Twenty-One Grammars of Diagnostic Evidence

Memory Dumps, Traces, and Logs through Chinese, Korean, Japanese, Arabic, Sanskrit, Russian, German, Irish, French, Classical Greek, Latin, Southern Quechua, Swahili, Turkish, Māori, Cherokee, Tibetan, Tamil, Ancient Egyptian, Basque, and Udmurt

Twenty-One Grammars of Diagnostic Evidence

Memory Dumps, Traces, and Logs through Chinese, Korean, Japanese, Arabic, Sanskrit, Russian, German, Irish, French, Classical Greek, Latin, Southern Quechua, Swahili, Turkish, Māori, Cherokee, Tibetan, Tamil, Ancient Egyptian, Basque, and Udmurt

存—迹—志
상태—자취—기록
状態—軌跡—記録
حالة—أثر—سجل
अवस्था—पद—अभिलेख
состояние—след—запись
Zustand—Spur—Protokoll
staid—rian—taifead
état—trace—journal
κατάστασις—ἴχνος—ὑπόμνημα
status—vestigium—commentarius
kay—yupi—qillqa
hali—alama—kumbukumbu
durum—iz—kayıt
āhua—tapuwae—rangitaki / mauhanga
ᎤᏍᏗᏓᎥ—ᎠᏍᏓᏩᏛᏍᏗ—ᏕᎦᎥᎥᏛᎢ
གནས་སྟངས་—རྗེས་ཤུལ་—ཐོ་ཡིག་
நிலை—தடம்—பதிவு
ꜥ—rd—zẖꜣ.w · 𓂝𓏤—𓂋𓂧𓂾—𓏞𓏛
egoera—aztarna—erregistro
югдур—пытьы—гожъям

Idea by Dmitry Vostokov, DumpAnalysis.org

Developed with AI assistance (GPT-5.6 Sol and Opus 5 Extra) and checked against the sources listed in this article.

29 July 2026

Abstract

Memory dumps, traces, and logs are usually introduced as three technical artifact types. A dump captures memory, a trace records execution, and a log stores messages. This description is operationally correct but conceptually thin. It says how the artifacts are produced, not what kind of knowledge each artifact can support, what each one omits, or how an analyst should move among them.

This article approaches the same triad through twenty-one linguistic perspectives: Chinese, Korean, Japanese, Arabic, Sanskrit, Russian, German, Irish, French, Classical Greek, Latin, Southern Quechua, Swahili, Turkish, Māori, Cherokee, Tibetan, Tamil, Ancient Egyptian, Basque, and Udmurt. The purpose is not to claim that a language mechanically determines thought, nor to replace established technical terminology with invented vocabulary. It is to use the semantic resources made visible by translation as analytical instruments. Across the twenty-one perspectives, a stable diagnostic structure appears. A memory dump is associated with state, presence, or a captured condition. A trace is associated with following, trajectory, footprint, residue, or consequence. A log is associated with inscription, record, journal, memorandum, or narrated account.

The comparison also reveals important extensions to the familiar three-artifact model. Tracing can be separated into the act of following, the path that is reconstructed, and the marks left by execution. Residual effects can remain in system state after their originating events have disappeared from observable sequences. Logs can be understood as designed testimony: statements selected and emitted by instrumentation, useful but never identical with the event itself. These distinctions lead to a practical diagnostic cycle: read the record, follow the marks, reconstruct the path, inspect the state, search for residual effects, and require the artifacts to corroborate or challenge one another.

> *Memory dumps preserve what the system was; traces reveal how it moved and what that movement left behind; logs preserve what the system was designed and able to say.*

Contents

Research note

This article is a unified development of the original comparative conversation initiated by Dmitry Vostokov and extended language by language. The first six language sections preserve the conceptual substance of that discussion while removing conversational repetition, harmonizing structure, and placing the results inside one larger diagnostic argument. The remaining fifteen sections extend the same method through documented technical usage and authoritative lexicographic sources. Sanskrit, Classical Greek, Latin, Southern Quechua, and Ancient Egyptian renderings are explicitly conceptual proposals rather than claims of universally standardized computing terminology. The label *Altaic* is retained only to answer the comparative prompt; Turkish is treated accurately as a Turkic language, and no genetic Altaic family is assumed. The Cherokee chapter distinguishes forms attested in the Cherokee Nation word list[60] and Microsoft computer terminology from the article's analytical use of those forms. The Tibetan, Basque, and Udmurt chapters likewise distinguish attested lexical meanings from proposed analytical compounds or workflow language. All twenty-one triads should be read as analytical formulations, not replacements for established product documentation or local technical usage.

Part I — From Artifacts to Evidence

1. The technical triad is also an epistemic triad

A system failure is never available to an analyst in its entirety. The original execution has already passed; even a live incident is observed through interfaces, buffers, snapshots, counters, messages, and models. Diagnostic work therefore begins with representations. Memory dumps, traces, and logs do not merely contain different data. They expose different relationships between an event and what can later be known about it.

A memory dump arrests a changing system and preserves a selected internal configuration. Its characteristic strength is coexistence: it shows which threads, objects, registers, references, buffers, and partially completed operations were present together. It is spatial and configurational even when the entities it contains were produced by temporal processes. Its characteristic limitation is history. The state may contain the decisive result of earlier activity without preserving the entire sequence that produced it.

A trace makes movement observable. It can reveal ordering, transitions, propagation, latency, calls, retries, and causal adjacency. Yet the English word *trace* compresses several ideas. It can mean the act of following, the route that is followed, a message emitted during that route, or a mark that remains after motion has ended. Those meanings should not be treated as interchangeable. An instrumentation event is not yet a reconstructed path; a reconstructed path is not the whole state; and a residual effect may survive even when no explicit trace message remains.

A log is a deliberately produced account. Its entries are chosen, named, classified, serialized, buffered, filtered, and retained according to the design and operating conditions of instrumentation. A log can provide the most immediately intelligible description of an incident, but that intelligibility can create misplaced authority. The message “connection failed” is not the connection failure itself. It is a component’s statement about what it detected, when it detected it, and how its developers taught it to speak.

These three relations can be summarized as state, movement, and account. The twenty-one-language comparison sharpens the summary by adding preserved existence, trajectory, footprint, consequence, inscription, testimony, memorandum, investigation, and culturally distinct ways of relating condition, path, and retained memory.

2. Translation as an analytical instrument

Technical translation often begins with equivalence: which term in language B corresponds to the established term in language A? That question is necessary for documentation, interfaces, education, and communication. It is not the only useful question. Once an artifact is named in another language, nearby words and distinctions may expose conceptual structure that English compresses or leaves implicit.

This method is neither linguistic mysticism nor a claim that speakers of one language diagnose software in a predetermined way. Modern technical vocabularies are hybrid. They combine borrowings, transliterations, calques, older literary terms, native words, and specialized compounds. The analytical value lies precisely in moving between these layers. A borrowed noun may identify the standard artifact, while a native or historically layered term may explain what the analyst is doing with it.

The method used throughout this article follows four rules:

- Preserve conventional technical terms and distinguish them from interpretive proposals.
- Separate the production of an artifact from the analytical relation it represents.
- Treat every artifact as partial, selected, and situated.
- Prefer cross-artifact corroboration over the authority of any single representation.

The result is a comparative diagnostic vocabulary, not a new localization standard.

3. The common triad

Evidence family	What it makes observable	Principal diagnostic question	Characteristic limitation
Memory dump or snapshot	Captured internal state and coexistence	What was present, and how was it configured?	Usually incomplete history
Trace	Movement, ordering, path, marks, and propagation	How did execution arrive here, and what did it leave behind?	Selected events and values
Log	Intentionally recorded statements and classifications	What did the system record or claim happened?	Designed account, not reality itself

The commonality is strong, but the differences among the languages matter. Chinese foregrounds preserved existence and the written account. Korean moves between a technical loan, a Sino-Korean analytical term, and a native experiential word for a trail left behind. Japanese offers a particularly clear progression from following to trajectory to residual trace to evidential audit trail. Arabic connects trace with effect. Sanskrit adds the internal impression that prior action leaves in present state. Russian connects the trace, the act of following it, causal consequence, and investigation. German makes configuration, physical trace, and procedural record unusually explicit. Irish distinguishes the act of

tracking from the remaining trail and from information deliberately placed on record. French connects state, inscription, and the chronological journal. Classical Greek distinguishes settled condition, footprint, and memorandum. Latin joins standing state, vestige, and the analyst's collected notes. Southern Quechua joins being, footprint, and writing while grammatically marking the source and certainty of knowledge. Swahili separates active following from the mark that remains and from the record retained for later use. Turkish distinguishes condition, trace, and registration while technical compounds preserve the boundary between a trace and a trace log. Māori connects condition and form with footprints that can be followed and a log that its technical and everyday registers name differently. Cherokee makes condition, trace, tracer, record, and a compound technical term for a log file visibly distinct. Tibetan separates circumstance, a trail left afterward, and an organized register. Tamil distinguishes condition, the mark or route of tracing, and a deliberately retained record. Ancient Egyptian supplies a historically bounded analogy of condition, footprint, and written record. Basque connects state, vestige, and record while treating the computing dump as an overturning or discharge. Udmurt joins situation, footprint, written note, and a living technical register in which native terms coexist with Russian-derived loans.

4. The comparative matrix

Language	Memory dump lens	Trace lens	Log lens
Chinese	存 — preserved existence	迹 — mark or remnant of movement	志 — intentionally written account
Korean	상태 — captured state	추적 / 궤적 / 자취 — following, trajectory, trail	기록 — preserved record
Japanese	状態 — captured condition	追跡 / 軌跡 / 痕跡 / 証跡 — following, trajectory, residual and evidential trace	記録 — intentional record
Arabic	حالة — condition or state	تتبّع / مسار / أثر — following, route, trace or effect	سجل — organized record
Sanskrit	अवस्था — state in which something stands	अनुसरण / मार्ग / पद / संस्कार — following, path, footprint, residual impression	अभिलेख / वृत्तान्त — inscription and account
Russian	состояние / снимок — state and snapshot	трассировка / трасса / след — tracing, route, remaining mark	запись / журнал — entry and accumulated journal
German	Zustand / Speicherabbild — condition and captured memory image	Ablaufverfolgung / Spur — tracing and the mark left behind	Protokoll / Aufzeichnung — procedural record and recording
Irish	staid / dumpáil cuimhne — state and memory dump	rianú / rian / lorg — tracking, trace, and trail	taifead / loga — stored record and operational log
French	état / vidage — condition and captured dump	traçage / trace — tracing and persistent mark	journal / enregistrement — chronological log and recorded entry

Language	Memory dump lens	Trace lens	Log lens
Classical Greek	κατάστασις — settled condition or configuration	ἀνίχνευσις / ἴχνος — tracking and footprint	ὑπόμνημα / ἀναγραφή — memorandum and formal registration
Latin	status — standing, posture, or condition	vestigatio / vestigium — tracking and remaining footprint	commentarius / acta — working record and official proceedings
Southern Quechua	kay / imayna kasqan — being and the way something is	yupinay / yupi — tracking and footprint	qillqa / qillqasqa — writing, document, and what has been written
Swahili	hali — state, condition, circumstance, or case	ufuatiliaji / alama / nyayo — following, mark, and footprint	kumbukumbu / rekodi — retained record and operational entry
Turkish	durum / bellek dökümü — condition and memory dump	izleme / iz — tracing and the mark or route followed	kayıt / günlük — registered entry and accumulated log
Māori	āhua / tūnga — condition, form, and position	whai / tapuwae — following and footprint	rangitaki / mauhanga — log in technical and everyday registers
Cherokee	ᎤᏍᏗᏓᏅ / ᎠᏅᏓᏗᏍᏙ — condition and memory	ᎠᏍᏓᏩᏛᏍᏗ / ᎠᏍᏓᏩᏕᎩ — trace and tracer	ᏕᎦᏅᏅᏛᎢ / ᏄᎵᏍᏔᏂᏙᎲ ᎧᏃᏢᏍᎩ ᎠᏍᏆᏂᎪᏙᏗ — record and log file
Tibetan	གནས་སྟངས་ — condition, situation, or state	རྗེས་ཤུལ་ — trace, trail, remains, or aftermath	ཐོ་ཡིག་ / ཐེམ་ཐོ་ — register, catalogue, and record
Tamil	நிலை / நினைவகக் கொட்டல் — state and memory dump	தடமறிதல் / தடம் — tracing and track or mark	பதிவு / பதிவேடு — record and register or logbook
Ancient Egyptian	ꜥ — condition or state of affairs	rd — foot or footprint	zẖꜣ.w — writing, record, or depiction
Basque	egoera — condition, situation, or state	aztarna — trace, vestige, mark, or indication	erregistro — register and computing record
Udmurt	югдур — condition, situation, or state	пыты — track or footprint	гожъям / журнал — written record and operational journal

The matrix should not be read as a set of exact one-word translations. It is a map of analytical centers of gravity. Each row uses a language to distribute the work that English often assigns to the single nouns *dump*, *trace*, and *log*.

Part II — Twenty-One Language Perspectives

The following twenty-one chapters apply a deliberately recurrent structure. Each language is asked to illuminate the same three artifacts so that similarities and differences remain directly comparable. Repetition of the core questions—what existed, how execution moved, and what was recorded—is therefore methodological rather than accidental.

Chapter 5 — Chinese: 存—迹—志

The Chinese perspective produces the compact formulation **存—迹—志**: preserved existence, marks left by movement, and an intentionally written account. Its diagnostic cycle is equally compact: use the account to question the trail, follow the trail toward preserved state, and use state to test the account. The formulation gives the three artifacts distinct epistemic roles without isolating them from one another.

Sources for this chapter: [1, 2, 3] in Sources and Further Reading.

存—迹—志: A Chinese-Language View of Memory Dumps, Traces, and Logs

The usual Simplified Chinese technical terms are:

- **Memory dump:** 内存转储 (*nèicún zhuǎnchǔ*)
- **Trace:** 跟踪 (*gēnzōng*), or sometimes 轨迹 (*guǐjì*) when referring to the resulting path
- **Log:** 日志 (*rìzhì*)

Traditional Chinese terminology often uses **記憶體傾印** for memory dump, **追蹤** for trace, and **日誌** for log. The contrast is revealing: mainland **转储** suggests transferring stored contents elsewhere, while Taiwanese **傾印** evokes pouring the contents out. Both describe making hidden internal state externally inspectable.

1. Memory Dump as 存 — preserved existence

The character **存** (*cún*) means to exist, remain, preserve, or retain. It can also carry the older sense of attending to or examining something.

A memory dump can therefore be understood as:

> ***系统之存 — what remains of the system***

It preserves a selected portion of the system's internal existence at a particular moment:

- memory contents,
- objects and structures,
- registers and stacks,
- handles and references,
- partially completed computation.

Microsoft's Chinese documentation[1] explicitly describes a dump as a file containing a **snapshot of a process at the moment the dump is created**.

This suggests several more expressive Chinese descriptions:

- **内存现场** — memory scene
- **状态存像** — preserved image of state
- **系统存态** — retained system state
- **瞬时之存** — momentary existence

A dump is therefore primarily **spatial and existential**. It answers:

> ***当时有什么？***
> *What was there at that moment?*

It does not naturally tell the complete story of how the system reached that state.

2. Trace as 迹 — marks left by movement

The character **迹** (*jì*) means a footprint, mark, impression, remnant, or something left behind from which prior action can be inferred.

This fits trace analysis unusually well:

> ***系统之迹 — the marks left by the system's movement***

There is also a useful distinction between two Chinese trace terms:

- **跟踪 / 追踪** — the activity of following
- **轨迹 / 踪迹** — the path or evidence being followed

Thus, tracing is not merely recording events. It is both:

1. the system leaving marks;
2. the analyst following those marks.

A trace log stores messages produced during one or more tracing sessions, according to Microsoft's terminology[2].

A trace is primarily **temporal and relational**. It answers:

> ***它怎样走到这里?***
> *How did it travel to this point?*

Possible conceptual names include:

- **执行之迹** — traces of execution
- **运行轨迹** — runtime trajectory
- **因果踪迹** — causal trail
- **系统行迹** — system movement marks

Unlike a dump, a trace reveals movement, ordering, transitions and interaction—but usually preserves only selected aspects of the state through which the system moved.

3. Log as 志 — an intentionally written account

The final character of **日志**, **志** (*zhì*), can mean an intention or aspiration, but it can also mean a written record, chronicle or systematically collected account.

A log can therefore be interpreted as:

> ***系统之志 — what the system was designed to record***

This does not mean that the system possesses human intention. It means that logging reflects the **intention of its instrumentation**:

- what developers considered important,
- which vocabulary they selected,
- which events they named,
- which values they exposed,
- which conditions they classified as errors or warnings.

A log is consequently more than residue. It is a **designed narrative**.

It answers:

> ***系统说发生了什么?***
> *What did the system say happened?*

Microsoft's documentation[1] also notes that logging and tracing sometimes overlap terminologically, which reflects the fact that both may produce sequences of diagnostic messages even though their analytical roles can differ.

The three epistemic forms

The Chinese-language view produces a compact diagnostic triad:

Artifact	Character	Diagnostic form	Principal question
Memory Dump	**存**	Preserved existence	What was there?
Trace	**迹**	Residual movement	How did it happen?
Log	**志**	Recorded account	What did the system report?

Or more poetically:

> ***Dump preserves the system's 存.***
>
> ***Trace reveals the system's 迹.***
>
> ***Log expresses the system's 志.***

These are not merely three storage formats. They are three different ways of knowing a software system.

存 is not explanation

A dump may contain the decisive evidence, but it does not automatically identify which earlier action produced it.

迹 is not complete existence

A trace may reconstruct execution flow, but it cannot preserve every value, object, hidden dependency or corrupted structure.

志 is not necessarily truth

A log is what the implementation was able and instructed to say. It may be incomplete, delayed, misleading, incorrectly classified, or produced before the actual failure.

A Chinese diagnostic cycle

The three forms can be connected into a method:

以志问迹 — use the account to question the trail

Begin with logs. Their semantic descriptions suggest components, operations, failures and approximate times that should be investigated in traces.

以迹索存 — follow the trail towards preserved state

Use traces to identify the relevant process, thread, request, transaction or causal transition. Then locate an appropriate dump or snapshot.

以存证志 — use preserved state to test the account

Inspect the dump to determine whether the state agrees with what the logs claimed and what the trace appears to show.

The full movement is:

> ***志 → 迹 → 存 → 证***
> *Account → trail → preserved state → evidence*

But analysis can also run in reverse:

> ***存→迹→志→义***
> *State → preceding path → recorded statements → reconstructed meaning*

A second, more forensic formulation

Another natural Chinese formulation is:

> ***系统现场 — 系统踪迹 — 系统自述***
> *System scene — system trail — system self-account*

Here:

- the **memory dump is the scene**;
- the **trace is the trail through and around the scene**;
- the **log is the system's account of what occurred**.

The investigator should trust none of them in isolation. The scene may be incomplete, the trail may contain gaps, and the account may be mistaken. Diagnostic truth emerges from their **互证** (*hùzhèng*)—mutual corroboration.

Chapter 6 — Korean: 상태—자취—기록

The Korean perspective is especially valuable because it moves across three lexical layers. English-derived technical terms identify the familiar artifacts; Sino-Korean terms express formal analytical concepts; the native Korean word **자취**, together with Sino-Korean **흔적**, makes the remaining trail experientially concrete. The result separates the activity of tracking from the trajectory produced by motion and from the marks that remain after passage.

The conventional Korean technical terms are:

- **Memory dump:** 메모리 덤프
- **Trace:** 추적, 트레이스, 실행 추적, 추적 로그
- **Log:** 로그, 기록

Unlike the Chinese terminology, modern Korean computing vocabulary frequently retains English loanwords such as **메모리**, **덤프**, **트레이스**, and **로그**. However, when these terms are explained through Korean and Sino-Korean vocabulary, a richer conceptual structure emerges:

> ***상태 — 자취 — 기록***
> *State — trail — record*

This triad can be understood as three different forms of diagnostic knowledge.

Sources for this chapter: [4, 5] in Sources and Further Reading.

1. Memory Dump as 상태 — captured state

Korean technical documentation normally calls a memory dump **메모리 덤프** or simply **덤프**. Microsoft's Korean documentation[4] defines a dump as a file containing a snapshot of a process at the time the dump is created, allowing the application's state to be inspected. A process dump contains no earlier execution history by itself.

The borrowed word **덤프** retains the English image of emptying or depositing contents. In Korean diagnostic language, however, the artifact can be interpreted more naturally as:

- **상태의 단면** — a cross-section of state
- **순간 상태** — momentary state
- **메모리 현장** — memory scene
- **상태를 떠낸 것** — a casting or impression taken from the state

The last expression is metaphorical. Just as one can make an impression or cast of an object, a dump can be viewed as something that has **떠낸**, or captured, the internal configuration of a process.

A full user-mode dump may contain the process memory space, executable images, handle information, stacks, threads, and other data needed to reconstruct the process's condition at the capture point.

A memory dump therefore primarily answers:

> ***그때 시스템은 어떤 상태였는가?***
> *What state was the system in at that moment?*

Or more concretely:

> ***그 순간 무엇이 메모리에 남아 있었는가?***
> *What remained in memory at that moment?*

The Korean concept **상태** emphasizes configuration rather than narrative. A dump preserves the arrangement of things, but not necessarily the sequence of actions that produced that arrangement.

2. Trace as 추적, 궤적 and 자취

Korean offers several related words for *trace*, and their differences are diagnostically important.

추적 — following the trail

추적 is written 追跡:

- **追** — to follow or pursue
- **跡** — a footprint, trace, or trail

The National Institute of Korean Language[5] defines **추적하다** as following the traces of an event or person in order to find something.

Thus, **추적** primarily describes an activity:

> *Someone follows evidence that has already been left behind.*

In software diagnostics, tracing can similarly mean following:

- execution flow,
- requests,
- calls,
- messages,
- transactions,
- causal relationships,
- movement across processes or services.

The Korean word therefore places the analyst inside the concept. A trace is not only generated by a system; it is also **followed by an investigator**.

궤적 — the path produced by movement

궤적 is written 軌跡:

- **軌** — track or wheel track
- **跡** — trace or footprint

The Korean Basic Dictionary[5] defines it both as the track left by a moving object and as the course through which some activity has developed.

This makes **궤적** particularly suitable for the resulting trace artifact:

> ***실행 궤적*** *— execution trajectory*
>
> ***요청 궤적*** *— request trajectory*
>
> ***오류 전파 궤적*** *— error-propagation trajectory*

A trace does not merely list events. It reveals that execution has travelled through a structured path.

자취 and 흔적 — what remains after passage

The native Korean word **자취** means a mark or place left behind by something. **흔적** similarly refers to evidence or a mark indicating that something happened or existed.

These words are less technical than **추적** and **궤적**, but perhaps more evocative:

> ***시스템이 움직이며 남긴 자취***
> *The trail left as the system moved*

The Korean trace concept can therefore be divided into two complementary sides:

Korean term	Diagnostic meaning
추적	The act of following
궤적	The structured path
자취 / 흔적	The residual marks left behind
트레이스	The technical artifact containing those marks

Official Korean technical documentation uses expressions such as **이벤트 추적**, **추적 로그**, and **이벤트 추적 로그 파일** for captured diagnostic event sequences.

A trace therefore primarily answers:

> ***시스템은 어떤 길을 거쳐 여기까지 왔는가?***
> *By what path did the system arrive here?*

It may also answer:

> ***무엇이 어떤 순서로 움직였는가?***
> *What moved, and in what order?*

Unlike a dump, a trace is temporal. It shows movement, order and interaction. But it usually preserves only the state values selected by its instrumentation.

3. Log as 기록 — an intentionally preserved account

The ordinary technical word is **로그**, but its closest conceptual Korean equivalent is **기록**.

기록 is written 記錄. The Korean Basic Dictionary[5] defines it as writing or otherwise preserving facts or thoughts so that they will remain for later, as well as the resulting document or representation.

This definition exposes an important property of software logs:

> *A log is not merely something that remains accidentally.*
> *It is something that was deliberately selected for preservation.*

A log records what instrumentation was designed to notice:

- named events,
- warnings and errors,
- state changes,
- identifiers,
- decisions,
- durations,
- outcomes,
- values considered diagnostically relevant.

The system does not independently decide what matters. Developers, architects, operators and frameworks determine what can be logged and how it will be described.

A log can therefore be understood as:

- **시스템 기록** — system record
- **실행 기록** — execution record
- **운영 기록** — operational record
- **시스템의 자기 기록** — the system's self-record
- **계측된 진술** — instrumented account

Korean technical documentation describes diagnostic logs as information recorded by an application or platform and made available for debugging and analysis.

A log primarily answers:

> ***시스템은 무엇을 기록했는가?***
> *What did the system record?*

A deeper diagnostic question is:

> ***시스템은 무엇을 기록하도록 설계되었는가?***
> *What was the system designed to record?*

The second question matters because absence from a log does not prove that an event did not occur. It may simply mean that the event was not instrumented, was filtered out, was overwritten, or could not be emitted.

The three Korean epistemic forms

The resulting Korean diagnostic triad is:

Artifact	Korean concept	Diagnostic form	Principal question
Memory dump	**상태**	Captured configuration	What state existed then?
Trace	**자취 / 궤적**	Movement made observable	How did the system arrive there?
Log	**기록**	Intentionally preserved account	What did the system record?

In compact form:

> ***덤프는 상태를 붙잡는다.***
> *A dump captures state.*
>
> ***트레이스는 움직임의 자취를 드러낸다.***
> *A trace reveals the trail of movement.*
>
> ***로그는 선택된 사실을 기록한다.***
> *A log records selected facts.*

These are not simply different file formats. They embody three different relationships to software reality:

- **상태** concerns what existed.
- **자취** concerns what passed and changed.
- **기록** concerns what was deliberately made communicable.

상태 is not history

A memory dump may contain the decisive corrupted object, blocked thread, invalid pointer, exhausted resource or inconsistent structure.

But the dump normally does not reveal the full sequence of events that produced it. Microsoft's Korean documentation[4] explicitly contrasts process dumps with execution histories: a process dump is a memory snapshot and does not itself contain earlier information.

Therefore:

> ***상태는 결과를 보여 주지만 경로 전체를 보여 주지는 않는다.***
> *State shows the result, but not necessarily the complete path.*

자취 is not complete state

A trace may reveal calls, transitions, messages and timing, but only for the events and attributes that were captured.

It can show:

- where execution went,
- when operations occurred,
- which components interacted,
- how long transitions took,
- how an error propagated.

But it may omit:

- hidden object state,
- overwritten memory,
- uninstrumented branches,
- values not included in trace messages,
- corruption that occurred below the tracing layer.

Therefore:

> ***자취는 움직임을 보여 주지만 존재 전체를 보존하지는 않는다.***
> *A trail shows movement, but does not preserve the whole state of existence.*

기록 is not necessarily truth

A log is shaped by vocabulary, assumptions and implementation decisions.

A message such as “connection failed” may be:

- correct but incomplete,
- emitted by the wrong layer,
- based on a secondary symptom,
- delayed,
- duplicated,
- incorrectly classified,
- disconnected from the original cause.

The log records what the code knew—or believed it knew—at the point where the message was produced.

Therefore:

> ***기록은 진술이지만 진실 전체는 아니다.***
> *A record is an account, but not the whole truth.*

A Korean diagnostic cycle

The three evidence forms can be arranged into a diagnostic method:

기록에서 자취를 찾는다

Find the trail through the record

Begin with logs. Their terminology, timestamps, identifiers and reported conditions suggest which execution path should be reconstructed.

자취를 따라 상태에 이른다

Follow the trail towards the state

Use traces to identify the relevant process, request, transaction, thread, component or causal transition. Then locate or capture the corresponding dump.

상태로 기록을 검증한다

Test the record against the state

Use the dump to determine whether the internal state supports what the log reported and what the trace appears to imply.

The cycle is:

> ***기록 → 추적 → 상태 → 검증***
> *Record → tracing → state → verification*

A reverse analytical movement is also possible:

> ***상태 → 역추적 → 기록 → 해석***
> *State → backward tracing → records → interpretation*

The Korean word **역추적**, meaning to trace backwards, is particularly appropriate for postmortem analysis: the analyst begins with a failure state and reconstructs the earlier path that produced it.

A forensic Korean formulation

A second, more concrete triad is:

> ***현장 — 동선 — 기록***
> *Scene — route of movement — record*

Here:

- the **memory dump is the 현장**, the preserved scene;
- the **trace is the 동선 or 자취**, the path through the scene and towards it;
- the **log is the 기록**, the written account produced along the way.

This resembles a forensic investigation:

- the scene may have been altered;
- the trail may contain gaps;
- the record may be mistaken;
- timestamps may disagree;
- witnesses—in this case, components—may have only partial viewpoints.

Diagnostic confidence emerges through:

> ***상호 검증*** *or* ***교차 검증***
> *mutual or cross-verification*

No single evidence source should automatically be treated as authoritative.

A specifically Korean linguistic insight

The Korean perspective is distinctive because it provides three linguistic layers at once.

English-derived technical terms

- 메모리
- 덤프
- 트레이스
- 로그

These name the engineering artifacts directly.

Sino-Korean analytical terms

- 상태
- 추적
- 궤적
- 기록
- 흔적

These provide abstract conceptual structure.

Native Korean experiential terms

- 자취
- 남다
- 따라가다
- 드러나다

These describe how evidence is encountered by an investigator.

Together they produce a complete diagnostic movement:

> ***무언가가 남고, 분석가는 그것을 따라가며, 그 과정에서 상태를 드러낸다.***
> *Something remains; the analyst follows it; and through that process the state is revealed.*

This makes the Korean formulation more dynamic than a simple classification of artifacts.

The system:

1. **상태를 가진다** — possesses state;
2. **자취를 남긴다** — leaves a trail;
3. **사건을 기록한다** — records events.

The analyst:

1. **기록을 읽는다** — reads the record;
2. **자취를 추적한다** — follows the trail;
3. **상태를 재구성한다** — reconstructs the state.

Chapter 7 — Japanese: 状態—軌跡—記録

The Japanese perspective provides the clearest staged transformation of trace evidence: **追跡 → 軌跡 → 痕跡 → 証跡**. Tracing begins as an activity of following, yields a trajectory, discovers residual signs, and becomes evidential only after preservation and verification. This sequence prevents the common mistake of treating raw instrumentation output as already established proof.

The conventional Japanese technical terms are:

- **Memory dump:** メモリダンプ
- **Trace:** トレース, 実行トレース, イベントトレース
- **Log:** ログ, ログ記録

Modern Japanese computing terminology commonly retains the English-derived katakana words **ダンプ**, **トレース**, and **ログ**. Yet Japanese kanji vocabulary exposes several distinctions hidden inside the English terminology:

> ***状態 — 軌跡 — 記録***
> *State — trajectory — record*

This triad represents three different forms of diagnostic knowledge:

- the dump captures the system's **状態**;
- the trace reveals its **軌跡** and **痕跡**;
- the log preserves a selected **記録**.

Sources for this chapter: [6–10] in Sources and Further Reading.

1. Memory Dump as 状態 — captured condition

Japanese technical documentation normally uses **メモリダンプ** or simply **ダンプ**. Microsoft's Japanese documentation[6] describes a dump as a file containing a snapshot of a process at the moment it was created, enabling examination of the application's state.

The word **状態** (*jōtai*) means the condition or configuration in which something exists. A dictionary explanation even uses the analogy of taking a photograph at one particular moment to describe a state at time (t).

This makes **状態** especially appropriate for a memory dump:

> ***ある瞬間におけるシステムの状態***
> *The state of the system at a particular moment*

A dump may preserve:

- memory contents,
- threads and stacks,
- registers,
- objects and data structures,
- handles and references,
- loaded modules,
- partially completed computations.

A live kernel dump is similarly described as capturing system state and preserving a consistent snapshot of kernel memory for later analysis.

Several Japanese expressions can therefore illuminate the concept:

- **状態の写し** — a copy or image of state
- **瞬間状態** — momentary state
- **メモリの断面** — a cross-section of memory
- **システムの現場** — the scene of the system
- **一瞬を切り取ったもの** — something that cuts out one instant

The phrase **切り取る**, "to cut out" or "extract," is particularly suggestive. A memory dump separates one instant from the continuous life of a running system and makes that instant available for inspection.

A dump primarily answers:

> ***その瞬間、システムはどのような状態だったのか。***
> *What state was the system in at that moment?*

It may also answer:

> ***そのとき、メモリには何が残っていたのか。***
> *What remained in memory at that time?*

A memory dump is therefore predominantly **spatial and configurational**. It presents objects, relationships and structures as they existed at a selected moment.

2. Trace as 追跡, 軌跡 and 痕跡

Japanese does not offer only one conceptual equivalent for *trace*. It offers several, and their differences are diagnostically valuable.

追跡 — the activity of following

追跡 is read *tsuiseki*:

- **追** — to pursue or follow;
- **跡** — a mark, footprint or trace.

The word means pursuit, tracking or tracing. In computing contexts, it can describe following execution or monitoring an evolving process.

Thus, **追跡** emphasizes an activity:

> ***残された跡を追うこと***
> *Following the marks that were left behind*

In software diagnostics, the analyst may pursue:

- a request,
- a transaction,
- a thread,
- a message,
- a function call,
- an error,
- a state transition,
- a causal chain.

This means that **追跡** places the observer inside the concept. The system produces evidence, but the investigator actively follows it.

A useful formulation is:

> ***実行を追跡する。***
> *Trace the execution.*

Here, trace is something one **does**.

軌跡 — the path travelled

軌跡 is read *kiseki*. Its literal imagery is that of a track or course produced by movement. It can refer to the path of a moving point or object and, more generally, to the course through which an activity or development has passed.

In diagnostics, this produces natural expressions such as:

- **実行軌跡** — execution trajectory
- **要求の軌跡** — request trajectory
- **処理の軌跡** — processing trajectory
- **障害伝播の軌跡** — fault-propagation trajectory
- **状態遷移の軌跡** — state-transition trajectory

A trace is therefore not merely a collection of messages. It represents the route through which execution moved.

Where **追跡** is the analytical action, **軌跡** is the path being reconstructed:

> ***追跡する者が、軌跡をたどる。***
> *The investigator traces by following the trajectory.*

A trace primarily answers:

> ***システムはどの経路を通って、この状態に至ったのか。***
> *Through what path did the system arrive at this state?*

痕跡 — evidence that something happened

痕跡 is read *konseki*. It refers to a remaining mark or sign from which a former presence or event can be inferred. The English word *trace* is similarly translated into Japanese as **痕跡**, **形跡**, **跡**, or **足跡** when it denotes something left after passage or action.

This gives trace analysis a more forensic interpretation:

> ***システムの活動が残した痕跡***
> *The traces left by system activity*

Examples include:

- an error message left after a failed operation;
- a timing gap left by contention;
- repeated retries left by an unavailable dependency;
- an unexpected transition left by corrupted state;
- an identifier appearing across several services;
- an incomplete sequence left by a terminated process.

The distinction is:

Japanese term	Diagnostic meaning
追跡	The act of following
軌跡	The path through which execution moved
痕跡	Residual evidence that movement or activity occurred
トレース	The technical mechanism or collected artifact

Microsoft's Japanese documentation[6] describes event trace log files as containing trace messages generated during one or more trace sessions. The messages are first placed in trace-session buffers and then delivered to consumers or written to trace logs.

Thus, the technical **トレース** contains selected **痕跡** from which an execution **軌跡** can be reconstructed through **追跡**.

証跡 — trace as evidence

Japanese technical and security language adds another important word:

> ***証跡*** *(shōseki) — evidential trace or audit trail*

A dictionary describes **証跡** as a trace that later helps demonstrate that something was factual or true.

In computing, **監査証跡** refers to retained information that supports auditing, compliance and investigation. Microsoft documentation, for example, describes an audit trail as a collection of textual log files recording system interactions and changes.

The Information-technology Promotion Agency of Japan also discusses preserving logs so that evidence of security compromise remains available and can later be analysed for signs of intrusion.

This produces an important progression:

> ***痕跡 → 証跡***
> *Residual trace → evidential trace*

A **痕跡** merely remains. A **証跡** has been collected, preserved and contextualized so that it can support an inference or claim.

Not every trace message automatically becomes evidence. It becomes useful **証跡** only when its provenance, time, context and integrity are sufficiently reliable.

3. Log as 記録 — intentional preservation

The ordinary Japanese computing term is **ログ**, but its main conceptual equivalent is **記録** (*kiroku*).

記録 means writing down, representing or otherwise preserving facts so that they remain available for future use. It can refer both to the act of recording and to the resulting preserved material.

This reveals a defining feature of logs:

> *Logs do not simply remain; they are produced because something was selected to be recorded.*

A system log may record:

- named events,
- warnings and errors,
- identifiers,
- decisions,
- state changes,
- resource usage,
- durations,
- security operations,
- success and failure outcomes.

A log is consequently not a neutral copy of system reality. It reflects instrumentation decisions:

- what developers considered important;
- which events received names;
- which values were exposed;
- which severity levels were assigned;
- which messages were filtered;
- which events were never instrumented.

A log primarily answers:

> ***システムは何を記録したのか。***
> *What did the system record?*

The deeper question is:

> ***システムは何を記録するように設計されていたのか。***
> *What was the system designed to record?*

The difference matters because an event's absence from a log does not establish that it never occurred. It may not have been selected for logging, may have been filtered, dropped, overwritten or lost during failure.

記録, 履歴 and 日誌

Japanese provides several related concepts around logging.

記録 — recorded fact

記録 emphasizes deliberate preservation. It is the broadest conceptual equivalent of logging.

> ***イベントを記録する***
> *Record an event.*

履歴 — accumulated history

履歴 (*rireki*) refers to what something or someone has passed through over time. In computing, it naturally appears in expressions such as:

- **操作履歴** — operation history
- **変更履歴** — change history
- **実行履歴** — execution history
- **アクセス履歴** — access history

Where **記録** emphasizes individual recorded facts, **履歴** emphasizes their accumulated temporal continuity.

A log can therefore be understood as a set of records that forms a history:

> ***記録の蓄積が履歴になる。***
> *An accumulation of records becomes a history.*

日誌 — operational journal

日誌 (*nisshi*) means a regular chronological account, especially an official or operational one; it is distinguished from the more personal **日記**.

Although Japanese computing normally uses the katakana term **ログ**, the journal-like meaning survives conceptually:

> *A log is a chronologically maintained operational record.*

The terms can therefore be arranged as follows:

Japanese term	Emphasis
記録	Individual facts intentionally preserved
履歴	The accumulated course of past operations
日誌	A chronological operational account
ログ	The technical artifact or recording mechanism

The three Japanese epistemic forms

The resulting Japanese diagnostic triad is:

Artifact	Japanese concept	Diagnostic form	Principal question
Memory dump	**状態**	Captured configuration	What state existed then?
Trace	**軌跡・痕跡**	Observable movement and its remains	How did the system arrive there?
Log	**記録**	Intentionally preserved account	What did the system record?

In compact form:

> ***ダンプは状態を切り取る。***
> *A dump cuts out a state.*
>
> ***トレースは実行の軌跡を示す。***
> *A trace shows the trajectory of execution.*
>
> ***ログは選ばれた事実を記録する。***
> *A log records selected facts.*

These are not merely three file formats. They represent three different ways of knowing a software system:

- **状態** concerns what existed;
- **軌跡** concerns how it moved;
- **痕跡** concerns what its movement left behind;
- **記録** concerns what was intentionally preserved.

状態 is not 経緯

A memory dump may expose the decisive evidence:

- a corrupted object;
- a deadlocked set of threads;
- a stale pointer;
- a leaked resource;
- an inconsistent data structure;
- an unexpected queue;
- a partially updated state.

But the dump does not ordinarily contain the complete earlier sequence that produced that state.

Therefore:

> ***状態は結果を示すが、そこに至る経緯のすべてを示すわけではない。***
> *State shows the result, but not necessarily the complete course that led to it.*

The Japanese word **経緯** is useful here because it means the circumstances and progression through which something came about.

A dump preserves the **結果としての状態**, the resulting state, but analysis must often reconstruct the **そこに至る経緯**, the course by which it arose.

軌跡 is not complete existence

A trace may show:

- execution order;
- calls and returns;
- service interactions;
- request propagation;
- timing and latency;
- retries;
- transitions;
- error propagation.

But it normally contains only events and values selected by tracing instrumentation.

It may omit:

- uninstrumented branches;
- hidden memory state;
- overwritten values;
- silent corruption;
- state changes below the tracing layer;
- events dropped because of buffer or resource limits.

Therefore:

> ***軌跡は動きを示すが、存在していた状態のすべてを保存するわけではない。***
> *A trajectory shows movement but does not preserve every state that existed.*

記録 is not necessarily truth

A log message is a statement generated from one implementation viewpoint.

For example, a message saying “connection failed” may be:

- accurate but incomplete;
- a secondary symptom;
- based on stale information;
- emitted by the wrong abstraction layer;
- incorrectly classified;
- separated in time from the original cause.

A log says what a component was programmed and able to say at that point.

Therefore:

> ***記録は事実への手掛かりであって、事実そのものとは限らない。***
> *A record is a clue to the facts, but is not necessarily the facts themselves.*

Or more compactly:

> ***記録されたことと、実際に起きたことは同一とは限らない。***
> *What was recorded and what actually happened are not necessarily identical.*

A Japanese diagnostic cycle

The three evidence forms can be connected into a diagnostic method.

記録を読む

Read the records

Begin with logs. Their timestamps, identifiers, terminology and reported events suggest which components and time intervals require investigation.

軌跡をたどる

Follow the trajectory

Use traces to reconstruct the relevant execution path:

- which request moved where;
- which thread performed which operation;
- which transition preceded the failure;
- where latency, retries or divergence began.

The verb **たどる** is particularly expressive. It means following a route, sequence or set of clues step by step.

状態を調べる

Inspect the state

Use a dump or snapshot to examine the internal configuration at the significant point identified by trace analysis.

相互に照合する

Cross-check the evidence

Determine whether:

- the dump supports the trace interpretation;
- the trace explains the logged statements;
- the logs identify the same entities visible in the dump;
- timestamps and causal relationships are consistent.

The complete movement is:

> ***記録 → 追跡 → 軌跡 → 状態 → 照合***
> *Record → tracing → trajectory → state → corroboration*

A reverse movement is equally important:

> ***状態 → 遡及 → 痕跡 → 記録 → 再構成***
> *State → backward inquiry → traces → records → reconstruction*

The Japanese verb **遡る** (*sakanoboru*), "to go back upstream or backwards in time," is especially appropriate for postmortem analysis.

The analyst begins with the visible failure state and works upstream towards its origin:

> ***障害状態から原因へ遡る。***
> *Trace backwards from the failure state towards its cause.*

A forensic Japanese formulation

A second, more concrete triad is:

> ***現場 — 足取り — 記録***
> *Scene — movements or trail — record*

Here:

- the **memory dump is the 現場**, the preserved scene;
- the **trace is the 足取り**, the sequence of movements leading to and through the scene;
- the **log is the 記録**, the written account produced along the way.

The word **足取り** can mean footsteps, progress or the route taken. It is therefore a natural forensic metaphor for reconstructing execution.

The investigator must remember:

- the scene may be incomplete;
- the trail may contain gaps;
- the record may reflect only one viewpoint;
- timestamps may be imprecise;
- some evidence may have been overwritten;
- different components may describe the same event differently.

Diagnostic confidence emerges through:

> ***相互検証*** *— mutual verification*
>
> ***照合*** *— comparison and reconciliation*
>
> ***裏付け*** *— corroboration*

No single artifact is automatically authoritative.

A specifically Japanese linguistic insight

The Japanese perspective operates through three linguistic layers.

Katakana technical artifacts

- メモリダンプ
- トレース
- ログ

These identify the engineering mechanisms and file types.

Kanji analytical concepts

- 状態
- 追跡
- 軌跡
- 痕跡
- 証跡
- 記録
- 履歴

These classify the different epistemic functions of the artifacts.

Native Japanese analytical movement

- **残る** — to remain
- **たどる** — to follow step by step
- **追う** — to pursue
- **遡る** — to trace backwards
- **切り取る** — to capture one portion or instant
- **照らし合わせる** — to compare and reconcile

Together they form a complete diagnostic process:

> ***システムが痕跡を残し、分析者がその軌跡をたどり、状態を調べ、記録と照らし合わせる。***
> *The system leaves traces; the analyst follows their trajectory, examines the state, and compares it against the records.*

The system:

1. **状態を持つ** — possesses state;
2. **軌跡を描く** — describes a trajectory;
3. **痕跡を残す** — leaves traces;
4. **出来事を記録する** — records events.

The analyst:

1. **記録を読む** — reads the records;
2. **痕跡を見つける** — finds the traces;
3. **軌跡をたどる** — follows the trajectory;
4. **状態を再構成する** — reconstructs the state;
5. **証跡として裏付ける** — establishes corroborating evidence.

Chapter 8 — Arabic: حالة—أثر—سجل

The Arabic perspective adds a crucial causal bridge. أثر can denote a mark left behind and also an effect or consequence. A trace can therefore be read not only as evidence *about* movement but as what movement has done to the diagnostic world. The language joins tracking, route, residue, consequence, record, and testimony in a way that naturally supports forensic reasoning.

The conventional Arabic technical terms are:

- **Memory dump:** تفريغ الذاكرة، ملف تفريغ الذاكرة
- **Trace:** تتبّع، أثر، آثار، تتبّع التنفيذ
- **Log:** سجل، سجلات، سجلّ الأحداث

Arabic technical documentation commonly uses تفريغ الذاكرة for a memory dump, التتبّع for tracing, الآثار for collected traces, and السجلات for logs. Microsoft's Arabic-localized documentation[11], for example, renders a memory dump as تفريغ الذاكرة and describes such a dump as a snapshot of a process at the moment it is created, while its observability documentation describes traces as capturing the journey of a request or workflow by recording events and state changes. Note that Arabic localization on Microsoft Learn covers Azure and product documentation; the .NET conceptual pages resolve under the ar-sa locale but their body text is served in English.

When these expressions are examined through Arabic vocabulary and word roots, a useful diagnostic triad emerges:

> حالة — أثر — سجل
> *State — trace/effect — record*

These are three different forms of knowledge about a software system:

- the memory dump captures its حالة;
- the trace reveals its أثر and مسار;
- the log preserves a selected سجل.

Sources for this chapter: [11, 12] in Sources and Further Reading.

1. Memory Dump as حالة — captured condition

The standard expression تفريغ الذاكرة consists of:

- تفريغ — emptying, unloading or transferring contents;
- الذاكرة — memory.

The technical metaphor is therefore more active than the English word *snapshot*. The contents of memory are conceptually **emptied out** or transferred from the running system into an external artifact that can be examined later.

Microsoft's .NET documentation describes a dump as a snapshot of a process at the time the dump was created and says that it permits the process state to be captured and examined separately; the Arabic-localized pages name that artifact تفريغ الذاكرة.

The central Arabic concept for this diagnostic form is:

> حالة النظام
> *the state or condition of the system*

The Cairo Arabic Language Academy[12] defines حال الشيء in terms of its quality, configuration and manner of being. This makes حالة an appropriate word for the arrangement of memory, threads, objects and resources at one moment.

A memory dump may preserve:

- thread and stack state;
- registers;
- objects and data structures;
- memory contents;
- loaded modules;
- handles and references;
- partially completed operations;
- corruption or inconsistency visible at capture time.

Microsoft's documentation[11] distinguishes several dump sizes, from small dumps containing module, thread, exception and stack information to full dumps containing all memory, including module images.

A dump therefore primarily answers:

> ما حالة النظام في تلك اللحظة؟
>
> *What was the state of the system at that moment?*

It may also answer:

> ماذا كان موجودًا في الذاكرة؟
>
> *What was present in memory?*

الذاكرة as remembrance

The Arabic word ذاكرة belongs to the lexical family of ذِكر, ذِكرى, تـذكّر and مـذكّرة—remembering, remembrance, recollection and memorandum. The Arabic Academy[12] groups these forms under the same root family.

This creates a productive metaphor:

> تفريغ الذاكرة هو إخراج ما كان النظام "يتذكّره" في تلك اللحظة.
>
> *A memory dump externalizes what the system "remembered" at that moment.*

Of course, computer memory does not remember in the human sense. Yet the Arabic terminology makes visible the relationship between:

- internal retention;
- external preservation;
- later recollection by an investigator.

Possible conceptual descriptions include:

- صورة الحالة — image of the state;
- لقطة الذاكرة — memory snapshot;
- مقطع من حالة النظام — a cross-section of system state;
- مشهد الذاكرة — the memory scene;
- ذاكرة مجمّدة — frozen memory.

The last expression is metaphorical: execution has stopped or been conceptually frozen, while its internal arrangement remains available for inspection.

A memory dump is therefore mainly **spatial and configurational**. It shows what coexisted at one point, but not necessarily how those things came to be arranged that way.

2. Trace as تتبّع, مسار and أثر

Arabic provides several distinct words around the English term *trace*. Their separation is diagnostically significant.

تتبّع — the act of following

تتبّع means following something through successive signs, stages or locations.

In software diagnostics, one may:

- follow a request;
- follow a thread;
- follow calls between functions;
- follow messages across services;
- follow the propagation of an error;
- follow a sequence of state transitions.

Thus, التتبّع emphasizes an activity performed either by instrumentation or by the analyst:

> تتبّع تنفيذ البرنامج
> *tracing program execution*
> تتبّع رحلة الطلب
> *tracing the journey of a request*

Arabic Microsoft documentation uses التتبّع for technical tracing and describes traces as recording function calls, values, system events and state changes across the journey of a request or workflow.

The investigator is therefore implicitly present in the word:

> *Someone is following something that moves.*

مسار — the route travelled

مسار means a path, route or line of movement. The Cairo Arabic Language Academy[12] gives expressions such as مسار الرحلة and مسار الطائرة, meaning the route of a journey or aircraft.

This makes it suitable for the reconstructed trajectory of software execution:

- مسار التنفيذ — execution path;
- مسار الطلب — request path;
- مسار المعاملة — transaction path;
- مسار الخطأ — error path;
- مسار انتشار العطل — fault-propagation path;
- مسار انتقال الحالة — state-transition path.

The distinction is:

> التتبّع هو الفعل، والمسار هو ما نتتبّعه.
> *Tracing is the activity; the path is what we follow.*

A trace therefore answers:

> ما المسار الذي سلكه النظام حتى وصل إلى هذه الحالة؟
>
> *What path did the system take to reach this state?*

أثر — the mark left behind

The most distinctive Arabic word in this analysis is أثر.

It may refer to:

- a mark;
- a footprint;
- a remnant;
- a trace;
- an indication of earlier presence;
- an effect or consequence.

Classical lexicographic sources describe the root as involving a mark or trace made or left upon something. The Cairo Arabic Language Academy[12] also preserves expressions such as جاء في أثره—he came after him or followed in his wake.

This makes أثر exceptionally appropriate for software diagnostics:

> **الأثر هو ما يتركه التنفيذ بعد مروره.**
>
> *A trace is what execution leaves after it passes.*

Examples include:

- a diagnostic message left by a failed operation;
- a latency gap left by contention;
- repeated calls left by retry behaviour;
- an identifier propagated through several services;
- an unexpected state transition;
- a partial sequence left when a process terminates;
- altered memory left by corruption.

The terms can be separated as follows:

Arabic term	Diagnostic meaning
تتبّع	The activity of following execution
مسار	The route or trajectory followed
أثر	A mark or consequence left by activity
آثار	A collected set of traces
سجل التتبّع	The technical artifact containing trace events

Microsoft's Arabic Application Insights documentation[11] uses آثار for trace telemetry and explains that these records contain application events, custom diagnostic messages and trace statements useful for understanding behaviour over time.

أثر as both trace and effect

Arabic adds a conceptual dimension that is weaker in the corresponding English terminology:

> أثر *can be both a residual mark and an effect.*

This means that an execution event may leave two related kinds of أثر:

1. أثر دلالي أو تسجيلي — an observable mark in traces or logs;
2. أثر سببي — an actual consequence in system state or behaviour.

For example:

- increased latency leaves a timing trace;
- that latency also causes retries;
- retries leave repeated-message traces;
- retries also increase resource pressure;
- resource pressure leaves further traces and changes memory state.

Thus, an أثر may be evidence of a cause while simultaneously being the consequence of an earlier cause.

This suggests a diagnostic chain:

> سبب → أثر → سبب جديد → أثر جديد
> *Cause → effect/trace → new cause → new effect/trace*

The Arabic word naturally joins **causality** and **observability**. The mark we observe may itself be part of the mechanism that produces the next mark.

3. Log as سجل — deliberately retained record

The standard Arabic term for a log is سجل, with the plural سجلات.

Common expressions include:

- سجل النظام — system log;
- سجل الأحداث — event log;
- سجل الأخطاء — error log;
- سجل التشغيل — operational log;
- سجل التدقيق — audit log;
- سجل التتبّع — trace log.

A سجل is not merely something that happens to remain. It is an organized record into which information is entered and preserved.

The Cairo Arabic Language Academy's[12] entry for سجّل includes the idea of entering something into a special register so that it is preserved from loss.

This reveals the essential character of software logging:

> السجل هو أثر اختير وحُفظ ونُظّم.
> *A log is a trace that was selected, preserved and organized.*

A system log may contain:

- named events;
- warnings and errors;
- identifiers;
- decisions;
- durations;
- state changes;
- security operations;
- resource measurements;
- success or failure outcomes.

Microsoft's Arabic documentation[11] distinguishes execution logging from trace logging: execution logs contain statistics, audit and performance information, while trace logs contain errors and general diagnostic information.

A log primarily answers:

> ماذا سجّل النظام؟
> *What did the system record?*

A deeper question is:

> ما الذي صُمّم النظام ليسجّله؟
> *What was the system designed to record?*

That distinction matters because absence from a log does not prove absence from execution. An event may have been:

- uninstrumented;
- filtered;
- dropped;
- overwritten;
- emitted too late;
- lost during failure;
- described using another vocabulary.

سجل, تسجيل and رواية

Several related Arabic concepts sharpen the interpretation.

تسجيل — the act of recording

تسجيل emphasizes the mechanism or process:

> تسجيل الأحداث
> *recording events*
> تسجيل رسائل التشخيص
> *recording diagnostic messages*

سجل — the resulting organized record

سجل is the artifact or body of retained entries:

> سجل الأحداث يحتوي على تسجيلات متعددة.
>
> *An event log contains multiple recorded entries.*

رواية — an account or narrative

A log may also be understood metaphorically as:

> رواية النظام لما حدث
>
> *the system's account of what happened*

But it is not an independent or complete narrative. It is an account constrained by:

- the component's viewpoint;
- the available information;
- instrumentation choices;
- vocabulary;
- timing;
- severity classification.

A log message is therefore not identical to reality. It is a statement produced from one location within reality.

The three Arabic epistemic forms

The resulting Arabic diagnostic triad is:

Artifact	Arabic concept	Diagnostic form	Principal question
Memory dump	حالة	Captured internal configuration	What existed at that moment?
Trace	أثر / مسار	Movement and what it left behind	How did the system arrive there?
Log	سجل	Intentionally preserved account	What did the system record?

In compact form:

> تفريغ الذاكرة يلتقط الحالة.
>
> *A memory dump captures the state.*
>
> التتبّع يكشف المسار والآثار.
>
> *Tracing reveals the path and its traces.*
>
> السجل يحفظ الوقائع المختارة.
>
> *The log preserves selected facts.*

These are not simply three technical formats. They are three relationships to software reality:

- الحالة concerns what existed;
- المسار concerns how execution moved;
- الأثر concerns what that movement left or caused;
- السجل concerns what was intentionally retained and made readable.

الحالة ليست تاريخًا

A dump may contain decisive evidence:

- a corrupted object;
- a blocked thread;
- an invalid pointer;
- a resource leak;
- an inconsistent structure;
- an unfinished update.

Yet it does not ordinarily contain the complete earlier history that produced that condition.

Therefore:

> الحالة تُظهر النتيجة، لكنها لا تُظهر دائمًا الطريق الكامل إليها.
>
> *State shows the result, but not always the complete road leading to it.*

الأثر ليس الحالة كلها

A trace may reveal:

- ordering;
- calls;
- transitions;
- timings;
- dependencies;
- retries;
- propagation across services.

But it normally includes only events and values selected by tracing instrumentation.

It may omit:

- hidden memory values;
- silent corruption;
- uninstrumented branches;
- overwritten state;
- events lost through buffering;
- changes below the tracing layer.

Therefore:

> الأثر يدلّ على الحركة، لكنه لا يحتفظ بكل ما كان موجودًا.
>
> *A trace indicates movement, but does not preserve everything that existed.*

السجل ليس الحقيقة كلها

A log message reports what a component was programmed and able to report.

For example, a message saying فشل الاتصال—"connection failed"—may be:

- accurate but incomplete;
- a secondary symptom;
- based on stale state;
- emitted at the wrong abstraction layer;
- separated from the original cause;
- assigned an inappropriate severity.

Therefore:

> السجل شهادة جزئية، لا صورة كاملة للحقيقة.
>
> *A log is partial testimony, not a complete picture of reality.*

This leads to another important Arabic concept:

> شهادة — *testimony or evidence*

The Arabic Academy[12] defines الشاهد as one who gives testimony. In the diagnostic metaphor, each component behaves like a partial witness: it reports what it could observe from its own position.

A log may therefore be treated as شهادة النظام, the system's testimony—but testimony that must be corroborated.

An Arabic diagnostic cycle

The three evidence forms can be arranged into a method.

قراءة السجل

Read the record

Begin with logs. Their timestamps, identifiers, component names and reported conditions suggest where to investigate.

تتبّع الأثر

Follow the trace

Use tracing to reconstruct:

- request movement;
- function and service interactions;
- transitions;
- retries;
- timing;
- error propagation.

The phrase is especially natural in Arabic:

> تتبّع الأثر
>
> *Follow the trace.*

إعادة بناء المسار

Reconstruct the trajectory

Individual trace events become a meaningful execution path only after they are ordered and related.

فحص الحالة

Inspect the state

Use a dump or snapshot to examine the internal configuration at the significant point identified by trace analysis.

مقابلة الأدلة

Compare the evidence

Determine whether:

- the dump supports the trace interpretation;
- the trace explains the log messages;
- identifiers refer to the same entities;
- timestamps and causal relations agree;
- one artifact contradicts another.

The complete movement is:

> سجل → تتبّع → مسار → حالة → تحقّق
>
> *Record → tracing → path → state → verification*

The reverse movement is equally important:

> حالة → رجوع في الأثر → سجل → إعادة بناء
>
> *State → following traces backwards → record → reconstruction*

The investigator may begin with a crash state and move backwards through the آثار until the original causal region is found.

A forensic Arabic formulation

A more concrete triad is:

> مشهد — أثر — شهادة
>
> *Scene — trace — testimony*

Here:

- the **memory dump is the** مشهد, the preserved scene;
- the **trace is the** أثر, what movement left behind;
- the **log is the** شهادة, the system's partial testimony.

The diagnostic investigator must remember:

- the scene may be incomplete;
- the trace may contain gaps;
- the witness may be mistaken;
- timestamps may disagree;
- evidence may have been overwritten;
- different components may describe one event differently.

Diagnostic confidence emerges through:

> التثبّت — *careful verification*
>
> المقارنة — *comparison*
>
> الترابط — *correlation*
>
> تضافر الأدلة — *convergence of evidence*

No artifact should automatically be treated as the entire truth.

A specifically Arabic linguistic insight

Arabic exposes diagnostic relationships through families of words formed from shared roots.

The family of memory

> ذاكرة — ذكر — تذكّر — ذكرى — مذكّرة

This family connects retention, recollection, remembrance and written reminders.

The family of following

> تبع — اتّبع — تابع — تتبّع

This family connects sequence, following and systematic pursuit.

The family of traces and effects

> أثر — آثار — تأثير — تأثّر

This family connects the mark left behind with the influence or consequence produced.

The family of records

> سجل — سجّل — تسجيل — مسجّل

This family connects the record, the act of recording and the entity that performs or contains the recording.

Together they describe the entire diagnostic process:

> يحتفظ النظام بحالة، ويترك أثرًا، ويسجّل بعض ما يحدث؛ ثم يقرأ المحلّل السجل، ويتتبّع الأثر، ويعيد بناء الحالة والمسار.
>
> *The system retains a state, leaves a trace and records part of what happens; the analyst then reads the record, follows the trace, and reconstructs the state and path.*

Actor	Diagnostic sequence
System	تكون له حالة → يسلك مسارًا → يترك أثرًا → يسجّل أحداثًا
Analyst	يقرأ السجل → يتتبّع الأثر → يعيد بناء المسار → يفحص الحالة → يتحقّق بتضافر الأدلة

Chapter 9 — Sanskrit: अवस्था—पद—अभिलेख

Modern Sanskrit has no single universally standardized vocabulary for every computing artifact. The compounds and formulations in this chapter are therefore conceptual renderings. Their analytical value is substantial: पद connects step and footprint, मार्ग identifies a path, and संस्कार names an impression formed by earlier action that continues to shape present state. The perspective makes internal residue as important as externally emitted events.

Possible Sanskrit-oriented expressions are:

- **Memory dump:** मेमरी-डम्प; conceptually, स्मृत्यवस्थाप्रतिरूपम् — representation of the memory state
- **Trace:** ट्रेस; अनुसरणम्, अनुगमनम्, मार्गः, पदम्, पदचिह्नम्
- **Log:** लॉग; अभिलेखः, घटनाभिलेखः
- **Execution history or account:** वृत्तान्तः

From these terms, a useful diagnostic triad emerges:

> अवस्था — पद — अभिलेख
> *State — trace or footprint — recorded account*

A more philosophically suggestive version is:

> अवस्था — संस्कार — अभिलेख
> *State — residual impression — inscription*

These formulations describe three ways in which a software system becomes knowable:

- a memory dump preserves its अवस्था, its condition at one moment;
- a trace reveals its पद, the steps or marks left by execution;
- a log preserves an अभिलेख, an intentionally recorded account.

Sources for this chapter: [13, 14, 15] in Sources and Further Reading.

1. Memory Dump as अवस्था — the state in which the system stood

The Sanskrit noun अवस्था (*avasthā*) means a state, condition, situation, stage or degree.

Its structure is diagnostically evocative. It is related to the verbal idea of **standing or remaining in a particular condition**:

> यस्याम् अवस्थायां तिष्ठति
> *the condition in which something stands*

A memory dump can therefore be viewed as:

> तत्कालीनावस्था
> *the state belonging to that particular time*

It preserves a selected arrangement of:

- memory contents;
- threads and stacks;
- registers;
- objects and data structures;
- references and handles;
- loaded modules;
- partially completed operations.

The dump asks:

> तस्मिन् क्षणे तन्त्रस्य अवस्था का आसीत्**?**
> *What was the state of the system at that moment?*

Or:

> तदा स्मृतौ किं स्थितम् आसीत्**?**
> *What was situated in memory then?*

स्मृति — memory and what can be recalled

The Sanskrit word स्मृति (*smṛti*) means recollection, remembrance or memory and derives from the verbal root associated with remembering.

Its classical meaning is broader and more human than computer memory. It also names remembered tradition. Nevertheless, as a conceptual metaphor for computing, it suggests that memory is:

> यत् धार्यते, तत् पश्चात् स्मर्यते
> *that which is retained and can later be recalled*

A memory dump moves part of this retained internal condition into an external diagnostic object.

Possible descriptive expressions include:

- स्मृत्यवस्थाचित्रम् — image of the memory state;
- अवस्थाप्रतिरूपम् — representation of state;
- क्षणावस्थाचित्रम् — image of a momentary state;
- तन्त्रावस्थाखण्डः — a cross-section of system state;
- स्थगितस्मृतिः — suspended or frozen memory.

These are analytical coinages rather than established universal technical terms.

The most important idea is:

> डम्पः प्रवाहस्य मध्ये अवस्थां स्थिरीकरोति।
> *A dump stabilizes a state within an ongoing flow.*

Execution is normally changing continuously. The dump separates one configuration from that continuity and makes it inspectable.

2. Trace as अनुसरणम् — the act of following

The Sanskrit term अनुसरणम् (*anusaraṇam*) means following or going after.

This corresponds closely to tracing as an activity:

- following a request;
- following a thread;
- following calls;
- following messages across processes;
- following the propagation of an error;
- following a causal sequence.

Thus:

> निष्पादनस्य अनुसरणम्
> *following the execution*

The analyst is implicitly present in this formulation. Tracing is not only something produced by instrumentation. It is also something performed by an observer.

The investigator asks:

> घटनाक्रमः कथम् अनुसर्तव्यः**?**
> *How should the sequence of events be followed?*

Or:

> दोषस्य मूलं यावत् कथम् अनुगन्तव्यम्**?**
> *How can one follow the fault back to its origin?*

Where the English word *trace* can denote both the activity and its result, Sanskrit can separate them:

- अनुसरणम् **/** अनुगमनम् — the act of following;
- मार्गः — the path followed;
- पदम् **/** पदचिह्नम् — the mark or footprint left behind.

3. Trace as मार्ग — the route taken by execution

The Sanskrit noun मार्गः (*mārgaḥ*) means a path, road, way, track, course or method. It can refer both to a physical route and to a figurative course.

This makes it an appropriate concept for:

- निष्पादनमार्गः — execution path;
- अनुरोधमार्गः — request path;
- सन्देशमार्गः — message path;
- दोषप्रसारमार्गः — fault-propagation path;
- अवस्थापरिवर्तनमार्गः — state-transition path.

A trace therefore reveals more than isolated events. It reveals a route:

> तन्त्रं केन मार्गेण अस्याम् अवस्थायाम् आगतम्?
> *By what path did the system arrive at this state?*

The distinction can be expressed compactly:

> अनुसरणं क्रिया; मार्गः अनुसरणीयः क्रमः।
> *Tracing is the activity; the path is the sequence being followed.*

The log or trace file may contain separate messages. Only analysis turns those messages into a reconstructed मार्ग.

4. Trace as पद — the footprint of execution

The Sanskrit noun पदम् (*padam*) has a remarkably wide semantic range. Among its meanings are a step, footstep, trace, vestige and mark.

This makes पदम् especially powerful for trace analysis.

A trace message may be treated as:

> निष्पादनस्य पदम्
> *a footprint of execution*

A sequence of messages becomes:

> पदक्रमः
> *a sequence of steps or footprints*

And trace analysis becomes:

> पदानुसरणम्
> *following the footprints*

This formulation unifies execution and investigation:

1. the system moves;
2. its movement leaves पदानि, steps or marks;
3. the analyst follows those marks;
4. the path is reconstructed.

Possible related expressions are:

- पदचिह्नम् — footprint or step-mark;
- कार्यपदम् — operational step;
- घटनापदम् — event footprint;
- दोषपदम् — trace of a fault;
- कारणपदानि — causal footprints.

The trace asks:

> कानि पदानि तन्त्रेण त्यक्तानि**?**
> *What footprints were left by the system?*

The word पदम् also means a word or textual unit in Sanskrit grammatical traditions. This creates an additional analogy: a trace consists of discrete diagnostic "words," but their meaning depends on their position within a larger sequence.

A single message is like an isolated पदम्. Its full diagnostic meaning appears only when related to the preceding and following messages.

5. चिह्न — trace as a sign

Another useful Sanskrit concept is चिह्नम् (*cihnam*): a mark, sign or distinguishing indication.

A पदचिह्नम् is literally a foot-mark or footprint. Dictionaries gloss the compound as a footprint or footmark.

This gives trace analysis a semiotic form:

> चिह्नं दृश्यते; तस्य कारणं अनुमीयते।
> *A sign is observed; its cause is inferred.*

A retry message, latency gap or corrupted value is not necessarily the cause of a problem. It is often a चिह्न, a sign pointing beyond itself.

Thus:

- a log entry may be a चिह्न;
- a timing anomaly may be a चिह्न;
- an unexpected transition may be a चिह्न;
- a repeated call may be a चिह्न of an unavailable dependency;
- an absent message may itself become a diagnostic चिह्न.

A trace is therefore not automatically an explanation. It is a field of signs from which explanation must be constructed.

6. संस्कार — the impression left by prior action

The most distinctive Sanskrit contribution may be the concept of संस्कार (*saṃskāra*).

The term has many philosophical, psychological and ritual meanings. In one important philosophical use, it denotes a latent disposition or impression formed through previous action or experience.

Applied metaphorically to software diagnostics:

> पूर्वक्रियया वर्तमानावस्थायां यत् चिह्नं निहितम्, तत् संस्कारवत्।
> *A mark deposited in the present state by earlier action is like a saṃskāra.*

For example:

- an allocation leaves an object in memory;
- repeated allocations leave fragmentation;
- contention leaves accumulated delay;
- retries leave increased queue depth;
- a failed write leaves partially updated state;
- corruption leaves altered data that persists after the corrupting event has ended.

The earlier action may no longer be present, but its impression remains.

This gives trace analysis two kinds of evidence:

बाह्यपदम् — externally visible footprint

A message, event, timestamp or span explicitly emitted by instrumentation.

आन्तरसंस्कारः — internal residual impression

A change left inside state:

- modified memory;
- resource depletion;
- stale cache contents;
- accumulated counters;
- queue growth;
- latent corruption.

The first appears primarily in traces and logs. The second may become visible only in a memory dump.

This produces an important diagnostic relationship:

> क्रिया → संस्कारः → अवस्था
> *Action → residual impression → state*

The present state is not merely a static arrangement. It contains impressions created by earlier execution.

7. Log as अभिलेख — the deliberately inscribed record

The Sanskrit word अभिलेखः (*abhilekhaḥ*) refers to something written or inscribed and is used for records, documents and inscriptions. Modern lexicographic resources also use it for a written record or recording.

A software log may therefore be interpreted as:

> घटनाभिलेखः
> *an event record*

or:

> तन्त्राभिलेखः
> *a system record*

The root image is stronger than merely "keeping data." An event is made externally legible by being inscribed.

A log may preserve:

- named events;
- warnings and errors;
- identifiers;
- durations;
- decisions;
- state changes;
- success or failure outcomes;
- resource measurements.

A log asks:

> तन्त्रेण किं अभिलिखितम्**?**
> *What was recorded by the system?*

The deeper question is:

> तन्त्रं किं अभिलेखितुं निर्मितम् आसीत्**?**
> *What was the system designed to record?*

This distinction matters because the log is shaped by instrumentation.

A system may experience an event without recording it. An entry may be:

- filtered;
- dropped;
- overwritten;
- emitted late;
- incorrectly classified;
- expressed through misleading vocabulary;
- absent because execution failed before logging occurred.

Therefore:

> अनभिलिखितं न अवश्यं अनभूतम्।
> *What was not recorded did not necessarily fail to occur.*

8. लेख, अभिलेख and वृत्तान्त

Sanskrit offers several related concepts for the log.

लेखः — writing or written item

लेखः emphasizes the act or product of writing.

A single log message can be viewed as one लेखांशः, a portion of writing.

अभिलेखः — preserved formal record

अभिलेखः suggests an entered, inscribed or durable record.

This is especially appropriate for an event log or audit log.

वृत्तान्तः — account of what happened

वृत्तान्तः can mean a story, history, occurrence, account, news or course of events.

This term exposes the narrative dimension of logging:

> घटनानां वृत्तान्तः
> *an account of events*

A log is therefore both:

- an अभिलेख, a collection of preserved entries;
- a partial वृत्तान्त, an account of how events unfolded.

But the account is never neutral. It is narrated from the viewpoints of the instrumented components.

The three Sanskrit epistemic forms

The resulting triad is:

Artifact	Sanskrit concept	Diagnostic form	Principal question
Memory dump	अवस्था	Captured internal configuration	What existed then?
Trace	पद / मार्ग / संस्कार	Steps, trajectory and residual impressions	How did the system arrive there?
Log	अभिलेख / वृत्तान्त	Deliberately preserved account	What did the system record?

In compact form:

> डम्पः अवस्थां गृह्णाति।
>
> *A dump captures the state.*
> ट्रेस् निष्पादनस्य पदानि मार्गं च दर्शयति।
>
> *A trace reveals the footprints and path of execution.*
> लॉग् चयनितघटनानाम् अभिलेखं रचयति।
>
> *A log constructs a record of selected events.*

These are not merely three file formats. They represent three relationships to system reality:

- अवस्था concerns what stood or existed;
- मार्ग concerns how execution moved;
- पद concerns the marks left by that movement;
- संस्कार concerns the persistent impressions produced by earlier action;
- अभिलेख concerns what was deliberately preserved;
- वृत्तान्त concerns the account reconstructed from those records.

अवस्था is not मार्ग

A memory dump may expose:

- a corrupted object;
- a blocked thread;
- an invalid reference;
- a resource leak;
- an inconsistent structure;
- an unfinished state transition.

But it does not normally contain the complete path that created that condition.

Therefore:

> अवस्था परिणामं दर्शयति, न तु सर्वं पूर्वमार्गम्।
> *State reveals the result, but not the entire preceding path.*

The dump shows where execution arrived. Trace analysis must often reconstruct how it arrived there.

पद is not complete अवस्था

A trace may show:

- ordering;
- calls;
- transitions;
- timings;
- request propagation;
- retries;
- component interactions.

But traces normally contain only what instrumentation selected.

They may omit:

- hidden memory values;
- uninstrumented branches;
- silent corruption;
- overwritten state;
- dropped events;
- activity below the tracing layer.

Therefore:

> पदानि गतिं सूचयन्ति, किन्तु सम्पूर्णाम् अवस्थां न धारयन्ति।
> *Footprints indicate movement, but do not preserve the complete state.*

अभिलेख is not complete सत्य

A log message expresses what one component was programmed and able to report.

For example, “connection failed” may be:

- accurate but incomplete;
- a secondary symptom;
- based on stale state;
- emitted by the wrong layer;
- separated from the initiating cause;
- assigned an inappropriate severity.

Therefore:

> अभिलेखः साक्ष्यांशः अस्ति, न तु सम्पूर्णं सत्यम्।
> *A log is a portion of evidence, not the whole truth.*

A log should be treated not as reality itself, but as one representation of reality.

A Sanskrit diagnostic cycle

The three forms can be arranged into a method.

अभिलेखपठनम् — reading the records

Begin with logs. Their timestamps, identifiers, event names and descriptions suggest the region of execution that requires investigation.

पदानुसरणम् — following the footprints

Use traces to connect individual events and follow the request, thread, transaction or error.

मार्गपुनर्निर्माणम् — reconstructing the path

Transform separate trace events into an ordered and causally meaningful execution trajectory.

अवस्थापरीक्षणम् — examining the state

Use a dump or snapshot to inspect the internal configuration at a significant point on that trajectory.

प्रमाणसमन्वयः — coordinating the evidence

Determine whether:

- the dump supports the trace interpretation;
- the trace explains the log entries;
- identifiers refer to the same entities;
- timestamps agree;
- one artifact contradicts another.

The complete movement is:

> अभिलेखः → पदानुसरणम् → मार्गः → अवस्था → सत्यापनम्
> *Record → following footprints → path → state → verification*

The reverse movement is equally important:

> अवस्था → संस्कारः → पदानि → अभिलेखः → कारणपुनर्निर्माणम्
> *State → residual impressions → traces → records → reconstruction of cause*

The analyst may begin with a crash state, identify persistent impressions of earlier activity, find corresponding trace events and then compare them with the recorded account.

A forensic Sanskrit formulation

A more concrete triad is:

> घटनास्थलम् — पदचिह्नानि — साक्ष्यलेखः
> *Scene — footprints — written testimony*

Here:

- the **memory dump is the** घटनास्थलम्, the preserved scene;
- the **trace consists of** पदचिह्नानि, the footprints leading through it;
- the **log is a** साक्ष्यलेखः, a written evidential account.

The investigator must remember:

- the scene may be incomplete;
- the footprints may contain gaps;
- the written account may reflect only one viewpoint;
- evidence may have been overwritten;
- different components may name the same event differently.

Diagnostic confidence arises through:

> परस्परप्रमाणीकरणम्
> *mutual corroboration*

No single artifact should automatically be treated as complete or authoritative.

A specifically Sanskrit insight

The Sanskrit perspective introduces a layered causal structure:

> कर्म — संस्कार — अवस्था — स्मृति
> *Action — residual impression — present state — retained memory*

An action occurs. It leaves an impression. The impression helps shape the state. The state may later be retained in memory and captured in a dump.

At the same time:

> गति — पद — मार्ग — वृत्तान्त
> *Movement — footprint — path — reconstructed account*

Execution moves. Its movement leaves steps. The steps reveal a path. Logs and traces allow the path to be narrated.

This yields two complementary diagnostic dimensions.

The internal dimension

> कर्म → संस्कारः → अवस्था

What action did to the system.

The external dimension

> कर्म → पदम् → अभिलेखः

What observable evidence the action left.

These dimensions do not always coincide.

An action may leave:

- an internal impression but no log message;
- a log message but no lasting memory change;
- a trace event that misrepresents the resulting state;
- a memory alteration long after the original trace event has disappeared.

Diagnosis requires relating the internal संस्कार to the external पद.

Chapter 10 — Russian: состояние—след—запись

The Russian perspective brings together state, snapshot, route, trace, record, and investigation. Its most distinctive chain is **след → следовать → следствие**: a mark is left, an analyst follows it, and the result is understood both as a causal consequence and as an object of investigation. This links execution residue to forensic method with unusual economy.

The conventional Russian technical terms are:

- **Memory dump:** дамп памяти, файл дампа
- **Trace:** трассировка, трасса выполнения, сообщения трассировки
- **Log:** лог, журнал, журнал событий
- **Log entry:** запись журнала

Russian technical vocabulary combines English-derived terms—**дамп** and **лог**—with native or assimilated analytical terms such as **снимок**, **состояние**, **трассировка**, **след**, **запись**, and **журнал**.

From these terms, a useful diagnostic triad emerges:

> ***состояние — след — запись***
> *state — trace — record*

A more artifact-oriented version is:

> ***снимок — трасса — журнал***
> *snapshot — trajectory — logbook*

These describe three different ways in which a software system becomes knowable:

- a memory dump captures its **состояние**;
- a trace reveals its **трассу** and the **следы** left by execution;
- a log preserves selected events as **записи** in a **журнале**.

Sources for this chapter: [16–19] in Sources and Further Reading.

1. Memory Dump as состояние — the condition in which the system existed

The established Russian term is **дамп памяти**.

Microsoft's Russian-language documentation[16] describes a dump as a file containing a **моментальный снимок процесса** at the time of capture and explains that it allows the state of a problematic process to be recorded and examined separately.

The central Russian concept is therefore:

> ***состояние системы***
> *the state of the system*

The word **состояние** denotes the position or condition in which something exists. In technical and physical usage, it can also describe the arrangement and interaction of elements within a system.

A memory dump may preserve:

- memory contents;
- threads and stacks;
- registers;
- objects and data structures;
- handles and references;
- loaded modules;
- partially completed operations;
- visible corruption or inconsistency.

A dump primarily answers:

> ***В каком состоянии находилась система в тот момент?***
> *What state was the system in at that moment?*

It may also answer:

> ***Что находилось в памяти?***
> *What was present in memory?*

Снимок — something taken out of continuous time

Russian diagnostic language commonly describes a dump as a **снимок состояния** or **моментальный снимок процесса**. The ordinary word **снимок** denotes an image obtained by capturing something, especially photographically.

This gives the dump a strong temporal meaning:

> ***снимок вырезает один момент из непрерывного выполнения***
> *a snapshot extracts one moment from continuous execution*

The usual Russian expression **снять дамп** is especially evocative.

The verb **снять** can mean to take a photograph or capture an image. Thus, although **дамп** is a borrowed technical noun, its ordinary Russian verb places it within the conceptual world of photography and observation:

> ***снять дамп процесса***
> *take a snapshot of the process*

The system normally exists as an uninterrupted flow of changing states. The dump arrests one configuration and makes it inspectable.

Possible conceptual descriptions include:

- **снимок памяти** — memory snapshot;
- **срез состояния** — cross-section of state;
- **мгновенное состояние** — instantaneous state;
- **зафиксированное состояние** — fixed or captured state;
- **цифровое место происшествия** — digital scene of the incident.

A dump is therefore mainly **spatial and configurational**. It shows which entities coexisted and how they were related at one point.

Память — storage and remembrance

The Russian word **память** has both human and computational meanings.

In ordinary usage, it means the ability to retain and reproduce earlier impressions or information. In computing, dictionaries define it as the part of a computer used to place, store, and transfer information in digital form.

This creates a productive diagnostic analogy:

> ***Дамп извлекает то, что система удерживала в памяти.***
> *A dump extracts what the system retained in memory.*

Computer memory is not human remembrance, but both meanings involve:

- retention;
- preservation;
- later retrieval;
- the possibility of loss;
- the reconstruction of something no longer directly present.

A memory dump may therefore be interpreted as:

> ***внешняя память о внутреннем состоянии системы***
> *an external memory of the system's internal state*

Once execution has continued, terminated, or crashed, the original configuration no longer exists as a live state. The dump remains as its preserved image.

2. Trace as трассировка — the act of revealing a route

The ordinary Russian technical term for tracing is **трассировка**.

Official Windows documentation[17] uses expressions such as:

- **сеанс трассировки** — trace session;
- **сообщение трассировки** — trace message;
- **журнал трассировки** — trace log;
- **поставщик трассировки** — trace provider.

A trace log stores messages generated during one or more tracing sessions. Those messages may first be held in trace buffers and then delivered to a consumer or written to the trace log.

Thus, **трассировка** describes a process:

> ***делать ход выполнения наблюдаемым***
> *make the course of execution observable*

One may trace:

- a request;
- a thread;
- a function call;
- a message;
- a transaction;
- an error;
- a state transition;
- a causal chain.

The trace asks:

> ***Как выполнялась система?***
> *How did the system execute?*

But the Russian vocabulary allows this question to be divided more precisely.

Трасса — the path or line of movement

The noun **трасса** means a route, road, path, or a line indicating the direction in which something proceeds. It may also denote a visible trail left by a moving projectile.

This makes it a useful metaphor for execution:

- **трасса выполнения** — execution trajectory;
- **трасса запроса** — request trajectory;
- **трасса транзакции** — transaction trajectory;
- **трасса распространения ошибки** — error-propagation trajectory;
- **трасса переходов состояния** — state-transition trajectory.

The distinction is:

> ***Трассировка — это действие; трасса — путь, который обнаруживается.***
> *Tracing is the activity; the trace route is the path that becomes visible.*

A raw trace file may contain separate events. Analysis turns those events into a meaningful route.

A trace therefore primarily answers:

> ***По какой трассе система пришла к наблюдаемому состоянию?***
> *Along what path did the system arrive at the observed state?*

След — the mark remaining after execution

The most important Russian word for the conceptual analysis of tracing is **след**.

Russian dictionaries give it several closely related meanings:

- a footprint or impression;
- a line left behind by a moving object;
- a scratch, mark, or sign remaining after something;
- the result or consequences of an event;
- an indication that testifies to something;
- a remnant of something that previously existed.

This is almost a complete vocabulary of software diagnostics.

> ***След — это то, что выполнение оставляет после себя.***
> *A trace is what execution leaves behind.*

Examples include:

- a message left after a failed operation;
- a latency interval left by contention;
- repeated calls left by retry behaviour;
- an identifier propagated through multiple services;
- a partially completed sequence left by termination;
- an altered object left by memory corruption;
- an increased counter left by resource pressure;
- the absence of an expected message.

A trace message is therefore not merely a record of motion. It is a **след выполнения**—a residual sign of execution.

Trace as mark, remnant, and consequence

The Russian word **след** is broader than a footprint.

It can also mean the **result or aftermath** of an event:

- **следы сбоя** — traces or consequences of a failure;
- **следы перегрузки** — traces of overload;
- **следы повреждения** — traces of corruption;
- **следы предыдущего выполнения** — remnants of previous execution.

This produces two forms of diagnostic trace.

Наблюдаемый след — observable trace

An explicitly emitted event:

- trace message;
- span;
- timestamp;
- log entry;
- counter update.

Остаточный след — residual trace

A persistent change remaining in the system:

- modified memory;
- leaked resources;
- increased queue depth;
- stale cache contents;
- fragmented storage;
- corrupted metadata;
- an incomplete transaction.

The first is mainly visible in traces and logs. The second may be visible only in a dump or snapshot.

This yields a causal chain:

> ***действие → след → состояние***
> *action → residual trace → state*

The present state contains traces of earlier activity even when the original event is no longer directly observable.

Следить, отследить, проследить

Russian also separates different modes of following evidence.

Следить — observe continuously

Следить can mean watching something that moves or changes, as well as exercising ongoing observation or control.

In diagnostics, this corresponds closely to monitoring:

> ***следить за состоянием системы***
> *monitor the state of the system*

Отследить — follow a process through its development

Отследить can mean observing the course or development of a process or phenomenon.

This is appropriate for:

> ***отследить распространение ошибки***
> *track the propagation of an error*

Проследить — reconstruct or follow from one point to another

Проследить suggests following an object, movement, or sequence through its course.

This corresponds to analytical reconstruction:

> ***проследить путь запроса***
> *trace the path of a request*

The sequence can therefore be expressed as:

> ***система оставляет след; трассировка его фиксирует; аналитик его прослеживает***
> *the system leaves a trace; tracing captures it; the analyst follows it*

Следствие — consequence and investigation

One of the most distinctive Russian linguistic contributions is the word **следствие**.

It has two major meanings:

1. something that follows from something else—a result, consequence, or conclusion;
2. an investigation that establishes and verifies the circumstances of an incident.

This dual meaning is almost perfectly adapted to diagnostic work.

A software failure may be:

> ***следствие более раннего события***
> *a consequence of an earlier event*

At the same time, analysing it requires:

> ***провести следствие***
> *conduct an investigation*

Thus, the Russian language places **causal consequence** and **forensic inquiry** inside the same word.

The diagnostic process becomes:

> ***найти следствие и провести следствие***
> *identify the consequence and investigate it*

Or more precisely:

> ***по следам восстановить причину следствия***
> *use the traces to reconstruct the cause of the consequence*

This creates a strong connection between:

- trace;
- causality;
- consequence;
- investigation;
- proof.

3. Log as запись — something deliberately fixed

The borrowed technical word **лог** is common in Russian, but its main conceptual equivalent is **запись**.

A dictionary defines **запись** both as the process of fixing information on a medium and as the resulting item that has been recorded or officially registered.

This is important because a log entry does not simply remain accidentally.

It is:

> ***событие, выбранное для фиксации***
> *an event selected for recording*

A system may record:

- named events;
- warnings and errors;
- identifiers;
- decisions;
- timings;
- state changes;
- resource measurements;
- security operations;
- successful or failed outcomes.

A log primarily answers:

> ***Что система записала?***
> *What did the system record?*

The deeper diagnostic question is:

> ***Что систему спроектировали записывать?***
> *What was the system designed to record?*

This distinction matters because the absence of a record does not prove the absence of an event.

An event may have been:

- uninstrumented;
- filtered;
- dropped;
- overwritten;
- emitted too late;
- lost during failure;
- described using unexpected terminology.

Therefore:

> ***То, что не записано, не обязательно не происходило.***
> *What was not recorded did not necessarily fail to occur.*

Журнал — an ordered accumulation of records

The more traditional Russian equivalent of a software log is **журнал**.

A journal is defined as a book or notebook used for periodic recording of events, decisions, operations, or observations.

This gives it several important properties:

- entries are accumulated;
- entries usually have an order;
- the journal is maintained over time;
- it has an operational purpose;
- its contents are meant to be consulted later.

Common technical expressions include:

- **журнал событий** — event log;
- **системный журнал** — system log;
- **журнал ошибок** — error log;
- **журнал аудита** — audit log;
- **журнал трассировки** — trace log;
- **запись журнала** — log entry.

The relationship is:

> ***Запись — отдельный зафиксированный факт; журнал — организованная последовательность записей.***
> *A record is one fixed fact; a journal is an organized sequence of records.*

A log is therefore both:

- a collection of individual statements;
- an evolving chronological account.

Протокол — formalized record of proceedings

Russian technical language also uses **протоколирование** for systematic logging.

Outside computing, a **протокол** is a document containing a concise or structured record of what occurred during a meeting, hearing, examination, or other proceeding.

As a diagnostic metaphor, this is useful:

> ***лог — это протокол работы системы***
> *a log is a record of the system's proceedings*

However, the system's protocol is selective. It contains only what its instrumentation was instructed and able to enter.

A log can therefore be viewed as:

- **запись** — an individual fact;
- **журнал** — an accumulated operational record;
- **протокол** — a formalized account of proceedings;
- **рассказ системы** — the system's partial narrative.

The three Russian epistemic forms

The resulting diagnostic triad is:

Artifact	Russian concept	Diagnostic form	Principal question
Memory dump	**состояние / снимок**	Captured internal configuration	What existed at that moment?
Trace	**трасса / след**	Movement and its remaining marks	How did the system arrive there?
Log	**запись / журнал**	Intentionally preserved account	What did the system record?

In compact form:

> ***Дамп фиксирует состояние.***
> *A dump captures state.*
> ***Трассировка раскрывает трассу и оставленные следы.***
> *Tracing reveals the route and the traces left behind.*
> ***Журнал сохраняет выбранные записи.***
> *A log preserves selected records.*

These are not merely three file formats. They represent three different relationships to software reality:

- **состояние** concerns what existed;
- **трасса** concerns how execution moved;
- **след** concerns what that movement left behind;
- **запись** concerns what was deliberately fixed;
- **журнал** concerns how records accumulated through time.

Состояние is not предыстория

A memory dump may expose decisive evidence:

- a corrupted object;
- a deadlocked thread;
- an invalid pointer;
- a leaked resource;
- an inconsistent structure;
- an unfinished update.

But it does not ordinarily contain the complete history that produced the state.

Therefore:

> ***Состояние показывает результат, но не обязательно всю предысторию.***
> *State shows the result, but not necessarily the complete history behind it.*

The dump tells us where execution arrived. Traces and logs are needed to reconstruct the road to that point.

След is not complete состояние

A trace may reveal:

- event ordering;
- calls and returns;
- transitions;
- timing;
- request propagation;
- retries;
- component interaction.

But it normally contains only the events and values selected by instrumentation.

It may omit:

- hidden memory values;
- silent corruption;
- uninstrumented branches;
- overwritten state;
- events lost from buffers;
- activity below the tracing layer.

Therefore:

> ***След указывает на движение, но не сохраняет всё состояние.***
> *A trace indicates movement but does not preserve the whole state.*

Запись is not complete truth

A log entry reports what one component was programmed and able to report.

For example, the message:

> ***Ошибка соединения***
> *Connection error*

may be:

- accurate but incomplete;
- a secondary symptom;
- based on stale information;
- emitted by the wrong abstraction layer;
- separated from the original cause;
- incorrectly classified.

Therefore:

> ***Запись — это свидетельство, а не сама действительность.***
> *A record is testimony, not reality itself.*

Or:

> ***Записанное событие и произошедшее событие не всегда совпадают.***
> *The recorded event and the event that actually occurred are not always identical.*

A Russian diagnostic cycle

The three evidence forms can be arranged into a diagnostic method.

Читать журнал

Read the log

Begin with timestamps, identifiers, component names, severity levels, and reported conditions.

Искать следы

Look for traces

Identify messages, timing anomalies, repeated operations, omissions, and state changes that indicate a path through execution.

Прослеживать трассу

Follow the trajectory

Connect events across:

- threads;
- processes;
- services;
- requests;
- transactions;
- causal transitions.

Исследовать состояние

Inspect the state

Use a memory dump or snapshot to examine the internal configuration at the relevant point.

Сопоставлять свидетельства

Correlate the evidence

Determine whether:

- the dump supports the trace interpretation;
- the trace explains the log messages;
- identifiers refer to the same entities;
- timestamps agree;
- one artifact contradicts another.

The complete movement is:

> ***журнал → след → трасса → состояние → проверка***
> *log → trace → trajectory → state → verification*

The reverse movement is equally important:

> ***состояние → остаточные следы → трасса → записи → реконструкция причины***
> *state → residual traces → trajectory → records → reconstruction of cause*

A forensic Russian formulation

A more concrete triad is:

> ***место происшествия — следы — протокол***
> *scene of the incident — traces — formal record*

Here:

- the **memory dump is the место происшествия**, the preserved scene;
- the **trace consists of следы**, the marks leading through and towards it;
- the **log is the протокол**, the system's recorded account.

The investigator must remember:

- the scene may be incomplete;
- the traces may contain gaps;
- the protocol may omit important events;
- timestamps may disagree;
- evidence may have been overwritten;
- different components may describe the same event differently.

Diagnostic confidence emerges through:

> ***сопоставление*** *— comparison*
> ***проверка*** *— verification*
> ***подтверждение*** *— corroboration*
> ***совокупность свидетельств*** *— convergence of evidence*

A trace may become an **улика** only after its relevance and relation to the event have been established. In ordinary Russian, an улика is an object or circumstance that testifies to culpability or involvement.

In software diagnostics, the equivalent is not proof of guilt but evidence supporting a causal explanation.

A specifically Russian linguistic insight

The Russian perspective reveals two parallel diagnostic sequences.

The system sequence

> ***состояние → действие → следствие → след***

The system has a state. An action occurs. It produces a consequence. That consequence leaves observable or residual traces.

The analyst sequence

> ***запись → след → трасса → следствие***

The analyst reads a record, finds traces, reconstructs the trajectory, and conducts an investigation.

The word **следствие** stands at the intersection:

- it is what happened as a consequence;
- it is also the inquiry into why it happened.

This produces a compact diagnostic principle:

> ***Всякое следствие оставляет следы, а всякое расследование начинается со следов.***
> *Every consequence leaves traces, and every investigation begins with traces.*

Chapter 11 — German: Zustand—Spur—Protokoll

The German perspective is built around **Zustand—Spur—Protokoll**. **Zustand** names the way something exists at a particular moment; **Spur** joins footprint, persistent alteration, and investigative lead; **Protokoll** is both an account of proceedings and, in computing, a recording of operations. The triad makes diagnosis a movement from configured condition through material indications to a deliberately maintained procedural record.

Sources for this chapter: [20–24] in Sources and Further Reading.

Conventional technical vocabulary

German technical writing readily combines localized terms with the English loan *Dump* and *Trace*. The analytical formulation in this chapter does not attempt to displace that usage. It uses the established vocabulary to expose three different relationships to evidence.

Artifact or activity	Common German expression	Analytical emphasis
Memory dump	**Speicherabbild**, **Dump**, **Dumpdatei**	A captured image of process memory and application condition
Tracing	**Ablaufverfolgung**, **Tracing**	The activity of following execution
Trace	**Spur**, **Ablaufspur**, **Trace**	The resulting sequence, indication, or residue
Log	**Protokoll**, **Protokolldatei**, **Aufzeichnung**	A deliberately maintained record of operations

Official German technical documentation[20] describes a dump as a file containing a **Momentaufnahme** of a process and uses it to investigate the application's **Zustand**. That pairing is analytically precise. The artifact is an image; its object is a condition.

1. Memory dump as Zustand and Speicherabbild

Zustand names the way in which a person or thing exists at a particular moment: its condition, configuration, or presently obtaining situation. In diagnostics, the word directs attention away from isolated values and toward the relations that make those values meaningful.

A **Speicherabbild** is therefore not simply a mass of copied memory. **Abbild** presents it as an image or representation. The image may contain threads, stacks, registers, objects, handles, queues, modules, and partially completed work, but those elements become evidence only when reconstructed as a **Systemzustand**.

The distinction suggests three questions:

- What portion of the system was made visible in the image?
- Which relations among the visible entities define the relevant condition?
- What history is absent even though its consequences remain in the captured state?

German compounds make it easy to specialize the idea: **Speicherzustand**, **Prozesszustand**, **Systemzustand**, and **Ausgangszustand** distinguish memory, process, system, and initial condition. This compositional clarity helps prevent a common analytical error. A dump does not preserve “the system” without qualification. It preserves a selected representation of a condition under a particular capture policy.

2. Tracing as Ablaufverfolgung, trace as Spur

Ablaufverfolgung is structurally revealing. **Ablauf** is a course, sequence, or process; **Verfolgung** is following or pursuit. The term therefore emphasizes tracing as an activity performed over execution. Instrumentation follows selected operations and emits observable events.

Spur shifts the focus from activity to what can be followed. It can name a line of footprints, a visible alteration testifying to an earlier process, a technical track, or an investigative lead. A **Spur** is not the past event itself. It is a present indication constrained by what the past left behind.

That distinction creates four analytical layers:

- **Verfolgung** — the mechanism or activity of observing execution;
- **Ablauf** — the process or course that actually occurred;
- **Spur** — the marks or indications available afterward;
- **Rekonstruktion** — the analyst's ordered explanation of the course.

The same vocabulary naturally supports forensic caution. A Spur may be fresh or old, complete or interrupted, relevant or misleading. It can be secured, erased, overlooked, or placed by a secondary effect. Diagnostic traces have the same vulnerabilities: sampling, buffering, clock skew, correlation error, schema changes, and missing providers can break the apparent route.

3. Log as Protokoll and Aufzeichnung

Protokoll has a particularly useful double life. It can mean a written account of proceedings or results, and in computing it can mean a recording of operations. This is distinct from the other familiar technical sense

of a communication protocol. The same surface word can refer either to rules governing interaction or to the record produced while interaction occurs; the context must make the difference explicit.

As an evidential concept, **Protokoll** foregrounds procedure and selection. A protocol or record is maintained according to a purpose. It identifies what counts as an entry, which vocabulary is legitimate, who or what is speaking, and how events are ordered. **Aufzeichnung** foregrounds the act and product of recording, while **Protokolldatei** foregrounds the stored container.

A log should therefore be read as an **instrumentierter Bericht**: an instrumented account. It may be accurate without being causally complete. A component can correctly record a timeout while remaining unable to observe the queue growth, lock contention, retry amplification, or corrupted state that produced it.

4. The German diagnostic grammar

The German formulation can be summarized as follows.

The system:

1. **befindet sich in einem Zustand** — exists in a condition;
2. **durchläuft einen Ablauf** — passes through a course of execution;
3. **hinterlässt Spuren** — leaves traces;
4. **erzeugt Aufzeichnungen** — produces records.

The analyst:

1. **liest das Protokoll** — reads the record;
2. **verfolgt die Spuren** — follows the traces;
3. **rekonstruiert den Ablauf** — reconstructs the execution course;
4. **untersucht den Zustand** — examines the captured condition;
5. **gleicht die Belege ab** — reconciles the evidence.

> *German distinguishes the condition in which the system stood, the indications its execution left behind, and the procedural account that instrumentation chose to maintain.*

The compact triad is **Zustand—Spur—Protokoll**. Its special contribution is procedural discipline. The state is configured, the trace is evidentially followed, and the log is recognized as a maintained record rather than an unmediated transcript of reality.

Chapter 12 — Irish: staid—rian—taifead

The Irish perspective uses **staid—rian—taifead** while retaining the conventional technical forms **dumpáil cuimhne**, **rianú**, and **loga**. **Rian** and **lorg** can name a trace, trail, or signs left behind, while **rianú** names the activity of tracing. **Taifead** names stored information and a record, including a computing record. The vocabulary separates action, remnant, and deliberate preservation with particular clarity.

Sources for this chapter: [25–30] in Sources and Further Reading.

Conventional technical vocabulary

Irish provides both localized technical forms and a rich native vocabulary for state, following, trace, trail, and record. The national English-Irish dictionary gives **dumpáil cuimhne** for *memory dump*, **rianú** for tracing, **loga an chórais** for *system log*, and **taifead** for a computing record. These forms support the analytical triad **staid—rian—taifead**.

Artifact or activity	Irish expression	Analytical emphasis
Memory dump	**dumpáil cuimhne**	A dumping or capture of computer memory
State	**staid**, also **riocht** or **bail** by context	Machine setting, condition, or present configuration
Tracing	**rianú**, verb **rianaigh**	The activity of tracing, locating, or following development
Trace or trail	**rian**, **lorg**	A remaining mark, track, scent, indication, or evidential trail
Record or log	**taifead**, **loga**, **cuntas**	Stored information, an operational log, or an account

The alternatives matter. Irish does not force every analytical relation into one noun. **Rianú** names an activity; **rian** and **lorg** name what remains or can be followed; **taifead** names what has been placed on record; **cuntas** can emphasize an account.

1. Memory dump as staid captured in cuimhne

Staid can refer to circumstances or condition and is specifically available for a machine setting. That makes it a direct analytical lens for a memory dump. **Cuimhne** names both human memory and computer data storage, while **dumpáil cuimhne** names the technical act or artifact of dumping memory.

The semantic conjunction is productive without implying that computer memory works like autobiographical recollection. A process dump is a capture from **cuimhne** whose value lies in the **staid** it permits an analyst to reconstruct. The dump preserves coexistence: which threads were waiting, which

objects were reachable, which buffers were full, which references connected structures, and which operation remained incomplete.

Other Irish words refine the condition. **Riocht** can emphasize the state or shape in which something is found; **bail** can emphasize the condition something is in. The analyst can therefore distinguish abstract machine state from the concrete condition of an artifact at capture time.

The diagnostic questions are:

- Cén staid ina raibh an córas? — In what state was the system?
- Cad a bhí sa chuimhne ag an nóiméad gabhála? — What was in memory at the moment of capture?
- Cén stair nach bhfuil sa dumpáil ach a bhfuil a lorg fós sa staid? — What history is absent from the dump but still leaves a trail in the state?

2. Tracing as rianú; trace as rian and lorg

The verb **rianaigh** and verbal noun **rianú** cover acts of tracing, locating an origin, describing development, and following a line. This is the active side of diagnostics: instrumentation follows a call, request, message, identifier, or transition.

The nouns **rian** and **lorg** name the available remainder. They can refer to marks, remains, a line or pattern, a scent, a trail, or related evidence. Expressions such as **lorg a fhágáil ina dhiaidh** make the temporal relation explicit: something passes and leaves a trail behind it.

This produces a useful separation:

- **rianú** — the act or mechanism of tracing;
- **cosán** or **conair** — the route or path reconstructed;
- **rian** — a trace, line, or remaining indication;
- **lorg** — a track or evidential trail that can be followed;
- **fianaise** — evidence after relevance and support have been established.

The last distinction is important. A raw trace event is not automatically **fianaise** in the strong sense. It becomes evidence through identity, integrity, context, and corroboration. A present **rian** may identify a route; an absent expected rian may identify a gap in instrumentation or an operation that never completed.

3. Log as taifead, loga, and cuntas

Taifead is especially well suited to diagnostic reasoning because it means stored information and is used for a record in computing. The verb **taifead** names the act of recording. **Loga** is the conventional log or logbook term, including **loga an chórais**, while **cuntas** can present the log as an account of activities.

These terms distinguish container, entry, and narrative function. A log file can be a **loga**; an individual stored item is a **taifead**; the larger interpretation offered by those entries is a **cuntas**. None is identical with the event itself.

The distinction exposes three sources of omission:

- the event was never selected for recording;
- it was selected but the record was filtered, delayed, dropped, or overwritten;
- the record exists but its account is interpreted at the wrong abstraction layer.

Irish therefore reinforces the idea that logging is an act of deliberate preservation. **Taifead a dhéanamh** is to make a record. The system does not merely shed logs; instrumentation makes choices about what will be kept.

4. The Irish diagnostic grammar

The system:

1. **tá sé i staid áirithe** — it is in a particular state;
2. **téann sé trí chonair** — it moves through a path;
3. **fágann sé rian nó lorg ina dhiaidh** — it leaves a trace or trail behind;
4. **déanann sé taifid** — it makes records.

The analyst:

1. **léann sé an loga** — reads the log;
2. **leanann sé an lorg** — follows the trail;
3. **rianaíonn sé an t-imeacht** — traces the event or development;
4. **atógann sé an chonair** — reconstructs the path;
5. **scrúdaíonn sé staid an chórais** — examines the system state.

> *Irish makes the diagnostic movement concrete: an action passes, a trail remains behind it, and selected information is deliberately placed on record.*

The compact triad is **staid—rian—taifead**. Its special contribution is the separation of tracing as activity, trail as remainder, and record as stored information. The vocabulary also keeps **fianaise**, established evidence, conceptually beyond the raw mark from which an investigation begins.

Chapter 13 — French: état—trace—journal

The French perspective gives the compact formulation **état—trace—journal**. A **vidage** captures the application's **état**; **traçage** produces events from which a **trace** and trajectory are reconstructed; a **journal** accumulates entries through time. French also makes **empreinte** available for the impression an action leaves on later state, strengthening the bridge between execution and residue.

Sources for this chapter: [31–35] in Sources and Further Reading.

Conventional technical vocabulary

French technical documentation uses **vidage** or **dump** for a dump, **traçage** for the activity of tracing, **trace** for the resulting mark or sequence, and **journal** for a log. Official .NET documentation[31] describes a dump as a snapshot useful for examining the application's **état**. Windows documentation[32] uses **journal de trace** for a trace log.

Artifact or activity	French expression	Analytical emphasis
Memory dump	**vidage**, **vidage mémoire**, **fichier de vidage**	Captured process memory and application state
Tracing	**traçage**, **suivi**	Instrumented following or monitoring
Trace	**trace**, **piste**, **empreinte**	Mark, trail, or impression left by prior activity
Log	**journal**, **fichier journal**, **journalisation**	Chronological accumulation of selected entries
Record	**enregistrement**, **entrée**	An individual stored item or act of recording

The analytical formulation is **état—trace—journal**. Each term names a different temporal relation: present configuration, surviving indication of prior activity, and a record accumulated through time.

1. Memory dump as état and instantané

État names a way of being that may be stable or subject to variation. It is conventionally contrasted with action or movement. That opposition is exactly what makes the term valuable for dump analysis. A dump arrests action and exposes a configuration.

The frequent explanatory word **instantané** adds a necessary qualification. The state belongs to a moment of capture. It may reveal the result of a long process without showing the process itself. A deadlock dump can expose a cycle of waits; it does not necessarily preserve the complete series of acquisitions and releases that produced the cycle.

French supports several diagnostic distinctions:

- **état initial** and **état final** distinguish points in a transition;
- **état courant** names the configuration presently observed;
- **état de fait** emphasizes an actually obtaining situation;
- **état des lieux** evokes a structured inventory of what is present.

The last phrase is a productive metaphor for a dump. An **état des lieux** is not a history of every action in a place. It is a disciplined inventory of condition at a defined time. Dump analysis similarly reconstructs a local world from objects, owners, waiters, references, counters, and incomplete work.

2. Traçage, trace, piste, and empreinte

Traçage is the activity or mechanism of tracing. **Trace** is the mark by which earlier existence or action can be recognized. It may be a physical mark, a remnant, a very small quantity, or figuratively an indication or proof. This semantic range fits observability unusually well: a trace event is often a small, selected remainder from a much larger execution.

Piste adds the idea of a trail to be followed during an inquiry. **Empreinte** adds the idea of an impression left on a receiving medium. The three terms can therefore distribute diagnostic work:

- **traçage** — instrumentation that observes selected activity;
- **trace** — the surviving mark or event;
- **piste** — the investigative route suggested by connected traces;
- **empreinte** — the persistent effect impressed on later state;
- **preuve** — evidential standing after verification.

An execution can leave traces in a telemetry store and an empreinte in memory, queues, caches, or resource ownership. The telemetry may expire while the stateful impression remains. Conversely, a trace may be recorded even when the operation leaves no durable state change.

French also encourages a distinction between **tracé** and **trace**. A **tracé** is a plotted line or resulting layout; in diagnostics it can metaphorically represent the reconstructed course. A **trace** is the indication from which that course is inferred. The plotted route should not be confused with the raw marks that constrain it.

3. Log as journal and journalisation

Journal is naturally chronological. It presents a sequence of entries maintained from day to day or operation to operation. In computing, **journalisation** names the process of generating or maintaining those entries, while **entrée de journal** identifies the individual record.

This vocabulary separates three layers:

- the logging policy and activity, **journalisation**;
- the individual recorded statement, **entrée** or **enregistrement**;
- the accumulated operational record, **journal**.

A journal is valuable precisely because it persists and orders selected statements. Its limitation is also built into that design: only what the instrumentation was prepared to formulate can enter the journal. Severity, vocabulary, timestamps, field selection, correlation, and retention are editorial decisions embedded in software.

The journal should therefore be read as a **récit instrumenté**, an instrumented account. It can report the system's local view truthfully while failing to describe remote causes, hidden state, uninstrumented branches, or consequences below its abstraction layer.

4. The French diagnostic grammar

The system:

1. **se trouve dans un état** — exists in a state;
2. **suit un parcours d'exécution** — follows an execution course;
3. **laisse des traces et des empreintes** — leaves traces and impressions;
4. **inscrit des événements dans un journal** — records events in a log.

The analyst:

1. **lit le journal** — reads the accumulated record;
2. **suit la piste** — follows the investigative trail;
3. **reconstruit le tracé** — reconstructs the course;
4. **examine l'état capturé** — examines captured state;
5. **confronte les indices** — compares and challenges the indications.

> *French distinguishes the state that has been captured, the mark from which prior activity is inferred, the course reconstructed from those marks, and the journal that accumulates the system's selected account.*

The compact triad is **état—trace—journal**. Its special contribution is the temporal discipline created by **instantané**, **trace**, **empreinte**, and **journal**: a moment is captured, the past remains in marks and impressions, and selected statements accumulate into a chronology.

Chapter 14 — Classical Greek: κατάστασις—ἴχνος—ὑπόμνημα

Classical Greek has no inherited computing terminology for dumps, traces, and logs. This chapter therefore offers an explicitly conceptual formulation: **κατάστασις—ἴχνος—ὑπόμνημα**. **Κατάστασις** can mean an established or settled condition; **ἴχνος** is a footprint or track that can be followed; **ὑπόμνημα** is a reminder, note, memorandum, or record prepared for later use. The triad foregrounds stabilization, inferential following, and externalized memory.

Sources for this chapter: [36–39] in Sources and Further Reading.

Scope and status of the vocabulary

Classical Greek did not contain a historical technical vocabulary for computer memory dumps, distributed tracing, or software logs. The terms in this chapter are therefore conceptual mappings grounded in attested lexical meanings, not claims about standardized modern Greek localization. The proposed triad is **κατάστασις—ἴχνος—ὑπόμνημα**: settled condition, footprint, and memorandum.

Diagnostic relation	Classical Greek lens	Analytical emphasis
Captured state	**κατάστασις**	Establishment, settled condition, fixedness, or configuration
Act of tracking	**ἀνίχνευσις / ἀνιχνεύειν**	Searching out or following by tracks
Trace	**ἴχνος**	Footstep, footprint, track, or line of conduct
Sign	**σημεῖον**	A mark or sign interpreted as indicating something
Record	**ὑπόμνημα**	Reminder, note, memorandum, commentary, or record for later use
Registration	**ἀναγραφή**	Inscription, entry, list, or formal registration

The chapter keeps the Greek words visible because their internal relations matter. It does not recommend them as direct replacements for current product terminology.

1. Captured state as κατάστασις

Κατάστασις has a dynamic history inside a noun of condition. It can refer to establishment or institution, the act of setting something in order, restoration or calming, and the resulting settled condition, fixedness, state, or system. This makes it more than a static label.

Applied analytically, a memory dump presents a **κατάστασις** because it shows a configuration that has come to stand in a certain way. The captured state is the outcome of earlier transitions. A blocked thread,

reachable object, open handle, full queue, or corrupted structure is not merely present; it has been established by prior action.

This double aspect corrects two opposing errors. The state is not timeless: it was produced. Yet it is not merely another event: it is the configuration in which many effects coexist at capture time.

The dump can therefore be approached through three questions:

- What has been set or arranged into the present configuration?
- Which relations are settled enough to be observed together?
- Which prior motions are known only through what they established?

The relation between establishment and condition is especially useful for persistent failures. A transient action can establish a durable bad state: an ownership cycle, leaked resource, poisoned cache, inconsistent index, or altered agent context. The event ends; the **κατάστασις** remains.

2. Trace as ἴχνος and investigation as following

Ἴχνος is a footprint, footstep, or track. Its force lies in indirect access. The walker is gone, but a mark remains from which motion can be inferred. The plural **ἴχνη** readily evokes a series of marks that constrains a route.

The related activity of tracking can be expressed through **ἀνιχνεύειν**, to track out or search by traces. This separates the mark from the investigator's action. A system leaves **ἴχνη**; an analyst searches and follows them. The route itself may be expressed through **ὁδός** or **πορεία**, path or course.

This yields four layers:

- **ἴχνος** — the individual footprint or remaining mark;
- **ἴχνη** — the available sequence or field of traces;
- **πορεία** — the course reconstructed through them;
- **ἀνίχνευσις** — the activity of tracking and investigation.

Σημεῖον adds a semiotic caution. A mark becomes a sign only in an interpretive relation: it signifies something to an observer under a model. A timestamp is not causality; a repeated identifier is not identity without scope; an absent event is not proof of absence without knowledge of the instrumentation.

The footprint metaphor also accommodates loss. Tracks may be erased, overlap, lead to the wrong object, or survive after the agent has disappeared. Diagnostic traces may likewise be sampled, reordered, duplicated, miscorrelated, or retained after the state they once described has changed.

3. Log as ὑπόμνημα and ἀναγραφή

Ὑπόμνημα can mean a reminder, note, memorandum, commentary, draft, or public record. Its central diagnostic value is externalized memory. Something is written down so that it may be recalled, consulted, or developed later. A software log serves exactly this broad function: it preserves selected statements for future readers who were not present at execution time.

The term also helps distinguish a raw note from a completed history. A collection of **ὑπομνήματα** is material for reconstruction, not necessarily the reconstruction itself. The entries may preserve observations, decisions, and warnings without explaining their causal relations.

Ἀναγραφή provides a complementary lens: inscription, entry on a list, or formal registration. It foregrounds the act by which something is admitted to an official record. In software, logging policy plays the same gatekeeping role. It decides which events receive names, fields, severities, and durable storage.

Together the terms distinguish:

- what is written as a reminder or working note;
- what is formally entered or registered;
- what is later organized into an explanatory account;
- what gains evidential status through verification.

4. The Classical Greek diagnostic grammar

The system can be represented as:

1. a **κατάστασις** is established;
2. execution proceeds along a **πορεία**;
3. the movement leaves **ἴχνη** and produces **σημεῖα**;
4. selected observations are preserved as **ὑπομνήματα** or **ἀναγραφαί**.

The analyst:

1. reads the memoranda;
2. searches out the footprints;
3. reconstructs the course;
4. inspects the settled condition;
5. tests whether the signs support the same account.

> *The condition is what prior action established; the footprints are what its passage left; the memorandum is what was written so that an absent event could later be recalled and examined.*

The compact triad is **κατάστασις—ἴχνος—ὑπόμνημα**. Its special contribution is the distinction between event, established condition, physical or logical footprint, and written aid to later memory. It also warns that notes about an event are materials for history, not the event's complete history.

Chapter 15 — Latin: status—vestigium—commentarius

Latin likewise supplies an analytical vocabulary rather than a historical localization standard. **Status—vestigium—commentarius** links the system's standing condition, the footprint or vestige left by prior action, and a working record compiled for later consultation. The related activity **vestigatio** makes investigation literally a search through vestiges, while **acta** distinguishes formal proceedings from the analyst's evolving commentary.

Sources for this chapter: [40–43] in Sources and Further Reading.

Scope and status of the vocabulary

Latin offers a powerful analytical vocabulary but no inherited standardized terminology for modern dumps, traces, and logs. The proposed triad **status—vestigium—commentarius** is conceptual. It uses attested meanings to clarify evidence relations while leaving current product terminology untouched.

Diagnostic relation	Latin lens	Analytical emphasis
Captured state	**status**	Standing, posture, position, condition, or established order
Memory image	**imago memoriae**	A conceptual image of memory, not a claimed standard term
Trace	**vestigium**	Footprint, track, trace, mark, remnant, or indication
Investigation	**vestigatio**	Tracking, searching out, or inquiry through traces
Working record	**commentarius** / **commentarii**	Notebook, memoranda, notes, diary, or explanatory record
Official record	**acta**	Things done and the registered proceedings that preserve them

The distinction between **commentarii** and **acta** is analytically useful. One presents an evolving working record or commentary; the other emphasizes formally recorded proceedings. Software systems produce both kinds: exploratory diagnostic notes and operational or audit records maintained under explicit rules.

1. Memory dump as status and imago

Status derives its force from standing. It can name posture, position, condition, public order, or the established situation of a person or thing. A dump captures a system in such a standing configuration.

The conceptual phrase **imago memoriae** emphasizes representation. The dump is an image of memory and associated process state, not memory's self-explanation. As with any **imago**, resemblance and selection must be evaluated. The image may omit unmapped regions, external services, historical transitions, network state, storage state, and activity outside the capture boundary.

Status analysis asks how the visible world hangs together:

- Which entities stand in relations of ownership, reference, waiting, or reachability?
- Which values represent stable configuration and which represent an interrupted transition?
- Which vestiges of earlier activity remain embodied in the current condition?

Latin also makes it natural to distinguish **status** from **motus**, state from movement. Diagnostics needs both. The dump foregrounds status; tracing foregrounds motus and **iter**, the path or journey of execution. Neither can substitute for the other.

2. Trace as vestigium; inquiry as vestigatio

Vestigium can name the sole or footprint, a track, trace, mark, vestige, remnant, or indication. It is one of the clearest lexical models for diagnostic evidence. Something has moved or acted; its direct presence is gone; a mark remains.

The activity **vestigatio** is tracking or searching out. The same semantic family therefore joins system residue and analytical method. Execution leaves **vestigia**; the investigator performs **vestigatio**; the proposed route is an **iter** reconstructed under evidential constraints.

The family supports a disciplined sequence:

- **motus** — the actual movement or action;
- **vestigium** — the individual mark left by it;
- **vestigia** — the plural field of marks available to inquiry;
- **iter** — the route reconstructed through them;
- **vestigatio** — the investigation that follows and tests the marks.

This structure resists a common interface illusion. A rendered trace waterfall looks like an **iter**, a coherent route. Its data, however, consists of selected **vestigia**. The route is a model assembled from marks, correlation rules, and assumptions about time and identity.

Vestigium also includes residue. A resource leak, elevated queue, altered cache, stale pointer, or incomplete transaction can be a vestige of an action whose telemetry has disappeared. The surviving state may contain stronger evidence of causal effect than the original emitted message.

3. Log as commentarius and acta

Commentarius can name a notebook, memorandum, notes, diary, or record prepared for use. **Commentarii** suggests accumulated working material: observations preserved so that events can be recalled, organized, and interpreted. It is therefore a good lens for diagnostic logs that collect entries without themselves supplying a complete causal history.

Acta, literally things done, also names recorded proceedings. The word brings action and official preservation close together while still distinguishing them. The **acta** are not every action; they are actions as admitted to a formal record.

This distinction maps well to observability:

- an emitted log entry is a selected **nota** or item;
- the accumulated working record resembles **commentarii**;
- an audit or authoritative proceedings record resembles **acta**;
- the final diagnostic explanation is a later **narratio** constrained by all available evidence.

A log can thus be truthful and partial. Its entry reports what an instrumented observer was able and designed to say. It may omit silent corruption, unobserved branches, remote dependencies, or effects that became visible only later.

4. The Latin diagnostic grammar

The system:

1. **statum habet** — has a state;
2. **iter peragit** — proceeds through a path;
3. **vestigia relinquit** — leaves traces;
4. **acta consignat** — records selected proceedings.

The analyst:

1. **commentarios legit** — reads the records;
2. **vestigia sequitur** — follows the traces;
3. **iter restituit** — reconstructs the route;
4. **statum inspicit** — examines the captured state;
5. **testimonia inter se confert** — compares the evidence with itself and across sources.

> *Latin joins the system's standing condition, the vestiges left by its movement, the inquiry that follows them, and the notes through which absent events are made available to later judgment.*

The compact triad is **status—vestigium—commentarius**. Its special contribution is the tight relation between vestige and investigation and the distinction between what was done, what was officially recorded as done, and what the analyst later constructs from the surviving record.

Chapter 16 — Southern Quechua: kay—yupi—qillqa

The Southern Quechua perspective is an analytical formulation rather than a claim of standardized computing terminology. **Kay—yupi—qillqa** brings together being or existing, the footprint left by passage, and writing or a written document. It also adds an evidential discipline that is especially relevant to diagnostics: Quechua grammar can distinguish direct knowledge, reported knowledge, and conjecture. The analyst must therefore ask not only what an artifact says, but how the claim became known.

Sources for this chapter: [44, 45] in Sources and Further Reading.

Scope and terminology

Quechua is a family of related Indigenous languages rather than a single uniform code. This chapter uses broadly Southern Quechua forms and cites a Bolivian bilingual dictionary[44] whose orthography includes forms such as **qhatipay**. Spellings and preferences differ across regions. The triad **kay—yupi—qillqa** is therefore an analytical lens, not a proposed localization standard for software products.

Artifact or activity	Southern Quechua lens	Analytical emphasis
State or condition	**kay**, **kaynin**, **imayna kasqan**	Being, existence, nature, or the way something presently is
Memory preservation	**yuya** / **yuyay**, **waqaychay**	Memory or recollection, and the act of keeping or preserving
Footprint or trace	**yupi**	The mark left by a foot where it passed
Tracking	**yupinay**, **yupipay**, **qhatipay** / **qatipay**	Following a trail or investigating through footprints
Writing or record	**qillqa**, **qillqasqa**	Writing, document, and that which has been written
Report or account	**willakuy**	Information, report, or narrated account

The source dictionary[44] defines **yupi** as a footprint and gives **yupinay**, **yupipay**, and **qhatipay** for following a trail. It defines **qillqa** as writing and document, and **qillqasqa** as something written. Those meanings are well attested. By contrast, a compound such as “Quechua for memory dump” would depend on a localization community and product context. This chapter therefore keeps *memory dump* as the technical artifact name and uses Quechua to analyse what the artifact does.

1. Memory dump as kay and preserved condition

Kay is the verb of being, existing, and having presence. It also participates in expressions that name qualities or states: a condition can be understood as the way something is at a moment. That makes it a

productive lens for a dump. The artifact does not preserve being in the abstract. It preserves a bounded and represented **kaynin**: the system's condition within a selected process, address space, capture mode, and instant.

The vocabulary of memory and preservation adds a second layer. **Yuya** or **yuyay** concerns memory and recollection; **waqaychay** concerns keeping, guarding, conserving, or preserving. A dump can be understood as an engineered act of preservation, but what is preserved is not a human recollection and not the complete past. It is a machine-readable configuration from which an analyst later reconstructs relations.

This perspective suggests four questions:

- What aspect of the system's **kay** was actually captured?
- Which relations make the captured values a coherent **kaynin** rather than an inventory?
- What did the capture mechanism preserve, and what did it exclude?
- Which earlier actions remain visible only through consequences in present state?

The distinction between existence and preservation guards against a familiar error. A dump is not the system's reality without remainder. It is a preserved representation of a limited condition. Threads outside the process, remote services, network history, prior scheduler decisions, overwritten buffers, and lost events may all remain absent.

2. Trace as yupi and the work of qhatipay

Yupi is a footprint: a sign left where a foot pressed the ground. The metaphor fits diagnostic trace evidence with unusual precision. Execution itself has passed. What remains is a field of selected marks—timestamps, events, identifiers, counters, spans, and altered structures—from which movement may be inferred.

The verbs **yupinay**, **yupipay**, and **qhatipay** or **qatipay** separate the footprint from the activity of tracking it. This matters because a trace file is not automatically a causal path. Instrumentation emits marks. An analyst follows them, tests their order, accounts for gaps, and reconstructs one or more possible routes.

The footprint image also makes absence legible. If a request should cross five boundaries and only four footprints are present, the missing mark may identify sampling, loss, an uninstrumented branch, or an action that never occurred. The analyst should not fill the gap simply because the surrounding sequence looks plausible.

Quechua also offers an important grammatical reminder about evidence. Many Quechua varieties mark whether information is directly known, reported, or inferred. The details vary by variety, but the diagnostic lesson is general: every assertion should carry an implicit answer to “How is this known?” A dump field may be directly inspected; a component failure may be reported by a log; a causal path may be inferred from footprints. Those are different evidential statuses.

3. Log as qillqa, qillqasqa, and willakuy

Qillqa names writing and a document; **qillqasqa** names what has been written. A log is therefore best approached as an inscription produced under rules. It contains selected statements in a schema, ordered and retained according to an instrumentation policy.

Willakuy adds the sense of information, report, or account. The combination distinguishes inscription from narration. A timestamped record may be accurately written while its message still expresses only one component's local interpretation. “Database unavailable” can be a faithful **qillqasqa** and an incomplete **willakuy** if the deeper cause is pool exhaustion, DNS delay, or a caller-side timeout.

This distinction supports three levels of analysis:

- the **qillqa** as the designed form and schema of the record;
- the **qillqasqa** as the concrete entry that was actually written;
- the **willakuy** as the account an analyst constructs from one or more entries.

The record should be read for both content and provenance. Which component wrote it? Was the knowledge direct, reported, or inferred? What was omitted by severity filters, buffering, sampling, retention, or failure of the logging path itself?

4. The Southern Quechua diagnostic grammar

The compact analytical sequence is deliberately presented as a vocabulary map rather than as standardized product phrasing.

The system:

1. **kay / kaynin** — exists in a particular condition;
2. **ñan** — proceeds along a path;
3. **yupi** — leaves footprints or traceable signs;
4. **qillqa / qillqasqa** — produces selected written records.

The analyst:

1. reads the **qillqasqa** and asks how its claim is known;
2. performs **qhatipay**, following the available **yupi**;
3. reconstructs the **ñan** while leaving gaps and uncertainty visible;
4. inspects the preserved **kaynin**;
5. compares direct observation, report, and inference before accepting a causal **willakuy**.

> *Southern Quechua adds an evidential question to the diagnostic triad: not only what state, footprint, or writing survives, but whether the resulting claim is observed, reported, or inferred.*

The triad is **kay—yupi—qillqa**. Its distinctive contribution is the combination of being, footprint, and inscription with an explicit concern for the source of knowledge.

Chapter 17 — Swahili: hali—alama—kumbukumbu

The Swahili perspective uses **hali—alama—kumbukumbu**: condition, a mark or trace, and a retained record. Its vocabulary sharply separates **kufuata** and **ufuatiliaji**, the acts of following and investigating, from **alama** and **nyayo**, the signs or footprints that can be followed. **Kumbukumbu** joins memory with records and minutes, making a log something preserved for later recollection and accountability rather than a transparent copy of events.

Sources for this chapter: [46, 47] in Sources and Further Reading.

Conventional and analytical vocabulary

Swahili technical practice combines established Swahili vocabulary with international loans such as *logi*, *rekodi*, and *dampo*. Product and regional usage varies. The analytical formulation here is **hali—alama—kumbukumbu**, supported by dictionary meanings rather than presented as a mandatory interface translation.

Artifact or activity	Swahili expression	Analytical emphasis
State or condition	**hali**	State, condition, essence, circumstance, or case
Memory dump	**memory dump**, **dampo ya kumbukumbu**	Conventional or loan-based technical naming; a captured condition
Following or investigation	**kufuata**, **ufuatiliaji**, **kufuatilia**	To follow, pursue, follow up, or investigate
Trace or mark	**alama**, **nyayo**	Sign, mark, clue, trace, vestige, or footprint
Record or log	**kumbukumbu**, **rekodi**, **logi**	Retained record, entry, or operational log

The TUKI Swahili-English dictionary[46] defines **hali** as state, condition, circumstance, or case. It gives **alama** as sign, token, mark, clue, trace, and vestige. **Kumbukumbu** joins remembrance and memory with records, minutes, and annals. The verb family around **fuata** distinguishes following from **fuatilia**, following up or investigating. Together these terms form a particularly practical diagnostic grammar.

1. Memory dump as hali

Hali directs attention to condition rather than substance. A dump contains memory, but its diagnostic object is the condition in which structures stand together: which threads wait, which locks are owned, which objects remain reachable, which queues have accumulated, and which operations are incomplete.

The word also accommodates scale. One can ask about the **hali ya mchakato**, the condition of a process, or the **hali ya mfumo**, the state of a system. This prevents the analyst from treating a process dump as if it were a complete system image. The captured boundary matters.

Dump analysis through **hali** asks:

- Which part of the condition was preserved?
- What relations among visible entities define the failure state?
- Which values are stable configuration and which are interrupted transitions?
- What earlier activity changed the present condition without leaving a complete event sequence?

The dump is strongest at coexistence. It can show several waits, references, counters, and partial operations together. It is weaker at duration and order unless historical information has also been retained in buffers or data structures. **Hali** therefore needs the temporal evidence supplied by tracing and the designed account supplied by records.

2. Trace as alama and nyayo; analysis as ufuatiliaji

Alama ranges across mark, sign, clue, trace, and vestige. **Nyayo** makes the physical metaphor more specific: footprints. The two terms distinguish an individual indication from the path an analyst may infer through a series of indications.

The verb **kufuata** means to follow, come after, or pursue. The dictionary example **kufuata nyayo** is literally to follow someone's footsteps. **Kufuatilia** adds follow-up and investigation. This makes it possible to state the diagnostic separation cleanly:

- the system leaves **alama** or **nyayo**;
- instrumentation stores selected marks;
- the analyst performs **ufuatiliaji**;
- a path or sequence is reconstructed only after identity, order, and missing evidence are tested.

This separation is crucial in distributed tracing. A span is a mark with attributes and timing. A trace visualization is a model that orders marks. Neither necessarily captures queueing outside instrumented code, clock error, dropped spans, unsampled retries, or consequences stored only in state.

The Swahili vocabulary also keeps investigation active. **Ufuatiliaji** is work: following up, comparing, checking, and revising. It is not the passive consumption of a preassembled story.

3. Log as kumbukumbu

Kumbukumbu connects memory with retained records, including minutes and annals. A log is thus a designed aid to later recollection. It makes certain events available after execution has passed, but it does so through choices about what deserves an entry, which name it receives, and how long it is kept.

This dual sense supports an important distinction. A system can have a **rekodi** or individual record; many entries accumulate into **kumbukumbu** that an investigator later consults. Yet retention does not guarantee completeness. Logging may fail under the same pressure that caused the incident. Buffers may overflow, clocks may disagree, entries may be sampled, and sensitive fields may be deliberately removed.

The analyst should therefore ask:

- Who or what created the record?
- Is the entry a direct observation, a classification, or a report from another layer?
- Which identifiers connect it to trace marks and captured state?
- What operational policy controlled filtering, buffering, and retention?

Kumbukumbu is valuable because it preserves. It is limited because preservation is selective.

4. The Swahili diagnostic grammar

The system:

1. **ina hali** — has a state or condition;
2. **hupitia njia** — proceeds through a path;
3. **huacha alama na nyayo** — leaves marks and footprints;
4. **huweka kumbukumbu** — keeps selected records.

The analyst:

1. **anasoma kumbukumbu** — reads the records;
2. **anafuatilia alama** — follows up the marks;
3. **anaunda upya njia** — reconstructs the path;
4. **anakagua hali** — inspects the condition;
5. **analinganisha ushahidi** — compares the evidence.

> *Swahili separates the condition that exists, the marks that remain, the active work of following them, and the records retained so that the incident can be investigated later.*

The compact triad is **hali—alama—kumbukumbu**. Its distinctive contribution is the clear movement from a remaining sign to active follow-up and from memory to an accountable retained record.

Chapter 18 — Turkish: durum—iz—kayıt

The Turkish perspective is **durum—iz—kayıt**: state or situation, trace or mark, and registered record. Conventional technical compounds such as **bellek dökümü**, **izleme dosyası**, **izleme günlüğü**, and **olay günlüğü** make the artifact boundaries explicit. Turkish is a Turkic language. It occupies the requested "Altaic" position only in the historical and areal sense of that label; the chapter does not assume a generally accepted Altaic genetic family.

Sources for this chapter: [48–51] in Sources and Further Reading.

Classification and conventional technical vocabulary

Turkish is a Turkic language. Older literature often placed Turkic, Mongolic, and Tungusic languages in an "Altaic" family, sometimes with Korean and Japanese. That genetic proposal is disputed and is rejected by many historical linguists; areal contact explains many shared features. This chapter uses Turkish for the requested historical "Altaic" slot without treating Altaic as an established family.

Turkish technical documentation supplies a stable practical vocabulary:

Artifact or activity	Turkish expression	Analytical emphasis
State or situation	**durum**	Present condition, configuration, or situation
Memory dump	**bellek dökümü**, **döküm dosyası**	A file into which memory contents are poured or written out
Tracing	**izleme**, **takip**	Monitoring, following, or tracking execution
Trace	**iz**	Mark, track, route, or residual indication
Trace log	**izleme günlüğü**, **olay izleme günlüğü**	A chronological log of selected trace events
Record or log	**kayıt**, **günlük**	Registered entry and accumulated journal or log

Microsoft's Turkish documentation[48] consistently uses **bellek dökümü** for memory dump and **izleme dosyası** or **izleme günlüğü** for trace files and trace logs. The vocabulary makes visible what English can blur: **izleme** is the activity, **iz** is the mark or track, and **günlük** is the accumulated log.

1. Memory dump as durum and bellek dökümü

Durum names a state, condition, or situation. It can refer to a single component or to a configuration of several interacting conditions. That breadth is useful in dump analysis, where the important evidence often lies in relations rather than isolated values.

Bellek dökümü is transparent as a technical compound: **bellek** is memory and **döküm** is a pouring out, casting, or dump. The metaphor suggests extraction, but the artifact remains a representation. A full memory dump, kernel dump, minidump, and application dump preserve different boundaries and degrees of detail.

The analyst should distinguish three levels:

- **bellek içeriği** — the visible contents of memory;
- **süreç durumu** — the process state reconstructed from those contents;
- **sistem durumu** — the larger system condition, only partly represented by a process artifact.

This separation prevents overclaiming. A dump may faithfully contain a process's pages and still omit scheduler history, remote dependencies, network queues, storage behavior, and prior transitions. It answers “What condition can be reconstructed here?” more reliably than “Everything that happened.”

2. Trace as iz; tracing as izleme

İz is a mark, track, sign, or trace. It can be followed, left behind, erased, or interpreted. **İzleme** is the act of monitoring or tracing. Turkish therefore preserves the same crucial distinction seen in several other chapters: the system leaves or emits traces; the analyst follows them and reconstructs a path.

Technical compounds sharpen the distinction further. An **izleme dosyası** is a trace file. An **olay izleme günlüğü** is an event-tracing log. A tool can convert, filter, correlate, or render such a file, but those operations do not guarantee that the resulting view is the complete route of execution.

A disciplined trace analysis asks:

- Which **izler** were emitted, retained, sampled, or lost?
- Which identity links the marks to the same request, process, thread, or transaction?
- Does their temporal order survive clock differences and buffering?
- Which **yol**, route, is directly observed and which part is inferred?
- What **etki**, effect, remains in state after the trace event has passed?

The last question prevents a narrow event-only analysis. A retry, allocation, lock acquisition, or configuration change may leave a durable **iz** in later state even if the original event is absent from the trace log.

3. Log as kayıt and günlük

Kayıt is a record, registration, or recorded entry. **Günlük** is a diary, journal, or log that accumulates entries through time. Their difference mirrors the relationship between one event record and the larger log in which many records are retained.

This distinction also exposes design. A **kayıt** exists because a component decided to register something under a schema. A **günlük** exists because an operating policy collected, ordered, retained, and made such entries available. Both are selective.

The phrase **olay günlüğü**, event log, encourages the analyst to ask whether an entry records the event itself or a component's interpretation of it. “Authentication failed” may be a correct local classification, while the causal path involves clock skew, expired credentials, unavailable identity service, or corrupted cache state. The entry is evidence, not final judgment.

Good analysis connects **kayıt** to **iz** and **durum**:

- correlate record identifiers with trace identifiers;
- compare log time with capture time and monotonic event order;
- test whether reconstructed state supports the message;
- retain contradictions instead of forcing all artifacts into one narrative.

4. The Turkish diagnostic grammar

The system:

1. **bir durumu vardır** — has a state;
2. **bir yol izler** — follows a path;
3. **iz bırakır** — leaves a trace;
4. **olayları kaydeder** — records selected events.

The analyst:

1. **günlükleri okur** — reads the logs;
2. **izleri takip eder** — follows the traces;
3. **yolu yeniden kurar** — reconstructs the path;
4. **durumu inceler** — examines the state;
5. **kanıtları karşılaştırır** — compares the evidence.

> *Turkish distinguishes the condition in which the system stands, the traces left or emitted by its movement, and the registered entries accumulated into an operational journal.*

Chapter 19 — Māori: āhua—tapuwae—rangitaki / mauhanga

The Māori perspective is **āhua—tapuwae—rangitaki / mauhanga**: condition or form, footprint, and a record that the language can name in either of two registers. **Tapuwae** makes trace evidence concrete as steps left by passage. For the log lens Māori offers a genuine choice: **rangitaki**, with the compounds kōnae rangitaki and rangitaki hapa, is the attested technical term for a log, a log file, and an error log, while **mauhanga** is the ordinary word for a record, documentation, or archive. The triad links what a system looked like, where its activity left footprints, and what was deliberately stored for later inquiry.

Sources for this chapter: [52–59] in Sources and Further Reading.

Conventional and analytical vocabulary

Māori supplies more than one term for the log lens, and the difference between them is instructive rather than accidental. Taiuru's Dictionary of Māori Computer Related Terms[55, 58] and the Microsoft Language Portal[59] both give **rangitaki** for log, glossed as a record of transactions or events that take place within an IT managed environment, with kōnae rangitaki and kōnae pūrongo for log file, rangitaki hapa for error log, and takitaki for logging[55, 58]. Te Aka[56, 57], by contrast, records only the everyday sense of **rangitaki** as blog — a frequently updated online journal or column — and Paekupu[57] lists it under blog and weblog beside hauhiko. Both usages are documented and neither cancels the other: the word carries the technical sense inside computing terminology and the everyday sense outside it, much as English log moves between a ship's logbook and a diagnostic file. Alongside these, **mauhanga** names a record, documentation, or archive in ordinary Māori. This chapter therefore treats rangitaki and mauhanga as parallel names for the same analytical position rather than choosing between them. The trace and dump lenses are analytical extensions grounded in established dictionary meanings.

Artifact or activity	Māori expression	Analytical emphasis
State, condition, or form	**āhua**, **tūnga**, **tūranga**	Condition, character, shape, position, situation, or stance
Memory	**pūmahara**	Memory in computing terminology
Trace or footprint	**tapuwae**, **tapuae**	Footprint, tread, or steps left by passage
Following	**whai**, **whāia**	To pursue or follow the available footprints
Record or log (general)	**mauhanga, tuhi**	Record, documentation, or archive; to write or record
Log, log file, logging (technical)	**rangitaki, kōnae rangitaki, kōnae pūrongo, takitaki**	Log, log file and logging in Taiuru and the Microsoft Language Portal[59]; rangitaki hapa is error log

Artifact or activity	Māori expression	Analytical emphasis
Blog (everyday usage)	**rangitaki**	The everyday sense of the same word: blog in Te Aka and Paekupu[56, 57]

Āhua can name shape, appearance, condition, character, likeness, nature, figure, or form. **Tapuwae** names a footprint. **Mauhanga** names a record, documentation, or archive and carries the record lens in ordinary Māori, while **rangitaki** carries it inside computing terminology, where kōnae rangitaki and kōnae pūrongo both name a log file. Neither word is a mistake for the other; they belong to different registers of the same language.

1. Memory dump as āhua and tūnga

Āhua does more than name an abstract state. It brings together condition and form: how something is and the shape in which it appears. That is a strong model for dump analysis. A dump is a representation of the system's visible form at capture time, and the analyst must infer the relations that make that form a meaningful condition.

Tūnga and **tūranga** add position, situation, site, status, and stance. Threads, objects, handles, and buffers do not merely possess values; they occupy positions within a structure of ownership, waiting, reachability, and responsibility.

The lens suggests four questions:

- What **āhua** of the system does this capture present?
- What is the **tūnga** of each relevant entity within that configuration?
- Which part of the form is stable and which part records an interrupted transition?
- What earlier path explains why the system now has this shape?

The term **pūmahara** supplies the computing word for memory, but the dump is not memory in the human sense of a narrated past. It is a captured form from which limited claims about prior action can be inferred.

2. Trace as tapuwae

Tapuwae or **tapuae** is a footprint or tread. Dictionary examples explicitly describe following the footprints of people and examining them to infer what happened. The diagnostic analogy is direct: system activity passes; footprints remain; an investigator follows and compares them.

Footprints have several useful properties as a model:

- they are effects of movement, not the movement itself;
- they are spatially or temporally distributed;
- some are clear, some partial, and some erased;
- a sequence of prints suggests a path but does not automatically prove identity or cause;
- later movement can overwrite earlier marks.

In software, a **tapuwae** may be an emitted event, changed counter, span, timestamp, identifier, allocation, queue entry, or residual state change. Some footprints live in trace files; others survive only in dumps, storage, caches, or external systems.

The analyst's task is **whai**, following. The distinction between footprint and following prevents the trace visualization from becoming unquestioned truth. A rendered path is an interpretation constrained by the available **tapuwae**, capture policy, correlation rules, and uncertainty.

3. Log and record: rangitaki and mauhanga

Rangitaki is the attested technical term for a log, glossed in the Microsoft Language Portal[59] as a record of transactions or events that take place within an IT managed environment; **kōnae rangitaki** and kōnae pūrongo name a log file, rangitaki hapa an error log, and takitaki the act of logging[55, 58]. **Mauhanga** names a record, documentation, or archive in ordinary Māori. The apparent conflict with Te Aka[56, 57], which gives rangitaki as **blog**, is a difference of register rather than a contradiction, and the chapter keeps both words in view.

Mauhanga, record or documentation, broadens the lens. It asks what is made durable enough to consult later. **Tuhi**, to write or record, emphasizes the producing act. Together the terms distinguish event, inscription, accumulated log, and later documentation.

A log is designed evidence. It records actions according to the capabilities and priorities of the system that writes it. It may include a component's interpretation rather than a direct measurement. It may omit events because of sampling, privacy rules, buffer loss, or failure of the logger. Its chronological appearance should not be confused with complete causality.

The analyst should therefore connect the **rangitaki** to the **tapuwae** and **āhua**:

- Which entry describes the same entity seen in the trace and dump?
- Does the captured condition support the recorded claim?
- Which footprints have no corresponding entry?
- Which entries describe intentions or classifications rather than completed effects?

4. The Māori diagnostic grammar

The system:

1. **he āhua tōna** — has a condition or form;
2. **ka whai i te ara** — proceeds along a path;
3. **ka waiho i ngā tapuwae** — leaves footprints;
4. **ka tuhi ki te rangitaki** — writes selected events to the log.

The analyst:

1. **pānuihia te rangitaki** — reads the log;
2. **whāia ngā tapuwae** — follows the footprints;
3. **hangaia anō te ara** — reconstructs the path;
4. **tirohia te āhua me te tūnga** — examines condition and position;
5. **whakatairitea ngā taunakitanga** — compares the evidence.

> *Māori presents diagnosis as attention to form and position, the disciplined following of footprints, and the reading of a record deliberately stored for later inquiry.*

Chapter 20 — Cherokee: ᏅᏍᎫᏓᎤ—ᎠᏍᏓᏩᏅᏍᏗ—ᏚᎦᎤᏙᏢᏥ

The Cherokee perspective is **ᏅᏍᎫᏓᎤ—ᎠᏍᏓᏩᏅᏍᏗ—ᏚᎦᎤᏙᏢᏥ** (*usdidanv—asdawadvsdi—deganvnvdvi*): condition, trace, and record. The vocabulary is grounded in entries labeled as Cherokee Nation word-list[60] material and Microsoft computer terminology in the Cherokee-English Dictionary database[62]. The paired entries for **trace** and **tracer** make the remaining indication and the role that follows it separately nameable, while the technical expression for **log file** presents logging as deliberate preservation rather than as an opaque loan.

Sources for this chapter: [60, 61, 62] in Sources and Further Reading.

Scope, sources, and orthography

Cherokee is a Southern Iroquoian language written in the syllabary developed by Sequoyah. Contemporary usage spans communities and dialects, and technical terminology continues to develop through language departments, educators, translators, and technology-localization work. This chapter uses forms shown in the Cherokee-English Dictionary database[62], which identifies material from the Cherokee Nation word list[60] with **cn** and Microsoft computer terms with **msct**. Romanizations are reproduced as search aids, not presented as complete pronunciation guides.

The compact triad is analytical, but each of its terms is attested in the consulted sources:

Artifact or activity	Cherokee expression	Source and analytical emphasis
Condition	**ᏂᏍᎫᏓᏁ** (*nusdidanvi*); **ᏅᏍᎫᏓᎤ** (*usdidanv*)	Cherokee Nation “condition of things”; Microsoft computer term “condition”
Memory	**ᎠᎤᏁᎫᏍᏙ** (*anvdadisdo*); **ᎠᎤᏁᎫᏍᏗ** (*anvdadisdi*)	Technical and general terms for memory
Trace	**ᎠᏍᏓᏩᏅᏍᏗ** (*asdawadvsdi*)	Cherokee Nation word-list[60] entry for “Trace”
Tracer	**ᎠᏍᏓᏩᏚᎩ** (*asdawadegi*)	Cherokee Nation word-list[60] entry for “Tracer”
Record	**ᏚᎦᎤᏙᏢᏥ** (*deganvnvdvi*); **ᎯᏰᏍ** (*gowelv*)	Technical record and a recorded or written item
Log file	**ᏧᎵᏍᎳᏂᏙᏍ ᎧᏃᎵᏍᎩ ᎠᏍᏆᏂᎪᏙᏗ** (*nulistanidolv kanohesgi asquanigododi*)	Microsoft computer terminology for “log file”

The chapter does not claim that a single triad exhausts Cherokee technical usage. Its purpose is narrower: to use attested vocabulary to separate the condition preserved in memory, the trace available for following, the role of the tracer, and the record deliberately retained in a log file.

1. Memory dump as condition and memory

No exact standardized Cherokee equivalent for *memory dump* was established by the consulted entries. The artifact name should therefore remain recognizable in technical contexts, while the analytical relation can be expressed through **ᎠᎤᏂᎯᏙ**, memory, and **ᎤᏍᏗᏂᎤ**, condition.

This distinction is useful. A memory dump contains selected memory, but diagnosis is rarely interested in bytes merely because they were stored. The analyst wants to reconstruct a **condition of things**: which threads were waiting, which objects remained reachable, which handles were owned, which queues had accumulated, and which operation was interrupted. The dump's meaning lies in the relations that coexist inside the capture.

The memory term should not encourage an autobiographical reading. A dump is not a narrated recollection of everything the process experienced. It is a technically bounded capture whose omissions depend on operating system, runtime, dump type, permissions, timing, and capture policy. It supports direct inspection of some values and only inference about the path that produced them.

The condition lens suggests four questions:

- Which **ᎤᏍᏗᏂᎤ** is actually represented by this dump: a thread, process, runtime, host, or distributed service?
- Which structures were preserved, and which were excluded or unreadable?
- Which relationships make the captured values diagnostically meaningful?
- What earlier activity could plausibly have established this condition?

The resulting rule is simple: describe the capture boundary before describing the system's condition. A partial dump can accurately preserve a local condition without constituting a complete image of the incident.

2. Trace and tracer

The Cherokee Nation entries provide a particularly valuable pair: **ᎠᏍᏓᏩᏗᏍᏗ** for *Trace* and **ᎠᏍᏓᏩᏚᎩ** for *Tracer*. Without forcing a morpheme-by-morpheme analysis, the lexical pair keeps the object or activity called a trace distinct from the role or mechanism that traces.

That distinction is often lost in English observability vocabulary. A trace may mean an emitted event, a trace file, a rendered waterfall, the reconstructed route of a request, or the act of instrumenting execution. A tracer may be a library, agent, collector, debugger, analyst, or correlation process. These are related, but they are not interchangeable.

In diagnostic work, a tracer selects what becomes visible. It chooses instrumentation points, fields, identifiers, timing sources, sampling rules, and failure behavior. The resulting trace is therefore designed evidence. It can be precise about what it captured while remaining silent about uninstrumented branches, dropped buffers, unsampled work, and effects preserved only in state.

The analyst should separate four stages:

- the **tracer** and its capture policy;
- the emitted trace events or marks;
- the path reconstructed from those marks;
- the condition and effects left after the path was executed.

This makes absence interpretable. A missing event may mean that an operation did not occur, but it may also mean that the tracer never observed it, sampling excluded it, a buffer was lost, or correlation failed. The trace constrains explanation; it does not manufacture completeness.

3. Record and log file

The technical record term **ᏕᏚᏅᏅᏛᎢ** names an individual or retained record. The full technical expression **ᏧᎱᏍᏔᏂᏙᏧ ᏫᏃᏺᏍᎩ ᎠᏍᏆᏂᎪᏙᏗ** is listed for *log file*. The longer expression is analytically revealing even when treated as a whole rather than assigned a speculative word-by-word gloss: a log file is explicitly named technical content, not an unexamined mirror of reality.

A record exists because a component decided that something was worth retaining under a schema. A log file exists because a system collected, ordered, encoded, stored, rotated, protected, and eventually discarded records under an operating policy. The difference between record and log file therefore exposes two levels of design: the local decision to emit an entry and the larger policy that makes many entries available for later inquiry.

The analyst should ask:

- Which component created each **ᏕᎦᏅᏁᏛᎢ**?
- Does the record describe an observation, an interpretation, an intention, or a completed effect?
- Which identifiers connect the record to **ᎠᏍᏓᏩᏛᏍᏓᏁ** and captured **ᏗᏍᏗᏓᏰᏁ**?
- What filtering, privacy, buffering, rotation, and retention rules governed the log file?
- Which expected records are absent, and what alternative causes could explain the absence?

Chronology alone does not settle causality. A log can be ordered yet incomplete, accurate at one abstraction layer yet misleading at another, or internally consistent while sharing the same faulty clock as the trace. Its authority grows through corroboration with marks and state.

4. The Cherokee diagnostic grammar

The following is an analytical workflow using attested terms, not a set of invented Cherokee interface sentences.

The system:

1. has an inspectable **ᏗᏍᏗᏓᏰᏁ** (*usdidanv*) — condition;
2. contains and changes **ᎠᏅᏗᏍᎪᏍᏙ** (*anvdadisdo*) — memory;
3. exposes **ᎠᏍᏓᏩᏛᏍᏓᏁ** (*asdawadvsdi*) — trace;
4. creates selected **ᏕᎦᏅᏁᏛᎢ** (*deganvnvdvi*) — records;
5. retains them in a technical log file under an explicit policy.

The analyst:

1. reads the retained records without treating them as verdicts;
2. identifies the **ᎠᏍᏓᏩᎶᏍᎩ** (*asdawadegi*) — tracer — and its capture limits;
3. reconstructs a path from the available trace marks;
4. inspects memory to establish the captured condition;
5. compares record, trace, tracer policy, and state before accepting a causal explanation.

> *Cherokee makes the condition, the trace, the tracer, the record, and the retained log file separately visible. Diagnosis becomes a comparison between what was preserved, what was followed, who or what performed the tracing, and what the system deliberately placed on record.*

Chapter 21 — Tibetan: གནས་སྣང་—རྗེས་ཤུལ་—ཐོ་ཡིག་

The Tibetan perspective is གནས་སྣང་—རྗེས་ཤུལ་—ཐོ་ཡིག་ (*gnas stangs—rjes shul—tho yig*): condition, the trail left afterward, and an organized register. The terms are lexically attested, but the triad is analytical rather than a claim that one Tibetan localization standard uses precisely these three forms for all computing artifacts. Its value lies in the movement from situated condition, through aftermath and remains, to an account arranged for later consultation.

Sources for this chapter: [63, 64, 65] in Sources and Further Reading.

Scope, sources, and transliteration

Tibetan computing vocabulary develops across institutions, regions, and technical communities. This chapter therefore does not present its triad as a single mandatory localization standard. It uses entries documented in the Tibetan–English–Sanskrit Dictionary[63–65], the only Tibetan lexicographic resource listed in Sources and Further Reading, to build an analytical vocabulary. Wylie transliteration is included as a stable search aid; it is not a phonetic transcription.

Artifact or activity	Tibetan expression	Attested range and analytical emphasis
Condition or state	གནས་སྣང་ (*gnas stangs*)	Situation, condition, circumstance, position, state, or status
Trace or trail	རྗེས་ཤུལ་ (*rjes shul*)	Trace, trail, remains, ruins, legacy, or aftermath
Register or catalogue	ཐོ་ཡིག་ (*tho yig*)	List, catalogue, register, notes, or manual
Record	ཟིན་ཐོ་ (*zin tho*)	A record or written note retained for reference

No exact, broadly standardized Tibetan equivalent for *memory dump* was established in the consulted sources. In technical communication, the artifact name should remain recognizable. The analytical lens is གནས་སྣང་: the condition that a capture allows an investigator to reconstruct.

1. Memory dump as གནས་སྣང་

གནས་སྣང་ gathers several senses useful to diagnostic work: condition, circumstance, situation, position, and state. A dump is not important merely because it contains memory. It is important because relationships among captured values establish a situation: threads wait on locks, objects retain references, queues hold unprocessed work, registers point into interrupted code, and buffers preserve incomplete exchanges.

The situation lens makes scope explicit. A process dump may show the condition of one runtime while saying little about a remote dependency. A kernel dump may preserve operating-system structures but omit application-level meaning. A partial dump may capture the decisive object but not the history that created it. The evidence is real within its boundary; the analyst's responsibility is to state that boundary.

Four questions follow:

- Which གནས་སྣང་ is represented: thread, process, runtime, host, or distributed service?
- Which relationships, rather than isolated values, define that condition?
- Which earlier actions are visible only through their remaining effects?
- What information was excluded by the dump type, permissions, timing, or capture policy?

This vocabulary discourages an undifferentiated claim that the dump “contains the failure.” It contains a selected condition from which the failure may be reconstructed.

2. Trace as རྗེས་རྒྱལ་

རྗེས་རྒྱལ་ is especially productive because its attested range includes trace, trail, remains, legacy, and aftermath. It directs attention to what is available *after* movement or action. A trace event is thus one possible mark; a rendered timeline is one possible reconstruction; a changed cache entry, orphaned resource, or stale flag may be an aftermath preserved in state.

The term helps separate three things that English often calls a trace:

- instrumentation that observes selected operations;
- a route reconstructed from emitted events;
- the remains or aftermath that execution leaves behind.

These layers do not always coincide. Sampling may omit the critical event while state still preserves its effect. A trace may show a request ending without showing the resource it leaked. Conversely, a residual object in a dump may strongly constrain the history without proving the precise route by which it was created.

The analyst should therefore follow རྗེས་ཐུལ་ in both temporal and material senses. Follow identifiers through event order, but also inspect what the path altered. The trail is not merely a line on a timeline; it is the total set of available indications left by passage.

3. Register and record

ཐོ་ཡིག་ can denote a list, catalogue, or register. ཟིན་ཐོ་ is used for a record or retained note. Together they distinguish an individual act of recording from the organized collection that makes many entries useful later. That is a strong model for logs.

A log entry is created under a local decision: a component selects fields, severity, wording, identifiers, and time. A log as register is governed by broader policy: collection, ordering, buffering, redaction, storage, rotation, access, and retention. Neither level is neutral. Both shape what a future analyst can know.

The register lens suggests asking:

- Who or what was authorized to create each record?
- What schema and vocabulary constrained the account?
- Which entries report observation, interpretation, intention, or completed effect?
- Which records were dropped, filtered, redacted, or retained elsewhere?
- Can the register be aligned with the trail and the captured condition?

An organized register is valuable precisely because it is designed. That design must be examined as part of the evidence rather than hidden behind the apparent clarity of the prose.

4. The Tibetan diagnostic grammar

The following is an analytical workflow using attested terms, not a proposed set of Tibetan interface commands.

The system:

1. has a changing གནས་སྟངས་ — condition or situation;
2. moves through operations that leave རྗེས་ཕྱུལ་ — a trail and aftermath;
3. creates selected ཟིན་ཐོ་ — records;
4. arranges retained entries as a ཐོ་ཡིག་ — register.

The analyst:

1. reads the register as a designed account;
2. follows the trail across events and residual effects;
3. reconstructs the route within the limits of instrumentation;
4. inspects captured memory to establish the relevant condition;
5. accepts an explanation only when register, trail, and state mutually constrain it.

> *Tibetan frames diagnosis as the reconstruction of a situated condition from the trail, remains, and aftermath of execution, tested against records deliberately arranged for later consultation.*

Chapter 22 — Tamil: நிலை—தடம்—பதிவு

Tamil, selected here as the Dravidian-language perspective, offers நிலை—தடம்—பதிவு (*nilai—taṭam—pativu*): state, track or trace, and record. Unlike a purely historical comparison, this chapter can also draw on attested Tamil computing vocabulary, including நினைவகக் கொட்டல் for a memory dump and பதிவுக் கோப்பு for a log file. The resulting grammar distinguishes a condition, the mark by which movement is followed, and an act or object of deliberate recording.

Sources for this chapter: [66, 67] in Sources and Further Reading.

Scope, sources, and the Dravidian perspective

Tamil is used here as one Dravidian-language perspective. It has a long literary history and an active modern technical register. The Tamil Computer Encyclopedic Dictionary[66] attests நினைவகக் கொட்டல் (*niṉaivakak koṭṭal*) for *memory dump*, while technical usage also provides பதிவுக் கோப்பு (*pativuk kōppu*) for *log file*. The compact analytical triad is நிலை—தடம்—பதிவு.

Artifact or activity	Tamil expression	Analytical emphasis
State or condition	நிலை (*nilai*)	Standing, condition, position, or state
Memory dump	நினைவகக் கொட்டல் (*niṉaivakak koṭṭal*)	A dump or discharge of memory contents
Trace or track	தடம் (*taṭam*)	Track, trace, route, mark, or impression
Tracing	தடமறிதல் (*taṭamaṟital*)	Finding, tracing, or tracking a trail
Record	பதிவு (*pativu*)	Record, entry, registration, or recording
Register or logbook	பதிவேடு (*pativēṭu*)	A book or register of retained entries
Log file	பதிவுக் கோப்பு (*pativuk kōppu*)	A file of records or log entries

These terms are not exact synonyms. Their analytical power comes from preserving the differences among captured content, the state inferred from it, the mark left by movement, and the record intentionally retained.

1. Memory dump and நிலை

The expression நினைவகக் கொட்டல் names the artifact through an action performed on memory: its contents are dumped or discharged into a form that can be inspected. நிலை names the object of analysis: the state or condition reconstructed from those contents.

This distinction prevents the container from being confused with the condition it represents. A dump may include millions of bytes without preserving every relevant relationship. It may omit pages, external services, prior events, or data protected by another boundary. Conversely, a small set of thread stacks and ownership links may be sufficient to establish a deadlock. Diagnostic value depends on structure, not volume.

The analyst can proceed through four questions:

- What memory was included in the நினைவகக் கொட்டல், and what was omitted?
- Which objects, threads, registers, and resources jointly establish the நிலை?
- Which parts of that state are direct observations and which are decoded or inferred?
- What earlier path could have produced the captured condition?

நிலை also resists a purely temporal reading. A snapshot has a timestamp, but its key evidence is coexistence: facts that were simultaneously true within the selected boundary.

2. Tracing, track, and mark

தடம் can name a track, trace, mark, route, or impression. தடமறிதல் names the activity of finding or following such a track. The pair separates the analyst's method from the indication on which the method operates.

In software systems, instrumentation emits events. The analyst correlates them into a route. The route leaves effects in state. All three may be described in relation to a trace, but they have different evidential status. An event is a recorded observation at an instrumentation point; a reconstructed path is an inference across events; an altered object or resource is a remaining mark.

The Tamil lens yields a useful sequence:

- establish how தடமறிதல் was performed and what it could observe;
- identify the தடம் preserved by event identifiers, timing, and order;
- test that track against marks left in captured state;
- distinguish silence from proof that no action occurred.

A missing trace event may reflect non-execution, sampling, dropped buffers, clock disagreement, or broken correlation. The track becomes persuasive when independent state and records agree with it.

3. பதிவு, பதிவேடு, and பதிவுக் கோப்பு

பதிவு names a record, entry, registration, or act of recording. பதிவேடு presents many such entries as a register or logbook. பதிவுக் கோப்பு identifies the technical file in which records are retained. The three forms expose a hierarchy often hidden by the English noun *log*.

The analyst should ask:

- Does a பதிவு report a direct observation, an interpretation, an intention, or a completed effect?
- What identifiers connect it to the தடம் and captured நிலை?
- How did the பதிவுக் கோப்பு handle buffering, failure, rotation, and redaction?
- Which expected entries are absent, and what competing explanations fit that absence?

The record is testimony generated by software. Its clarity is useful, but it earns authority only through corroboration.

4. The Tamil diagnostic grammar

The following workflow uses attested vocabulary as analytical labels.

The system:

1. has a changing நிலை — state;
2. moves through operations that leave a தடம் — track or mark;
3. creates selected பதிவு entries — records;
4. retains them in a பதிவுக் கோப்பு — log file;
5. may preserve memory through a நினைவகக் கொட்டல் — memory dump.

The analyst:

1. reads each record according to its source and schema;
2. performs தடமறிதல் by following identifiers and marks;
3. reconstructs the route without assuming that every event was captured;
4. derives the relevant நிலை from the dump's relationships;
5. compares state, track, and record before assigning cause.

> *Tamil distinguishes the dumped contents of memory from the state inferred within them, the activity of tracing from the track it follows, and an individual record from the register and file that preserve it.*

Chapter 23 — Ancient Egyptian: ꜥ—rd—zḫꜣ.w · 

The Ancient Egyptian perspective is an explicitly historical analogy: **ꜥ—rd—zḫꜣ.w**, condition, footprint, and written record, shown in the TLA headword spellings as  . No claim is made that Ancient Egyptian possessed or anticipates a computing terminology. The point is narrower and more disciplined: attested lexical categories from the Thesaurus Linguae Aegyptiae[68–70] provide a triad with which to distinguish a state of affairs, the physical sign of passage, and a record that preserves information beyond the event.

Sources for this chapter: [68, 69, 70] in Sources and Further Reading.

Scope, period, and method

Ancient Egyptian is included as a historical conceptual perspective, not as a source of computing terminology. The language spans millennia and several written stages; meanings, spellings, and grammatical behavior vary by period and genre. The three forms used here are lemmata documented by the Thesaurus Linguae Aegyptiae[68–70]. Transliteration remains the primary analytical notation, while the hieroglyphic sequences shown beside it reproduce the TLA lemma-page headword spellings. Corpus spellings can vary by period and occurrence.

Analytical object	Egyptian lemma	Attested lexical range
Condition or state of affairs	**ꜥ** — 	Condition or state of affairs
Foot or footprint	**rd** — 	Foot; footprint
Writing or record	**zḫꜣ.w** —	Writing, record, or depiction

The transliterated triad **ꜥ—rd—zḫꜣ.w** and its hieroglyphic display  are deliberately modest. They do not project computers into antiquity. They ask how three attested lexical categories clarify the relation among a captured condition, the sign left by passage, and information preserved through writing.

1. Captured condition as ꜥ

The lemma **ꜥ**, written in the TLA headword spelling, denotes in the relevant sense a condition or state of affairs. It therefore supplies an analytical equivalent for what an investigator seeks in a memory dump: not memory as a substance, but the configuration that obtained at capture time.

Applied conceptually, the term focuses attention on coexistence. A thread owns a resource while another waits; a pointer refers to a damaged structure; a queue contains work that has not advanced; an exception object coexists with partially updated state. These relations define the state of affairs. The dump is the medium by which some of them become inspectable.

That object remains bounded. A captured ᶜ may be local rather than global, partial rather than complete, and downstream of actions no longer represented. The state constrains history without narrating it.

2. Trace as rd, the footprint

rd, shown as  in the TLA headword spelling, can mean a foot or footprint. The footprint is a powerful but disciplined model of trace evidence. It is caused by passage, remains after the moving agent has gone, and supports inference about direction, sequence, contact, and sometimes identity. Yet it is not the journey itself.

The same distinction applies to software traces. An emitted event marks one observation point. A series of events can be ordered into a route. A changed resource can be a footprint left in state. None preserves the entire execution. Each constrains what could have happened.

The footprint lens encourages five checks:

- determine what mechanism could make the mark;
- establish whether the mark is contemporaneous, displaced, or overwritten;
- distinguish one footprint from the route inferred across many footprints;
- inspect the surface—the instrumentation and storage system—that preserved it;
- test the inferred route against the captured condition.

It also clarifies absence. No visible footprint does not prove that no passage occurred. The surface may not preserve marks, another action may erase them, or the observer may have examined the wrong area.

3. Written record as zḫꜣ.w

The lemma **zḫꜣ.w**, shown as  in the TLA headword spelling, includes writing, record, and depiction. For the present comparison, it names information deliberately externalized so that an event, classification, or observation can remain available after the occasion of its production.

A software log is likewise a designed inscription. It selects what will be named, how it will be classified, which context will accompany it, and where it will be retained. The record may be truthful within its

viewpoint and still omit the decisive effect. It may report an intention before completion, or summarize a complex sequence under one label.

Reading a log as **zḫꜣ.w** therefore requires attention to production:

- Which component acted as writer?
- What conventions determined the form of the record?
- Does it preserve observation, interpretation, command, result, or depiction?
- What was not writable under the available schema and policy?
- Does the record agree with the footprints and state of affairs?

Writing extends memory but does not abolish selection. The record's permanence is not the same as completeness.

4. The Ancient Egyptian diagnostic analogy

No Egyptian interface sentences or unattested technical compounds are proposed. The workflow remains in English while preserving the attested analytical terms.

The system, by analogy:

1. has an inspectable **ꜥ** () — a condition or state of affairs;
2. moves through operations that leave **rd** () — footprints;
3. externalizes selected observations as **zḫꜣ.w** () — written records.

The analyst:

1. reads the record as a produced inscription;
2. identifies each footprint and the surface that preserved it;
3. reconstructs a route without confusing it with the marks;
4. examines the captured state of affairs;
5. requires writing, footprints, and condition to corroborate one another.

> *Ancient Egyptian contributes a historically bounded but analytically exact analogy: a condition can be inspected, a footprint can be followed, and a written record can outlast the event—yet none alone is the whole event.*

Chapter 24 — Basque: egoera—aztarna—erregistro

The Basque perspective is **egoera—aztarna—erregistro**: state, trace or vestige, and record. Elhuyar's dictionary[72–75] also attests **iraulketa** in computing for a dump or *volcado*, allowing the artifact to be named while **egoera** names what the analyst reconstructs from it. The chapter therefore keeps localization and analysis separate: a dump is produced, a condition is inferred, a trace is followed, and a record is tested.

Sources for this chapter: [71–75] in Sources and Further Reading.

Scope, sources, and technical usage

Basque technical terminology is documented through resources including the public Euskalterm terminology bank and Elhuyar's dictionaries[71–75]. Elhuyar attests[72–75] **egoera** for condition or state, **aztarna** for trace or vestige, **erregistro** for a register and an informatics record, and **iraulketa** in computing for a dump. The analytical triad is **egoera—aztarna—erregistro**.

Artifact or activity	Basque expression	Analytical emphasis
State or condition	**egoera**	Situation, condition, circumstances, or state
Computing dump	**iraulketa**	Dump or *volcado* in computing
Trace or vestige	**aztarna**	Trace, vestige, mark, indication, or footprint
Record	**erregistro**	Register; record in informatics

The compositional phrase **memoria-iraulketa** may transparently describe a memory dump, but this chapter does not present it as a universal standardized interface label. The attested term **iraulketa** names the operation or artifact family; **egoera** names the condition reconstructed from it.

1. Dump as iraulketa, evidence as egoera

Iraulketa evokes emptying, overturning, or dumping content into another form. That is an apt operational description of dump creation. **Egoera** answers the analytical question: what state, situation, or condition do the resulting contents support?

The distinction keeps capture and interpretation apart. Dump creation is a technical operation with a scope and policy. State reconstruction is an evidential argument. The same bytes may support different claims depending on type information, symbol quality, runtime knowledge, and external context.

The **egoera** lens directs the analyst to ask:

- Which system boundary does this state describe?
- Which values coexisted, and which relations connect them?
- Which apparent values may be stale, undecoded, or outside the capture?
- What earlier operation would explain both the state and the visible residue?

A dump can faithfully preserve a local **egoera** while failing to capture the distributed incident. Precision begins with naming the level at which the state claim is made.

2. Trace as aztarna

Aztarna covers trace, vestige, mark, and indication; it is also used in expressions for a digital footprint. The term therefore brings together physical residue and inferential sign. An **aztarna** exists now because something happened earlier, but its interpretation depends on the process that could have produced and preserved it.

For software diagnosis, this yields four layers:

- the instrumentation that makes a mark observable;
- the individual event or state change treated as an **aztarna**;
- the route reconstructed by correlating multiple marks;
- the residual effects that remain after the visible sequence ends.

The analyst should not assume that a displayed trace is identical to execution. It is a reconstruction from selected marks. Sampling, buffering, clock domains, and correlation rules influence the route. State may preserve an important vestige even where explicit trace events are missing.

The trace becomes strong evidence when its marks agree with independent condition and record. A route that exists only because one visualization joined ambiguous identifiers should be treated as a hypothesis, not a fact.

3. Record as erregistro

Erregistro names a register and, in informatics, a record. That double sense is analytically useful. A record is a structured unit; a register is an organized collection maintained under rules. A log participates in both.

Each record has a producer and a schema. The accumulated register has collection and retention policy. The analyst must examine both levels:

- What component created the **erregistro**?
- Which fields were observed and which were calculated or classified?
- Does the entry describe an attempted action or a completed effect?
- What buffering, redaction, rotation, and loss could affect the register?
- Which identifiers connect it to an **aztarna** and the captured **egoera**?

This prevents prose from receiving automatic priority over less readable evidence. “Operation succeeded” is a component's recorded classification. A changed resource, response from a dependency, or surviving object may confirm or contradict it.

4. The Basque diagnostic grammar

The following workflow uses attested words as analytical terms rather than prescribing a product localization.

The system:

1. has a changing **egoera** — condition or state;
2. moves through operations that leave **aztarnak** — traces or vestiges;
3. produces selected **erregistroak** — records;
4. may expose captured memory through an **iraulketa** — dump.

The analyst:

1. reads the records as designed statements;
2. follows the traces and identifies how they were preserved;
3. reconstructs the route while tracking uncertainty;
4. inspects the dump to establish the relevant state;
5. compares **erregistro**, **aztarna**, and **egoera** before accepting cause.

> *Basque separates the operation that dumps information from the state reconstructed within it, then joins vestige and record in a method that treats every diagnostic artifact as selected evidence.*

Chapter 25 — Udmurt: югдур—пытьы—гожъям

The Udmurt perspective is **югдур—пытьы—гожъям** (*jugdur—pyt'y—gožjam*): condition or situation, footprint, and something written down as a record. The vocabulary is grounded in the Udmurt National Corpus[76] dictionary and the Udmurt–Russian electronic dictionary[77, 78]. Because a standardized Udmurt equivalent for *memory dump* was not established in the consulted sources, the chapter retains recognizable technical terminology where necessary and uses the native triad as an analytical grammar.

Sources for this chapter: [76, 77, 78] in Sources and Further Reading.

Scope, sources, and a living technical register

Udmurt is a Uralic language of the Permic branch. This chapter draws on the dictionary of the Udmurt National Corpus[76] and the Udmurt–Russian electronic dictionary[77, 78] produced from lexicographic work of the Udmurt Institute. The resources attest **югдур** for condition or situation, **пытьы** for a footprint or track, **гожъям** for what has been written down as a record, and **гожтон** for the act of recording or writing.

Artifact or activity	Udmurt expression	Analytical emphasis
Condition or situation	**югдур** (*jugdur*)	State, condition, situation, or circumstances
Track or footprint	**пытьы** (*pyt'y*)	A footprint or track left by passage
Recording or writing	**гожтон** (*gožton*)	The action of writing or recording
Written record	**гожъям** (*gožjam*)	Something written down; a note or record
Journal or log	**журнал** (*žurnal*)	A borrowed but integrated term for a journal

No exact standardized Udmurt term for *memory dump* was established in the consulted sources. Russian-derived and international technical terms are common in specialist contexts. The analytical method does not conceal that fact: it keeps the artifact recognizable and uses the Udmurt triad **югдур—пытьы—гожъям** to state what kind of evidence is being examined.

1. Captured memory as югдур

Югдур can express a condition, situation, or set of circumstances. That breadth is useful because a memory dump rarely describes an isolated variable. It preserves a local situation consisting of relationships: ownership, waiting, reachability, configuration, accumulated work, and interrupted control.

The analyst should therefore avoid treating the dump as an automatic history. It is evidence of a **югдур** at a capture boundary. Some earlier actions may be inferable through their consequences, while the actions

themselves are absent. Other facts may lie in a remote service, storage system, or excluded memory range.

The condition lens creates four obligations:

- name the process, runtime, host, or service whose **югдур** is being claimed;
- distinguish direct values from decoded structures and higher-level inferences;
- identify relationships that make the captured facts diagnostically significant;
- state which history and external context the dump does not preserve.

This is especially important in multilingual technical environments. The artifact may retain an international or Russian-derived label, but the evidential claim can still be made precisely in the language's own terms.

2. Trace as пытьы

Пытьы names a footprint or track. Like other footprint metaphors in the comparison, it makes two facts visible at once: movement is no longer present, and a sign of that movement remains. The sign can be followed, but it must be interpreted.

A software **пытьы** may be an emitted event, a correlated identifier, a timing relation, a modified record, or a residue in memory. These signs belong to different layers. An event is produced by instrumentation; a path is reconstructed from several events; a residual state is an effect left by the path.

The analyst should ask:

- What mechanism created and preserved this footprint?
- Could sampling, buffering, clock skew, or identifier reuse distort it?
- Which route is directly observed and which is inferred?
- What effect should the route have left in the captured condition?
- Which alternative path could produce the same available footprints?

A footprint narrows possibility without eliminating uncertainty. Its evidential force grows when the expected state and written records support the same route.

3. Recording, written record, and journal

The distinction between **гожтон** and **гожъям** is valuable: one names the activity of writing or recording, while the other names what has been written down. The loan **журнал** names an accumulated journal and

demonstrates how Udmurt technical usage can integrate a wider regional vocabulary without losing native analytical distinctions.

This produces three levels:

- **гожтон** — the act and policy of recording;
- **гожъям** — an individual written note or record;
- **журнал** — the accumulated operational journal in which entries are consulted.

At each level, design choices matter. A component chooses whether to emit; a schema chooses what can be expressed; a journal pipeline chooses what is buffered, transformed, retained, or lost. A clear record may still be an interpretation rather than a direct observation.

The analyst should connect each **гожъям** to its producer, viewpoint, time source, and correlation identifiers. Then the entry can be tested against **пытьы** and **югдур**. If they disagree, the disagreement is evidence about instrumentation, timing, or the system—not a nuisance to be averaged away.

4. The Udmurt diagnostic grammar

The following is an analytical workflow. It does not invent Udmurt product commands or claim a standardized localization for every artifact.

The system:

1. has a changing **югдур** — condition or situation;
2. moves through operations that leave **пытьы** — footprints;
3. performs selective **гожтон** — recording;
4. creates **гожъям** — written records;
5. may accumulate them in a **журнал** — operational journal.

The analyst:

1. identifies the producer and policy behind each written record;
2. follows the footprints while preserving alternative explanations;
3. reconstructs the route within the limits of instrumentation;
4. inspects captured memory to establish the relevant condition;
5. lets condition, footprints, and records corroborate or challenge one another.

> *Udmurt adds a clear distinction between recording and what has been recorded, while the footprint and condition lenses keep the operational journal connected to movement and surviving state.*

Part III — Comparative Synthesis

26. What remains invariant

Across twenty-one very different linguistic histories, three invariants recur.

First, the dump is associated with a condition that can be held still: preserved existence, state, a snapshot, or the configuration in which the system stood. The emphasis is not on memory as a bag of bytes. It is on a temporary stabilization of relations. A thread is meaningful in relation to its stack, wait object, registers, and owner. An object is meaningful in relation to its fields, references, allocator, and surrounding structures. Dump analysis is therefore the reconstruction of a local world from a captured configuration.

Second, the trace is associated with motion and its legibility. The languages repeatedly distinguish the act of following from the route that is followed and from the residue that makes the route inferable. This is a foundational analytical distinction. Instrumentation performs capture; the system executes a path; the path generates events and effects; the analyst reconstructs an explanation. Collapsing all four into "the trace" hides where interpretation enters.

Third, the log is associated with deliberate preservation: a written account, a record, an entry in a journal, an inscription, or a protocol of proceedings. The repeated idea of writing matters. Logging is not passive sedimentation. Someone selected a vocabulary, severity model, schema, sampling policy, retention period, and failure behavior. Even automatically generated logs inherit intention from their instrumentation.

These invariants support a concise epistemic triad:

> ***State tells us what was there. Path and residue tell us how it came to be. Record tells us what the instrumented system said about it.***

27. What each language adds

Chinese: existence, mark, and account

The Chinese formulation is ontologically economical. **存** concerns what remains or is preserved. **迹** concerns the visible remnant from which prior movement can be inferred. **志** concerns an account deliberately committed to writing. The three concepts form a disciplined cycle of questioning rather than a hierarchy of sources. A log is used to locate a trail; the trail is used to locate relevant state; the state is used to test the account.

Korean: movement across lexical registers

Korean makes the layered nature of technical reasoning unusually visible. A borrowed artifact name can coexist with a formal Sino-Korean analytical term and a native word that conveys lived experience. **추적** is the activity of tracking. **궤적** is the trajectory. **자취** is the native word for what remains after something has passed, and **흔적** its Sino-Korean counterpart. The movement among registers resembles diagnostic work itself: from product terminology, through formal model, to concrete evidence.

Japanese: evidence is produced through verification

Japanese provides a disciplined ladder. **追跡** is following; **軌跡** is the travelled trajectory; **痕跡** is a remaining sign; **証跡** is an evidential trail. The last term should not be reached automatically. A message becomes evidence only when identity, order, context, integrity, and relevance have been established. The ladder is therefore not only lexical. It is a quality model for trace analysis.

Arabic: the trace is also an effect

Arabic أثر prevents an overly textual understanding of tracing. An execution may leave an explicit span, but it may also leave queue growth, memory pressure, cache pollution, a leaked capability, or corrupted metadata. These are not descriptions of the action; they are effects of the action. They may later become visible in a dump even when the originating trace messages have been lost. Arabic therefore supplies the bridge from temporal execution to spatial state.

Sanskrit: the present contains impressions of the past

Sanskrit संस्कार deepens the same bridge. Earlier action deposits an impression that helps shape subsequent state and behavior. In software, the metaphor applies to persistent and semi-persistent consequences: allocated objects, warmed caches, fragmented heaps, retry counters, learned parameters, altered context windows, accumulated contention, or poisoned state. The event can disappear while its impression remains active. Diagnostics must therefore correlate external footprints with internal impressions.

Russian: following traces becomes investigation

Russian **след** is a mark, remnant, indication, or consequence. Verbs built around the same semantic family express continuous observation and analytical reconstruction. **Следствие** then joins consequence and investigation. The system produces a consequence; the analyst conducts an investigation; both are linked by the traces that remain. The linguistic family models diagnosis as a causal and forensic practice rather than a search for a single error message.

German: configured condition and procedural record

German **Zustand—Spur—Protokoll** emphasizes disciplined procedure. A **Speicherabbild** is an image from which a system condition must be reconstructed. **Ablaufverfolgung** is the activity of following a process, while **Spur** is the indication it leaves. **Protokoll** is explicitly a maintained record of operations. The German perspective therefore highlights the distinction between a process, the tracks that survive it, and the procedural account created under an instrumentation policy.

Irish: tracking, trail, and stored record

Irish distinguishes **rianú**, the activity of tracing, from **rian** and **lorg**, the trace or trail left behind. **Taifead** then names stored information and a computing record, while **loga** names the operational log. This vocabulary makes the analyst's movement concrete: follow the trail, reconstruct its route, inspect the state, and decide when a mark has enough context and integrity to count as **fianaise**, evidence.

French: state, impression, and chronology

French **état—trace—journal** organizes evidence temporally. The dump is an **instantané** of state. A **trace** is a surviving indication, a **piste** is the investigative route suggested by connected traces, and an **empreinte** is an impression left on later state. The **journal** accumulates selected entries through time. French therefore connects snapshot, residue, inferred course, and chronology without collapsing them into one artifact.

Classical Greek: established condition and externalized memory

Classical Greek **κατάστασις** carries both establishment and settled condition: present state is something prior action has brought about.”**Ἴχνος** is the footprint from which absent movement is inferred.‘**Ὑπόμνημα** is a memorandum or record made for later recollection and use. The perspective distinguishes what happened, what it established, what footprints it left, and what was written so that later inquiry could begin.

Latin: vestige and investigation

Latin **status—vestigium—commentarius** joins standing condition, surviving footprint, and accumulated working record. **Vestigatio**, investigation by following vestiges, places analytical method in the same semantic family as the traces it examines. **Acta** adds a further distinction between actions and actions formally admitted to the record. Latin therefore exposes the gap between event, vestige, recorded proceeding, and later commentary.

Southern Quechua: footprint and evidential source

Southern Quechua **kay—yupi—qillqa** joins being or existing, the footprint left by passage, and writing or document. The tracking verbs around **yupi** distinguish the mark from the investigator's active work.

Quechua evidential systems add another discipline: a claim may be directly known, reported, or inferred. Diagnostic reports should preserve the same distinction even when English grammar does not require it.

Swahili: condition, follow-up, and retained memory

Swahili **hali—alama—kumbukumbu** separates state from the mark that indicates prior activity and from the record retained for later consultation. **Kufuatilia** makes diagnosis an active follow-up or investigation rather than passive viewing. **Kumbukumbu** connects records with remembrance, clarifying that logs are aids to institutional memory whose coverage depends on deliberate selection and preservation.

Turkish: trace activity and registered entry

Turkish **durum—iz—kayıt** distinguishes condition, remaining or emitted trace, and registered record. Technical compounds such as **izleme dosyası** and **izleme günlüğü** keep the tracing activity, trace file, and trace log visible as related but different objects. **Kayıt** and **günlük** then distinguish an individual entry from the accumulated operational journal.

Māori: captured form, footprint, and technical log

Māori **āhua—tapuwae—rangitaki / mauhanga** connects a system's condition and form with footprints left by movement and a record its technical and everyday registers name differently. **Tūnga** adds the position of an entity within the captured configuration. The footprint model insists that the reconstructed route remains an interpretation of surviving marks, while **rangitaki** anchors the account in attested computing terminology and mauhanga anchors it in the ordinary language of records.

Cherokee: condition, trace, tracer, and retained record

Cherokee **ᎤᏍᏗᏓᏅᏙ—ᎠᏍᏓᏩᏛᏍᏗ—ᏚᎦᏅᎤᏛᎢ** joins condition, trace, and record. The attested pair **ᎠᏍᏓᏩᏛᏍᏗ** (*Trace*) and **ᎠᏍᏓᏩᏕᎩ** (*Tracer*) keeps captured indication separate from the role or mechanism that follows and records it. The longer technical term for *log file* makes retention visibly designed. Cherokee therefore adds the tracer's viewpoint and capture policy to the comparison: evidence must be read together with the instrument or analyst that made it available.

Tibetan: condition, aftermath, and organized register

Tibetan གནས་སྟངས་—རྗེས་ཤུལ་—ཐོ་ཡིག་ joins a situated condition with the trail, remains, or aftermath left by action and with a register arranged for later use. རྗེས་ཤུལ་ widens trace analysis beyond emitted events

to include legacy and surviving effects. The distinction between ཟིན་ཐོ་, a record, and ཐོ་ཡིག་, an organized list or register, makes the design and curation of a log independently visible.

Tamil: dumped memory, track, and recorded entry

Tamil நிலை—தடம்—பதிவு distinguishes the state inferred by an analyst, the track or impression left by movement, and the act or object of recording. The attested technical expression நினைவகக் கொட்டல் names the memory dump operationally, while நிலை names its epistemic object. பதிவு, பதிவேடு, and பதிவுக் கோப்பு separate entry, register, and log file.

Ancient Egyptian: condition, footprint, and writing

Ancient Egyptian **ḥ—rd—zḫꜣ.w** supplies a deliberately bounded historical analogy. A condition can be established, a footprint can survive a movement, and writing can preserve a record after the event. The distance from modern computing is methodologically useful: it forces the comparison to remain conceptual and prevents an analytical metaphor from being presented as a historical technical term.

Basque: state, vestige, and informatics record

Basque **egoera—aztarna—erregistro** separates the state reconstructed from a dump from the vestiges by which a path is inferred and the records designed for consultation. The computing sense of **iraulketa** names the dump operation, while **egoera** names what the investigator seeks within it. **Aztarna** keeps a trace connected to the material or digital mark that survives passage.

Udmurt: situation, footprint, and what was written down

Udmurt **югдур—пытьы—гожъям** joins a situated condition, the footprint of prior activity, and the written record that remains available afterward. The distinction between **гожтон**, recording as an activity, and **гожъям**, what has been recorded, separates instrumentation policy from its output. The loan **журнал** shows that a living technical register can integrate borrowed artifact names while retaining native analytical distinctions.

28. From three artifacts to five analytical objects

The comparison suggests that the conventional triad should be expanded into five analytical objects. The artifacts remain three; the objects of reasoning are five.

Analytical object	Meaning	Typical evidence
State	A captured configuration of entities and relations	Memory dump, snapshot, checkpoint

Analytical object	Meaning	Typical evidence
Emitted trace	An explicitly observable event or measurement	Span, trace message, timestamp, counter update
Reconstructed path	An ordered and causal interpretation of events	Correlated trace, request graph, execution sequence
Residual effect	A lasting impression produced by earlier activity	Queue depth, altered memory, leaked resource, stale cache
Recorded account	A designed statement about events or conditions	Log entry, journal, audit record, protocol

The distinction between emitted trace and reconstructed path is essential. A trace file is not a path in the same way that scattered footprints are not yet a route. Correlation, ordering, missing-event analysis, and causal interpretation turn messages into a path.

The distinction between emitted trace and residual effect is equally important. Some actions emit messages but leave no durable state. Others leave major state changes but emit no messages. Many failures arise in the gap: the trace says what was attempted, while the dump shows what the attempt actually left behind.

29. A worked diagnostic example

Consider a distributed checkout service that intermittently returns “payment timeout.” The application log supplies a readable account: a payment call exceeded its deadline. If the log is treated as complete truth, the investigation may stop at the payment service.

The trace tells a larger story. A request enters checkout, invokes inventory, calls payment, retries twice, and spends increasing time waiting for a worker. The reconstructed path reveals that the final timeout is preceded by a retry loop and delayed scheduling. Yet the trace still does not show why workers are unavailable.

A memory dump taken during the incident exposes the state: most worker threads are blocked on a shared pool; a queue has grown; several request objects remain reachable; and the retry mechanism has multiplied outstanding work. Residual effects connect the sequence to the state. Earlier slow responses caused retries; retries increased queue depth; queue depth delayed scheduling; delayed scheduling produced more timeouts. The “payment timeout” message was accurate as an observation and misleading as a causal summary.

Read through the twenty-one-language model:

- The log is the **record or account** that tells the analyst where to begin.
- The trace provides **steps, marks, and trajectory** that show how the request moved.
- Retry amplification appears as **effect and residual impression**, not merely as another message.
- The dump captures the **state** in which those effects coexist.
- Corroboration turns the collected marks into **evidence**.
- The final explanation is a **causal investigation**, not a paraphrase of the first error entry.

This example also shows why time and state must be analysed together. The trace explains the production of the dump state; the dump explains the consequences that make the later trace intelligible.

30. A unified diagnostic cycle

The strongest cycles proposed in the twenty-one perspectives can be combined into one method.

Step 1 — Read the account

Begin with logs because they usually contain the most accessible vocabulary: component names, operation names, severities, identifiers, and approximate times. Treat them as orientation, not verdict. Ask what the system was designed to record, which layer is speaking, and what could have been filtered or lost.

Step 2 — Find and follow the marks

Locate trace messages, spans, counters, identifiers, timestamps, and absences relevant to the account. Distinguish active tracking from the artifacts it produces. A missing expected event may be as important as a present one.

Step 3 — Reconstruct the path

Order events, correlate identities, account for clock uncertainty, and test alternative causal sequences. Do not confuse chronology with causality. A meaningful trajectory is an analytical construction constrained by evidence.

Step 4 — Inspect captured state

At significant points on the path, inspect a dump, snapshot, checkpoint, or other preserved state. Look for structures that support or contradict the proposed path: blocked threads, object lifetimes, partial updates, resource ownership, queue contents, loaded modules, and corrupted relations.

Step 5 — Search for residual effects

Ask what earlier execution left behind. This includes memory changes, leaked resources, accumulated delay, stale data, context growth, counters, fragmented storage, and altered configuration. Residual effects often bridge the gap between a trace event and a later failure state.

Step 6 — Corroborate and challenge

Require each evidence family to question the others. Does state support the reconstructed path? Does the path explain the log? Do identifiers and timestamps refer to the same entities? Could the log describe a symptom while the dump preserves the cause? Contradiction is not a nuisance to be discarded; it is often the most informative evidence in the case.

The forward movement is:

> ***record → marks → path → state → residual effect → verification***

The reverse movement is equally important:

> ***state → residual effect → preceding action → emitted traces → recorded account → reconstructed cause***

31. Confidence as mutual corroboration

Diagnostic confidence should not be inherited from artifact prestige. A complete memory dump can be irrelevant, a distributed trace can be sampled, and an audit log can be semantically wrong. Confidence arises from converging constraints.

Useful corroboration dimensions include:

- **Identity:** the same request, thread, process, object, model invocation, or transaction is being discussed.
- **Time:** timestamps and capture times are compatible after accounting for clock and buffering behavior.
- **Order:** the proposed sequence respects known ordering constraints.
- **State:** the captured configuration is a plausible result of the proposed actions.
- **Semantics:** messages and fields mean what the interpretation assumes they mean.
- **Integrity:** evidence has not been dropped, overwritten, sampled away, or altered.
- **Perspective:** the artifacts describe the same event from compatible abstraction layers.

No single dimension proves the explanation. Together they can make alternative explanations increasingly difficult to sustain.

Part IV — Design Implications

32. Observability should distinguish message, path, and effect

Observability platforms often use *trace* as the name of a stored hierarchy of spans. That representation is valuable, but the linguistic comparison suggests a richer design vocabulary. Systems should distinguish:

- the activity of tracing;
- the individual emitted trace event;
- the path reconstructed from correlated events;
- the residual effect left in system state;
- the evidential status achieved after verification.

This distinction improves both schemas and user interfaces. A platform can show raw events separately from inferred causality. It can label gaps, sampling, and uncertainty. It can connect a span to stateful consequences such as queue growth or heap retention. It can prevent a visually coherent waterfall from appearing more complete than the data warrants.

33. Logs should expose their viewpoint

If logs are designed accounts, their viewpoint should be explicit. An entry should make it possible to determine which component observed the condition, which abstraction layer classified it, which identifiers were available, and whether the message describes a cause, a local symptom, or an attempted response.

Good logging design therefore includes more than adding context fields. It includes epistemic metadata:

- observer or component identity;
- event time and recording time;
- correlation identity and scope;
- confidence or classification basis where appropriate;
- causal parent or triggering condition when known;
- versioned schema and vocabulary;
- indication of sampling, aggregation, or loss.

The goal is not to make every entry verbose. It is to make the limitations of the account inspectable.

34. Dumps should be connected to the path that made them necessary

A dump is often collected after a threshold, exception, crash, watchdog action, or manual intervention. The trigger itself should be correlated with traces and logs. Capture metadata should include the reason, relevant identifiers, process and host identity, monotonic and wall-clock time, software version, and the diagnostic policy that selected the dump type.

This turns an isolated snapshot into a point on a reconstructed trajectory. It also supports comparison among multiple snapshots: which relations changed, which residual effects accumulated, and which expected recovery did not occur?

35. Agentic and AI systems

The model becomes especially useful for AI and agentic systems because their “state,” “trace,” and “record” are frequently conflated.

An agent log may narrate an intention: “I will query the database.” A tool trace may show that a call was attempted. A state snapshot may reveal the actual context, memory, tool permissions, pending tasks, cached results, or partial outputs. Residual effects may include context growth, duplicated work, altered memory, rate-limit consumption, or a changed external resource. The narration, action, and consequence can diverge.

The twenty-one-language distinctions therefore support several safeguards:

- Treat model-generated explanations as accounts, not privileged truth.
- Separate planned action from executed path.
- Preserve tool-call and environment traces independently of natural-language narration.
- Snapshot relevant agent state at decision boundaries.
- Track residual effects on context, memory, budgets, and external systems.
- Require cross-artifact evidence before assigning causal responsibility.

For learning systems, Sanskrit संस्कार is particularly suggestive. A prior action may alter a cache, retrieval index, prompt state, model memory, or fine-tuned parameter and thereby shape later behavior after the original event has disappeared. Arabic أثر and Tibetan རྗེས་ཐུབ་ emphasize that consequential aftermath

may be more diagnostically important than the emitted trace. Japanese **証跡** reminds us that a stored event becomes evidence only through integrity and verification. Russian **следствие**, Latin **vestigatio**, Irish **rianú**, Swahili **ufuatiliaji**, Quechua **qhatipay**, Cherokee **ᎠᏍᏓᏩᏗᏍᏗ**, Tamil தடமறிதல், Basque **aztarna**, Egyptian **rd**, and Udmurt **пытьы** keep trace-following connected to investigation and surviving marks. German **Protokoll**, French **journal**, Greek **ὑπόμνημα**, Latin **commentarius**, Turkish **günlük**, Māori **rangitaki**, Cherokee **ᏚᎦᎤᏅᎢ**, Tibetan ཇོ་ཡིག་, Tamil பதிவு, Basque **erregistro**, and Udmurt **гожъям** emphasize that a stored account is an aid to later judgment, not privileged access to the model's actual causal process.

36. Limits and cautions

This comparative method has limits that should remain visible.

First, the proposed triads are analytical constructions. Speakers, documentation teams, and technical communities use many terms, and usage differs by country, product, period, and specialization. A memorable formulation should not be mistaken for a universal lexical rule.

Second, the article does not claim that language determines diagnostic ability. The same distinctions can be expressed in English, and experts routinely make them without multilingual analysis. Translation is used here as a device for discovering and stabilizing distinctions.

Third, etymology does not settle present technical meaning. A word’s history can generate a productive metaphor, but current usage, system architecture, and evidence quality remain decisive.

Fourth, no artifact is inherently primary. Some incidents are solved almost entirely from traces; others require a dump; others depend on logs, metrics, network captures, storage images, model checkpoints, or external business evidence. The triad is a core, not a boundary.

Finally, corroboration does not guarantee certainty. Evidence can share the same blind spot because several artifacts were produced by the same faulty clock, schema, component, or instrumentation library. Independent perspectives matter.

37. Conclusion

Memory dumps, traces, and logs are often placed beside one another as items in a diagnostic toolbox. The multilingual comparison shows that they are better understood as three families of evidence and as several distinct analytical operations.

The dump concerns preserved existence and captured state. The trace concerns following, movement, route, footprint, residue, and consequence. The log concerns inscription, record, journal, memorandum, and account. Around this stable triad, the twenty-one languages add complementary insights: Chinese provides a compact cycle of account, trail, and state; Korean moves among technical, formal, and experiential registers; Japanese distinguishes following, trajectory, trace, and evidential trail; Arabic joins trace with effect; Sanskrit joins action with persistent impression; Russian joins trace, consequence, and investigation; German joins configured condition with procedural recording; Irish separates tracing, trail, stored record, and evidence; French separates mark, investigative route, impression, and chronology; Classical Greek joins established condition, footprint, and externalized memory; Latin joins vestige with investigation and working commentary; Southern Quechua adds evidential source to footprint and writing; Swahili joins condition with active follow-up and retained memory; Turkish separates trace activity, mark, entry, and journal; Māori joins captured form, footprint, and a log its technical and everyday registers name differently; Cherokee separates condition, trace, tracer, individual record, and retained log file; Tibetan joins situated condition with trail, aftermath, and register; Tamil separates a memory dump from the state inferred in it and a track from its tracing; Ancient Egyptian supplies the bounded analogy of condition, footprint, and writing; Basque distinguishes dump operation, state, vestige, and record; Udmurt separates recording from what was written down while connecting both to footprint and condition.

Together they produce a more precise diagnostic discipline. Read what the system said. Follow what it left. Reconstruct how it moved. Inspect what remained. Identify what earlier activity changed. Then let every representation test the others.

> *The system's state is not its history. Its trace is not its whole state. Its log is not the whole truth. Diagnosis begins where these partial forms of evidence meet.*

Sources and Further Reading

The links are a selected subset of the localized technical documentation and lexicographic resources. Accessed July 2026.

- [1] Microsoft Learn — Dumps (.NET), Simplified Chinese
- [2] Microsoft Learn — Trace logs, Simplified Chinese
- [3] Chinese Text Project — 志
- [4] Microsoft Learn — Dumps (.NET), Korean

- [5] National Institute of Korean Language — Basic Korean Dictionary
- [6] Microsoft Learn — Dumps (.NET), Japanese
- [7] Microsoft Learn — Trace logs, Japanese
- [8] Kotobank — 軌跡
- [9] Kotobank — 痕跡
- [10] Kotobank — 証跡
- [11] Microsoft Learn — Dumps (.NET), ar-sa locale (interface Arabic; body text not translated)
- [12] Arabic Language Academy in Cairo — dictionary search
- [13] Monier-Williams Sanskrit-English Dictionary
- [14] Sanskrit Dictionary — अवस्था
- [15] Internet Encyclopedia of Philosophy — Yoga (discussion of saṃskāra)
- [16] Microsoft Learn — Dumps (.NET), Russian
- [17] Microsoft Learn — Trace logs, Russian
- [18] Gramota.ru — след
- [19] Gramota.ru — следствие
- [20] Microsoft Learn — Dumps (.NET), German
- [21] Microsoft Learn — Trace logs, German
- [22] Duden — Zustand
- [23] Duden — Spur
- [24] Duden — Protokoll
- [25] Foclóir.ie — dump
- [26] Foclóir.ie — state
- [27] Foclóir.ie — trace
- [28] Foclóir.ie — trail

- [29] Foclóir.ie — record
- [30] Foclóir.ie — log
- [31] Microsoft Learn — Dumps (.NET), French
- [32] Microsoft Learn — Trace logs, French
- [33] CNRTL — état
- [34] CNRTL — trace
- [35] CNRTL — journal
- [36] LSJ — κατάστασις
- [37] Logeion — ἴχνος
- [38] LSJ — ὑπόμνημα
- [39] Logeion — ἀναγραφή
- [40] Logeion — status
- [41] Logeion — vestigium
- [42] Logeion — commentarius
- [43] Logeion — acta
- [44] Bolivian Ministry of Education — Iskay Simipi Yuyayk'ancha: Quechua–Spanish bilingual dictionary
- [45] Peruvian Ministry of Education — Diccionario escolar del quechua central (Central Quechua; cited for contrast, not as a Southern Quechua source)
- [46] TUKI — Kamusi ya Kiswahili–Kiingereza
- [47] Institute of Kiswahili Research — Kamusi ya Kiswahili Sanifu
- [48] Microsoft Learn — Memory dump file options, Turkish
- [49] Microsoft Learn — netsh trace, Turkish
- [50] Microsoft Learn — EventSource introduction, Turkish
- [51] Annual Review of Linguistics — The Unity and Diversity of Altaic
- [52] Te Aka Māori Dictionary — āhua

- [53] Te Aka Māori Dictionary — tapuwae
- [54] Te Aka Māori Dictionary — mauhanga
- [55] Taiuru — A Dictionary of Māori Computer Related Terms, Edition 2
- [56] Te Aka Māori Dictionary — rangitaki (blog; everyday register)
- [57] Paekupu — Hangarau Matihiko (Māori to English)
- [58] Taiuru — A Dictionary of Māori Computer Related Terms, Edition 1 (kōnae pūrongo for log file)
- [59] Glosbe entry attributed to Microsoft Language Portal — rangitaki: log and blog; kōnae rangitaki: log file; rangitaki hapa: error log
- [60] Cherokee Nation Language Department — Word List
- [61] Cherokee Nation — Cherokee Language Technology
- [62] Cherokee-English Dictionary — Cherokee Nation and Microsoft computer terms
- [63] Tibetan–English–Sanskrit Dictionary — གཞན་སྣང་
- [64] Tibetan–English–Sanskrit Dictionary — རྗེས་ཐུལ་
- [65] Tibetan–English–Sanskrit Dictionary — ཐོ་ཡིག་
- [66] Tamil Computer Encyclopedic Dictionary — memory dump terminology
- [67] Tamil Wiktionary — memory dump
- [68] Thesaurus Linguae Aegyptiae — ḏ, condition or state of affairs
- [69] Thesaurus Linguae Aegyptiae — rd, foot or footprint
- [70] Thesaurus Linguae Aegyptiae — zḫꜣ.w, writing or record
- [71] Euskalterm — Basque public terminology bank
- [72] Elhuyar Dictionary — egoera
- [73] Elhuyar Dictionary — aztarna

- [74] Elhuyar Dictionary — erregistro
- [75] Elhuyar Dictionary — iraulketa
- [76] Udmurt National Corpus — dictionary
- [77] Udmurt–Russian electronic dictionary — 2008 academic edition
- [78] Udmurt Institute — Udmurt–Russian electronic dictionary project