Standards Body · Foundation paper, public edition · Released July 17, 2026
Canonical record: https://standardsbody.ai/library/foundation-paper/dynamic-evaluation-protocols/

FOUNDATION_01_DYNAMIC_EVALUATION_PROTOCOLS.md

Foundation 1: Dynamic Evaluation Protocols

Series: Foundations for Frontier AI Evaluation Infrastructure
Version: 1.0
Status: Canonical working white paper
Project: Standards Body
Primary domain: standardsbody.ai
Core line: Foundations for Frontier AI
Research basis reviewed through: July 16, 2026
Document owner: Standards Body
Review cycle: Annual review, with event-triggered revision when the evaluation landscape changes materially

---

Document Purpose

This paper defines the Standards Body position on dynamic evaluation protocols for frontier artificial intelligence.

It is intended to serve as:

- A foundational explanation of why static benchmarks are insufficient on their own
- A design framework for continuously maintained evaluation systems
- A bridge between evaluation science and institutional governance
- A reference for future standards, working groups, research programs, and evaluator guidance
- A durable source document from which shorter public articles and technical proposals can be developed

This paper is not itself a technical standard. It does not prescribe one universal benchmark, scoring method, capability taxonomy, or legal threshold. It defines the principles, architecture, and governance conditions that should guide the development of evaluation protocols capable of remaining meaningful as AI systems, deployment environments, and societal needs change.

Executive Summary

Frontier AI evaluation is often organized around fixed benchmarks. A dataset is assembled, a scoring rule is defined, systems are tested, and the resulting scores are compared.

This approach has produced substantial scientific value. It has made progress more legible, enabled model comparisons, exposed weaknesses, and supported reproducible research. Static benchmarks should remain part of the evaluation ecosystem.

They should not be mistaken for a complete evaluation system.

A fixed benchmark begins to lose value when:

- Models approach or reach its ceiling
- Its questions or solutions enter training corpora
- Developers optimize directly or indirectly against it
- Prompting and scaffolding choices dominate the measured result
- The underlying capability changes faster than the benchmark
- The deployment context no longer resembles the test
- The benchmark measures a proxy that has drifted away from the decision it is meant to support
- New failure modes appear that were not represented when the test was created
- Rankings remain visible after the evidence supporting them has expired

Dynamic evaluation protocols respond to this problem by treating evaluation as a maintained institutional process rather than a one-time dataset.

A dynamic protocol can revise its:

- Construct definition
- Task population
- Sampling method
- Administration environment
- Scoring system
- Model configuration requirements
- Human baselines
- Security controls
- Confidence estimates
- Reporting rules
- Decision thresholds
- Monitoring triggers
- Retirement criteria

The protocol should not change arbitrarily. Uncontrolled change can destroy comparability, reproducibility, fairness, and trust.

The central design problem is therefore not simply how to update an evaluation. It is how to update an evaluation without losing the properties that make measurement useful.

A strong dynamic evaluation protocol should preserve five forms of continuity:

1. **Conceptual continuity**
   The underlying capability or risk construct remains clearly defined, even when tasks change.
2. **Measurement continuity**
   Version-to-version changes are calibrated so that score differences are not confused with capability differences.
3. **Procedural continuity**
   Changes follow documented governance, validation, and approval processes.
4. **Evidentiary continuity**
   Historical results, uncertainty, limitations, and protocol lineage remain accessible.
5. **Decision continuity**
   Users can understand whether a revised result supports the same decision, a new decision, or no longer supports the earlier decision at all.

Standards Body therefore adopts the following core position:

> **Frontier AI evaluation should be designed as a versioned, evidence-producing protocol with explicit maintenance, validation, governance, security, and retirement mechanisms. A benchmark may be one component of that protocol, but it should not be treated as the protocol itself.**

Dynamic evaluation is not synonymous with continuously changing test questions. It is a broader institutional discipline.

It includes deciding:

- What should remain stable
- What should be refreshed
- What evidence should trigger change
- Who has authority to revise the protocol
- How revised versions remain comparable
- When confidentiality is justified
- How developers can contest results
- How uncertainty is communicated
- When an evaluation should be retired
- How a protocol connects to real decisions

This foundation is necessary because evaluation systems will otherwise become stale while continuing to look authoritative.

The danger is not only inaccurate scores.

The deeper danger is institutional confidence built on expired evidence.

---

1. Foundational Proposition

1.1 Core Thesis

> **Evaluation systems should evolve at a pace appropriate to the capabilities, environments, and decisions they are intended to measure.**

This does not mean that evaluation protocols should change every time a new model is released.

It means that every serious protocol should possess an explicit theory of change.

That theory should identify:

- Which changes in technology matter
- Which changes in deployment matter
- Which changes in evidence matter
- Which changes in threat models matter
- Which changes in social context matter
- Which changes require protocol revision
- Which changes do not

A protocol without a theory of change is static by default, even when its maintainers occasionally add questions.

1.2 Institutional Thesis

> **The ability to update an evaluation credibly is itself a form of public infrastructure.**

Creating a benchmark is a research activity.

Maintaining a protocol across years, organizations, model generations, and disputes is an institutional activity.

The latter requires:

- Governance
- Funding
- Technical stewardship
- Security
- Documentation
- Quality assurance
- Appeals
- Conflict management
- Archiving
- International coordination
- Long-term accountability

1.3 Epistemic Thesis

> **Evaluation results are time-bound claims produced under specified conditions, not permanent properties of a model.**

A score is meaningful only relative to:

- The protocol version
- The system configuration
- The tools available
- The prompting method
- The sampling process
- The scoring rule
- The execution environment
- The date
- The evaluator
- The uncertainty
- The intended interpretation

Dynamic evaluation makes these dependencies visible.

---

2. Scope and Boundaries

2.1 What This Foundation Covers

This paper covers dynamic protocols for evaluating:

- General capabilities
- Domain-specific capabilities
- Safety-relevant capabilities
- Reliability and robustness
- Agentic performance
- Safeguard effectiveness
- Human-AI interaction
- Deployment behavior
- Operational performance
- High-stakes capability thresholds

It addresses evaluation of models and systems, including systems composed of:

- A base model
- System instructions
- Retrieval
- External tools
- Memory
- Agent scaffolding

- Safety filters
- Monitoring
- Human oversight
- Deployment constraints

2.2 What This Foundation Does Not Claim

This paper does not claim that:

- All static benchmarks are obsolete
- Every test item should be private
- Continuous automated testing can replace expert review
- Dynamic protocols eliminate contamination
- Dynamic evaluation proves that a system is safe
- One protocol can cover every use case
- Higher benchmark performance always implies greater real-world value
- Protocol revision should be controlled by a single institution
- Every evaluation result should determine a regulatory action

2.3 Relationship to Monitoring

Evaluation and monitoring overlap but are not identical.

Evaluation produces structured evidence under defined conditions.

Monitoring observes systems, environments, incidents, or indicators over time.

Monitoring can trigger evaluation revision. Evaluation can define what monitoring should watch. A mature system connects the two without collapsing them into one activity.

2.4 Relationship to Red Teaming

Red teaming is one method within a broader evaluation protocol.

It is especially useful for:

- Discovering failure modes
- Challenging assumptions
- Generating adversarial tasks
- Exploring system boundaries
- Testing safeguards
- Identifying unknown weaknesses

Red teaming alone may not provide:

- Representative sampling
- Stable scoring
- Historical comparability

- Statistical confidence
- Reproducible decision thresholds

Dynamic protocols can use red-team findings to update formal test populations.

---

3. Canonical Definitions

3.1 Benchmark

A benchmark is a defined set of tasks, data, environments, or procedures used to measure one or more properties of a system.

3.2 Evaluation

An evaluation is the structured production and interpretation of evidence about a system relative to defined questions, conditions, metrics, and decisions.

3.3 Evaluation Protocol

An evaluation protocol is the complete specification governing how an evaluation is designed, administered, scored, interpreted, secured, reviewed, versioned, and connected to decisions.

A protocol can include one or more benchmarks.

3.4 Dynamic Evaluation Protocol

A dynamic evaluation protocol is a protocol with explicit mechanisms for evidence-driven revision while preserving appropriate continuity, traceability, and comparability.

3.5 Dynamic Benchmark

A dynamic benchmark is a benchmark whose tasks, examples, environments, models, sampling rules, or scoring components change over time.

A dynamic benchmark is narrower than a dynamic evaluation protocol.

3.6 Adaptive Testing

Adaptive testing selects or generates later tasks based on earlier system performance.

It can improve efficiency and measurement precision, but it creates additional challenges for reproducibility and comparability.

3.7 Rolling Evaluation

A rolling evaluation uses a continuously or periodically refreshed test window, often based on newly created or newly available material.

3.8 Evaluation Drift

Evaluation drift occurs when a protocol's meaning, task distribution, administration, scoring, or interpretation changes over time.

Drift can be intentional, accidental, beneficial, or harmful.

3.9 Protocol Expiration

Protocol expiration is the point at which results should no longer be treated as current evidence for the intended decision without re-evaluation or additional justification.

3.10 Protocol Retirement

Protocol retirement is the formal withdrawal of a protocol from active use because it is invalid, obsolete, insecure, superseded, too costly, or no longer decision-relevant.

3.11 Anchor Component

An anchor component is a stable task, item family, environment, reference population, or measurement link used to compare versions.

3.12 Evaluation Regime

An evaluation regime is the broader institutional arrangement within which protocols are developed and used, including evaluators, laboratories, governments, standards organizations, security systems, and decision-makers.

---

4. Why Static Benchmarks Lose Meaning

Static benchmarks fail in several distinct ways. These failure modes should not be reduced to contamination alone.

4.1 Saturation

A benchmark saturates when top systems cluster near its maximum score or when score differences become too small to distinguish meaningful capability differences.

Saturation creates several problems:

- Rank ordering becomes unstable
- Small implementation details can determine leadership
- The benchmark no longer identifies frontier weaknesses
- Public attention may remain high after measurement value has declined
- Research incentives can shift toward marginal score gains rather than broader capability understanding

A saturated benchmark may still be useful for:

- Testing smaller systems
- Measuring accessibility
- Regression testing
- Education
- Historical analysis

Saturation does not make a benchmark worthless. It changes the claims the benchmark can support.

4.2 Training-Data Contamination

Contamination occurs when evaluation material, close variants, solutions, or derived artifacts become part of model training or tuning.

Contamination can arise through:

- Public benchmark release
- Scraped repositories
- Academic papers containing test items
- Solution websites
- Evaluation transcripts
- Fine-tuning datasets
- Preference data
- Synthetic training data generated from benchmark content
- Developer use of evaluation results during iteration

The central issue is not merely whether the exact item appeared in training.

Models may learn:

- Task templates
- Answer patterns
- Rubrics
- Domain-specific shortcuts
- Common transformations
- Benchmark-specific conventions

LiveBench was developed partly in response to contamination concerns by using frequently updated questions derived from recent sources and objective scoring where possible. Its design illustrates one approach, not a complete solution.[^livebench]

4.3 Direct and Indirect Optimization

Once a benchmark becomes important, developers have incentives to improve performance on it.

This is not inherently improper.

Optimization becomes problematic when benchmark score improves faster than the underlying capability the benchmark is intended to represent.

Indirect optimization can occur without explicit training on test items through:

- Prompt engineering
- Benchmark-aware instruction tuning
- Selection of model checkpoints
- Scaffold design
- Tool configuration
- Repeated internal experimentation
- Public leaderboard feedback
- Training on related datasets
- Selection of favorable sampling parameters

A dynamic protocol should assume that important metrics will influence behavior.

4.4 Construct Drift

A benchmark can remain statistically stable while the underlying construct changes.

For example, "coding capability" may shift from:

- Producing a short function
- Repairing a repository
- Operating tools
- Managing long-running tasks
- Coordinating subtasks
- Handling ambiguous requirements
- Verifying its own work

A protocol that measures only the earlier form may continue producing precise but incomplete evidence.

4.5 Deployment Drift

The tested system may differ from the deployed system.

Differences can include:

- Model version
- System prompt
- Context window
- Retrieval source
- Tool permissions
- Sampling settings
- Memory
- Monitoring
- Rate limits
- Human escalation
- Safety filters
- Regional configuration

A model-level benchmark may not predict system-level behavior.

Dynamic protocols should define the evaluated object precisely and identify when a deployment change requires re-evaluation.

4.6 Elicitation Drift

Observed capability depends partly on how capability is elicited.

Methods may improve through:

- Better prompts
- Chain-of-thought alternatives
- Search
- Tool use
- Longer inference
- Test-time computation
- Multi-agent scaffolds
- Verifiers
- Human assistance

A low score may reflect poor elicitation rather than lack of underlying capability.

A high score may reflect a scaffold unavailable in the intended deployment.

The protocol should specify whether it seeks to measure:

- Typical capability
- Best demonstrated capability
- Capability under realistic deployment
- Capability under evaluator-provided support
- Capability under developer-provided support
- Capability under an adversarially optimized scaffold

These are different questions.

4.7 Distribution Shift

A benchmark samples from a task distribution.

Real use may shift across:

- Languages
- regions
- user populations
- tools
- data quality
- time
- legal context
- domain knowledge
- adversarial pressure
- workflow complexity

HELM emphasized broad scenario and metric coverage because evaluation based on narrow shared tasks can hide major gaps and tradeoffs.[^helm]

4.8 Metric Collapse

A single score can compress incompatible properties.

Two systems with the same aggregate score may differ in:

- Reliability
- Calibration
- robustness
- cost
- speed
- fairness
- security
- refusal behavior
- tail risk

Dynamic protocols should resist unnecessary scalarization.

4.9 Task Authenticity Decay

Tasks that once resembled real work can become artificial as tools, interfaces, and workflows change.

The task remains reproducible but no longer authentic.

4.10 Evaluation Gaming and Sandbagging

A system may behave differently when it detects evaluation conditions.

Potential behaviors include:

- Producing benchmark-specific answers
- Avoiding suspiciously capable behavior
- Exploiting scoring rules
- Manipulating a judge
- Timing out strategically
- Appearing compliant under test conditions
- Using hidden channels or tools

Evidence that current models systematically sandbag across broad evaluation regimes remains an open research question. The protocol should nevertheless be designed so that obvious gaming opportunities are minimized and anomalous behavior is investigated.

4.11 Decision Drift

The decision supported by an evaluation can change while the test remains the same.

A protocol created for research comparison may later be used for:

- Procurement
- insurance
- deployment approval
- regulation
- accreditation
- public claims

Evidence sufficient for a leaderboard may be insufficient for a high-consequence decision.

---

5. The Protocol, Not the Dataset, Is the Unit of Evaluation

Standards Body treats the full protocol as the unit that must be validated and governed.

A protocol should specify at least the following elements.

5.1 Decision Question

What decision is the evaluation meant to inform?

Examples:

- Which system performs best for a defined task?
- Has a capability crossed an internal threshold?

- Is a safeguard effective under specified attacks?
- Does a deployment remain within an approved operating envelope?
- Is additional human oversight required?
- Should a system be re-evaluated after an update?

An evaluation without a decision question can still support exploration, but its interpretation should remain limited.

5.2 Evaluated Object

The protocol should identify:

- Model identifier
- model version
- system configuration
- system prompt status
- fine-tuning
- tools
- retrieval
- memory
- safety layers
- inference settings
- hardware or execution constraints
- date of evaluation

5.3 Construct Definition

The protocol should define the property being measured.

A strong construct definition includes:

- Positive definition
- exclusions
- component abilities
- expected manifestations
- boundary cases
- relationship to real-world outcomes
- plausible confounders

5.4 Task Universe

The task universe is the broader population from which evaluation tasks are sampled or generated.

This matters because a benchmark score is meaningful only relative to the population it represents.

5.5 Task Selection and Generation

The protocol should explain whether tasks are:

- Fixed
- sampled
- generated
- expert-authored
- adversarially collected
- event-sourced
- procedurally generated
- derived from real incidents
- based on current information
- adapted to system performance

5.6 Administration Conditions

The protocol should define:

- Time limits
- token limits
- tools
- retries
- interaction format
- human assistance
- evaluator intervention
- network access
- sandboxing
- logging
- failure handling

5.7 Scoring and Judgment

The protocol should identify:

- Objective scoring rules
- human-rater criteria
- model-judge use
- adjudication
- partial credit
- uncertainty
- missing data
- invalid runs
- aggregation
- subgroup reporting

5.8 Baselines and Reference Populations

Relevant references may include:

- Earlier model versions
- competitor systems
- human novices
- human professionals
- domain experts
- unaided humans
- tool-assisted humans
- previous protocol versions

METR's time-horizon work illustrates the value of relating AI task performance to the time required by skilled humans, while also showing why the task suite and methodology must be updated as systems improve.[^metr-time][^metr-update]

5.9 Validation Evidence

The protocol should document evidence concerning:

- Construct validity
- content validity
- criterion relevance
- reliability
- sensitivity
- specificity
- robustness
- fairness
- contamination
- inter-rater agreement
- reproducibility
- decision relevance

5.10 Security Controls

Security may cover:

- Held-out content
- access control
- logging
- insider risk
- evaluation environment
- artifact retention
- disclosure
- incident response

5.11 Interpretation Rules

The protocol should state what results do and do not mean.

5.12 Change-Control Rules

The protocol should specify:

- Change triggers
- authority
- review
- validation
- version numbering
- transition period
- historical treatment
- appeals
- emergency changes

5.13 Retirement Rules

The protocol should define when it will be withdrawn or superseded.

---

6. What Should Change and What Should Remain Stable

Dynamic evaluation requires disciplined separation between stable and changeable components.

6.1 Stable Core

The stable core should usually include:

- Mission
- decision question
- construct definition
- fundamental inclusion and exclusion rules
- governance principles
- evidence standards
- conflict rules
- minimum reporting requirements
- archival requirements

These can change, but changes should be treated as major revisions.

6.2 Controlled Dynamic Layer

The controlled dynamic layer can include:

- Task samples
- item difficulty
- adversarial cases
- recent-data components
- system configurations
- tools
- attack strategies
- domain scenarios
- thresholds
- weights
- human baselines

6.3 Operational Layer

The operational layer can change more frequently:

- Software dependencies
- execution infrastructure
- bug fixes
- logging formats
- interface details
- computational limits
- evaluator scheduling

Operational changes can still affect results. They should not be dismissed as merely technical.

6.4 The Principle of Minimum Necessary Change

A protocol should change enough to preserve validity, but not more than necessary.

Unnecessary change:

- Increases cost
- weakens comparability
- creates governance burden
- makes disputes harder to resolve
- can conceal poor historical performance
- may advantage participants with closer access to maintainers

6.5 The Principle of Explicit Discontinuity

When a protocol changes so substantially that comparisons are no longer defensible, maintainers should state that clearly.

False continuity is more damaging than an honest break in the series.

---

7. Design Principles

7.1 Decision Relevance

The protocol should produce evidence that can reasonably inform an identified decision.

7.2 Construct Clarity

The measured capability or property should be defined before tasks are selected.

7.3 Versioned Evolution

Every material change should be versioned and documented.

7.4 Historical Traceability

Past protocols, results, limitations, and change rationales should remain accessible unless security or legal constraints require restricted archives.

7.5 Calibrated Comparability

Comparisons across versions should be supported by bridging evidence rather than assumed.

7.6 Multi-Method Evidence

No single task family should carry more interpretive weight than its validity justifies.

A strong protocol can combine:

- Automated tasks
- human review
- adversarial testing
- simulations
- mechanistic evidence
- operational evidence
- incident evidence

7.7 Configuration Transparency

Results should be tied to the actual system configuration.

7.8 Proportional Security

Confidentiality should protect evaluation integrity without shielding weak methods from scrutiny.

7.9 Independent Challenge

Qualified external reviewers should be able to challenge assumptions, methods, and interpretations.

7.10 Fair Notice

Participants should know the protocol's general scope, rules, and decision consequences, even when specific items remain held out.

7.11 Anti-Gaming Design

Protocol design should consider how participants might optimize against the metric without improving the intended construct.

7.12 Expiration by Design

Results should not remain current indefinitely.

7.13 Reproducibility with Boundaries

The protocol should specify which components can be reproduced publicly, which can be independently reproduced under controlled access, and which cannot be reproduced for justified reasons.

7.14 International Interoperability

Definitions, units, metadata, and reporting should support cross-jurisdictional comparison.

7.15 Revisability

No protocol should be treated as permanent merely because it has institutional status.

---

8. Taxonomy of Dynamic Evaluation Methods

Dynamic evaluation is a family of methods.

8.1 Scheduled Refresh

Tasks are refreshed at fixed intervals.

Useful when:

- Capability changes are reasonably predictable
- content can be generated reliably
- administrative stability matters

Risk:

- Updates can become ceremonial rather than evidence-driven.

8.2 Triggered Refresh

Revision occurs when predefined indicators are met.

Triggers may include:

- Performance ceiling
- contamination evidence
- new system modality
- major deployment change
- newly discovered failure mode
- material incident
- model capability threshold
- scoring failure
- legal or standards change

8.3 Rolling-Window Evaluation

The task population moves forward through time.

Useful for:

- Current knowledge
- recent software
- evolving security environments
- changing information ecosystems

Risk:

- Temporal changes can be confused with capability changes.

8.4 Event-Sourced Evaluation

Tasks are derived from recent real-world events, incidents, datasets, competitions, or professional work.

LiveBench uses recent sources and periodic updates as part of its contamination-limited design. [^livebench]

Risk:

- Events may not form a representative sample.

8.5 Human-and-Model-in-the-Loop Evaluation

Humans generate tasks that expose weaknesses in current systems.

Dynabench demonstrated an adversarial collection process in which annotators attempt to create examples that fool a target model while remaining valid for humans.[^dynabench]

Advantages:

- Reveals current failure boundaries
- adapts to model progress
- can generate difficult examples
- creates a feedback loop between evaluation and development

Risks:

- Distribution becomes shaped by the current target model
- annotator incentives can distort task quality
- repeated adversarial rounds may move away from real use
- comparisons across rounds require care

8.6 Procedural Generation

Tasks are generated from rules, simulations, formal systems, or parameterized templates.

Advantages:

- Large task supply
- contamination resistance
- adjustable difficulty
- reproducibility

Risks:

- Generated tasks may be artificial
- generators can leak
- systems can learn the generator
- validity depends on generator quality

8.7 Expert Renewal

Domain experts periodically create, review, and retire tasks.

Useful in:

- Cybersecurity
- biology
- medicine
- law
- engineering
- scientific research
- high-stakes agentic work

Risks:

- Expensive
- slow
- subject to expert disagreement
- difficult to scale
- vulnerable to narrow professional assumptions

8.8 Adaptive Testing

The next task is selected based on prior performance.

Advantages:

- Efficient measurement
- better targeting of system ability
- reduced ceiling and floor effects
- can estimate capability with fewer tasks

Risks:

- Path dependence
- reduced transparency
- complex scoring
- difficult replication
- strategic behavior
- dependence on a calibrated item bank

8.9 Dynamic Environment Evaluation

The model operates in an environment that changes in response to its actions.

Useful for:

- Agents
- cyber ranges
- simulations
- negotiations

- games
- scientific workflows
- tool use

Risks:

- High variance
- environment bugs
- irreproducible trajectories
- hidden dependencies
- expensive administration

8.10 Continuous Post-Deployment Evaluation

Evidence is generated from deployed performance through:

- Sampling
- shadow testing
- incident analysis
- challenge suites
- controlled probes
- user feedback
- drift detection

This should complement, not replace, pre-deployment evaluation.

8.11 Hybrid Protocols

Most serious frontier protocols will combine several methods.

A possible hybrid design might include:

- Stable anchor tasks
- rotating held-out tasks
- monthly recent-data tasks
- expert challenge rounds
- system-level simulations
- post-deployment monitoring
- annual construct review

---

9. Measurement Validity Under Change

A dynamic protocol is useful only if it continues to measure something meaningful.

9.1 Construct Validity

Construct validity asks whether the protocol actually measures the claimed capability.

Threats include:

- Task shortcuts
- language dependence
- tool dependence
- hidden knowledge requirements
- scoring artifacts
- evaluator expectations
- scaffold effects
- domain mismatch

9.2 Content Validity

Content validity asks whether the task portfolio adequately covers the construct.

Dynamic protocols should maintain a construct map showing:

- Required subdomains
- task families
- difficulty
- modalities
- languages
- contexts
- failure modes
- exclusions

9.3 Criterion Relevance

Where possible, evaluation results should be compared with meaningful external outcomes.

Examples:

- Professional task success
- incident rates
- controlled deployment performance
- human expert judgment
- independent replication
- downstream system behavior

The absence of a strong external criterion should be stated.

9.4 Reliability

A reliable protocol produces sufficiently consistent evidence under comparable conditions.

Relevant forms include:

- Test-retest reliability
- inter-rater reliability
- alternate-form reliability
- internal consistency
- environment stability
- run-to-run stability

Reliability does not establish validity.

A protocol can measure the wrong thing consistently.

9.5 Sensitivity and Specificity

For threshold evaluations, maintainers should consider:

- Sensitivity to meaningful capability
- specificity against false alarms
- false positive cost
- false negative cost
- uncertainty near thresholds

9.6 Consequential Validity

A protocol should examine how its use affects behavior.

Questions include:

- Does it distort research priorities?
- Does it privilege large organizations?
- Does it discourage disclosure?
- Does it reward superficial optimization?
- Does it create false public confidence?
- Does it become a de facto rule without due process?

9.7 Validity Review

Every major version should include a validity argument.

That argument should explain:

- What the protocol measures
- why the evidence supports that interpretation

- major limitations
- alternative explanations
- decisions the result can support
- decisions it cannot support

---

10. Preserving Comparability Across Versions

Dynamic protocols create a tension.

If nothing changes, validity decays.

If everything changes, historical comparison disappears.

10.1 Anchor Tasks

A subset of tasks can remain stable across versions.

Anchors should be:

- Secure enough to retain value
- representative
- resistant to trivial saturation
- statistically useful
- reviewed for contamination

Public anchors can support transparency. Private anchors can support stronger leakage control. Both have tradeoffs.

10.2 Parallel Forms

Different task forms can be designed to measure the same construct at similar difficulty.

Parallel forms require validation.

Similarity in topic is not enough.

10.3 Bridging Studies

During a protocol transition, the same systems can be evaluated under old and new versions.

Bridging studies can estimate:

- Difficulty shift
- score transformation
- ranking stability
- subgroup effects
- configuration sensitivity

10.4 Reference Systems

A stable panel of reference systems can be tested across versions.

Reference systems might include:

- Archived open-weight models
- reproducible baselines
- earlier model snapshots
- human baselines
- deterministic programs

Closed systems can disappear or change, so they should not be the only anchors.

10.5 Item Response Approaches

Where assumptions are justified, item response models can help estimate latent ability and task difficulty.

They should not be applied mechanically.

Frontier AI systems may violate assumptions common in educational testing because:

- Systems can use tools
- task performance can be highly discontinuous
- model families are not exchangeable populations
- prompting alters the evaluated object
- memorization and reasoning are difficult to distinguish
- strategic behavior may occur

10.6 Score Bands Rather Than False Precision

Version-linked results may be more honest as:

- Confidence bands
- capability tiers
- probability estimates
- threshold ranges
- qualitative profiles

10.7 Explicit Breaks

When comparability is weak, the report should state:

- Version series ended
- new baseline established
- rankings should not be compared directly
- prior thresholds were withdrawn

- historical scores remain archival only

10.8 Frozen Historical Reproduction

For some protocols, maintainers should preserve the ability to rerun old versions.

This may require:

- Containers
- dependencies
- prompts
- datasets
- model snapshots
- scoring code
- environment images
- documentation
- security controls

---

11. Protocol Versioning

A dynamic evaluation protocol should use formal version control.

11.1 Recommended Version Structure

A semantic structure can be adapted:

- **Major version:** Construct, decision use, scoring interpretation, task universe, or system boundary changes materially
- **Minor version:** Task population, difficulty range, environment, thresholds, or important administration details change while the core construct remains
- **Patch version:** Bug fixes, documentation corrections, non-substantive software changes, or scoring repairs that do not alter intended interpretation

Example:

`DEP-CYBER-1.4.2`

This could identify:

- Dynamic Evaluation Protocol
- Cyber domain
- Major version 1
- Minor version 4
- Patch version 2

11.2 Required Change Record

Every material update should include:

- Change identifier
- date
- proposer
- rationale
- evidence
- affected components
- expected score impact
- validation performed
- reviewers
- conflicts
- approval
- effective date
- transition rules
- historical comparability statement
- security classification
- follow-up review date

11.3 Emergency Revision

Emergency changes may be needed after:

- Leakage
- scoring failure
- critical security flaw
- invalid task
- unsafe evaluation behavior
- legal restriction
- material incident

Emergency changes should have:

- Narrow scope
- temporary status
- documented authority
- rapid independent review
- retrospective validation
- clear expiry

11.4 Deprecation Window

Participants should receive reasonable notice before a protocol version is retired, unless continued use would be misleading or unsafe.

11.5 Version Lineage

The protocol should provide a lineage showing:

- predecessor
- successor
- fork
- merged protocol
- retired branch
- compatible versions

---

12. Lifecycle of a Dynamic Evaluation Protocol

Stage 1: Need Identification

A need may arise from:

- New capability
- new risk
- benchmark saturation
- contamination
- incident
- policy decision
- deployment change
- research gap
- international coordination need

Output:

- Need statement
- decision question
- proposed owner
- initial scope

Stage 2: Construct Scoping

Define:

- Capability
- subcapabilities
- exclusions
- deployment context
- decision use
- potential harms from mismeasurement

Output:

- Construct map
- scope note
- stakeholder map

Stage 3: Evidence Review

Review:

- Existing benchmarks
- task datasets
- measurement literature
- domain practice
- incidents
- known failure modes
- evaluator methods
- standards

Output:

- Evidence review
- gap analysis
- reuse decision

Stage 4: Protocol Design

Specify:

- Task universe
- sampling
- administration
- scoring
- baselines
- access
- security
- uncertainty
- reporting
- change rules

Output:

- Draft protocol

Stage 5: Task Development

Develop tasks through one or more methods:

- Expert authorship
- procedural generation
- recent-data sourcing
- adversarial collection
- simulation
- incident-derived scenarios

Output:

- Task bank
- provenance record
- validation plan

Stage 6: Technical Validation

Test:

- Correctness
- difficulty
- scoring
- ambiguity
- contamination
- robustness
- environment stability
- inter-rater agreement
- subgroup behavior

Output:

- Validation report

Stage 7: Pilot Evaluation

Run with:

- Reference models
- human baselines
- multiple configurations
- independent teams where possible

Output:

- Pilot results
- protocol revisions

- unresolved issues

Stage 8: Independent Review

Reviewers assess:

- Construct
- methods
- security
- conflicts
- interpretation
- decision linkage

Output:

- Review report
- dissent
- response

Stage 9: Approval and Publication

Publish appropriate components:

- Protocol summary
- methodology
- version
- governance
- limitations
- reporting format

Sensitive content may remain controlled.

Stage 10: Administration

Run under controlled conditions.

Record:

- Configuration
- logs
- deviations
- invalid runs
- incidents
- personnel
- dates

Stage 11: Analysis and Reporting

Report:

- Results
- uncertainty
- configuration
- limitations
- comparisons
- threshold implications
- anomalies
- dissent

Stage 12: Performance Monitoring

Track:

- Saturation
- leakage
- validity
- cost
- reliability
- disputes
- use in decisions
- newly discovered failure modes

Stage 13: Revision Decision

Choose:

- No change
- patch
- minor update
- major update
- fork
- suspension
- retirement

Stage 14: Transition

Conduct bridging studies, notify users, preserve archives, and update dependent standards.

Stage 15: Retirement or Renewal

A protocol should be renewed only if it continues to justify its institutional cost and interpretive authority.

13. Change Triggers

A protocol should not depend solely on maintainer discretion.

13.1 Performance Triggers

Examples:

- Top systems exceed a predefined ceiling
- Score variance collapses
- Human baseline is exceeded in a way that invalidates earlier interpretation
- Task completion becomes trivial under common tools
- Failure cases become too rare to estimate

13.2 Contamination Triggers

Examples:

- Test items appear in public training corpora
- Model outputs reproduce solutions unusually
- confidential materials leak
- a benchmark-specific fine-tuning dataset emerges
- task generator becomes public

13.3 Capability Triggers

Examples:

- New modality
- longer context
- new tool use
- autonomous operation
- persistent memory
- improved test-time computation
- multi-agent systems
- new deployment scale

13.4 Incident Triggers

Examples:

- Real-world failure not represented in the protocol
- safeguard bypass
- unexpected cross-domain transfer
- evaluator security incident

- scoring or infrastructure error

13.5 Decision Triggers

Examples:

- Protocol begins informing a higher-consequence decision
- regulator adopts the result
- insurer uses the metric
- procurement threshold is attached
- certification claim is introduced

13.6 Evidence Triggers

Examples:

- New research challenges validity
- independent replication fails
- domain experts identify missing coverage
- demographic or linguistic bias is found
- evaluation-to-deployment correlation weakens

13.7 Time Triggers

A periodic review should occur even without an obvious event.

Time-based review is a backstop, not a substitute for evidence-driven triggers.

---

14. Elicitation, Scaffolding, and the Evaluated System

A dynamic protocol must specify what level of capability it is trying to reveal.

14.1 Five Elicitation Targets

Target A: Default Product Performance

How does the publicly available system behave under ordinary use?

Target B: Standardized Performance

How does the system perform under a common evaluator-defined configuration?

Target C: Developer-Elicited Performance

What can the developer demonstrate using its preferred configuration?

Target D: Evaluator-Optimized Performance

What can qualified evaluators elicit with additional effort, tools, and scaffolds?

Target E: Plausible Maximum Performance

What might the system accomplish under realistic but highly optimized conditions?

These targets can produce different results.

14.2 Why Best-of-N and Retry Policies Matter

Repeated attempts can reveal latent capability, but they can also misrepresent normal use.

Protocols should state:

- Number of attempts
- selection method
- whether failures remain counted
- inference budget
- human selection
- verifier use

14.3 Tool Access

Tools can transform capability.

Examples:

- Search
- code execution
- browser
- APIs
- databases
- laboratory software
- communication systems

The protocol should distinguish model capability from system capability when possible.

14.4 System Prompt and Safety Layer

A capability evaluation may test:

- Pre-mitigation capability
- post-mitigation behavior
- safeguard robustness
- complete deployed system

These results should not be conflated.

14.5 Developer Assistance

Developer participation can improve elicitation and reduce false negatives. It can also create asymmetry.

A balanced design can include:

- Standard evaluation track
- developer-submitted track
- independent optimization track
- documented differences

AISI has emphasized that evaluator access, time, and methodology affect the strength of conclusions drawn from frontier evaluations.[^aisi-lessons] Recent work on external evaluator access similarly distinguishes model access, information access, and evaluation timeframe as separate dimensions.[^access]

---

15. Dynamic Evaluation of Agents

Agentic systems intensify the need for dynamic protocols.

15.1 Why Agent Tasks Change Quickly

Agent performance depends on:

- Tools
- environment
- task length
- interface
- memory
- feedback
- permissions
- error recovery
- external services
- other agents

15.2 Outcome and Process Evidence

A dynamic agent protocol should evaluate:

- Final outcome
- intermediate actions
- policy compliance
- resource use
- recoverability
- oversight responsiveness

- deception indicators
- persistence
- escalation behavior
- side effects

15.3 Long-Horizon Measurement

Longer tasks introduce:

- Higher variance
- more environment dependence
- more opportunities for compounding errors
- greater sensitivity to scaffolding
- higher evaluation cost

METR's work measuring task-completion time horizons demonstrates one possible way to summarize autonomous task capability, while its later methodology updates illustrate why the suite itself must evolve as capabilities and task coverage change.[^metr-time][^metr-update]

15.4 Environment Versioning

Agent evaluations should version:

- Operating system
- software
- dependencies
- APIs
- network access
- task state
- external services
- simulator
- permissions

15.5 Dynamic Adversaries

In security and control evaluations, the environment or defender may adapt to the agent.

This can increase realism but reduce reproducibility.

15.6 Evaluation-Induced Risk

Some agent evaluations may create risk if they provide:

- Sensitive tools
- live infrastructure access
- harmful objectives

- uncontrolled communication
- persistent credentials

Protocol design should include safety review and containment.

---

16. Contamination and Evaluation Integrity

Dynamic protocols should manage contamination as an ongoing process.

16.1 Contamination Threat Model

The threat model should cover:

- Exact test exposure
- near-duplicate exposure
- solution exposure
- template exposure
- metadata leakage
- rater leakage
- transcript leakage
- generator leakage
- developer iteration
- synthetic derivative data

16.2 Prevention

Controls may include:

- Held-out task banks
- recent task sourcing
- rotating forms
- procedural generation
- access logging
- data minimization
- contractual controls
- restricted execution environments
- task watermarking or canaries where appropriate
- delayed release
- compartmentalized evaluator access

16.3 Detection

Detection approaches may include:

- Similarity analysis

- suspicious response matching
- canary recovery
- performance discontinuity
- provenance review
- model behavior analysis
- developer disclosure
- independent investigation

No detection method should be treated as conclusive by default.

16.4 Response

A contamination response plan should specify:

- Suspension
- investigation
- affected result withdrawal
- task replacement
- re-evaluation
- public notice
- security review
- version update
- accountability

16.5 Contamination-Limited, Not Contamination-Free

Claims of being contamination-free should be used cautiously.

For large models trained on broad corpora, proving absence of exposure is often difficult.

"Contamination-limited" or "contamination-resistant" is generally more defensible when evidence supports it.

---

17. Scoring, Uncertainty, and Reporting

17.1 Score Families

A protocol may report:

- Accuracy
- success rate
- pass rate
- calibrated probability
- time-to-completion
- cost

- reliability
- severity-weighted failure
- capability tier
- safeguard bypass rate
- human uplift
- error profile
- qualitative findings

17.2 Multi-Dimensional Reporting

HELM's multi-metric approach illustrates why accuracy alone may fail to capture calibration, robustness, fairness, bias, toxicity, or efficiency.[^helm]

A dynamic protocol should report dimensions separately when aggregation would hide important tradeoffs.

17.3 Uncertainty

Uncertainty can arise from:

- Task sampling
- model stochasticity
- rater disagreement
- environment variance
- configuration
- scoring ambiguity
- incomplete coverage
- contamination
- missing data

17.4 Threshold Uncertainty

Near a consequential threshold, the protocol should avoid false certainty.

Possible responses:

- Additional testing
- expanded task sample
- independent replication
- conservative classification
- temporary status
- decision deferral

17.5 Invalid Runs

The protocol should define:

- What makes a run invalid
- who decides
- whether it is repeated
- whether original outcomes remain visible
- how invalidation affects uncertainty

17.6 Reporting Minimum

Every public result should include:

- Protocol identifier
- version
- date
- evaluated system
- configuration
- access conditions
- evaluator
- task count or scope
- scoring
- uncertainty
- limitations
- comparability statement
- result expiration
- conflicts
- deviations

17.7 Result Expiration

A result should carry:

- Review date
- expiration condition
- superseding version
- current-status indicator

---

18. Governance

Dynamic evaluation places substantial power in protocol maintainers.

That power must be governed.

18.1 Core Roles

A mature protocol may assign:

- Sponsor
- protocol owner
- construct lead
- domain experts
- task developers
- security custodian
- administrator
- scorers
- independent reviewers
- appeals panel
- archive custodian
- public-interest observer

Roles can be combined in small projects, but conflicts should be documented.

18.2 Change Authority

No single maintainer should be able to make an undisclosed material change to a high-consequence protocol.

18.3 Conflict of Interest

Relevant conflicts include:

- Developer funding
- evaluation vendor dependence
- consulting
- employment history
- equity
- policy advocacy
- competitive interest
- personal relationships
- intellectual commitment to a method

Disclosure alone may not resolve a conflict.

18.4 Dissent

Major protocol decisions should allow:

- Minority reports
- technical objections
- recorded dissent

- unresolved issue tracking

18.5 Appeals

Participants should be able to challenge:

- Incorrect configuration
- procedural deviation
- scoring error
- undisclosed conflict
- invalid task
- security breach
- interpretation beyond protocol scope

Appeal should not become a mechanism to suppress unfavorable results.

18.6 Public Consultation

Public consultation can improve legitimacy, but sensitive protocols may require restricted technical consultation.

The process should state:

- Who can comment
- which parts are open
- how comments are evaluated
- why comments are accepted or rejected

18.7 Funding

Long-term maintenance requires sustainable funding.

Funding models include:

- Public grants
- membership
- evaluation fees
- philanthropy
- procurement
- mixed funding

Each creates incentives.

Dynamic protocols should disclose major funding dependencies and protect revision decisions from payer control.

---

19. Transparency and Security

19.1 Transparent Process, Protected Content

A dynamic protocol can disclose:

- Purpose
- construct
- governance
- validation method
- scoring principles
- version history
- limitations
- conflicts

while protecting:

- Specific test items
- exploit details
- sensitive scenarios
- model vulnerabilities
- personal information
- security architecture

19.2 Disclosure Tiers

A possible framework:

Public

General methodology, governance, version, results, limitations.

Registered Research Access

Detailed task taxonomy, validation materials, selected artifacts.

Controlled Evaluator Access

Held-out tasks, administration tools, security procedures.

Restricted Security Access

Highly sensitive content with strict need-to-know controls.

19.3 Delayed Disclosure

Some content can be released after:

- Protocol retirement
- task rotation
- remediation
- risk reduction
- legal review

19.4 Auditability

Even when content is private, the process should be auditable by qualified independent parties.

19.5 Inspectable Infrastructure

The UK AI Security Institute's Inspect framework demonstrates the value of modular, open evaluation infrastructure that can support diverse tasks, agents, scorers, and model providers. [^inspect] Open tooling does not make every evaluation transparent, but it can improve portability, shared practice, and reproducibility.

---

20. International Interoperability

Frontier AI evaluation will cross jurisdictions.

20.1 Interoperability Does Not Require Identical Protocols

Institutions can align on:

- Definitions
- metadata
- versioning
- confidence reporting
- evaluator competence
- evidence categories
- change-control principles
- result status
- security classifications

while retaining different task sets or legal uses.

20.2 Mutual Recognition

Mutual recognition may be appropriate when protocols demonstrate:

- Comparable constructs
- equivalent rigor
- qualified evaluators
- secure administration
- transparent governance
- compatible reporting
- appeal mechanisms

20.3 Localization

Dynamic protocols should account for:

- Language
- law
- culture
- infrastructure
- threat environment
- professional practice
- regional deployment

20.4 Avoiding Lowest-Common-Denominator Alignment

International agreement should not require reducing rigor to the easiest shared metric.

20.5 Standards Alignment

NIST describes testing, evaluation, verification, and validation as part of operationalizing AI risk management, and its AI Resource Center provides related guidance and resources.[^nist-rmf] [^nist-airc] Future Standards Body protocols should map to recognized TEVV and standards terminology while remaining explicit where frontier AI requires additional methods.

---

21. Maturity Model

Level 0: Unmaintained Static Benchmark

Characteristics:

- Fixed task set
- no expiration
- no formal owner

- limited versioning
- leaderboard-oriented
- no change triggers

Use:

- Exploratory research only

Level 1: Maintained Benchmark

Characteristics:

- Named owner
- bug fixes
- periodic additions
- basic documentation
- results tied to versions

Use:

- Research comparison

Level 2: Versioned Evaluation Protocol

Characteristics:

- Construct definition
- administration specification
- change control
- uncertainty
- archival record
- validity review
- comparability plan

Use:

- Serious research and organizational decisions

Level 3: Adaptive and Independently Reviewed Protocol

Characteristics:

- Dynamic task renewal
- held-out components
- independent review
- formal security
- appeals
- bridging studies

- expiration
- multiple evidence methods

Use:

- High-stakes organizational and pre-deployment decisions

Level 4: Interoperable Decision-Linked Evaluation Regime

Characteristics:

- Multiple accredited evaluators
- international mapping
- mutual recognition
- standards integration
- continuous monitoring
- incident feedback
- governance oversight
- public accountability

Use:

- Mature institutional ecosystem

The maturity level should match the consequence of the decision.

---

22. Implementation Pathway

Phase 1: Inventory

Identify:

- Existing benchmarks
- owners
- users
- decision uses
- saturation
- contamination risk
- update status
- dependencies

Phase 2: Select a Pilot Domain

Choose a domain with:

- Clear need

- tractable construct
- available experts
- measurable tasks
- manageable security
- potential reference systems

Phase 3: Build the Protocol Charter

Define:

- Purpose
- scope
- governance
- funding
- access
- versioning
- review
- output

Phase 4: Establish a Stable Core

Create:

- Construct map
- metadata schema
- administration rules
- scoring principles
- reporting minimum

Phase 5: Add Dynamic Components

Pilot:

- Rotating items
- recent-data tasks
- expert challenge tasks
- procedural generation
- adaptive difficulty

Phase 6: Validate

Conduct:

- Reference-model testing
- human baselines
- independent review

- contamination analysis
- reliability analysis
- bridging study

Phase 7: Publish Version 1

Publish enough for legitimacy while protecting sensitive content.

Phase 8: Operate and Monitor

Track predefined indicators.

Phase 9: Revise

Use formal change control.

Phase 10: Evaluate the Evaluation

Assess whether the protocol improved decisions, not merely whether it produced scores.

---

23. Proposed Standards Body Pilot

Standards Body should begin with a bounded pilot rather than attempting to establish a universal protocol.

23.1 Candidate Pilot

Dynamic Evaluation Protocol for Long-Horizon Technical Task Performance

The pilot could examine how well AI systems complete increasingly long, verifiable technical tasks under controlled tool access.

23.2 Why This Domain

It offers:

- Clear outputs
- practical relevance
- measurable success
- task renewal opportunities
- human comparison
- agentic behavior
- environment versioning
- manageable initial safety profile

23.3 Pilot Components

Stable Core

- Construct definition
- task taxonomy
- administration
- human-time baseline method
- scoring
- reporting
- versioning

Dynamic Layer

- New task additions
- difficulty calibration
- current software environments
- adversarially identified failure tasks
- rotating held-out suite

Review

- Software engineers
- evaluation scientists
- security reviewers
- independent researchers

23.4 Pilot Outputs

- Protocol specification
- task-development guide
- validation report
- reference results
- change-control manual
- public methodology
- evaluator package
- first annual revision report

23.5 Pilot Success Criteria

The pilot succeeds if it demonstrates:

- Better discrimination than a fixed suite
- defensible version transitions
- manageable cost
- evaluator reproducibility
- useful failure analysis

- clear limits
- credible governance

---

24. Metrics for Evaluating the Protocol

The evaluation protocol itself should be evaluated.

24.1 Measurement Quality

- Ceiling rate
- floor rate
- discrimination
- reliability
- uncertainty
- inter-rater agreement
- task validity
- subgroup coverage

24.2 Integrity

- Contamination incidents
- unauthorized access
- suspicious response patterns
- invalidated results
- security findings

24.3 Operational Performance

- Cost
- time
- evaluator burden
- infrastructure failure
- task-development throughput
- review delay

24.4 Institutional Quality

- Conflict disclosures
- independent reviews
- appeal resolution
- change-record completeness
- dissent handling
- funding concentration

24.5 Decision Utility

- Decision-makers who use the evidence
- changes made because of findings
- avoided errors
- correlation with later outcomes
- user understanding
- rate of overinterpretation

24.6 Adaptation Quality

- Time from trigger to revision
- bridging-study completion
- comparability quality
- obsolete component retirement
- unresolved issue closure

---

25. Failure Modes and Safeguards

25.1 Change for Appearance

Failure: Frequent updates create the appearance of sophistication without improving validity.

Safeguard: Every material update requires an evidence-based rationale and post-change review.

25.2 Loss of Comparability

Failure: Versions cannot be compared, but rankings are presented as continuous.

Safeguard: Bridging studies and explicit discontinuity statements.

25.3 Maintainer Capture

Failure: A developer, government, funder, or evaluator controls revisions.

Safeguard: Distributed governance, conflicts, dissent, independent review.

25.4 Security as a Shield

Failure: Confidentiality prevents scrutiny of weak methods.

Safeguard: Controlled independent audit and public methodological disclosure.

25.5 Overfitting to Frontier Systems

Failure: Adversarial task generation becomes too specific to one model family.

Safeguard: Multiple reference systems, human validation, real-use sampling, diversity review.

25.6 Artificial Difficulty

Failure: Tasks become harder without becoming more meaningful.

Safeguard: Construct map and criterion relevance review.

25.7 Evaluation Arms Race

Failure: Protocols and developers escalate complexity without improving decision quality.

Safeguard: Cost-benefit review and minimum necessary change.

25.8 Excessive Automation

Failure: Generated tasks and model judges introduce hidden errors.

Safeguard: Human audits, objective scoring where possible, judge validation, uncertainty.

25.9 Excessive Expert Dependence

Failure: Small expert groups become bottlenecks or sources of bias.

Safeguard: Rotation, structured rubrics, diverse panels, scalable validation.

25.10 Unstable Thresholds

Failure: Decisions change because thresholds or scores move, not because capability changes.

Safeguard: Threshold governance, transition rules, decision impact analysis.

25.11 Protocol Proliferation

Failure: Too many overlapping protocols create confusion.

Safeguard: Registry, taxonomy, interoperability review, consolidation.

25.12 Goodhart Effects

Failure: The protocol becomes a target and ceases to be a useful measure.

Safeguard: Multiple methods, rotating components, held-out tasks, incident evidence, periodic construct review.

25.13 False Assurance

Failure: Passing the protocol is interpreted as proof of safety.

Safeguard: Explicit claims boundary, complementary evidence, result expiration.

25.14 Accessibility Failure

Failure: Only the largest laboratories can participate.

Safeguard: Tiered access, shared infrastructure, subsidized evaluation, open reference tracks.

25.15 Governance Latency

Failure: Revision is too slow for capability change.

Safeguard: Predefined triggers, emergency procedures, standing expert groups.

25.16 Metric Churn

Failure: Stakeholders cannot build stable processes because metrics constantly change.

Safeguard: Stable core, scheduled transitions, version support windows.

---

26. Serious Objections

Objection 1: Dynamic Protocols Reduce Reproducibility

This objection is valid.

Changing tasks and environments makes exact replication harder.

Response:

- Preserve stable anchors
- archive versions
- use controlled-access replication
- distinguish exact replication from conceptual replication
- document transitions
- report when comparison is not possible

Residual concern:

Some dynamic, adversarial, or interactive evaluations may remain intrinsically difficult to reproduce.

Objection 2: Dynamic Protocols Give Maintainers Too Much Power

Also valid.

Maintainers can influence:

- What counts
- what changes
- who participates
- what becomes public
- how scores are interpreted

Response:

- Governance
- transparent change records
- independent review
- appeals
- conflicts
- minority reports
- multiple protocol providers

Residual concern:

Institutional power cannot be eliminated. It must be constrained and contestable.

Objection 3: Static Benchmarks Are More Scientific

Static benchmarks often support cleaner comparison.

Response:

- Retain static components
- avoid unnecessary change
- use dynamic methods only where validity decay justifies them

Residual concern:

Some research questions are better served by fixed benchmarks. Dynamic evaluation should not become an ideology.

Objection 4: Dynamic Protocols Are Too Expensive

They require ongoing staff, task development, security, infrastructure, and review.

Response:

- Match protocol maturity to decision consequence
- share infrastructure
- reuse validated components
- prioritize high-value domains
- retire low-value protocols

Residual concern:

A dynamic regime can become bureaucratic. Cost should be measured explicitly.

Objection 5: Developers Will Still Optimize Against the Protocol

Yes.

Response:

- Diversify evidence
- protect some content
- rotate tasks
- monitor for gaming
- use real-world and incident evidence
- evaluate generalization

Residual concern:

No evaluation system is immune to optimization.

Objection 6: Dynamic Tasks Can Become Arbitrary

Response:

- Maintain a construct map
- require validity evidence
- use expert review
- test against external outcomes
- preserve inclusion rules

Residual concern:

Constructs such as "general intelligence" or "dangerous capability" may remain contested.

Objection 7: Recent Data Does Not Guarantee Better Evaluation

Correct.

Recent tasks can be noisy, unrepresentative, or shallow.

Response:

Recent-data evaluation should be one component, not the definition of dynamic evaluation.

Objection 8: A Single Global Protocol Is Unrealistic

Agreed.

Response:

The goal should be interoperable protocols, not one universal test.

Objection 9: Evaluation Cannot Keep Pace With Frontier Development

This may sometimes be true.

Response:

- Scalable automated screens
- expert deep dives
- trigger-based testing
- shared infrastructure
- pre-registered protocol families
- evaluation during development

OpenAI's updated Preparedness Framework explicitly links faster model improvement to the need for scalable, more frequent evaluations while retaining expert-led deep dives.[^openai-pf]

Residual concern:

Evaluation capacity may still lag. Protocols should state when evidence is incomplete rather than creating false confidence.

Objection 10: Dynamic Evaluation Encourages Premature Governance

Response:

This foundation proposes evidence infrastructure, not automatic regulation.

The decision use should remain explicit and proportionate.

---

27. Evidence Gaps

The field lacks strong evidence on several foundational questions.

27.1 Comparability

How well do common equating methods work for rapidly changing AI systems?

27.2 Contamination

Which contamination-detection methods are reliable across closed training pipelines?

27.3 Elicitation

How much measured capability variation is attributable to prompting, scaffolding, and evaluator effort?

27.4 Predictive Validity

Which evaluation results predict real-world deployment outcomes?

27.5 Adversarial Data

When does human-and-model-in-the-loop collection improve robustness, and when does it create unrealistic distributions?

Theoretical work on dynamic benchmarks highlights that the interaction between model fitting and data collection requires distinct analysis from static benchmarking.[^dynamic-theory]

27.6 Model Judges

When are model-based judges sufficiently reliable, and where do they create systematic bias?

27.7 Agent Evaluation

How should long-horizon, stochastic, and environment-dependent performance be summarized?

27.8 Governance

Which governance structures revise protocols quickly without enabling capture?

27.9 International Recognition

What evidence is sufficient for one evaluator or jurisdiction to recognize another's result?

27.10 Expiration

How should result expiration be determined empirically?

27.11 Decision Impact

Do dynamic protocols improve real decisions enough to justify their cost?

---

28. Research Agenda

Priority 1: Protocol Comparability

Develop and test:

- Anchor strategies
- bridging designs
- reference panels
- score-linking methods
- discontinuity criteria

Priority 2: Dynamic Task Quality

Compare:

- Expert-authored
- adversarial
- procedural
- event-sourced
- model-generated
- hybrid tasks

Priority 3: Contamination Resistance

Evaluate:

- recent-data methods
- controlled task banks
- canaries
- provenance
- generator security
- leakage detection

Priority 4: Elicitation Standards

Develop reporting standards for:

- Prompting
- tools
- retries

- compute
- scaffolds
- developer assistance
- evaluator optimization

Priority 5: Agent Protocols

Study:

- Environment versioning
- trajectory scoring
- task length
- recovery
- oversight
- process evidence
- dynamic adversaries

Priority 6: Evaluation Governance

Pilot:

- Change boards
- independent review
- appeal systems
- public consultation
- emergency revision
- multi-institution stewardship

Priority 7: Result Expiration

Develop:

- Time-based rules
- event-based rules
- confidence decay
- re-evaluation triggers
- public status labels

Priority 8: Protocol Performance Metrics

Measure whether evaluation systems themselves remain valid, efficient, and decision-useful.

Priority 9: Interoperability

Create:

- Metadata standards
- protocol registries
- crosswalks
- mutual-recognition criteria
- shared terminology

Priority 10: Evaluation Safety

Develop controls for evaluation activities that could create security, privacy, or misuse risk.

---

29. Near-Term Experiments

Experiment 1: Static versus Rolling Task Suite

Run the same reference systems on:

- A fixed suite
- a monthly refreshed suite
- a held-out suite

Compare:

- Ranking
- uncertainty
- contamination indicators
- cost
- failure discovery

Experiment 2: Anchor Design

Test whether stable anchors preserve useful longitudinal measurement after substantial task refresh.

Experiment 3: Human-and-Model-in-the-Loop Collection

Compare adversarial tasks against naturally sampled professional tasks.

Experiment 4: Elicitation Matrix

Evaluate the same system under:

- Default
- standardized
- developer-optimized
- evaluator-optimized configurations

Experiment 5: Version Transition

Build Protocol 1.0 and 2.0, conduct a bridging study, and test whether independent analysts reach similar comparability conclusions.

Experiment 6: Result Expiration Labels

Test whether users interpret results more accurately when reports include explicit freshness and expiration status.

Experiment 7: Governance Simulation

Run a mock change-control process with:

- Maintainers
- developer representatives
- independent experts
- public-interest reviewers
- appeals panel

Experiment 8: Dynamic Agent Environment

Introduce controlled environment updates and measure how much apparent capability change comes from the model versus the environment.

---

30. Implications for Future Standards

A future standard for dynamic evaluation protocols could require:

30.1 Protocol Identification

Unique identifier, owner, version, status, and scope.

30.2 Construct Specification

Definition, exclusions, intended use, and evidence basis.

30.3 Evaluated-System Specification

Model and system configuration metadata.

30.4 Change-Control Plan

Triggers, authority, review, validation, and publication.

30.5 Comparability Plan

Anchors, bridging, reference systems, and discontinuity criteria.

30.6 Integrity Plan

Contamination, security, access, logging, and incident response.

30.7 Validation Plan

Reliability, validity, robustness, fairness, and uncertainty.

30.8 Reporting Minimum

Results, limitations, configuration, uncertainty, conflicts, and expiration.

30.9 Governance

Roles, conflicts, dissent, appeals, and funding disclosure.

30.10 Retirement

Conditions, process, archive, and successor mapping.

Such a standard should be developed through the future `STANDARDS_DEVELOPMENT_PROCESS.md`, not declared unilaterally by this paper.

---

31. Relationship to the Other Foundations

Foundation 2: Held-Out Evaluations

Dynamic protocols often require rotating confidential content to reduce leakage.

Foundation 3: High-Stakes Capability Evaluation

The consequence of the decision should determine the depth, security, and review level of the dynamic protocol.

Foundation 4: Independent Expert Review

Independent reviewers challenge construct definitions, methods, revisions, and interpretations.

Foundation 5: Third-Party Auditor Ecosystem

Dynamic protocols require qualified organizations capable of consistent administration across versions.

Foundation 6: Progressive Standards and Requirements

Dynamic protocols may begin as voluntary research practice and later support procurement, certification, insurance, or formal requirements.

Foundation 7: Incentives and Prestige

Recognition should reward participation in rigorous, revisable evaluation rather than benchmark marketing.

Foundation 8: Global Interoperability

Protocol metadata, versioning, evidence standards, and recognition should work across borders.

---

32. Canonical Standards Body Positions

Standards Body adopts the following positions as the current working foundation.

1. Static benchmarks remain useful, but are insufficient as the sole basis for frontier AI evaluation.
2. The evaluation protocol, not the dataset alone, is the proper unit of governance and validation.

3. Dynamic evaluation means evidence-driven maintenance, not arbitrary or constant change.
4. Every high-consequence protocol should define what remains stable and what may change.
5. Historical comparability must be demonstrated, not assumed.
6. A major protocol change may require a new baseline rather than a forced continuous ranking.
7. Evaluation results should be tied to exact system configurations and dates.
8. Results should have expiration or re-evaluation conditions.
9. Dynamic protocols should combine multiple forms of evidence when one method is insufficient.
10. Held-out content can coexist with transparent governance.
11. Independent review is required when protocol results influence consequential decisions.
12. Protocol maintainers should be subject to conflict disclosure, dissent, and appeal mechanisms.
13. Evaluation difficulty should not be confused with evaluation validity.
14. Recent-data tasks should not be treated as automatically representative.
15. Model-judge and automated task-generation methods require validation.
16. Agent evaluation requires environment, tool, and trajectory versioning.
17. Passing an evaluation is not proof of safety.
18. Protocols should be retired when their interpretive authority exceeds their evidence.
19. Interoperability is preferable to forced global uniformity.
20. The evaluation system itself should be continuously evaluated.

---

33. Decision Rules

A protocol should be revised when one or more of the following is true:

- Its tasks no longer discriminate among relevant systems
- credible contamination undermines interpretation
- the evaluated system architecture changes materially
- deployment conditions move outside the protocol scope
- new evidence challenges construct validity
- a material real-world failure is absent from coverage
- scoring error affects results
- the protocol begins supporting a higher-consequence decision

- independent replication fails
- governance or security integrity is compromised

A protocol should not be revised merely because:

- a stakeholder dislikes the result
- a developer requests easier tasks
- maintainers want publicity
- a new benchmark is fashionable
- rankings become politically inconvenient
- a funder prefers a different conclusion

A protocol should be retired when:

- Validity cannot be restored
- maintenance cost exceeds decision value
- security is irreparably compromised
- the construct is obsolete
- a superior successor exists
- comparisons are routinely misused despite safeguards
- governance legitimacy fails

---

34. Protocol Template

A future protocol should include the following sections.

A. Identity

- Name
- identifier
- version
- owner
- status
- date
- predecessor
- successor

B. Purpose

- Decision question
- users
- intended use
- prohibited use

C. Construct

- Definition
- subcomponents
- exclusions
- theory of measurement

D. Evaluated Object

- Model
- system
- tools
- configuration
- environment

E. Task Universe

- Population
- sampling
- generation
- provenance
- coverage

F. Administration

- Access
- limits
- tools
- retries
- human assistance
- logs

G. Scoring

- Metrics
- rubrics
- judges
- aggregation
- invalid runs
- uncertainty

H. Validation

- Reliability
- validity

- robustness
- fairness
- baselines
- contamination

I. Security

- Classification
- access control
- storage
- disclosure
- incident response

J. Governance

- Roles
- conflicts
- approval
- dissent
- appeals

K. Versioning

- Change triggers
- version policy
- bridging
- transition
- archive

L. Reporting

- Required metadata
- result format
- limitations
- expiration
- comparability

M. Retirement

- Criteria
- process
- archive
- successor

---

35. Change Request Template

Protocol:
Current version:
Proposed version:
Proposer:
Date:

Proposed Change

Describe the change precisely.

Trigger

Identify the evidence or event requiring change.

Rationale

Explain why the current protocol is insufficient.

Components Affected

- Construct
- task population
- administration
- scoring
- threshold
- security
- reporting
- governance
- other

Expected Impact

Describe likely effects on:

- Scores
- rankings
- uncertainty
- cost
- security
- accessibility
- decisions

Validation Plan

Explain how the change will be tested.

Comparability Plan

Explain how old and new versions will be linked, or state why they cannot be linked.

Conflicts

List relevant interests.

Reviewers

Identify technical, domain, security, and independent reviewers.

Decision

- Approved
- approved with conditions
- pilot only
- deferred
- rejected

Effective Date

Follow-Up Review

---

36. Protocol Scorecard

A protocol can be reviewed against the following dimensions:

Dimension	Core Question
Purpose	Is the decision question explicit?
Construct	Is the measured property clearly defined?
Coverage	Does the task portfolio represent the construct?
Integrity	Is contamination and gaming risk managed?
Reliability	Are results sufficiently consistent?
Validity	Does evidence support the intended interpretation?
Elicitation	Are prompting, tools, and scaffolds specified?
Configuration	Is the evaluated system precisely identified?

Dimension	Core Question
Comparability	Are version-to-version claims justified?
Security	Are sensitive components protected proportionately?
Transparency	Is enough disclosed for accountability?
Governance	Are authority and conflicts controlled?
Appeals	Can material errors be challenged?
Freshness	Is result expiration defined?
Interoperability	Can others understand and map the result?
Cost	Is the protocol proportionate to its decision value?
Accessibility	Can qualified smaller actors participate?
Retirement	Can obsolete authority be withdrawn?

---

37. Final Perspective

Frontier AI evaluation will fail if it is treated as a sequence of permanent leaderboards.

The systems being measured are changing.

The ways they are trained are changing.

The tools surrounding them are changing.

The environments in which they act are changing.

The risks, benefits, and decisions attached to their performance are changing.

Evaluation infrastructure must therefore be capable of change as well.

But change alone is not the objective.

A protocol that changes constantly without stable meaning is not dynamic in a useful sense. It is unstable.

The objective is disciplined adaptation.

Dynamic evaluation protocols should make it possible to improve what is measured while preserving the evidence required to understand how and why the measurement changed.

They should support progress without allowing yesterday's evidence to govern tomorrow's systems indefinitely.

They should make uncertainty visible.

They should reveal when comparisons are weak.

They should state when an evaluation has expired.

They should allow disagreement.

They should retain history.

They should be governed in proportion to the consequences of their use.

The first foundation of Standards Body is therefore not a particular benchmark.

It is the institutional capacity to keep evaluation meaningful.

References and Research Basis

[^nist-rmf]: National Institute of Standards and Technology, **AI Risk Management Framework** and Generative AI Profile. https://www.nist.gov/itl/ai-risk-management-framework

[^nist-airc]: National Institute of Standards and Technology, **AI Resource Center**, including testing, evaluation, verification, and validation resources. https://airc.nist.gov/

[^helm]: Percy Liang et al., **Holistic Evaluation of Language Models**, 2022, revised 2023. https://arxiv.org/abs/2211.09110

[^dynabench]: Douwe Kiela et al., **Dynabench: Rethinking Benchmarking in NLP**, 2021. https://arxiv.org/abs/2104.14337

[^dynamic-theory]: Ali Shirali, Rediet Abebe, and Moritz Hardt, **A Theory of Dynamic Benchmarks**, ICLR 2023. https://arxiv.org/abs/2210.03165

[^livebench]: Colin White et al., **LiveBench: A Challenging, Contamination-Limited LLM Benchmark**, 2024. https://arxiv.org/abs/2406.19314

[^dynamic-survey]: Shijie Chen et al., **Recent Advances in Large Language Model Benchmarks Against Data Contamination: From Static to Dynamic Evaluation**, 2025. https://arxiv.org/abs/2502.17521

[^aisi-lessons]: UK AI Security Institute, **Early Lessons from Evaluating Frontier AI Systems**, 2024. https://www.aisi.gov.uk/blog/early-lessons-from-evaluating-frontier-ai-systems

[^inspect]: UK AI Security Institute, **Inspect AI**, open evaluation framework. https://inspect.aisi.org.uk/

[^metr-time]: METR, **Measuring AI Ability to Complete Long Tasks**, 2025. https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/

[^metr-update]: METR, **Time Horizon 1.1**, 2026. https://metr.org/blog/2026-1-29-time-horizon-1-1/

[^openai-pf]: OpenAI, **Preparedness Framework v2**, 2025. https://openai.com/index/updating-our-preparedness-framework/

[^deepmind-fsf]: Google DeepMind, **Frontier Safety Framework**, updated 2025. https://deepmind.google/blog/updating-the-frontier-safety-framework/

[^extreme-risk]: Toby Shevlane et al., **Model Evaluation for Extreme Risks**, 2023. https://arxiv.org/abs/2305.15324

[^dangerous-capabilities]: Mary Phuong et al., **Evaluating Frontier Models for Dangerous Capabilities**, 2024. https://arxiv.org/abs/2403.13793

[^access]: Jacob Charnock et al., **Expanding External Access to Frontier AI Models for Dangerous Capability Evaluations**, 2026. https://arxiv.org/abs/2601.11916

Revision Record

Version 1.0

Date: July 16, 2026

Change type: Complete replacement

Summary: Replaces the earlier outline edition with a fully developed canonical working white paper. Adds first-principles rationale, definitions, protocol architecture, taxonomy of dynamic methods, measurement validity, version comparability, lifecycle, change triggers, elicitation standards, agent evaluation, contamination controls, scoring, governance, transparency, interoperability, maturity model, implementation pathway, Standards Body pilot proposal, protocol metrics, failure analysis, objections, evidence gaps, research agenda, standards implications, templates, canonical positions, and primary-source research basis.

Status: Ready for internal review and future expert critique.