observations involved, and the declarations mark both zones as defaults, to be read as assumptions.

A record is mixed when one observation in its cluster is dated directly and another by inference. The committed pair has none: the day-month ambiguity is symmetric, so both copies of a message are ambiguous or neither is, which is also why the twelve overlap messages strict mode cannot collapse are quarantined on both sides. A mixed record needs fragments in different zones, so that a message falls on different calendar days; row h constructs that case, and the report counts one.

4.3 Deduplication settings and the risk of merging distinct messages

Deduplication is itself a resolution. Deciding that two observations are one message is a judgement made on the reader's behalf, and the argument against silent resolution applies to it as much as to dates. E5 measures how it depends on its settings over 2,160 runs: four deduplication windows, three clustering procedures, three densities of repeated messages, and a hypothetical one-minute jitter on a tenth of overlapping copies.

Without jitter, one-to-one clustering reaches a pairwise F1 of 1.0 in every cell, at every window up to 300 s. Complete and single linkage, which allow two similar records from one fragment into one cluster, do not. With many repeats their F1 falls to 0.83 at $\Delta t_{\text{granularity}} = 0$ and to 0.66 at 300 s. Recall stays at 1 in both cases, so the whole loss consists of distinct messages merged. Jitter moves the error the other way. At $\Delta t_{\text{granularity}} = 0$ the jittered copies are not recognised, and recall falls to about 0.90; a 60 s window recovers almost all of them under one-to-one clustering. The two errors pull the window in opposite directions, which is why CARTA requires the window to be declared rather than fixing its value.

The same-fragment merge needs a direct account. CARTA 1.0 permitted it, and the CHAMP-Pipeline release that implemented 1.0 did it: two identical messages sent by one person in one minute, both in one fragment, became one record. The report did not reveal it, because by its own rules nothing had gone wrong; we found it by checking the clustering rule against the objection that deduplication is a resolution. CARTA 1.1 makes the remedy structural: a cluster holds at most one observation from each fragment, so two similar records in one fragment are two messages by construction. Every remaining collapse is recorded in the provenance map, so a reader can audit any merge and undo it from the unmodified inputs.

4.4 Comparison with existing tools

TABLE 8: BENCHMARK ON THE COMMITTED PAIR (EVERY COLUMN SHOWN WAS IDENTICAL UNDER BOTH LOAD ORDERS)

Tool	Shown	Exact messages	Misdated	Overlap duplicates left (of 20)	Lost	Dropped without notice
keepsake	172	170	0	0	0	0
The-Omar-Files	183	128	53	11	0	0
tarikab7 (HTML app)	181	159	22	11	0	2
krishnarajt	190	190	0	20	0	2
tarikab7 (Next.js parser)	134	133	0	0	37	58
CHAMP-Pipeline	172	170	0	0	0	0

Would existing tools have done as well on the same input, and what would they have told the reader? We ran the included tools that read WhatsApp text exports: keepsake (sokie, 2026), The-Omar-Files (Akif-b-Atif, 2026), krishnarajt (2021) and tarikab7 (2026), whose two parsers make five implementations; one further tool is discussed below. Each ran with its code unmodified at a pinned commit and was scored against a ground truth parsed independently of CHAMP-Pipeline.

On the committed pair, keepsake matches CHAMP-Pipeline on every column of Table 8. The other tools misdate records, leave overlap duplicates or lose messages; the committed pair mixes date orders, so misdating partly reflects format support. None marks a misdated record, and only the HTML version of tarikab7 flags some duplicates. Over five seeds of scenario pairs in each tool's best-supported format, keepsake and CHAMP-Pipeline are equally accurate on the baseline, repeated-message and sender scenarios. krishnarajt silently merges five genuinely repeated messages, and both tarikab7 implementations merge five but keep the extra copies flagged as duplicates; the HTML version also leaves one unflagged duplicate per run. On easy input, accuracy does not separate CARTA from the best existing tool.

The undeclared-zone scenario (f) shows what does. With one fragment rendered in +05:30 and no zone declared, every tool, CHAMP-Pipeline included, makes the same errors: 99 overlap duplicates left and 650 records misdated over the five seeds. The errors are the same, but the outputs are not. CHAMP-Pipeline's report carries one zone-offset advisory per run, naming the two fragments and the suspected offset, which our scoring counts as a notice for every affected record. No other tool's output gives any sign that something went wrong; keepsake retries its matching at a shift of ±60 minutes, which cannot reach +05:30.

We did not run chat-history-manager (frozenspider, 2026): building it requires a Rust toolchain and build dependencies that our environment lacked, and its text loader accepts only two-person chats. Its code stops a merge when it detects a time shift between datasets, a refusal rather than a report, but the check relies on source identifiers that its WhatsApp text loader does not set, so it would not have fired on this input.

4.5 Defects exposed by implementation (CARTA 1.0 → 1.1)

Writing and testing a conforming implementation exposed seven defects: five in the specification (items 1, 2, 5, 6 and 7) and two in the implementation (items 3 and 4). They are reported as results because they show that reporting obligations can be tested by implementation in a way prose cannot.

1. A UTC instant with no offset to convert by. CARTA 1.0 required UTC instants, but an export carries no offset. CARTA 1.1 requires a declared zone per fragment, recorded in the report (Section 3.3).
2. Equation (5) reported ordinary conversation as contradiction. In 1.0 any two same-bucket records with different senders conflicted, flooding the log for a group chat; 1.1 requires identical bodies.
3. One export variant defeated deduplication. The doubled-filename attachment form adds a continuation line to the body, so copies failed to match. Normalization now canonicalises it.
4. An unreachable branch. A record whose established date order yields an impossible date cannot arise, because a record is ambiguous only when both leading fields lie in 1 to 12. The branch is now a guard, which confirms that the malformed and ambiguous reason codes partition this input format.
5. Two identical messages in one fragment were merged. CARTA 1.0 left clustering open, and the complete linkage of its implementation merged such messages without a report (Section 4.3). CARTA 1.1 allows at most one observation per fragment in a cluster.
6. A second attachment variant defeated deduplication. One anticipated export form writes an attachment's name with a trailing period in place of its extension, so the message was kept twice, with no conflict reported. Normalization now records each attachment's stem, and Equation (4) treats matching stems as congruent.
7. A declared zone did not make local time unique. Under daylight-saving time a fall-back hour occurs twice and a spring-forward hour never. CARTA 1.1 adds a sixth reason code, AMBIGUOUS_LOCAL_TIME, for the first case and treats the second as MALFORMED_TIMESTAMP.

4.6 Limitations

No real archive has been reconstructed. The ground truth is a construction of the project that wrote the implementation. That tests whether the specification is implementable and self-consistent, not how real archives behave; the 19 input-form categories record what the specification anticipates, not what WhatsApp has been observed to emit.

Fragments that are not internally complete lose an interpretive guarantee. A user may clear a conversation, or an export may come from an application that pruned old media. The mechanical guarantees survive, since Invariants 1 and 2 concern the records the pipeline saw, but the archive cannot distinguish a quiet week from a cleared conversation, and export formats do not record what they omitted.

Residual errors the report does not flag. Beyond the third group of Section 4.2, records ambiguous under daylight-saving time are quarantined and never rescued, and a zone change within one fragment is not handled; since the export documentation does not say

which zone it renders, one zone per fragment is an untested assumption. The advisory can raise false alarms on large disjoint fragments (Section 4.1).

The survey has a single coder. Recall is bounded by the search interface, and the codebook was formalised from coded examples after the first pass. Evidence extraction, initial coding and the verification pre-answers were AI-assisted; the author reviewed them and made the inclusion and coding decisions. Coding was therefore neither blind nor independent.

Generalisation is untested, and no forensic or legal standing is claimed. Normalization, Synthesis, the invariants and the report are defined over record types, but Ingestion is platform-specific, and a second platform may need reason codes or conflict kinds this taxonomy lacks. The report's resemblance to chain-of-custody practice is architectural, not a claim of admissibility.

5.0 Conclusion

A history split by a phone repair is not lost in the usual sense. Both halves survive, each intact on the device that holds it, and the vendor's documentation confirms that no operation joins them. We have called this a loss by non-union and defined it as Temporal Communication Fragmentation, a state fixed by three jointly necessary and sufficient conditions and distinct from replication, from multi-device use and from the fragmentation studied in personal information management.

The remedy we propose is a contract, not an algorithm. CARTA 1.1 does not promise the right archive. It promises that the reader will be told what was done: records that cannot be read unambiguously are kept with a reason, every record traces to its bytes, contradictions and uncertain merges are reported, the run is deterministic, and its settings are declared. Of 60 chat-merging tools found by a systematic search of GitHub, none fully provides more than 2 of these five properties. CARTA specifies all five, and CHAMP-Pipeline shows that they can be met together on synthetic input with known ground truth. Whether real exports behave as the generator assumes is still untested.

The clearest result is one in which CARTA does no better than anyone else. When a fragment's time zone is left undeclared, every tool we ran, CHAMP-Pipeline included, returns an archive with the same duplicates and the same misdated records. Only one of those archives comes with a report that says so. When the data owner is the only person who can check the result, silent resolution is the failure and visible uncertainty is the safeguard.

Declarations

Funding. This research received no external funding.

Conflicts of interest. None, including with WhatsApp, Meta Platforms and the maintainers of the surveyed tools.

Ethics and data. No human participants were involved. The author opened two of his own WhatsApp chat exports locally, solely to confirm the export format. Their content was not copied into the corpus, the repository or this paper, was not processed by any software described here, and was not provided to any AI system. All experimental data are synthetic.

Data availability. Code, synthetic corpus, experiment outputs, survey data and the platform-evidence index are archived as release v0.2.0 (https://doi.org/10.5281/zenodo.23526491; all versions: https://doi.org/10.5281/zenodo.22260521).

Prior publication. This manuscript has not been published or submitted elsewhere. Earlier drafts in the public repository have not been peer-reviewed or posted to a preprint server.

Declaration of AI use. No AI tool is credited as an author.

- Language correction and editing. An AI assistant (Claude, Anthropic) corrected grammar, punctuation and style in author-drafted text; the author checked and approved all wording, claims and citations.
- Programming assistance. An AI coding assistant (Claude Code, Anthropic) assisted in writing, refining and debugging the reference implementation, synthetic-data generator and experiment scripts to the author's specifications. The author specified each change and reviewed every commit, test and output. It also typeset Algorithm 1 and condensed the survey codebook into Table 4; the author reviewed both.
- Research assistance. AI assistants searched for literature and software, archived public documentation, and checked bibliographic records against Crossref. In the tool survey they located sources, extracted candidate evidence, produced first-pass codes and draft verification answers, and drafted the codebook from the author's coded examples. These were candidates, not findings: the author checked each against the source, made all inclusion and coding decisions, assigned the final codes, and approved the codebook. Coding was therefore neither blind nor independent.

The research questions, the framework and its design decisions, and the analysis and interpretation of all results are the author's. AI outputs were adopted only after the author's verification. No personal or confidential data were provided to any AI system. The author takes full responsibility for the content.

References

[1]Abraham, R., Schneider, J., & vom Brocke, J. (2019). Data governance: A conceptual framework, structured review, and research agenda. International Journal of Information Management, 49, 424–438. https://doi.org/10.1016/j.ijinfomgt.2019.07.008

[2] Akif-b-Atif. (2026). The-Omar-Files [Computer software, commit 27faf12]. GitHub. https://github.com/Akif-b-Atif/The-Omar-Files

[3] Almeida, P. S. (2025). Approaches to conflict-free replicated data types. ACM Computing Surveys, 57(2), 1–36. https://doi.org/10.1145/3695249

[4] Anglano, C. (2014). Forensic analysis of WhatsApp Messenger on Android smartphones. Digital Investigation, 11(3), 201–213. https://doi.org/10.1016/j.diin.2014.04.003

[5] Baas, J., Dastani, M., & Feelders, A. (2021). Exploiting transitivity constraints for entity matching in knowledge graphs (arXiv:2104.12589). arXiv. https://doi.org/10.48550/arXiv.2104.12589

[6] Breitinger, F., Studiawan, H., & Hargreaves, C. (2025). SoK: Timeline based event reconstruction for digital forensics: Terminology, methodology, and current challenges. Forensic Science International: Digital Investigation, 53, 301932. https://doi.org/10.1016/j.fsidi.2025.301932

[7] Chabot, Y., Bertaux, A., Nicolle, C., & Kechadi, M.-T. (2014a). A complete formalized knowledge representation model for advanced digital forensics timeline analysis. Digital Investigation, 11(Suppl. 2), S95–S105. https://doi.org/10.1016/j.diin.2014.05.009

[8] Chabot, Y., Bertaux, A., Nicolle, C., & Kechadi, M.-T. (2014b). Automatic timeline construction and analysis for computer forensics purposes. In 2014 IEEE Joint Intelligence and Security Informatics Conference (pp. 276–279). IEEE. https://doi.org/10.1109/JISIC.2014.54

[9] Chapman, A., Lauro, L., Missier, P., & Torlone, R. (2024). Supporting better insights of data science pipelines with fine-grained provenance. ACM Transactions on Database Systems, 49(2), 1–42. https://doi.org/10.1145/3644385

[10] Dearman, D., & Pierce, J. S. (2008). "It's on my other computer!": Computing with multiple devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 767–776). ACM. https://doi.org/10.1145/1357054.1357177

[11] Dreier, L. M., Vanini, C., Hargreaves, C. J., Breitinger, F., & Freiling, F. (2024). Beyond timestamps: Integrating implicit timing information into digital forensic timelines. Forensic Science International: Digital Investigation, 49, 301755. https://doi.org/10.1016/j.fsidi.2024.301755

[12] Du, X., Hargreaves, C., Sheppard, J., & Scanlon, M. (2021). TraceGen: User activity emulation for digital forensic test image generation. Forensic Science International: Digital Investigation, 38, 301133. https://doi.org/10.1016/j.fsidi.2021.301133

[13] Ehrlinger, L., & Wöß, W. (2022). A survey of data quality measurement and monitoring tools. Frontiers in Big Data, 5, 850611. https://doi.org/10.3389/fdata.2022.850611

[14] frozenspider. (2026). chat-history-manager [Computer software, commit d1f6f88]. GitHub. https://github.com/frozenspider/chat-history-manager

[15] Hornung, L., Bakker, C., Timbermont, B., van Dijk, R., van Beek, H., & Alechina, N. (2026). A formal model for forensically sound timestamp normalization. In 33rd International Symposium on Temporal Representation and Reasoning (TIME 2026) (OASIcs Vol. 146, pp. 10:1–10:21). Schloss Dagstuhl. https://doi.org/10.4230/OASIcs.TIME.2026.10

[16] Hubers, C., Schwanen, T., & Dijst, M. (2008). ICT and temporal fragmentation of activities: An analytical framework and initial empirical findings. Tijdschrift voor Economische en Sociale Geografie, 99(5), 528–546. https://doi.org/10.1111/j.1467-9663.2008.00490.x

[17] ISO/IEC. (2008). ISO/IEC 25012:2008 Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model. ISO. https://www.iso.org/standard/35736.html

[18] Karger, D. R., & Jones, W. (2006). Data unification in personal information management. Communications of the ACM, 49(1), 77–82. https://doi.org/10.1145/1107458.1107496

[19] Kistler, J. J., & Satyanarayanan, M. (1992). Disconnected operation in the Coda File System. ACM Transactions on Computer Systems, 10(1), 3–25. https://doi.org/10.1145/146941.146942

[20] krishnarajt. (2021). Whatsapp-Chat-File-Merger [Computer software, commit 138add0]. GitHub. https://github.com/krishnarajt/Whatsapp-Chat-File-Merger

[21] log2timeline/plaso. (2026). Plaso [Computer software, commit 3733f4d]. GitHub. https://github.com/log2timeline/plaso

[22] Moreau, L., Groth, P., Cheney, J., Lebo, T., & Miles, S. (2015). The rationale of PROV. Journal of Web Semantics, 35, 235–257. https://doi.org/10.1016/j.websem.2015.04.001

[23] Onik, A. R., Brown, J., Walker, C., & Baggili, I. (2025). A systematic literature review of secure instant messaging applications from a digital forensics perspective. ACM Computing Surveys, 57(9), 1–36. https://doi.org/10.1145/3727641

[24] Parker, D. S., Popek, G. J., Rudisin, G., Stoughton, A., Walker, B. J., Walton, E., Chow, J. M., Edwards, D., Kiser, S., & Kline, C. (1983). Detection of mutual inconsistency in distributed systems. IEEE Transactions on Software Engineering, SE-9(3), 240–247. https://doi.org/10.1109/TSE.1983.236733

[25] Priestley, M., O'Donnell, F., & Simperl, E. (2023). A survey of data quality requirements that matter in ML development pipelines. Journal of Data and Information Quality, 15(2), 1–39. https://doi.org/10.1145/3592616

[26] Saeedi, A., Nentwig, M., Peukert, E., & Rahm, E. (2018). Scalable matching and clustering of entities with FAMER. Complex Systems Informatics and Modeling Quarterly, 16, 61–83. https://doi.org/10.7250/csimq.2018-16.04

[27] Saito, Y., & Shapiro, M. (2005). Optimistic replication. ACM Computing Surveys, 37(1), 42–81. https://doi.org/10.1145/1057977.1057980

[28] Sapariya, M. (2026). CHAMP-Pipeline: Chat History Archival & Merge Pipeline (Specification and Reference Implementation of the CARTA Framework) (Version 0.2.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.23526491

[29] Sebnembasak. (2026). wa-rescue [Computer software, commit c6dc551]. GitHub. https://github.com/sebnembasak/wa-rescue

[30] Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). A comprehensive study of convergent and commutative replicated data types (Research Report RR-7506). INRIA. https://inria.hal.science/inria-00555588

[31] Sillence, E., Dawson, J. A., Brown, R. D., McKellar, K., & Neave, N. (2026). Digital hoarding and personal use digital data. Human–Computer Interaction, 41(1), 6–25. https://doi.org/10.1080/07370024.2023.2293001

[32] Sokie. (2026). keepsake [Computer software, commit 89cdfb6]. GitHub. https://github.com/sokie/keepsake

[33] Souibgui, M., Atigui, F., Zammali, S., Cherfi, S., & Ben Yahia, S. (2019). Data quality in ETL process: A preliminary study. Procedia Computer Science, 159, 676–687. https://doi.org/10.1016/j.procs.2019.09.223

[34] Tarikab7. (2026). whatsapp-chat-merger [Computer software, commit 403c98e]. GitHub. https://github.com/tarikab7/whatsapp-chat-merger

[35] Tseng, F.-C., Chiou, Z.-Y., Chuang, H.-H., Su, L.-T., Lin, Y.-H., Lin, Y.-R., Lee, Y.-C., Wang, P.-J., Chen, U.-D., & Chang, Y.-J. (2023). Multiple device users' actual and ideal cross-device usage for multi-stage notification-interactions: An ESM study addressing the usage gap and impacts of device context. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1–15). ACM. https://doi.org/10.1145/3544548.3580731

[36] WhatsApp. (2026a). Can't transfer chat history [Help Center article 1039228717453239]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015115509/https://faq.whatsapp.com/1039228717453239/?locale=en_US&cms_platform=android

[37] WhatsApp. (2026b). Can't move chats from an Android device to an iPhone [Help Center article 1195710084347483]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015131617/https://faq.whatsapp.com/1195710084347483/?locale=en_US&cms_platform=android

[38] WhatsApp. (2026c). How to delete chats from your old phone [Help Center article 1233566054551746]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015132100/https://faq.whatsapp.com/1233566054551746/?locale=en_US&cms_platform=android

[39] WhatsApp. (2026d). About inactive account deletion [Help Center article 828406668498455]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015153538/https://faq.whatsapp.com/828406668498455/?locale=en_US&cms_platform=android

[40] WhatsApp. (2026e). How to restore your chat history [Help Center article 618575946635920]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015144935/https://faq.whatsapp.com/618575946635920/?locale=en_US&cms_platform=android

[41] WhatsApp. (2026f). About linked devices [Help Center article 378279804439436]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015141959/https://faq.whatsapp.com/378279804439436/?locale=en_US&cms_platform=android

[42] WhatsApp. (2026g). About message history on linked devices [Help Center article 653480766448040]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015145726/https://faq.whatsapp.com/653480766448040/?locale=en_US&cms_platform=android

[43] WhatsApp. (2026h). How to export your chat history [Help Center article 1180414079177245]. Retrieved October 15, 2026, from https://web.archive.org/web/20261015131254/https://faq.whatsapp.com/1180414079177245/?locale=en_US&cms_platform=android

[44] Wunder, J., Wash, R., Renaud, K., Oliveira, D. A., & Benenson, Z. (2025). Achieving resilience: Data loss and recovery on devices for personal use in three countries. In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (pp. 1–26). ACM. https://doi.org/10.1145/3706598.3714202

[45] Yu, W., & Ignat, C.-L. (2020). Conflict-free replicated relations for multi-synchronous database management at edge. In 2020 IEEE International Conference on Smart Data Services (pp. 113–121). IEEE. https://doi.org/10.1109/SMDS49396.2020.00021