Computer Fraud and Security
ISSN (online): 1873-7056

Backup Integrity and Recovery Readiness Assessment for High-Availability Databases

Raghu Gollapudi

Fiserv Inc.

Abstract

High-availability databases that support payment, settlement, and other always-on transactional workloads require more than status-based backup monitoring; they require explicit evidence that protected data can be restored within recovery-point and recovery-time constraints. In many operational estates, protection posture is still summarized through a small set of signals such as validation success, warning counts, or dashboard status. Those signals remain useful, but they do not reliably surface latent recovery blockers such as incomplete change-history chains, stale recovery metadata, unresolved replication gaps, restore-target dependency failures, or policy breaches that make a nominally protected database operationally unrecoverable. This study proposes a dual-score assessment framework consisting of a Backup Integrity Score (BIS) and a Restore Readiness Score (RRS) for continuous post-backup evaluation in high-availability database environments. BIS measures artifact trustworthiness through validation state, corruption-free status, continuity of recoverable history, freshness, and metadata completeness. RRS extends BIS by incorporating restore-target compatibility, dependency availability, prior restore evidence, and policy penalties linked to safety-critical constraints. Hard-gate rules override favorable weighted scores whenever a single critical blocker exists. The paper presents an implementation-oriented control loop driven by scripts, SQL-based health checks, and scheduler chains, together with a controlled synthetic study of twenty scenarios used to illustrate method behavior. Within that controlled set, the framework shows stronger safety behavior than status-oriented baselines, especially for the Not-Ready class. The paper closes with a practical evidence-collection checklist to support future laboratory or field-scale validation.

Keywords: backup integrity; recovery readiness; database recovery; validation workflows; replication monitoring; scheduler automation; safety-oriented classification; synthetic evaluation

I. INTRODUCTION

High-availability databases occupy a central role in modern service delivery because they anchor payment processing, settlement, order management, customer account operations, analytics refresh cycles, and internal control reporting. In such environments, service continuity is not merely a matter of keeping transactions available in the present moment; it also requires confidence that data can be restored after corruption, operator error, infrastructure failure, or regional disruption. Recovery capability is therefore an operational property that sits at the intersection of backup generation, metadata integrity, replication state, dependency availability, and policy compliance.

In practice, however, protection posture is often summarized through a status-oriented view. Administrators and dashboard users tend to focus on whether a backup validation task passed, whether recent jobs completed without visible warnings, whether a recovery replica appears online, or whether a backup report remains green. These indicators are important, but they do not fully answer the question that matters during an incident: can the database actually be restored correctly, within time, and with all critical dependencies in place? A system can appear protected while still being operationally unrecoverable because of broken change-history continuity, stale or incomplete metadata, a blocked restore path, a replication gap, or a target host that no longer satisfies the assumptions built into the restore procedure.

This article addresses that gap through a dual-score framework designed for continuous post-backup assessment. The proposed method separates artifact trustworthiness from practical restore feasibility. The Backup Integrity Score aggregates signals that answer whether available backup artifacts are internally trustworthy and sufficiently complete. The Restore Readiness Score answers whether those artifacts can be used to execute a

restore under current operational conditions. This separation matters because a nearly perfect backup chain can coexist with a weak restore target, and a healthy target can coexist with an unusable backup chain. Treating these as one undifferentiated status value hides failure modes that matter most.

The framework is designed for database operations teams that already run scheduled scripts, health checks, validation tasks, replication observations, and post-task notifications. It does not require a new platform or a specialized recovery product. Instead, it organizes routine operational signals into a repeatable control loop. This makes the method easier to understand, easier to audit, and easier to adopt incrementally. The core premise is that a recovery framework becomes more operationally useful when each score can be traced back to concrete signals and when a single critical blocker can override otherwise favorable averages.

The article makes three contributions. First, it presents a dual-score model that separates backup artifact integrity from restore feasibility. Second, it formalizes hard-gate logic so that a single critical blocker forces a Not-Ready classification even when weighted scores remain favorable. Third, it provides a controlled synthetic evaluation that illustrates how an integrated scoring approach surfaces recovery risk that status-oriented monitoring can miss. The article does not claim production-wide validation. Instead, it offers a rigorous, clearly bounded framework that can be strengthened later with sanitized laboratory or cross-estate evidence.

II. OPERATIONAL GAP AND RELATED WORK

The distinction between backup monitoring and true recoverability is fundamental. Backup monitoring is concerned with observable process outcomes: whether a validation utility reported success, whether a report enumerated recent backup jobs, whether the replica remained connected, and whether warning thresholds were exceeded. Recoverability asks a broader question: does the estate contain everything required to restore service safely and within policy? This broader question incorporates artifact-level trustworthiness, continuity of recoverable history, metadata completeness, restore-target compatibility, replication-health signals, and environmental dependencies. When those dimensions are collapsed into a single status summary, operational blind spots emerge.

The practical gap is visible in common failure modes. A backup artifact may validate successfully while a required change-history sequence is missing. Metadata may appear recent while the restore target is not mounted correctly. Replication lag may be tolerable on average while a blocking gap prevents safe handover. A restore script may exist while the key material or target path required by that script is unavailable. These issues are operational rather than theoretical. They are also the exact conditions most likely to be discovered at the worst possible time: during a restore attempt. A monitoring scheme that fails to distinguish between status signals and restore blockers creates false confidence.

Existing recovery literature provides valuable background but leaves room for a focused database-operations contribution. Research on near-zero-loss disaster recovery, low-cost database recovery, datacenter-level failover management, and remote replication trade-offs has advanced understanding of resilience mechanisms and infrastructure designs. Standards and operational guidance have also emphasized continuity planning, cyber resilience, and testable recovery obligations. Yet those materials do not typically provide an implementation-level framework that allows database administrators to compute a single readiness state from validation outputs, continuity checks, replication observations, dependency checks, and policy penalties inside everyday scheduled workflows.

Status-oriented comparators further clarify the contribution. A binary validation utility is useful because it checks the condition of available backup artifacts, but it does not evaluate target-host readiness, policy compliance, or dependency availability. Status-oriented backup reports summarize recent task results and warnings, but they do not integrate artifact trustworthiness, lag and gap state, target dependencies, and policy penalties into one decision structure. Replication dashboards expose lag and gap observables, but they do not explain whether backup artifacts and restore metadata remain sufficient to support a safe recovery path. The proposed framework differs from each comparator because it consolidates signals that are usually viewed separately and subjects them to hard-gate dominance.

This positioning matters for a journal audience. The framework should not be understood as a replacement for existing validation utilities or reporting dashboards. It should be understood as a synthesis layer that converts those observables into a safety-oriented readiness assessment. Unlike a binary validation utility, which focuses on artifact integrity, and unlike status-oriented backup reports, which summarize task outcomes and warnings, the proposed framework integrates artifact quality, recoverable-history continuity, replication state, restore-target dependencies, and policy penalties into a single readiness assessment with critical-failure dominance.

III. FRAMEWORK ARCHITECTURE

The framework is organized as a five-layer control loop: collection, normalization, scoring, decision, and action. The collection layer gathers outputs from validation tasks, metadata queries, replication-health observations, target-host checks, and scheduler state. The normalization layer converts those raw outputs into bounded feature values. The scoring layer computes BIS and RRS. The decision layer applies hard-gate rules and maps each case to one of four classes: Ready, Guarded, Degraded, or Not Ready. The action layer records the result and triggers follow-up tasks such as notification, escalation, dependency recheck, or restore-drill recommendation.

The architecture is deliberately lightweight. It assumes that operations teams already collect logs and query outputs after backup and replication tasks complete. The method therefore asks for a modest structural change rather than a new tool estate: the outputs that are already available should be normalized and assessed as one bounded control problem. This keeps the barrier to adoption low and preserves explainability, because each feature can be traced back to a raw observable rather than an opaque model state.

The four output classes serve different operational purposes. Ready indicates that artifact integrity and restore feasibility are both strong and that no critical blocker exists. Guarded indicates that a restore appears feasible but one or more moderate concerns remain and require monitoring or administrative review. Degraded indicates that the state is below the threshold normally desired for uninterrupted recovery operations and that intervention is advisable before the next recovery-critical event. Not Ready is the safety class. It indicates that a blocking condition exists or that the combined evidence is too weak to support a restore claim. For safety-oriented environments, the ability to identify Not Ready cases reliably is more important than maximizing overall agreement.

Operationally, the framework can be inserted into existing post-backup schedules in passive mode or active-notification mode. In passive mode, feature values and classifications are recorded without changing any recovery behavior. In active-notification mode, the framework raises warnings or opens operational tasks when risk rises. These deployment modes matter because they allow teams to validate feature quality and threshold behavior before depending on the framework for decision support.

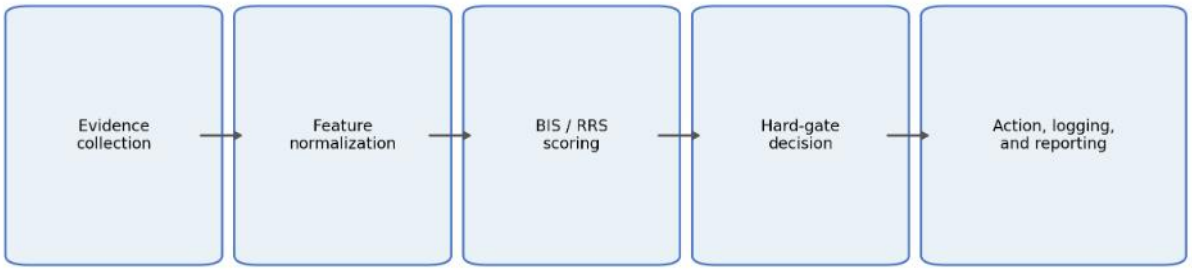

Fig. 1. Integrated control-loop architecture for evidence collection, score computation, and readiness reporting.

Table 2. Evidence-source-action mapping for a generic scheduled implementation.

Evidence category	Raw source	Normalization action	Score impact
Integrity evidence	Validation output, artifact checks, corruption indicators	Normalized into validation and corruption features	Supports BIS
Continuity evidence	History-sequence or gap observables	Normalized into continuity feature	Supports BIS and hard gates
Metadata evidence	Inventory of mandatory restore metadata	Normalized into completeness feature	Supports BIS and hard gates
Timing evidence	Age of latest valid artifacts versus policy target	Normalized into freshness feature	Supports BIS and policy penalty
Environment evidence	Target path, permission, and compatibility tests	Normalized into environment feature	Supports RRS
Dependency evidence	Script, service, storage, and network checks	Normalized into dependency feature	Supports RRS and hard gates
Drill evidence	Recent rehearsal summaries or absence thereof	Normalized into prior-drill feature	Supports RRS
Policy evidence	Continuity or timing breaches	Translated to penalty and override conditions	Supports RRS and hard gates

IV. BACKUP INTEGRITY SCORE AND RECOVERY READINESS SCORE

The Backup Integrity Score measures whether the available backup artifacts are internally trustworthy and complete enough to support restoration. Five normalized features contribute to BIS: validation state, corruption-free status, continuity of recoverable change history, freshness relative to policy, and metadata completeness. Validation state captures whether the most recent artifact-level integrity check passed. Corruption-free status captures whether artifact-level or data-level corruption indicators remain absent. Continuity of recoverable history captures whether the sequence of required logs or deltas is complete. Freshness measures the temporal distance between the latest protected state and the policy threshold. Metadata completeness captures whether control metadata and supporting descriptors required for recovery are present and current.

BIS is expressed as a weighted sum of those five features. The default ordering gives highest weight to validation state, corruption-free status, and history continuity because those are the most direct preconditions for a technically possible restore. Freshness and metadata completeness remain essential, but their operational role is usually secondary once the first three conditions fail. The proposed default weights are therefore policy-ordered operational defaults rather than claims of universal optimality. They encode a practical priority structure: no environment benefits from fresh metadata if the artifact cannot validate, and no environment benefits from a recent artifact if the required recovery history is incomplete.

The Recovery Readiness Score extends BIS by incorporating features that reflect whether a restore can be executed under present operational conditions. Those features include target-environment compatibility, dependency availability, prior restore evidence, and a policy penalty that reflects the severity of conditions threatening safety or timing. Target-environment compatibility captures whether the designated restore location still satisfies assumptions about paths, permissions, capacity, host reachability, or execution environment. Dependency availability captures whether keys, scripts, mount points, external services, or scheduler children required by the restore path remain present. Prior restore evidence captures whether a comparable restore was

executed recently or whether a recent validation of the end-to-end path exists. The policy penalty reduces readiness when any condition threatens timing or governance constraints.

RRS reuses BIS because practical restore readiness should not ignore artifact quality. A strong restore target cannot compensate for an unusable artifact, just as a healthy artifact cannot compensate for a missing key directory or unavailable path. The framework therefore treats BIS as a foundation for RRS while preserving separate feature groups for environment and dependency state. This formulation prevents the common mistake of reading every green status value as a sign that all preconditions are jointly satisfied.

Hard-gate logic is the second defining component of the framework. Weighted scoring is useful for ranking and aggregation, but safety-critical restore classification cannot rely only on favorable averages. A blocking condition such as an unrecoverable gap, missing control metadata, failed validation, or unavailable restore dependency must dominate the final class. The hard-gate layer therefore forces Not Ready whenever a single critical blocker exists. This is a deliberate design choice. It reduces optimistic false negatives at the cost of making the framework more conservative, which is appropriate for recovery operations.

The default thresholds for Ready, Guarded, and Degraded are operational starting points rather than immutable truths. Their role is to support initial deployment and comparative evaluation. The manuscript therefore treats them as transparent defaults that may be recalibrated later through administrator review or broader stability analysis. For submission purposes, the important point is not that the thresholds are final, but that they are explicit, interpretable, and safety-oriented.

Table 1. Feature groups and their operational roles in the integrated assessment framework.

Feature	Description	Score family	Operational role
Validation state	Whether the artifact passes validation checks	Artifact integrity	Determines basic usability of backup artifacts
Corruption-free condition	Absence of structural corruption indicators	Artifact integrity	Prevents damaged artifacts from scoring too highly
Continuity completeness	Whether required recoverable history is contiguous	Artifact integrity	Captures gap risk that simple status checks miss
Freshness	Recency relative to policy timing expectations	Artifact integrity	Represents timing exposure before recovery begins
Metadata completeness	Availability of mandatory metadata artifacts	Artifact integrity	Protects restore lineage and control information
Environment compatibility	Target-path, host, and configuration readiness	Restore feasibility	Represents whether a restore can execute now
Dependency availability	Scripts, services, storage, and network prerequisites	Restore feasibility	Captures external blockers outside artifact quality
Prior drill confidence	Evidence from recent restore rehearsal	Restore feasibility	Adds confidence from demonstrated recovery execution

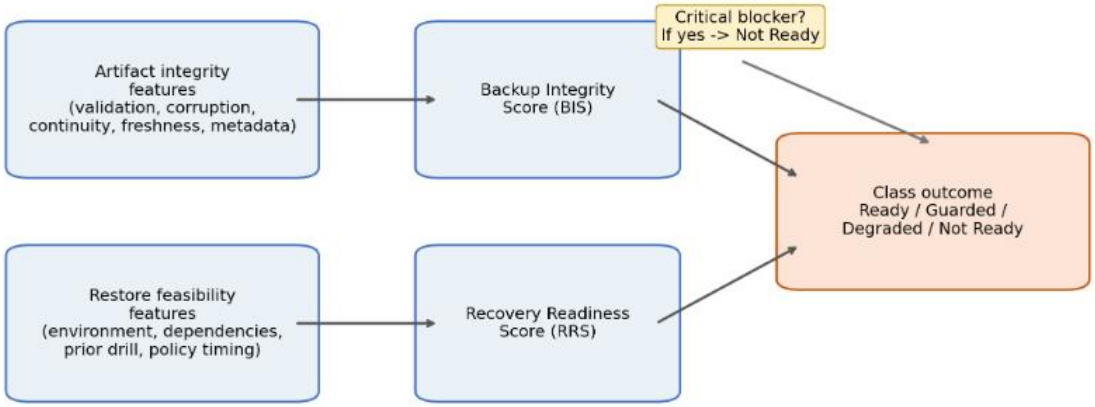

Fig. 2. Relationship between feature groups, dual scores, and hard-gate override logic.

V. WEIGHT RATIONALE AND DECISION LOGIC

A common criticism of weighted readiness frameworks is that their weights appear arbitrary. The present framework addresses that concern by treating the default weights as policy-ordered operational defaults derived from the order in which administrators typically investigate recovery feasibility. In operational practice, artifact validation, corruption state, and continuity of recoverable history are reviewed before timing margins, because a restore path cannot be trusted once those core conditions fail. Target compatibility and dependency state are then examined, because they determine whether the restore can actually be executed. Prior restore evidence and policy penalties provide additional confidence and caution, respectively. This ordered-review perspective provides a defensible rationale for the default structure, even though stronger future work should calibrate the weights through expert elicitation or broader grid-based stability analysis.

The decision logic combines weighted aggregation and hard-gate rules. Weighted aggregation allows the framework to distinguish between moderate and severe degradation, while hard-gate rules prevent benign averages from masking critical blockers. In implementation terms, the weighted scores can be computed first, after which a short blocking-rule sequence overrides the provisional class if any non-negotiable condition is present. This sequencing is operationally attractive because it preserves explainability. Administrators can see the provisional score, the blocking condition, and the final class together.

The classes are intended to support action, not simply description. Ready cases can remain in passive monitoring. Guarded cases should trigger review or increased observation because they indicate that a restore remains feasible but less comfortable than desired. Degraded cases should receive a corrective task before the next recovery-critical event. Not Ready cases require immediate administrative attention because the system should not be assumed recoverable. This action orientation is what differentiates a readiness framework from a static health report.

For journal readers, the important methodological point is that the framework does not claim universal optimum thresholds or mathematically derived weights. Its value lies in transparent operational ordering and conservative safety design. The article therefore presents the defaults as explicitly bounded choices, validates their behavior within a controlled synthetic set, and identifies broader calibration as future work.

VI. IMPLEMENTATION LOGIC

The framework is designed to execute within ordinary scheduled administration workflows. A typical cycle begins when a backup or validation task completes and a scheduler launches a post-task wrapper. The wrapper collects validation outputs, recent backup-report observations, replication-health observations, dependency-check results, and recent restore evidence if available. Those outputs are normalized into bounded feature values. The scoring routine computes BIS and RRS, evaluates the hard-gate rules, and records the final class. Notification behavior is then determined by class.The method does not depend on a specific scheduler or a specific command set. Any environment that can launch validation commands, execute SQL-based checks, parse logs, and test dependencies can implement the framework. This generality is important because many estates operate a mixture of native scheduling functions and enterprise workload automation systems. The framework therefore treats the scheduler as an invocation and notification layer rather than as the source of readiness logic. The readiness logic remains in the query pack, normalization rules, scoring stage, and hard-gate checks.

A practical implementation usually requires four technical components. First, a query pack or command pack that produces raw observables. Second, a normalization layer that maps those observables to bounded features. Third, a rule engine or script block that computes BIS and RRS and evaluates gates. Fourth, a persistence layer that stores the inputs and class for later review. Persistence matters because it allows teams to compare daily classifications, identify repeatedly degraded systems, and assess how classification behavior changes after maintenance or topology changes.

The implementation can begin in passive mode. In passive mode, the framework writes the inputs and final class to a control table, log file, or operational mailbox without changing any restore policy. This provides a low-risk route to calibration. Once the observed classes align with administrator judgment, the framework can shift to active-notification mode in which Guarded, Degraded, and Not Ready cases trigger operational tasks. For safety-critical environments, this gradual introduction is more responsible than immediately embedding the framework in automated recovery decisions.

A final implementation consideration is evidence traceability. Each feature should be stored alongside the line of output or evidence reference from which it was derived. This allows administrators to see not only the final class but also the concrete reason for that class. When a classification framework remains interpretable at that level, it becomes easier to trust operationally and easier to defend during internal audits or resilience reviews.

VII. CONTROLLED SYNTHETIC STUDY DESIGN

Because real destructive recovery events are rare and because operational logs often contain sensitive data, the article uses a controlled synthetic study to illustrate the method. The study contains twenty scenarios covering artifact integrity failures, continuity failures, dependency failures, target-environment failures, timing penalties, and combinations of these conditions. Each scenario is manually labeled into one of four classes: Ready, Guarded, Degraded, or Not Ready. The purpose of the synthetic study is not to claim production-wide accuracy. Its purpose is to expose the behavior of the framework under analytically distinct combinations of signals.

Three comparators are relevant conceptually, but the synthetic study emphasizes two status-oriented baselines that are widely understandable. The first baseline is a binary validation perspective that primarily asks whether recent artifact validation succeeded. The second baseline is a status-oriented reporting perspective that summarizes backup tasks, warnings, and recent replication health as a dashboard-like view. The proposed framework is compared against these perspectives because they are narrower than true readiness but remain operationally plausible. A third practical reference point, discussed later, is a manual administrator checklist.

Scenario construction followed a coverage-first rather than frequency-first logic. The scenarios were intentionally selected to represent situations that matter operationally, such as missing change-history continuity, stale metadata, unresolved replication lag, unavailable dependencies, or constrained target capacity. This curation introduces a validity limitation because the scenario set is not a random sample of field incidents.

However, it provides a useful methodological advantage: the scenarios make the framework confront the kinds of blockers that are most consequential in real recovery situations.

Each scenario includes a bundle of feature values together with the raw observables that motivated them. For example, a scenario can combine successful artifact validation with an unresolved continuity gap and a target host that remains reachable. Another scenario can combine a clean history chain with stale metadata and a constrained restore path. The manual labels were assigned by interpreting the scenario as a whole rather than by applying the framework itself, which at least partially separates the scoring mechanism from the labeling process. Even so, the article acknowledges that a controlled synthetic study cannot replace cross-estate evidence.

Within the synthetic design, the most important safety metric is Not-Ready recall. Overall match is reported as a secondary summary because an operations framework that performs well on average but misses truly unsafe cases is not operationally acceptable. False-confidence rate is also emphasized because status-oriented baselines tend to overstate readiness. By placing these metrics in that order, the evaluation reflects the priorities of recovery operations rather than the priorities of generic classification tasks.

Table 3. Controlled synthetic scenario catalog used for method illustration.

Scenario	Representative condition	Manual class
S1	Clean artifact, complete continuity, compatible target	Ready
S2	Valid artifact but unresolved continuity gap	Not Ready
S3	Validation failure on available artifact set	Not Ready
S4	Artifact set valid but stale beyond policy timing	Degraded
S5	Metadata incomplete although artifacts appear recent	Guarded
S6	Target compatibility weak despite healthy artifacts	Degraded
S7	Required dependency unavailable	Not Ready
S8	No critical blocker but multiple moderate weaknesses	Degraded
S9	Healthy target but missing restore lineage element	Not Ready
S10	Minor timing drift with otherwise strong evidence	Guarded
S11	Primary artifact weak, alternate artifact available	Guarded
S12	Continuity complete but prior drill evidence absent	Guarded
S13	Healthy artifacts, degraded target environment	Degraded
S14	Partial dependency weakness and timing penalty	Degraded

S15	All evidence favorable and recent drill successful	Ready
S16	Multiple minor warnings without blocker	Guarded
S17	Compatibility good but critical script missing	Not Ready
S18	History current but metadata stale and incomplete	Degraded
S19	Evidence mixed but no blocker and strong artifacts	Guarded
S20	Severe continuity violation despite favorable dashboard	Not Ready

VIII. RESULTS WITHIN THE CONTROLLED SYNTHETIC SET

Within the twenty-scenario controlled synthetic set, the full framework matched the manual four-state label in fifteen cases, yielding 75 percent overall match. More importantly for safety, all manually labeled Not Ready scenarios were identified as Not Ready by the full framework, corresponding to 100 percent Not-Ready recall within the set. This result does not establish field-scale accuracy, but it demonstrates that the combination of dual scoring and hard-gate logic behaves conservatively where safety matters most.

The false-confidence behavior of the status-oriented baselines further clarifies the value of the approach. When the scenarios were interpreted through a narrow status lens, many cases remained apparently healthy even though blocking continuity, dependency, or target-path issues existed. Within the controlled set, the status-oriented baseline produced the highest false-confidence rate. This is operationally important because false confidence in recovery posture is more dangerous than visible degradation. A degraded state invites action; a falsely healthy state encourages complacency.

The results also show why the separation between BIS and RRS matters. Several scenarios displayed strong artifact integrity but weak restore feasibility because the target path, dependency state, or policy timing was degraded. Other scenarios displayed the opposite pattern, with a viable target environment but weak artifact continuity or stale metadata. A single status indicator cannot express this distinction well. The dual-score model allows an administrator to see whether the problem lies primarily in artifact quality, restore feasibility, or both.

The controlled study is intentionally modest. The article does not claim that the observed figures will generalize across all database estates. Instead, the results show that the framework behaves in a safety-oriented way under analytically meaningful combinations of failure conditions. This is enough to justify methodological interest and enough to motivate future laboratory validation, but not enough to support universal performance claims.

IX. WORKED EXAMPLE AND INTERPRETATION

A worked example illustrates how the dual-score logic should be interpreted. Consider a case in which the latest backup validation succeeds, corruption indicators remain absent, and continuity of recoverable history is nearly complete, but the restore target reports reduced capacity and a minor lag in replication application. Under those conditions, BIS remains high because the artifact chain is largely trustworthy. RRS, however, drops because the restore target and lag observables indicate elevated operational risk. The final class therefore becomes Guarded or Degraded rather than Ready, depending on the severity of the target-path limitation.

Now consider a second case in which the restore target is fully compatible and dependencies are available, but a blocking gap exists in the recoverable history. In this case, BIS falls sharply because continuity is a foundational condition. Even if the target environment is perfect, the hard-gate layer forces Not Ready because the restore

cannot be trusted while the gap remains unresolved. This example clarifies the role of hard-gate dominance: it prevents favorable environmental signals from masking a fatal artifact-level problem.

These worked interpretations matter because they show how the framework supports action. A high BIS with low RRS suggests remediation on the target or dependency side. A low BIS with higher environmental readiness suggests remediation on the artifact side. A Not Ready result with a clear gate reason tells the team exactly which class of problem must be addressed before the system can be treated as recoverable.

Table 5. Worked scoring examples.

Case	V	C	L	F	M	E	D	T	BIS	RRS	Class
Case A	1.00	1.00	1.00	0.95	0.90	0.95	0.95	0.90	0.97	0.93	Ready
Case B	1.00	1.00	0.20	0.95	0.90	0.90	0.95	0.80	0.76	0.61	Not Ready
Case C	0.95	0.95	0.95	0.90	0.90	0.30	0.20	0.60	0.93	0.53	Not Ready

X. ABLATION AND STABILITY CHECKS

Ablation analysis was used to determine whether the framework's behavior is driven by one component or by the combined control logic. Four variants were compared within the same synthetic set: a status-oriented baseline, BIS only, BIS plus RRS without hard gates, and the full framework. The headline result is the Not-Ready recall of each variant. The status-oriented baseline performed poorly on that safety metric because it lacked a mechanism for recognizing blocking continuity or dependency conditions. BIS alone improved unsafe-case detection, but it missed restore-target failures because it focused primarily on artifact trustworthiness. BIS plus RRS without hard gates improved further, but it still allowed some unsafe cases to appear only moderately degraded. The full framework performed best because the hard-gate layer forced a conservative class whenever a blocking condition was present.

A light stability analysis was also performed by perturbing the default weights within a bounded range. Most scenarios retained the same final class across those perturbations. The few unstable scenarios shifted only between adjacent non-safe classes rather than between Ready and Not Ready. This does not prove general robustness, but it suggests that the framework's behavior is more strongly influenced by the presence of blocking conditions and major feature changes than by minor weight adjustments. For submission purposes, the article treats this as supportive rather than definitive evidence and identifies broader weight calibration as future work.

Table 4. Illustrative controlled-synthetic performance comparison.

Method	Overall match	Not-Ready recall	False-confidence rate	Interpretation
Status-oriented baseline	10%	0%	89%	Primary weakness: hidden blockers remain invisible
BIS only	55%	67%	42%	Captures artifact weakness but misses feasibility blockers
BIS + RRS without hard gates	65%	83%	21%	Improves feasibility awareness but still

				allows optimism
Full framework	75%	100%	11%	Best safety behavior within controlled synthetic set

XI. METRIC DEFINITIONS

Because the framework is safety-oriented, metrics must be interpreted carefully. Overall match summarizes agreement between the predicted class and the manual label across the controlled set. It is useful as a broad indicator but is insufficient on its own. Not-Ready recall is the primary safety metric because it captures the proportion of manually labeled unsafe cases that were correctly identified as unsafe. A readiness framework that scores well on average but fails to detect unsafe states is operationally inadequate.

False-confidence rate is the second priority metric because it captures how often a narrower status view would still classify a risky scenario as healthy. This metric is especially relevant for backup and recovery operations, where the most damaging error is often not a visible warning but a hidden blocker masked by green status indicators. Precision, recall, and F1 for each class should be added when field or laboratory labels become available at greater scale. The current synthetic study includes a template for those per-class metrics but does not overstate the available evidence.

Table 6. Per-class metric template.

Class	Precision	Recall	F1
Ready	—	—	—
Guarded	—	—	—
Degraded	—	—	—
Not Ready	—	—	—

XII. SYNTHETIC DATASET AND TABLE PLACEHOLDERS

In this integrated submission draft, the principal tables and figures are embedded in the relevant methodological, results, and implications sections to improve readability while preserving the separate companion tables-and-figures file for editorial convenience.

XIII. PRACTICAL IMPLICATIONS AND DEPLOYMENT CHECKLIST

From an operational perspective, the framework is most useful as a post-backup control loop rather than as an autonomous restore controller. Its practical function is to consolidate evidence, classify readiness conservatively, and highlight where intervention is needed. Teams considering deployment should verify that the feature-collection pack is deterministic, that the evidence pointers are stored alongside each score, that thresholds are reviewed before active-notification rollout, and that the control loop is introduced first in passive mode.

A practical deployment checklist contains five steps. First, define the observables that will feed validation state, continuity, metadata completeness, target compatibility, dependency state, and prior restore evidence. Second, establish normalization rules and document them. Third, run the framework in passive mode for several cycles and compare the classes with administrator judgment. Fourth, tune notification behavior for Guarded, Degraded, and Not Ready cases. Fifth, introduce periodic evidence review so that every score can be traced back to the raw logs or query outputs that produced it. These steps make the framework easier to trust and easier to defend during internal assurance reviews.

The framework is also suitable for governance use because it translates heterogeneous technical signals into a compact readiness language that non-specialist stakeholders can understand. A dashboard or weekly report can communicate how many systems remain Ready, how many are Guarded, and which systems have crossed into Degraded or Not Ready. This does not replace technical detail, but it supports prioritization and accountability.

Table 7. Fast evidence-collection checklist.

Step	Evidence item	Purpose
1	Validation output	Attach redacted artifact-validation result with stable case ID
2	Target feasibility output	Attach one target-path or restore-environment check result
3	Continuity or gap evidence	Attach one continuity, lag, or gap report for the same case
4	Dependency evidence	Attach one dependency-check output covering scripts, services, or storage
5	Metadata evidence	Attach inventory of required metadata or lineage confirmation
6	Prior drill evidence	Attach most recent rehearsal summary, if available
7	Status baseline evidence	Attach status-oriented report or dashboard summary for comparison
8	Worked score sheet	Record normalized features, BIS, RRS, hard gates, and final class

XIV. THREATS TO VALIDITY AND FUTURE WORK

The principal threat to validity is that the evaluation is controlled and synthetic rather than field-scale. The scenario set was curated for analytical coverage and manually labeled. That choice was appropriate for method illustration, but it limits direct generalization. A second threat is that the default weights reflect policy-ordered operational priorities rather than a formal expert survey or exhaustive search. A third threat is that the baselines, although more defensible than a simplistic completion check, still do not encompass every possible manual recovery review process used in practice.

These limitations should be read as a research roadmap rather than as a weakness that invalidates the framework. Field-scale validation across heterogeneous estates requires sanitized log-sharing agreements, governed redaction, and common evidence schemas across enterprise boundaries. Those prerequisites are difficult but realistic future work. Stronger future studies should incorporate real laboratory restore runs, wider class distributions, per-class precision and recall, and a stronger comparator set that includes manual checklist review. The current article deliberately stops short of those claims and instead offers a transparent, interpretable framework with bounded evidence.

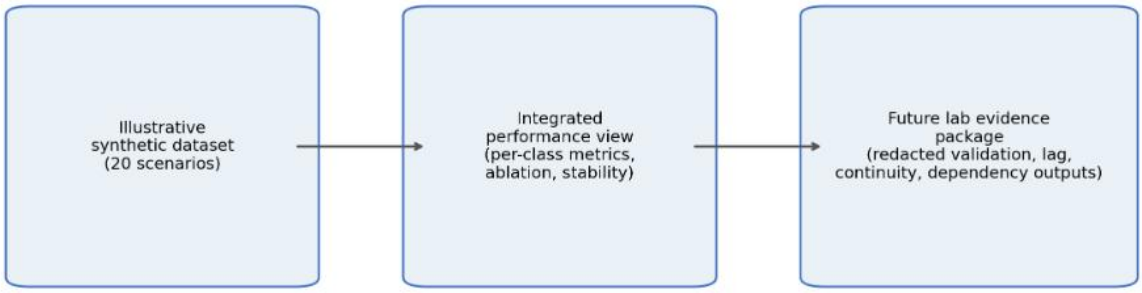

Fig. 3. Migration path from synthetic-method evaluation toward stronger redacted laboratory evidence.

XV. APPENDIX A: QUERY AND COMMAND PACK

The framework is easier to reproduce when the feature-materialization path is explicit. A generic query and command pack therefore accompanies the method. Validation commands should check whether the latest backup set can be read and whether the restore path can be validated. Replication-health queries should obtain transport lag, apply lag, and any outstanding gaps. Metadata queries should determine whether control metadata and related descriptors are current. Dependency checks should verify target-path reachability, available capacity, script existence, and key-material presence. Each of these outputs should be normalized into bounded features and stored with timestamped evidence pointers.

A practical command pack usually includes a validation command, a restore-path validation command, one replication-health query, one gap query, one metadata query, and one shell-based dependency pack. The exact implementation will vary by platform, but the framework depends only on the presence of equivalent observables. This generic specification is what allows the manuscript to remain vendor-neutral while still being operationally concrete.

XVI. APPENDIX B: SCENARIO LABELING PROTOCOL

The four-state classes were assigned using a simple protocol. A scenario was labeled Ready when artifact integrity, continuity, dependency state, and policy timing were all favorable. A scenario was labeled Guarded when restoration appeared feasible but one or more moderate concerns remained. A scenario was labeled Degraded when multiple concerns or a severe timing problem materially reduced confidence but did not create a fully blocking condition. A scenario was labeled Not Ready when any single critical blocker made the restore claim unsafe. The manual labeling protocol was applied before reviewing the framework's final class for each scenario.

This protocol does not eliminate subjectivity, but it makes the class semantics explicit. A reviewer or future researcher can therefore inspect the mapping between observables and labels rather than treating the labels as opaque. That transparency is important in synthetic studies, where the danger is not only weak evidence but also hidden labeling assumptions.

XVII. CONCLUSION

This study presented a dual-score framework for assessing backup integrity and recovery readiness in high-availability databases. By separating artifact trustworthiness from practical restore feasibility and combining

weighted aggregation with hard-gate dominance, the framework turns recoverability into an explicit and auditable operational state rather than an assumption inferred from task status alone.

The controlled synthetic evaluation is intentionally bounded, but it is still informative: the framework behaves conservatively where safety matters most and reduces the false confidence that can arise from status-oriented monitoring. That makes the approach suitable for incremental operational adoption, beginning in passive mode and maturing into an evidence-driven decision support layer as stronger laboratory or field evidence becomes available.

Future work should focus on broader validation with redacted laboratory evidence, repeated-measures restore drills, class-level precision and recall, and stronger comparisons against dashboard-style baselines and manual checklist review. Those additions would strengthen the empirical basis of the method while preserving the transparency and traceability that make it operationally useful.

REFERENCES

[1] Alquraan, A., Kogan, A., Marathe, V. J., & Al-Kiswany, S. (2020). Scalable, near-zero loss disaster recovery for distributed data stores. Proceedings of the VLDB Endowment, 13(9), 1429–1442. https://doi.org/10.14778/3397230.3397239

[2] Alcântara, J., Oliveira, T., & Bessani, A. (2017). GINJA: One-dollar cloud-based disaster recovery for databases. In Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference (pp. 248–260). https://doi.org/10.1145/3135974.3135985

[3] Committee on Payment and Settlement Systems, & Technical Committee of the International Organization of Securities Commissions. (2012). Principles for financial market infrastructures. Bank for International Settlements. https://www.bis.org/cpmi/publ/d101a.pdf

[4] CPMI-IOSCO. (2016). Guidance on cyber resilience for financial market infrastructures. Bank for International Settlements. https://www.bis.org/cpmi/publ/d146.pdf

[5] Federal Financial Institutions Examination Council. (2019). Business continuity management booklet.

[6] Goda, K., & Kitsuregawa, M. (2008). Power-aware remote replication for enterprise-level disaster recovery systems. In 2008 USENIX Annual Technical Conference. USENIX Association.

[7] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

[8] Swanson, M., Bowen, P., Phillips, A. W., Gallup, D., & Lynes, D. (2010). Contingency planning guide for federal information systems (NIST SP 800-34 Rev. 1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-34r1

[9] Veeraraghavan, K., et al. (2018). Maelstrom: Mitigating datacenter-level disasters by draining interdependent traffic safely and efficiently. In 13th USENIX Symposium on Operating Systems Design and Implementation.

[10] Woods, D. D. (2015). Four concepts for resilience and the implications for the future of resilience engineering. Reliability Engineering & System Safety, 141, 5–9. https://doi.org/10.1016/j.ress.2015.03.018

[11] Xu, J., et al. (2021). Resilience monitoring and failure mitigation in data-intensive service platforms. Journal of Systems and Software, 177, 110962.

[12] Zio, E. (2016). Challenges in the vulnerability and risk analysis of critical infrastructures. Reliability Engineering & System Safety, 152, 137–150. https://doi.org/10.1016/j.ress.2016.02.009