Quuipu

QUUIPU RESEARCH & ARCHITECTURE SERIES

WHITEPAPER 06

AI

Context Before Intelligence: Integrating LLMs into Cognitive Web Infrastructure



Foundational Whitepaper · English

Quuipu Edition 2 · v1

Publication identity

Publication
AI

Series
Quuipu Research & Architecture Series

Publication number
Whitepaper 06

Edition
Quuipu Edition 2 · v1

Language
English

Historical edition
Cognisphere Edition 1

This publication succeeds Cognisphere Edition 1. Historical PDFs remain immutable.

Contents

01. Context before intelligence

02. A non-LLM configuration can already deliver a cognitive experience through rules, semantic maps, source

03. LLM integration adds linguistic elasticity, synthesis, transformation and the ability to handle unanticipated

04. Reference architecture

05. AI Gateway

06. Model abstraction

07. Model routing

08. Context packaging

09. RAG as retrieval discipline

10. Memory

11. Prompt minimization

12. Token economics

13. Privacy architecture

14. Local models

15. Mistral as an adapter, not a dependency

16. Other providers

17. Tools and function calling

18. Structured outputs

19. Human control

20. AI transparency

21. Security and prompt injection

22. Evaluation

23. AI failure modes

24. Operating model

25. AI deployment patterns

26. AI invariants

27. The category boundary

28. The final principle

A. AI Gateway request envelope

B. Model Router decision matrix

C. AI risk / control matrix

D. AI-01 · AI is optional

Preamble

Context Before Intelligence: Integrating LLMs into Cognitive Web Infrastructure

> This publication succeeds Cognisphere Edition 1. Quuipu Edition 2 preserves the historical thesis while updating active brand and publication identity.

Preamble

Context Before Intelligence: Integrating LLMs into Cognitive Web Infrastructure

FOUNDATIONAL THESIS

Context before intelligence The LLM extends Quuipu; it does not define it.

Quuipu does not require an LLM to exist. An LLM can extend Quuipu's capabilities.

Context / Intent / Source / Policy Deterministic Rules, semantic maps, curated representations

Context / Intent / Source / Policy AI Gateway Model Router → Mistral / Other / Local

REFERENCE ARCHITECTURE

Deterministic + AI ↓

Decision: deterministic or AI ↓

AI Gateway ↓

Model Router ↓

Mistral / Other / Local ↓

Validation & Trust ↓

Experience Layer

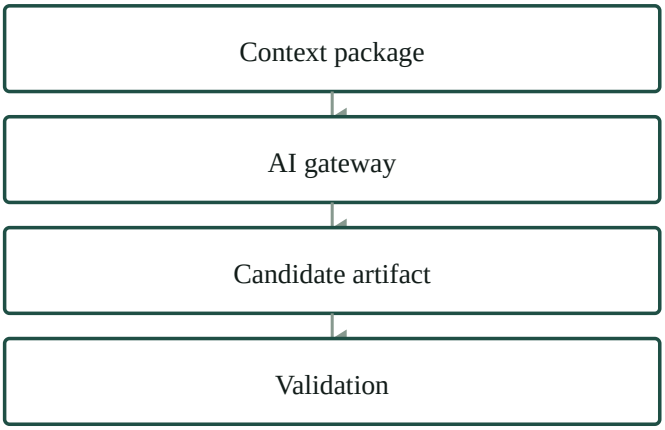


Figure 1 — Quuipu AI reference flow.

The model is downstream of context, source authority and policy - and upstream of validation, provenance and human control.

CHAPTER 01

Context before intelligence

Quuipu does not originate from a language model. It originates from the idea that a digital experience should resolve context, intent, continuity, meaning and source first, and only then decide whether generative intelligence adds value.

The thesis Quuipu does not require an LLM to exist. An LLM can extend Quuipu's capabilities. This separates the Cognitive Web category from the chatbot category.

Why context first A model can generate language without knowing which source has authority, which version is current or which objective the user explicitly declared. Quuipu resolves these constraints before inference.

AI as optional capability The AI Gateway is replaceable and can be disabled. The site, canonical source, navigation and deterministic transformations continue to exist without an LLM provider.

CHAPTER 02

A non-LLM configuration can already deliver a cognitive experience through rules, semantic maps, source

contracts, context resolution and curated representations.

Deterministic runtime Intent mapping, topic routing, source selection, relationship traversal and fallback can be deterministic.

Curated transformations Original, Simple or Step-by-step can use predefined editorial representations where the domain requires maximum control.

Semantic navigation Explore can traverse a canonical relationship graph without free generation.

Why this matters Operating without an LLM avoids lock-in, reduces cost and makes clear that Quuipu is infrastructure rather than a conversational UI.

Capability Without LLM With LLM

Understand Curated/simple variants Dynamic simplification within contract

Explore Canonical relationship graph Bounded synthesis over retrieved evidence

Navigation Rules + semantic mapping Natural-language intent interpretation

Fallback Canonical content Same canonical fallback remains

CHAPTER 03

LLM integration adds linguistic elasticity, synthesis, transformation and the ability to handle unanticipated

requests while preserving runtime boundaries.

Extension, not replacement The model extends capabilities; it does not replace source authority, the context engine, policy, provenance or fallback.

Candidate generation Generative output is a candidate validated against evidence, claimability and Semantic Integrity.

Bounded tasks The model may simplify, explain, summarize, compare or structure within an explicit task contract.

No chatbot requirement An LLM capability can be invoked behind Experience, Understand or Explore without exposing a general-purpose chat.

CHAPTER 04

Reference architecture

The architecture separates the deterministic path from the AI path. The Core decides when a request needs generative inference and sends it to the AI Gateway.

Core boundary

AI Gateway Mediates between runtime and provider: normalizes requests, policy, timeout, privacy, logging and the response envelope.

Model Router Selects a model according to task, privacy, latency, cost, modality and deployment policy.

Provider adapters Mistral, other providers or local models sit behind a common adapter contract.

Return path The response returns through the validation layer before it becomes an experience.

Layer Owns Must not own

Core Context, state, source, policy Provider-specific semantics

AI Gateway Request policy, privacy, limits Canonical truth

Model Router Model selection Experience authority

Adapter Provider protocol Domain logic

Validation Evidence, schema, integrity Free-form UX rendering

CHAPTER 05

AI Gateway

The AI Gateway is the control point that prevents the provider from becoming the architecture.

Normalized contract The runtime sends a provider-independent task package instead of building product logic around one API.

Policy enforcement The gateway applies model allowlists, data class, timeout, token ceilings, tool permissions and fallback.

Observability It records operational metrics and minimized decision trace without requiring behavioral profiling.

Failure containment Rate limits, provider outages or invalid output degrade to another model or to the deterministic runtime.

CHAPTER 06

Model abstraction

Model abstraction keeps Quuipu stable while models, versions and providers evolve.

Capability descriptors Each model can declare modality, context window, structured output, tool support, latency class, privacy class and cost class.

No semantic coupling The domain should request capabilities rather than know model names.

Version awareness Model ID and version become part of provenance when an output is AI-derived.

Lifecycle Deprecations and replacements are handled by router and release gates rather than scattered through experience code.

CHAPTER 07

Model routing

Routing means selecting the smallest adequate model for a task rather than sending everything to the largest model.

Task fit Classification, short rewriting, complex synthesis and multimodality can require different model classes.

Latency fit Inline interactions favor fast models; asynchronous processes may use deeper models.

Privacy fit Sensitive data may require a specific region, provider or local deployment.

Cost fit Budgets and token ceilings can be policy inputs.

Resilience Router and gateway can support provider fallback or deterministic fallback without changing the UI.

Routing dimension Example policy Effect

Task rewrite vs synthesis Choose capability class

Latency inline < asynchronous Prefer faster model

Privacy local-only / EU-region Restrict deployment

Cost token ceiling Small-first / fallback

Resilience provider health Alternate adapter

CHAPTER 08

Context packaging

Quuipu does not hand the model an entire session. It constructs the minimum context package required for the task.

Declared intent Explicit user intent is preferred to behavioral inference.

Relevant state Only state necessary to the current decision enters the package.

Source excerpts Retrieval supplies targeted evidence with identity, locator and version.

Policy envelope The package includes transformation contract, forbidden claims, output schema and disclosure requirements.

Context budget Every element must justify its token, privacy and risk cost.

Declared intent ↓

Relevant state ↓

Authority-filtered retrieval ↓

Evidence excerpts ↓

Transformation contract ↓

Output schema

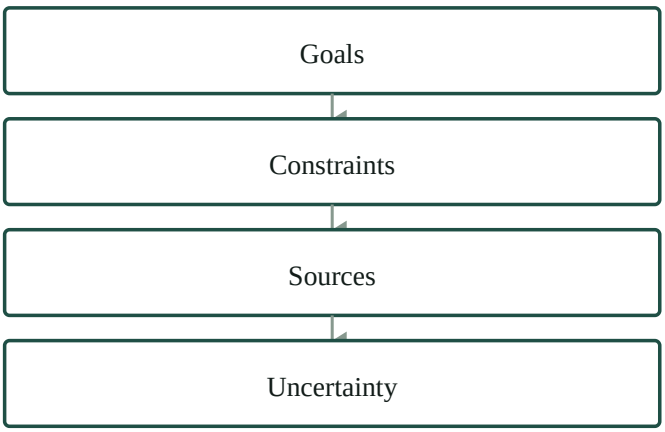


Figure 2 — Context package.

CHAPTER 09

RAG as retrieval discipline

RAG in Quuipu is not simply 'adding documents to the prompt'. It is a pipeline of source resolution, retrieval, evidence selection and provenance binding.

Authority-aware retrieval Search begins from allowed sources and authority scope rather than an indiscriminate corpus.

Chunk identity Every chunk retains source identity, version and locator.

Retrieval ranking Semantic relevance is combined with authority, freshness and scope.

Evidence map Selected passages are associated with the claims they can support.

Post-generation validation The response is checked against evidence and policy before emission.

Source authority ↓

Retrieval ↓

Chunk identity + version ↓

Evidence ranking ↓

LLM candidate ↓

Claim validation ↓

Provenance binding

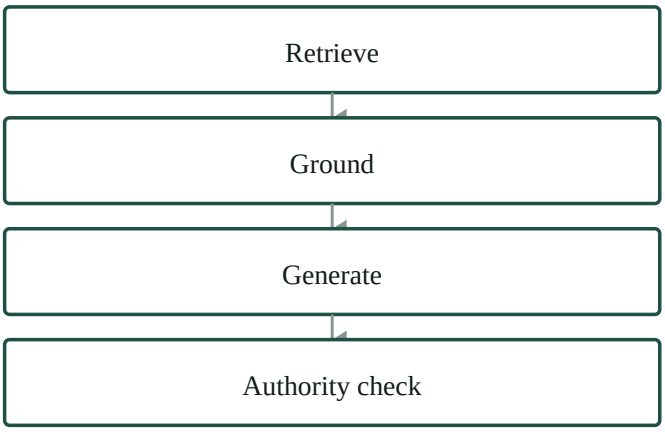


Figure 3 — Authority-aware RAG.

CHAPTER 10

Memory

Memory should serve continuity without becoming unlimited profiling.

Three memories It is useful to distinguish session state, explicit continuity memory and domain knowledge memory.

User-declared continuity Persistent preferences or goals should be explicit, revocable and separate from implicit behavior.

Model memory is not canonical memory A provider memory feature does not replace Quuipu's governed memory/state layer.

Retrieval over replay Where possible, retrieving structured state is preferable to resending long transcripts.

Expiry TTL, deletion and scope are part of the memory contract.

Memory class Typical lifetime AI use

Session state Minutes / session Current intent and navigation

Explicit continuity User/org governed Persistent goals or preferences

Domain knowledge Versioned Retrieval source

Provider memory Provider-specific Optional; never canonical

CHAPTER 11

Prompt minimization

The longest prompt is not necessarily the best prompt. Quuipu aims to send only the instructions and evidence required.

Stable system contract Invariant rules are compact, versioned and separated from content.

Structured context JSON or structured envelopes reduce ambiguity compared with concatenated transcripts.

No decorative context Unnecessary information is excluded even if it might make output appear more personalized.

Security benefit Less context reduces leakage and prompt-injection surface.

Economic benefit Fewer tokens reduce cost and latency.

CHAPTER 12

Token economics

AI cost should be a measurable architectural variable rather than a production surprise.

Cost per cognitive action Measuring input/output tokens and cost per capability makes Experience, Understand and Explore comparable.

Small-first routing Simple tasks can use small models or deterministic logic.

Cache carefully Caching is useful only when source version, context and policy make reuse safe.

Budget ceilings Tenant, session or capability can have cost limits.

Value threshold AI is invoked when expected benefit exceeds cost, latency and risk.

Cost lever Architecture response Trade-off

Input tokens Prompt minimization + retrieval May require better ranking

Output tokens Structured concise schema Less free prose

Model size Small-first routing Escalation needed for hard tasks

Repeated work Version-safe cache Invalidation complexity

Tenant budget Quota / ceiling Possible deterministic fallback

CHAPTER 13

Privacy architecture

The AI path must respect the same privacy boundaries as the rest of Quuipu.

Data minimization Only data necessary for the task is sent to a provider.

Classification Data classes can determine whether a request is cloud-eligible, region-bound or local-only.

No raw credentials Host tokens, cookies or secrets never enter the context package.

Provider controls Retention, training use, region and enterprise controls must be verified for each provider and plan.

Local option For some domains, a local model can reduce external transfers and increase control.

CHAPTER 14

Local models

Local models make it possible to move inference closer to data or device, while introducing new operational trade-offs.

Benefits Sovereignty, local latency, offline capability and lifecycle control.

Costs Hardware, upgrades, evaluation, security and capacity planning become deployment responsibilities.

Hybrid routing The router can use local-first and cloud escalation for more complex tasks.

Same contracts Local is neither trusted nor untrusted by default: provenance, evaluation and policy still apply.

CHAPTER 15

Mistral as an adapter, not a dependency

Mistral is a particularly relevant example for a European and hybrid strategy, but Quuipu should not become semantically dependent on a single vendor.

Why relevant Mistral offers APIs, open-weight and commercial models, RAG tooling and deployment/self-hosting options; current documentation also describes enterprise and privacy controls.

Adapter contract Integration should sit behind a MistralAdapter mapping capabilities, request envelope, errors, usage and provenance.

Routing choice A Mistral model is selected when it best satisfies task, cost, privacy or deployment constraints.

No endorsement lock-in Mistral is used as a concrete case; the Model Router remains open to Other and Local.

Mistral integration concern Quuipu abstraction Reason

Model family/version Capability descriptor Avoid domain coupling

API request MistralAdapter Provider isolation

RAG/embeddings Retrieval interface Keep authority-aware retrieval

Privacy/deployment Routing policy Cloud/self-host choices

Usage/cost Gateway telemetry Cross-provider economics

CHAPTER 16

Other providers

Multi-provider openness reduces dependency and allows different capabilities to be optimized independently.

Provider neutrality The Core assumes no proprietary API, tokenizer or provider semantics.

Comparative evaluation Models are evaluated on Quuipu tasks: fidelity, evidence use, structured output, latency, cost and safety.

Policy allowlists A tenant may authorize only selected providers or regions.

Exit strategy Prompt contracts, datasets and evaluation harnesses support replacement without rewriting the product.

CHAPTER 17

Tools and function calling

An LLM may orchestrate tools, but access to actions must be more constrained than access to text generation.

Read tools Retrieval and lookup can be granted within source scope.

Write tools Data-changing actions require authorization, schema, idempotency and often human confirmation.

Tool registry The gateway exposes only tools approved for the current capability.

No ambient agency The model does not autonomously discover new privileges or tools.

Trace Every material tool call enters the decision trace.

CHAPTER 18

Structured outputs

For runtime integration, structured outputs are often more useful than free-form text.

Schema-first The task contract defines fields, enums, evidence references and failure state.

Validation Non-conforming output is repaired in a bounded way, retried or rejected.

Rendering separation The model produces semantic payload; the Experience Layer decides how to render it.

Safer evolution Schema versioning allows model changes without breaking UI components.

CHAPTER 19

Human control

The AI path remains subordinate to choices made by the user and the organization.

Visible invocation Where useful or required, users can know that an AI capability was invoked.

Mode control Users can return to Original or a deterministic path.

Operator controls Tenant admins can disable providers, capabilities, memory or tools.

Escalation High-materiality claims can require human review.

Override An override must have a real, traceable effect.

CHAPTER 20

AI transparency

Transparency means making the role of AI understandable without granting it more authority than it has.

AI versus source The UI distinguishes canonical content from derived representation.

Provider versus capability Users should understand AI's function; provider identity matters where policy or disclosure requires it.

Generated content Marking and disclosure can be applied at emission without altering the canonical source.

EU context From 2 August 2026, Article 50 AI Act transparency obligations apply to the covered categories; Commission guidelines published on 20 July 2026 clarify scope and application.

Legal boundary Concrete applicability depends on role, system and use case and requires specific assessment.

Transparency object User-facing treatment Runtime anchor

AI invoked Disclosure/state where applicable capabilityRef

Derived representation Differentiate from canonical transformationRef

Source support Source/provenance link evidenceRefs

Uncertainty Qualification trustState

Generated content marking Metadata where applicable emissionPolicy

CHAPTER 21

Security and prompt injection

RAG and tool use expand the attack surface. Context-first also means treating sources and instructions as objects with distinct trust levels.

Instruction hierarchy Retrieved content cannot redefine system policy.

Source sanitization HTML, documents and metadata may contain malicious instructions or untrusted content.

Tool isolation The model sees only authorized capabilities.

Output validation URLs, action parameters and structured output are validated outside the model.

Least privilege Each request receives the minimum access necessary.

CHAPTER 22

Evaluation

Model selection is a verifiable product decision, not a generic benchmark ranking.

Task suites Representative datasets measure Simple, Step-by-step, synthesis, extraction, classification and conflict handling.

Semantic fidelity WP04 Semantic Integrity provides invariants for verifying that transformation does not alter meaning.

Trust tests WP05 adds provenance, uncertainty, policy, disclosure and fallback.

Cost and latency Quality is evaluated together with tokens, time-to-first-token and total latency.

Regression gates New model versions reach production only after comparative evaluation.

Evaluation axis Question Metric/example

Fidelity Did meaning change? Invariant pass rate

Grounding Are claims supported? Claim support ratio

Latency Is interaction responsive? TTFT / total latency

Cost Is routing efficient? Cost/action

Trust Are limits visible? Fallback/disclosure correctness

CHAPTER 23

AI failure modes

The architecture must assume that a model can fail even when the provider is available.

Hallucination Unsupported claims are intercepted by evidence and claimability controls.

Instruction drift Output outside the task contract is rejected or retried.

Provider outage The router tries another adapter or deterministic fallback.

Context overflow Packaging reduces evidence or changes strategy instead of silently truncating material elements.

Model deprecation Lifecycle registry and evaluation enable controlled migration.

CHAPTER 24

Operating model

Bringing LLMs into Quuipu requires clear operational responsibility.

AI capability owner Defines task, schema, evaluation and SLO.

Model governance Approves providers, versions, data classes and routing policy.

Domain owner Defines source authority and materiality.

Engineering Operates gateway, adapters, security, observability and cost controls.

Trust/compliance Defines disclosure, logging, retention and review requirements.

CHAPTER 25

AI deployment patterns

No-AI Deterministic runtime and curated content only.

Cloud AI Gateway to managed providers.

EU-region cloud Routing constrained by region and provider policy.

Self-hosted Inference inside organization-controlled infrastructure.

Hybrid Local/small-first with escalation to cloud or more capable models.

Pattern Where inference runs Best fit

No-AI None Maximum determinism

Managed cloud Provider API Fast adoption

Region-bound Selected cloud region Residency policy

Self-hosted Org infrastructure Sovereignty/control

Hybrid Local + cloud Balanced cost/privacy/capability

CHAPTER 26

AI invariants

These invariants prevent the addition of an LLM from turning Quuipu into an opaque chatbot or a model-centric system.

AI is optional The Core remains valid without an LLM.

Context precedes inference Intent, source, scope and policy are resolved before generation.

Model is replaceable Providers and versions remain behind adapters.

Source outranks model The LLM never becomes the source of truth.

Minimum necessary context The package contains only necessary data.

Fallback survives AI The deterministic path remains available.

CHAPTER 27

The category boundary

The strategic distinction is not 'does Quuipu use AI?'. It is where AI is placed in the architecture.

Chatbot pattern The model is the entry point and conversation is the product.

The cognitive experience is the product; the model is a capability invoked when useful.

Web-native The user remains inside the website, its sources and its journeys.

Governed intelligence Context, Semantic Integrity and Trust Architecture bound what the model can do.

Market message Quuipu is not another chatbot. It is Cognitive Web infrastructure that can use AI without becoming dependent on it.

Chatbot-centric Quuipu-centric

Conversation is the primary interface Website experience remains primary

Model holds conversational context Runtime packages bounded context

Model often selects knowledge Authority-aware retrieval selects sources

Provider behavior shapes product Provider sits behind adapter

Failure often becomes bad answer Failure can return to canonical web

CHAPTER 28

The final principle

Quuipu AI defines a way to integrate generative models without handing them the product identity, source of truth or control of the experience.

Before intelligence Resolve context, intent, source, scope and policy.

Use intelligence Invoke the smallest adequate model when it adds value.

After intelligence Validate evidence, semantic fidelity, trust state and provenance.

Keep control Preserve fallback, override and model replacement.

The thesis Quuipu does not require an LLM to exist. An LLM can extend Quuipu's capabilities.

CHAPTER A

AI Gateway request envelope

Field Example Purpose

taskType simplify / synthesize / classify Capability routing

context declared intent + bounded state Task relevance

evidence SourceRef + excerpts Grounding

policy allowed claims/tools/providers Governance

privacyClass cloud / region / local Data routing

budget max input/output/cost Economics

outputSchema versioned schema Validation

trace request + policy IDs Auditability

CHAPTER B

Model Router decision matrix

Condition Preferred route Fallback

Deterministic rule sufficient No model Canonical/curated

Low-complexity text transform Small/fast model Deterministic representation

Complex synthesis Capable model Reduced-scope answer

Sensitive local-only data Local model No-AI/refusal

Provider unavailable Alternate provider Deterministic fallback

Evidence insufficient No generation Original / clarify

CHAPTER C

AI risk / control matrix

Risk Control Safe outcome

Hallucination Evidence + claimability gate Block unsupported claim

Prompt injection Instruction hierarchy + sanitization Ignore hostile instructions

Context leakage Minimization + data class Exclude unnecessary data

Provider lock-in Adapters + eval harness Replace model/provider

Cost runaway Budget + routing Small-first/fallback

Model drift Versioned evaluation Release gate

Hidden AI role Disclosure policy Visible invocation where needed

Tool overreach Allowlisted tools + auth Bounded action

CHAPTER D

AI-01 · AI is optional

AI-02 · Context before inference Intent, scope, source and policy precede generation.

AI-03 · Source before model The model never becomes the canonical source.

AI-04 · Provider abstraction Provider protocol remains outside domain logic.

AI-05 · Minimum context Only task-necessary context is packaged.

AI-06 · Authority-aware RAG Retrieval respects source authority and version.

AI-07 · Candidate output Generative output is validated before experience emission.

AI-08 · Model-aware provenance Model and version are traceable for derived output.

AI-09 · Cost is bounded Token and model economics are policy inputs.

AI-10 · Privacy routes inference Data class can constrain provider, region or local execution.

AI-11 · Human control survives Users/operators can override, disable or fall back.

AI-12 · Deterministic fallback survives AI failure does not remove access to the canonical experience.

APPENDIX E

Relationship to the Quuipu series Whitepaper Contribution to Quuipu AI

01 · The Cognitive Web Defines context-aware experience as the category.

02 · Architecture Provides Core, Host, Experience and state boundaries.

03 · Quuipu Web Shows where AI may enhance website experience without replacing it.

04 · Semantic Integrity Constrains semantic transformation and evidence-backed claims.

05 · Trust Architecture Adds provenance, explainability, bounded intelligence and human control.

06 · Quuipu AI Places LLMs behind context, routing, validation and fallback.

REFERENCES

Selected external references European Commission. Guidelines on transparency obligations for providers and deployers of AI systems, 20 July 2026. The Commission states that Article 50 transparency obligations apply from 2 August 2026. Mistral AI documentation. Current documentation describes API access, RAG and embeddings, model families, workspaces and admin-

istration, privacy controls, and deployment options. These references are used only to ground the vendor example; Quuipu's architecture remains provider-neutral. Architecture note. Product capabilities, model names, licensing, pricing and provider controls evolve. Production routing policy should therefore rely on current provider documentation and versioned evaluation rather than static whitepaper assumptions.

Context Before Intelligence: Integrating LLMs into Cognitive Web Infrastructure

CONTEXT BEFORE INTELLIGENCE. AI EXTENDS THE SYSTEM; IT DOES NOT DEFINE IT.