Version 1.7 Last update: March 3, 2026

University of Texas System
UT Aspire
Security Controls & Safe Use Guide

For End Users

Disclaimer: As the UT System's UT Aspire Private GenAI services evolve, we will continue to update this document. Please check it frequently for updates.

1. Purpose

UT Aspire provides a private Generative AI (GenAI) environment for UT System institutions. This document explains UT Aspire's security controls, how to reduce the likelihood of automated safety actions, and how to request investigation of suspected false positives.

Reminder: GenAI responses can be incomplete or incorrect. Review outputs for accuracy and use iterative refinement by continually rephrasing and adding context to improve results.

2. Security Controls at a Glance

UT Aspire uses layered security controls to protect users and the UT System while enabling productive use.

2.1 Gateway-Level Detection & Response (HiddenLayer AIDR)

HiddenLayer is deployed on the UT Aspire Lite LLM gateway to detect potentially risky prompt or response patterns and may block, redact, or return safety messages when triggered.

2.2 Native / Default Guardrails

UT Aspire also applies default guardrails to handle unsafe, disallowed, or non-compliant content.

2.3 Operational Safeguards (Privacy, Logging, Monitoring)

UT Aspire keeps interactions within the environment. Activity logs may be retained for auditing, monitoring, and investigation purposes.

3. Why Controls Sometimes Take Action

Controls act on patterns and risk signals, not intent. Legitimate requests may occasionally resemble restricted patterns.

- Refusals or blocks
- Redactions
- Requests to rephrase
- Safer alternative responses

4. How to Reduce the Likelihood of Triggering Controls

Use clear, context-rich prompts that explain legitimate purpose.

5. Examples of Restricted Categories (Not Exhaustive)

This list is illustrative only and not complete.

- Instructions for wrongdoing or exploitation
- Evasion or bypass of controls
- Violence or self-harm instructions
- Harassment or hate content
- Explicit sexual content
- Exposure of confidential or regulated data
- Fraud, impersonation, or social engineering

6. Chat Deletion vs. Logging

Deleting a chat removes it from the user interface, but activity logs may still exist for authorized auditing and investigation.

7. False Positive Process

If you believe HiddenLayer or a native guardrail took action incorrectly and would like it investigated, work with your campus or institution IT/support staff to submit a ServiceNow ticket on your behalf in the UT Shared Information Services (SIS) ServiceNow instance. Tickets should be submitted under the Category: Generative Artificial Intelligence (GenAI) and include all relevant details.

Individual end users (including students) should not submit tickets directly to SIS. Your campus or institution support staff can submit the ticket on your behalf if escalation is appropriate.

Appendix A: Known-Good Prompt Templates

These templates are examples designed to work well within UT Aspire.

A. Policy Drafting

"Draft a plain-language policy section for a UT System audience on [TOPIC]. Focus on intent and compliance. Avoid operational detail."

B. Communications

"Draft a professional message for [AUDIENCE] explaining [PURPOSE]. Provide safe, high-level guidance."

C. Summarization

"Summarize this published, non-confidential document for a general audience. Do not add new facts."

D. Awareness Content

"Create an awareness-level overview of risks related to [TOPIC] with prevention best practices."

Appendix B: Code-Specific Known-Good Prompt Examples

These templates are designed to be clear, compliant, and aligned with HiddenLayer and native guardrails. Replace bracketed fields with your details and do not include sensitive or restricted data.

How to Redact Code Safely (Before You Paste)

To reduce risk and avoid guardrail actions, remove or replace sensitive details with placeholders such as [TOKEN], [HOST], or [ENDPOINT]:

- Secrets and credentials: passwords, API keys, OAuth tokens, session cookies, private keys, certificates.
- Environment variables and config values: .env contents, connection strings, service principals, tenant IDs when paired with credentials.
- Endpoints and internal details: internal hostnames, IPs, VPN routes, non-public URLs, identifiable infrastructure names.
- Sensitive data in examples: student/patient data, HR data, IDs, addresses, or any confidential/controlled data.
- Proprietary code you are not authorized to share: replace with minimal reproducible snippets or pseudocode.

B.1 Secure Code Review (Defensive)

```
You are a secure code reviewer. Review the following code for security
and reliability issues.
Context: This is a learning/demo example (not production). Provide
defensive recommendations only.
Output: (1) prioritized findings, (2) suggested fixes, (3) safer
alternative patterns.
Code:
[PASTE REDACTED CODE SNIPPET]
```

B.2 Refactor for Readability

```
Help me refactor this function for readability and maintainability.
Constraints: keep behavior the same; no external dependencies; include
comments.
Output: revised code + short explanation of changes.
Code:
[PASTE CODE]
```

B.3 Add Input Validation + Error Handling

```
Show how to add robust input validation and error handling to this
code.
```

```
Focus on defensive programming patterns; avoid anything that could be
misused.
Code:
[PASTE CODE]
```

B.4 Unit Tests

```
Write unit tests for the following function using [pytest/JUnit/etc.].
Include edge cases and negative tests.
Function:
[PASTE FUNCTION]
```

B.5 Debugging with Minimal Disclosure

```
I'm seeing this error in my local dev environment. Help diagnose likely
causes and safe fixes.
Do not ask for secrets/tokens. If more context is needed, tell me
exactly what NON-sensitive info to share.
Error:
[PASTE ERROR]
Minimal code path:
[PASTE REDACTED SNIPPET]
```

B.6 Documentation

```
Create docstrings and a short README section for this module.
Audience: UT staff with basic technical background.
Code:
[PASTE CODE]
```

B.7 Design First (Clarifying Questions)

```
Before writing any code, ask me 5–8 clarifying questions about
requirements and constraints for: [FEATURE].
After I answer, propose a design and then produce a minimal, safe
implementation.
```

B.8 Threat Modeling (Defensive)

```
Help me do a lightweight threat model for this component.
Output: assets, trust boundaries, threat list (STRIDE), and
mitigations.
Avoid exploit instructions and focus on defense.
Component description:
[DESCRIBE ARCH / DATA FLOWS]
```

Safe Reframe (If You Hit a Block)

```
My intent is educational/defensive. Provide a high-level overview and
best practices only; do not include step-by-step misuse instructions.
```

Revision History

Version	Description	Revision Date
1.1	Added Quick Reference and Known-Good Prompt Templates	February 5, 2026
1.3	Formatted Appendix B prompts as code blocks; added redaction checklist; refreshed Quick Reference branding	February 11, 2026
1.4	Reformatted Quick Reference to match standard document headings and layout	February 16, 2026
1.5	Final consistency sweep; PDF layout optimization; wording alignment with FAQ/Prompt Literacy	February 18, 2026
1.6	Clarified ServiceNow escalation process via campus/institution staff and SIS ServiceNow only	March 2, 2026
1.7	Branded with UT System UT Aspire header and footer	March 3, 2026