Keywords: application promises; explicit claims; implied claims; consumer protection; testing process; guidelines AMTSO

2026-09-22

Version 1.0

The cybersecurity industry's testing standard community

AMTSO Guidelines for Testing Application Promises

Abstract

These Guidelines provide definitions, methodology, and procedural guidance required for effective, fair, and unbiased testing of the explicit and implied promises that applications make to their customers, and of the controls that application vendors operate to honor those promises.

Notice and Disclaimer of Liability Concerning the Use of AMTSO Documents

This document is published with the understanding that AMTSO members are supplying this information for general educational purposes only. No professional engineering or any other professional services or advice is being offered hereby. Therefore, you must use your own skill and judgment when reviewing this document and not solely rely on the information provided herein.

AMTSO believes that the information in this document is accurate as of the date of publication although it has not verified its accuracy or determined if there are any errors. Further, such information is subject to change without notice and AMTSO is under no obligation to provide any updates or corrections.

You understand and agree that this document is provided to you exclusively on an as-is basis without any representations or warranties of any kind whether express, implied, or statutory. Without limiting the foregoing, AMTSO expressly disclaims all warranties of merchantability, non-infringement, continuous operation, completeness, quality, accuracy, and fitness for a particular purpose.

In no event shall AMTSO be liable for any damages or losses of any kind (including, without limitation, any lost profits, lost data, or business interruption) arising directly or indirectly out of any use of this document including, without limitation, any direct, indirect, special, incidental, consequential, exemplary, and punitive damages regardless of whether any person or entity was advised of the possibility of such damages.

This document is protected by AMTSO's intellectual property rights and may be additionally protected by the intellectual property rights of others.

Contents

1. Introduction

Every application makes promises. Some promises are explicit: a privacy policy that states what data the app collects and with whom it is shared, a feature list that describes what the app does, an install screen that explains what the user is agreeing to. Other promises are implied: a flashlight app implies that it will not harvest the user's contact list, a security product implies that it will not weaken the security of the device it runs on, and any app distributed through a major platform store implies that it complies with that store's policies. Consumers make installation and purchase decisions based on these promises, yet today there is no standardized, vendor-neutral methodology for verifying that an application honors them.

The gap matters. When an application breaks its promises, such as collecting data before consent is given, allowing its capabilities to be misused by third parties, or behaving differently than its description states, the harm falls on consumers who had no realistic way to detect the broken promise, whether it is deception or oversight. Anti-malware vendors, platform stores, and certification bodies all make blocking, removal, and certification decisions that depend on whether apps keep their promises, and independent testing of promise-keeping strengthens the evidence base for all of these decisions.

The expectation that applications honor their claims is already an established, cross-industry norm rather than the invention of any single organization:

- **Apple's App Review Guidelines** (§2.3, Accurate Metadata) require that an app's metadata, including descriptions and screenshots, accurately reflect the app's core experience, and Apple removes apps and expels developers who attempt to cheat the system or steal user data.
- **Google Play's Deceptive Behavior policy** prohibits apps that mislead users or enable dishonest behavior, requires that apps "perform as reasonably and accurately expected by the user," and requires explicit, reversible user consent for changes to device settings.
- **Microsoft Store Policies** (§10.1, Distinct Function & Value; Accurate Representation) require that a product and its metadata accurately and clearly reflect the source, functionality, and features of the product, and that a product must not attempt to mislead customers in any way; §10.2.2 further prohibits dynamically changing described functionality, and §10.5 requires that data practices match the published privacy policy.
- **AV vendor unwanted-software criteria** (for example, Microsoft's criteria for Unwanted and Malicious Software) treat misleading claims, lack of user consent, and behavior beyond stated functionality as grounds for detection.
- **AppEsteem's Application Certification Requirements (ACRs),** developed by AppEsteem working with the anti-virus community, encode promise-keeping at Deceptor level. ACR-015 (effective 1 October 2026) requires that an app "protects from app abuse, takeover, and unwanted misuse of its capabilities; honors the spirit of its claims and implications of how it will protect consumer and system data," with the intent that consumers can rely on all privacy and protection claims an app makes or implies.

These Guidelines define a horizontal methodology that sits above category-specific AMTSO guidelines. Where a promise falls within an area already covered by an existing AMTSO guideline, for example performance promises made by a VPN product, or protection promises made by an IoT security product, testers should apply the relevant category-specific guideline for the mechanics of that test and use this document for the surrounding methodology: identifying the promises, deciding what is testable, and reporting the results fairly. Worked

examples, including testing of residential proxy operator promises and of privacy consent and data handling promises, are provided in Appendix A. The examples are illustrative only. They do not make the referenced criteria, test plans, vendors, certifiers, or enforcement programs part of the Guidelines themselves.

These Guidelines are intended to comply fully with the AMTSO Fundamental Principles of Testing; Appendix B maps each Principle to the sections of this document that implement it.

1.1. Contributing Members

AMTSO acknowledges the contributions made to the construction of these Guidelines by the following individuals and companies along with the general membership of the AMTSO Application Promises Working Group.

Dennis Batchelder, Hong Jia (AppEsteem)

Luis Corrons (Gen Digital)

John Hawes, Scott Jeffreys (AMTSO)

2. Core Definitions

Application (or **app**) refers to any software product or service offered to customers, regardless of platform (desktop, mobile, browser extension, smart device, network appliance) or distribution channel (platform store, direct download, bundling, pre-installation). The entity that develops, publishes, or operates the application is referred to as the **vendor**.

Application promise refers to any commitment, made or implied by a vendor, on which a reasonable customer would rely when deciding to install, purchase, configure, or continue using an application. Promises may concern what the application does, what it does not do, what it protects, and how the vendor behaves in operating it.

Customer, consumer, and user should be used carefully. In many tests these roles are the same person, but in some application categories they differ. For example, in residential proxy testing, the paying proxy customer is not the same person as the consumer whose device or connection is used. Where the roles differ, the test methodology should identify which party each promise protects and whose expectations are being assessed.

Explicit promise refers to a promise stated by the vendor in any customer-facing material, including offer and landing pages, advertisements, store listings and metadata, install and consent flows, in-product user interface and messaging, documentation, support content, terms of service, and privacy policies.

Implied promise refers to a promise that a reasonable customer would understand to be in force even though the vendor has not stated it, arising from the application's category and presentation, from the policies of the platforms and stores through which it is distributed, from applicable regulation, or from established norms among comparable applications. Under these Guidelines, an implied promise may only be asserted by a tester when it is supported by documented evidence (see Section 3.2).

Promise inventory refers to the documented set of promises, explicit and implied, each with its provenance and evidence, that a tester compiles for an application before designing tests. The promise inventory serves the same role in promise testing that a validated sample set serves in detection testing.

Honoring, partial honoring, and violation. A promise is *honored* when the application's observed behavior and the vendor's observed conduct are consistent with the promise; *partially honored* when behavior is consistent in some tested scenarios but not others, or when controls exist but are materially incomplete; and *violated* when observed behavior or conduct contradicts the promise.

2.1. Promise Taxonomy

Promises fall into recurring types. The taxonomy below is intended to help testers build complete inventories and to help readers compare results across tests; it is not exhaustive, and a single promise may fall into more than one type.

Promise type	Examples of what it covers
Privacy and data handling	What data is collected, when collection begins, what consent is obtained, how data is used, stored, shared, sold, and retained, and how deletion requests are honored.
Protection and safety	What the application protects the customer or their device from, including security claims, parental or content controls, and claims of keeping the customer "safe."
Anti-abuse and misuse prevention	Controls that prevent the application or its capabilities from being abused, taken over, or misused by third parties, including the vendor's own customers, against the interests of the consumer hosting the application.
Performance and resource use	Claims about speed, efficiency, battery, bandwidth, and what device resources (CPU, GPU, network, storage) the application will and will not consume.
Functional and behavioral	That the application does what its metadata, description, and UI say it does; that it does not change device or browser settings without consent; that changes are reversible; that it uninstalls cleanly.
Sourcing and consent	That capabilities installed on a consumer's device (for example, a bundled residential proxy or SDK) were obtained with informed consent, and that the consented scope of use is respected after installation.

2.2. Failure Modes and Collateral Impact

Promise testing should distinguish between two primary promise failures and one additional collateral-impact dimension:

- **Under-delivery:** the application makes a promise and breaks it. Example: a privacy policy states data is collected only after consent, but network capture shows telemetry transmitted on first launch before the consent screen is shown.
- **Over-claiming:** the application makes a promise it cannot keep, or that is functionally impossible. Example: a claim to block "all" threats of a given type, or a capability claim

that the device hardware cannot support. Over-claiming is itself a form of deception even when no individual customer is observably harmed.

- **Over-blocking or collateral impact**: the application enforces a promise or protective control in a way that blocks legitimate activity, degrades normal use, or creates avoidable harm outside the promise being tested. Example: a safety control blocks ordinary public web access that the application claims to support.

A balanced test should address the relevant dimensions for the promises under test. A test that probes only for broken promises, without also verifying that legitimate promised functionality works and that protective controls do not over-restrict legitimate activity, is incomplete.

3. Building the Promise Inventory

The promise inventory is the foundation of the test. Just as detection testing depends on accurately classified samples (Fundamental Principle 5), promise testing depends on an accurately compiled and validated promise inventory. Errors at this stage, such as attributing a promise the vendor never made, misreading an ambiguous claim, or asserting an implied promise without evidence, invalidate the test results built on them.

3.1. Cataloging Explicit Promises

Testers should systematically review all customer-facing vendor materials, including:

- Offer pages, landing pages, and advertisements;
- Platform store listings and metadata (title, description, screenshots, category, declared permissions, data-safety declarations);
- Download, install, and consent flows, including bundled-offer screens;
- In-product user interface, settings, notifications, and upsell messaging;
- Documentation, FAQs, knowledge bases, and support responses;
- Terms of service, end-user license agreements, and privacy policies.

For each promise catalogued, the tester should record its provenance: the exact wording, where it appears, the date and application version at which it was captured, and a preserved copy (screenshot or archived page). Promises change over time; vendors revise policies, listings, and install flows, so the inventory must be anchored to a point in time, and the test report should state the capture dates. Where a promise appears in multiple places with inconsistent wording, the tester should record all variants and note the inconsistency, which may itself be a reportable finding.

Where relevant, testers should also account for regional variants, localized materials, platform-specific listings, cohort-specific or A/B tested flows, and time-limited promotions.

3.2. Asserting Implied Promises: the Evidence Standard

Implied promises are where promise testing is most powerful and most contestable. A tester who asserts that an application implicitly promised something the vendor never stated must be able to defend that assertion. Under these Guidelines, an implied promise may be included in the promise inventory only when it is backed by documented evidence. Acceptable categories of evidence include:

- **Platform and store policies** that the application is subject to by virtue of its distribution channel. An app distributed through Google Play implicitly promises to comply with Google Play policies (for example, that it performs as reasonably expected by the user and that device-settings changes require consent and are reversible); likewise for Apple's App Review Guidelines, Microsoft Store Policies, and comparable channel policies.
- **Anti-malware and certification criteria,** including AV vendors' published unwanted-software and PUA criteria, and certification or watchlist requirements such as AppEsteem's ACRs and ADAs. Detection of the application or its components by reputable AV vendors as PUA/unwanted software may be supporting evidence that the application is failing an industry expectation, but it is not sufficient on its own. Testers should consider the basis, consistency, and dispute status of such detections before using them to support an implied promise.
- **Applicable regulation,** such as consent and data-protection requirements under GDPR, CCPA, and comparable regimes, where the application is offered in the relevant jurisdictions. Testers should take care to frame regulatory expectations as testable behaviors (for example, "no personal data is transmitted before consent is obtained") rather than offering legal conclusions. Where legal interpretation would be required, testers should either rely on qualified legal analysis or avoid presenting the finding as a legal conclusion.
- **Category norms,** established by a documented survey of comparable applications. If a tester asserts that applications of a given category do not engage in a given behavior, the survey of comparators, which apps were examined, when, and what was found, should be retained and disclosed.
- **Consumer-side evidence,** including customer reviews and complaints, credible news reporting of application misbehavior, and regulator or consumer-protection actions, which can establish both what customers reasonably expected and where expectations were betrayed.

Each implied promise in the inventory must cite its supporting evidence. Implied promises asserted without evidence, or supported only by the tester's own intuition, must not be tested or reported as promises. Testers should also avoid stretching evidence: a platform policy implies compliance with that policy, not with the tester's preferred stricter standard.

3.3. Validating the Inventory

Consistent with Fundamental Principle 5, testers must take reasonable care to validate that each inventory entry is accurately classified before testing begins:

- Confirm the promise is genuine: it is a commitment a reasonable customer would rely on, not marketing puffery ("the world's best"), aspiration, or a forward-looking statement. Where the line is unclear, the tester should either resolve the ambiguity (for example, by querying the vendor contact point, qualify the promise, test only the unambiguous component) or exclude the entry and disclose the exclusion.
- Confirm the promise is current: it appears in materials in force for the application version under test, and has not been withdrawn or superseded.
- Confirm the promise applies to the configuration under test: some promises are scoped to specific platforms, regions, editions, or settings.

- Re-validate any promise whose test outcome is a violation, just as detection testers are encouraged to revalidate samples that produce false-positive or false-negative results, before reporting it.

The validated promise inventory, including exclusions and their reasons, should be published with or referenced from the test report (see Section 7), so that readers and vendors can assess whether the test measured promises the vendor actually made or reasonably owed.

4. Testability Assessment

Not every promise in the inventory can be tested, and not every testable promise can be tested the same way. Before designing test cases, the tester must classify each promise by how, and whether, it can be measured, and must disclose that classification in the test methodology. Conclusions may only be drawn from what was actually measured (Fundamental Principle 7).

4.1. Directly Testable Promises

A promise is directly testable when honoring or violation is observable in the behavior of the application or service itself, using techniques such as:

- Network traffic capture and analysis on the endpoint or from the network, including TLS interception in controlled environments;
- Filesystem, registry/settings, and process monitoring before, during, and after install, use, and uninstall;
- Resource-consumption measurement (CPU, GPU, memory, bandwidth) against claimed limits;
- Probe scenarios that attempt the behavior the promise forbids (for example, attempting to use an application's capability for a disallowed purpose and observing whether it is blocked);
- Functional verification that promised features exist and operate as described.

Directly testable promises should generally be preferred as the core of the test, since they support the strongest conclusions.

4.2. Indirectly Measurable Promises

Some promises concern vendor conduct or controls whose outcomes the tester cannot observe directly, for example: "we operate a robust vetting (KYC) program for customers requesting elevated access," "we log and monitor usage," "we promptly address vulnerabilities in our components," or "we maintain a public list of approved customers." For these, the tester evaluates observable proxies for the promise, such as:

- The documented existence, scope, and apparent operation of the claimed control (published program descriptions, public listings, transparency reports, audit attestations);
- The vendor's observable track record (update cadence, CVE response history, breach disclosures, responsiveness of the published contact point);
- Where appropriate and ethical, exercising the control itself (for example, submitting a data-deletion request, or applying through the vendor's published process) and observing how it is handled.

Testers are not required to exercise a control in order to assess it; however, the test methodology must disclose, for each indirectly measured promise, exactly how it was measured: what was examined, what was exercised, and what was taken on the strength of vendor documentation alone. The report must make clear that for indirectly measured promises the tester evaluated the control or its evidence, not the outcome, and must not present such results with the same strength as direct behavioral observations.

4.3. Promises That Cannot Be Tested

Some promises offer no observable surface at all within the tester's lawful and ethical reach, for example claims about purely internal vendor conduct for which no direct or proxy evidence channel exists. These should be recorded in the inventory, marked as not tested, and listed in the report with a clear statement that no judgment on them can be derived from the test results. This mirrors the handling of untested features in other AMTSO guidelines: silence about a promise is not endorsement, and readers must be told the difference between "honored" and "not evaluated."

Where a promise is only partially observable, testers should decompose it into sub-promises and classify each separately. For example, "we never sell your data" may be untestable as stated, but decomposes into testable components (no transmission of identifiable data to known data brokers; no appearance of seeded canary data in third-party datasets; data-sharing disclosures consistent with observed traffic) and indirectly measurable components (privacy-policy consistency, published audit results). The report should make clear which components were tested and that conclusions are limited to them. If a claimed control is reviewed but not exercised, the report should label the result as documentation-reviewed or evidence-reviewed, not as control-validated.

5. General Test Design Considerations

When designing a test, whether evaluating a single application or comparing several, close attention should be paid to ensuring the test is fair, balanced, and unbiased (Fundamental Principles 2 and 4). The aim is to provide useful, accurate information on whether the application honors the promises a reasonable customer relies on.

5.1. Defining the Test Scope

A complete promise inventory (Section 3) will frequently contain more promises than any single test can or should address. Before designing test cases, the tester must make and record an explicit scoping decision: which promises, grouped by category, are in scope for this test, and which are deliberately excluded, with the reason for each exclusion. This scoping step is distinct from the testability classification in Section 4. Testability asks whether a promise *can* be measured; scoping asks whether, for the purpose of this particular test, the tester *will* measure it. A promise may be perfectly testable yet reasonably excluded from a given test, and a test that silently omits whole categories of promise is neither transparent nor fair to the applications under comparison.

For each promise or promise category, the tester should record one of three dispositions, with the supporting reasoning:

- **In scope**: the promise will be tested, with a note on whether it will be measured directly or indirectly (Section 4).

- **Out of scope**: the promise will not be tested in this test, with a stated reason. Legitimate reasons include: the promise falls within an area covered by another AMTSO guideline and is being tested there instead; the test has a deliberately narrow stated purpose that the promise falls outside of; the promise cannot be tested without endangering the public or a third party (Section 6); resource or access constraints prevent fair measurement; or the promise is better addressed in a separate, dedicated test.
- **Not testable**: the promise offers no observable surface within the tester's lawful and ethical reach (Section 4.3), and so is recorded as evaluated for testability but not tested.

The scoping decision should be guided by the test's stated purpose (Fundamental Principle 6) and should not be used to quietly avoid promises that are inconvenient, unflattering to a favored product, or commercially sensitive. In a comparative test, scope must be defined at the category level and applied equally across all applications, so that one product is not credited for honoring a promise that was not examined in another. The scope, including the in-scope categories and the excluded promises with their reasons, must be disclosed in the test methodology and report (Section 7), so that readers understand the boundaries of the test and do not read a narrow test as a broad endorsement.

5.2. Application and Promise Selection

Applications should be selected for testing on a basis the tester can state and defend. The scope of a test, which applications and which of their promises, should be chosen to match the test's stated purpose (Fundamental Principle 6). Where an application makes no explicit promise in an area but a general customer would reasonably expect protection or restraint, the tester may include that area on the strength of an evidenced implied promise (Section 3.2), but the tester should state clearly that the vendor made no explicit claim. In comparative tests, the same promise types should be tested across all applications where comparable and applicable, and apparent differences in promises between products should be reported rather than silently normalized.

5.3. Test Environment

Test systems should be representative of those in general use, with current operating system versions and updates, and should be reverted to a known-clean state before each application is installed and before each test case is run, to avoid contamination from prior testing. Connectivity to any resources required by the application or by the test cases should be verified before testing. Because both promises and application behavior drift over time, the tester should pin and record the application version, the dates of testing, and the dates on which each promise was captured. Where the tester deliberately departs from representative versions or configurations, this should be disclosed and justified.

5.4. Designing Test Cases

Test cases should probe both failure modes defined in Section 2.2, under-delivery and over-claiming, and should include positive cases that confirm promised functionality works and that protective controls do not over-restrict legitimate activity. As in scam and phishing testing, test cases fall into two broad categories:

- **Real-world cases** observe the application's behavior under genuine conditions, real network destinations, real data flows, real usage, and generally yield the most representative results.

- **Artificial or simulated cases** introduce controlled stimuli, for example a tester-controlled endpoint that a promise says should not be reached, a seeded canary identifier, or a scripted attempt to misuse a capability, and allow precise, repeatable measurement. Where a test case is introduced differently than it would arise in the real world, the tester should note that the change of vector may affect the application's response and the accuracy of the result.

Test cases should be designed so that a clear, defensible inference can be drawn from each observed outcome, and so that the volume and quality of cases provide a sufficient evidentiary basis for the conclusions drawn, using statistical methods where the sampling model supports statistical claims (Fundamental Principle 8). Where a promise is tested across many scenarios (for example, many destinations or many sessions), the tester should use enough cases to distinguish a genuine pattern from noise and disclose the counts.

5.5. Confirmation and Verification

As with all testing, testers should confirm and verify their results before publication. Testers may include the application's vendor in this process, providing pre-publication access to findings and allowing the vendor to challenge unexpected or questionable outcomes through the published contact point (Fundamental Principle 9). Where a promise or test case is judged unsuitable after a dispute, for example the asserted promise was not in fact made, or was withdrawn before the version under test, the affected results should be removed from all tested applications to preserve balance. The AMTSO Testing Protocol Standard provides a framework for communications between testers and test subjects and should be referenced where applicable.

5.6. Reporting, Scoring and Rating

Measuring whether an application honors its promises can be more complex than measuring detection, because outcomes are often graded rather than binary: a promise may be honored, partially honored, violated, not evaluated, or evaluated only indirectly.. These Guidelines therefore prescribe a scoring discipline rather than a fixed scoring formula. A test may use a score, rating, certification outcome, pass/fail decision, or promise-by-promise reporting, provided the approach is documented, applied consistently, and does not imply broader conclusions than the evidence supports. Testers should:

- Develop a rigorous reporting or scoring process, as appropriate to the test purpose, that accounts for graded outcomes (honored / partially honored / violated / not evaluated) and that distinguishes directly observed results from indirectly measured ones;
- Document the reporting, scoring, rating or certification process in the test methodology and apply it equally across all applications tested;
- Where weights, scores, or ratings are used, weight promises according to their significance to consumer protection and state the weighting rationale. Where no score is used, explain how promise-level results are interpreted.
- Include both failure modes in scoring, broken promises (under-delivery) and impossible or unkept claims (over-claiming), together with positive verification that promised functionality works and that controls do not over-block;
- Base every published conclusion on the test results obtained, drawing no broader conclusion than the measured evidence supports (Fundamental Principle 7).

Where a simpler measurement suffices for a given promise, for example a binary "did identifiable data leave the device before consent: yes or no", a simpler scoring method may be used for that promise, provided it is documented and applied consistently.

6. Tester Conduct and Public Safety

Testing application promises can involve real consumers, real consumer devices, live vendor services, and live third-party websites. Consistent with Fundamental Principle 1 (testing must not endanger the public) and Principle 10 (all parties must act in good faith), testers must take care not to cause harm during testing.

Some tests may require live third-party destinations to verify category reachability, brand-specific behavior, or real-world routing. This is acceptable only when the test does not perform harmful actions against those destinations. Harmful actions, exploit attempts, destructive requests, abusive automation, or actions likely to damage reputation or availability should be performed only against tester-controlled or purpose-built infrastructure:

- Do not harm real consumers or their devices. Where a test touches applications or capabilities installed on genuine consumer endpoints, testers must not cause those endpoints to perform harmful, illegal, or unconsented actions.
- Do not harm live third-party websites or services. Use tester-controlled or purpose-built targets to demonstrate a forbidden capability rather than carrying it out against a live victim: test proof of concept, not proof of harm. For example, demonstrate that a capability is not blocked without completing an attack that would damage a real site.
- Use only the access a normal customer would have. Testers should obtain the same standard level of access granted to ordinary customers and must not rely on special access that would make results unrepresentative, unless the elevated access is itself the subject of the test and is disclosed.
- Do not release malware into the wild; follow industry-standard containment for any malicious samples or infrastructure used, in line with the Fundamental Principles.
- Use intrusive techniques responsibly. Techniques such as seeding canary data into intelligence-sharing systems, or submitting test cases to community feeds, can pollute shared resources and mislead other parties; testers should use the minimum necessary, avoid contaminating community systems with irrelevant data, and disclose such techniques.
- Handle any personal data encountered during testing lawfully and minimally; avoid collecting or retaining real consumers' personal information beyond what the test requires.

7. Transparency and Disclosure

Consistent with Fundamental Principles 2, 3, and 9, any test released to the public should be accompanied by, or reference the location of, the information a reader needs to assess its fairness and reliability. For promise testing this includes:

- The validated promise inventory, including the provenance and capture dates of explicit promises, the evidence supporting each implied promise, and any promises excluded from the inventory with the reasons for exclusion;
- The test scope: which promise categories were in scope, and which promises were placed out of scope, each with the reason (Section 5.1);
- The testability classification of each promise (directly testable, indirectly measurable, or not tested) and, for indirectly measured promises, exactly how each was measured;

- The applications and versions tested, how they were obtained and configured, and the test environment and dates;
- The methodology and scoring process, applied equally across all applications, and how results were calculated and interpreted;
- Any significant financial relationship between the tester or publisher and a tested vendor or affiliate, disclosed to avoid the appearance of bias;
- An active contact point for testing-related correspondence, and a good-faith commitment to respond to vendor questions and disputes within a reasonable time.

Transparency should not require disclosure that would create avoidable harm. Testers may withhold or generalize exact samples, exploit details, personal data, live harmful destinations, bypass details, or operational indicators where publishing them would enable abuse, compromise safety, or violate legal or contractual duties. In such cases, the report should disclose the category of information withheld and the reason for withholding it.

Where a promise area falls within an existing category-specific AMTSO guideline, the report should reference that guideline and follow its disclosure requirements in addition to those above.

Appendix A: Example Testing Scenarios

The following worked examples illustrate how the methodology above is applied to specific promise areas. They are illustrative, not prescriptive, and testers are invited to contribute additional scenarios to AMTSO as a community resource. They should not be read as AMTSO endorsement of any specific commercial program, certification scheme, enforcement action, or vendor position.

A.1. Residential Proxy Operator Promises

This example applies the methodology to a residential proxy operator: a vendor that obtains residential IP endpoints by bundling a proxy with consumer applications, obtains the consumer's informed consent to install it, and then resells web access through those endpoints to its own customers. The promises at issue are that the residential proxy was ethically sourced (installed with informed consent) and that the operator will keep the consumer and their device safe and limit the proxy's use to legitimate business activity. This worked example is based on one implementation approach and is included only as an illustration of how the methodology can be applied.

Scope

This test is scoped to the promises the operator makes about operating the residential proxy: ethical sourcing and consent, anti-abuse and misuse prevention, protection of the host endpoint, and the limits placed on what customers may do through the proxy. Promises unrelated to operating the proxy, such as billing accuracy, marketing claims about speed or coverage, or the data-handling practices of any companion application, are out of scope for this test and would be addressed separately; where a companion application bundles the proxy, its sourcing and consent flow may be cross-referenced but its broader privacy promises are not evaluated here. The scope is recorded so that a clean result is not read as an endorsement of the promises this test did not examine.

Promise inventory

- **Explicit:** the operator's claims of "ethical sourcing," its public statement of disallowed activities, and its description of customer vetting and access controls.
- **Implied (evidenced):** that a consumer who consented to host a proxy did not consent to becoming a node in a rentable botnet, supported by documented criteria such as certification requirements, AV PUA criteria, platform policies on device abuse, and the consumer-harm record. Four consumer harms motivate the test: access harm (the consumer's IP is flagged and their own browsing degrades), legal exposure, moral/consent harm, and endpoint takeover.

Expectations derived from the promises

The operator is expected to prevent the proxy from being used to attack others, to block access to malicious or dark-web destinations, to stop bypass attempts against blocked content, to protect the endpoint from being hijacked for resource harvesting, and, for non-vetted customers, to limit traffic to public web data retrieved by simple GET requests, blocking logins, account creation, posting, and access to fraud-prone site categories. The operator is also expected to run a vetting (KYC) program for elevated access, to publicly list customers granted elevated access and the justification, and to maintain a public statement of disallowed activities.

Test design (testability applied)

- **Directly testable:** attempting, as an ordinary customer, each disallowed action (credential validation, sending mail or posting, sandbox escape, reconnaissance, category/host bypass, ad-click fraud, account creation, reaching malware/breach or dark-web sites, launching attacks, CPU/GPU harvesting, peer-to-peer use) and observing whether it is blocked; and confirming that legitimate public-web-data retrieval is allowed (a positive case guarding against over-blocking).
- **Indirectly measurable:** the existence and apparent operation of the vetting program, usage logging, proxy updating, the public customer listing, and the public disallowed-activities statement, with the tester disclosing how each was assessed.

Tester conduct

Testers take only standard customer access and must not be granted special rights; test for both missed blocks and over-blocking; demonstrate forbidden capabilities against purpose-built targets (for example, betterbrowsing.org) rather than harming live sites, demonstrating proof of concept, not proof of a successful attack; and never cause live consumer endpoints to perform illegal actions.

Tester financial relationships

Tester certifies some of the residential proxies, so has a financial interest in ensuring they can pass this test. Tester has also placed some of the to-be-tested residential proxies on its active Deceptor lists; since some of these residential proxies will sign up for its certification or other remedy services, tester has a financial interest in finding violations.

A.2. Privacy Consent and Data-Handling Promises

This example applies the methodology to an application's privacy promises, drawing implied promises from platform policies and applicable data-protection regulation.

Scope

This test is scoped to privacy and data-handling promises only: what data is collected, when collection begins relative to consent, and whether actual data practices match what is disclosed. Other promise categories the application may make, such as protection, performance, or functional claims, are out of scope for this test and would be evaluated separately. Scoping the test to privacy alone keeps the consent-timing and disclosure-consistency findings focused and ensures the result is not read as a judgment on promises this test did not examine.

Promise inventory

- **Explicit:** statements in the privacy policy and store data-safety declaration about what is collected, when, for what purpose, and with whom it is shared; any "we do not sell your data" claim.
- **Implied (evidenced):** that personal data is not collected before consent and that data practices match disclosures, supported by Google Play and Apple privacy requirements, Microsoft Store §10.5, and GDPR/CCPA consent obligations where the app is offered in those jurisdictions.

Test design (testability applied)

- **Directly testable, consent timing:** capture network traffic from a clean install through first launch and the consent flow; determine whether identifiable or personal data is transmitted before the user grants consent. A transmission before consent is a direct violation of an evidenced implied promise.

- **Directly testable, disclosure consistency:** enumerate the third-party endpoints and SDKs the app actually contacts and compare them against the disclosed sharing; transmission to undisclosed third parties, or to known data brokers, is a finding. Seeded canary data (unique identifiers planted as the only copy) can be used to detect onward sharing where it later appears in third-party datasets.
- **Indirectly measurable:** the privacy policy's internal consistency and currency, the operation of data-access and deletion mechanisms (which the tester may choose to exercise), and any published audit, with disclosure of how each was assessed.
- **Not testable as stated:** a bare "we never sell your data" claim with no observable surface; decompose into the testable and indirect components above and report that the conclusion is limited to them.

Ethics

Canary and synthetic-data techniques must use data that is genuinely synthetic and harmless, must not pollute shared intelligence systems, and must not entail collecting real third parties' personal information. Where a deletion or access mechanism is exercised, it should be exercised with the tester's own test-account data, not a real consumer's.

Tester financial relationships

Tester certifies or wishes to certify/help remedy some of the to-be-tested apps, so has a financial interest in finding violations.

Appendix B: Mapping to the AMTSO Fundamental Principles

The following table shows where each of the AMTSO Fundamental Principles of Testing is implemented in these Guidelines.

Fundamental Principle	Implemented in
1. Testing must not endanger the public	Section 6 (Tester Conduct and Public Safety)
2. Testing must be unbiased	Sections 5.2, 5.5; Section 7 (financial-relationship and methodology disclosure)
3. Testing should be reasonably open and transparent	Section 7 (Transparency and Disclosure)
4. Effectiveness and performance measured in a balanced way	Section 2.2 (failure modes and collateral impact); Section 5.4, 5.6 (balanced cases and scoring)
5. Reasonable care to validate classification	Section 3 (building and validating the promise inventory), esp. 3.2 to 3.3
6. Methodology consistent with testing purpose	Sections 2.1, 5.1, 5.2 (scope and selection matched to purpose)
7. Conclusions based on the test results	Section 4 (testability); Section 5.6 (no conclusion beyond evidence)
8. Test results should be statistically valid	Section 5.4 (sufficient evidentiary basis, disclosed case counts where applicable)
9. Active contact point for correspondence	Sections 5.5, 7
10. All parties must act in good faith	Section 5.5 (dispute handling); Section 6 (tester conduct)

Appendix C: Reference Materials

The following materials are referenced in these Guidelines or provide useful background. References to third-party policies are provided as examples of the cross-industry norm that applications should honor their promises; they are not endorsements, and testers should consult the current version of any policy at the time of testing.

- AMTSO, The Fundamental Principles of Testing.
- AMTSO, Testing Protocol Standard.
- AppEsteem, Application Certification Requirements (ACRs), including any current or scheduled requirements relevant to application promises, and AppEsteem Deceptor Activities (ADAs).
- Apple, App Review Guidelines (§2.3 Accurate Metadata; §5.1 Privacy).
- Google Play, Developer Program Policy, Deceptive Behavior (including Misleading Claims and Deceptive Device Settings Changes) and User Data policy.
- Microsoft Store Policies (§10.1 Distinct Function & Value; §10.2 Security; §10.5 Personal Information).
- Microsoft, Criteria for Unwanted and Malicious Software.
- Applicable data-protection regulation, including the EU General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA).

DRAFT v1.0 approved and adopted by the AMTSO membership 2026-09-22.

