The Orchestrator's Starter Kit

Seven tools for builders who build with AI.

A working toolkit for builders, developers, and product designers. Use it to design, audit, and govern any AI product. No prior reading required.

Every template in this kit traces back to a specific chapter of Dark Decisions: The Orchestrator's Guide to Building AI Products That Work When You're Not Watching by Tom H. Olakitan.

INTRODUCTION

How to use this kit

Seven tools. One for every critical moment in building a reliable AI product.

This kit converts the Orchestrator Framework into tools you can use on a product you are building right now. Use the tools in the order that matches where you are. If you have an existing product, start with the 8-Gate Diagnostic Card. If starting from scratch, begin with the Constitutional Declaration. If preparing to launch, go straight to the Go-Live Gate Checklist.

#	TOOL	WHEN TO USE IT	FOR WHOM
1	**8-Gate Diagnostic Card** Assess any existing AI product against all eight gates.	When auditing an existing product	Builder, Product Leader
2	**Constitutional Declaration** Define what your product believes before the first pipeline is built.	Before building begins	Builder, Founder
3	**Pipeline Specification** Define every step of a pipeline before writing a line of code.	Before each pipeline is built	Builder, Developer
4	**Standing Rules Starter** Five rules that activate automatically before every task.	During architecture design	Builder, Developer
5	**Go-Live Gate Checklist** Twenty questions. The definitive answer to: is this product ready to	Before every launch	Product Leader, Builder
6	**AI Product Failure Map** Map the five most common AI failures to the gates they violate.	When debugging or post-incident	Developer, Builder
7	**Edge Case Inventory** Eight AI-specific failure states every product designer must plan for.	During UX and product design	Designer, Developer

> To fill in digitally: open in any web browser, click any field and type. To save as PDF: File > Print > Save as PDF. To use on paper: print at 100% on US Letter. Every blank field is intentional. A field you cannot fill in is a gap worth naming.

PART 1: DIAGNOSE

8-Gate Diagnostic Card

Assess any AI product, including one already in production. Chapter 8.

For each gate, answer honestly. Pass means the mechanism is present in the architecture and operating. Fail means it is missing, declared but not enforced, or untested. Any failure is a named gap: the first thing to build next.

1 **Input Quality Gate**

Does the system filter every input before the AI processes it, with a defined minimum quality threshold, injection scanning, and a validation message when input fails?

If fail: Define the threshold. Write the exact validation message. Run it on every input, without exception.

☐ Pass ☐ Fail

2 **Retrieval Verification**

Does the system check that information it draws on is accurate, current, and above a defined relevance threshold, with defined handling when retrieval returns nothing useful?

If fail: Set a relevance threshold. Define what happens when retrieval returns nothing. Do not fill empty retrieval with training patterns.

☐ Pass ☐ Fail

3 **Verification Engine**

Does the system check the AI's output before it reaches any user, with defined VERIFIED criteria, a substance check, grounding against source material, and a fixed retry protocol?

If fail: Define VERIFIED criteria before generation runs, not after. One retry. FLAGGED outputs never reach users.

☐ Pass ☐ Fail

4 **Confidence Signal System**

Does every output carry a visible VERIFIED, INFERRED, or FLAGGED confidence signal, assigned by defined criteria, that cannot be suppressed for any reason?

If fail: The signal is missing or suppressible. Make it architectural, not behavioral.

☐ Pass ☐ Fail

5 **Permission Classification**

Has every action been classified as CONTAINED, BOUNDARY-CROSSING, NEVER AUTONOMOUS, or STRATEGIC, with higher classifications blocked without explicit human approval?

If fail: Build the NEVER AUTONOMOUS list now. Ambiguous classifications resolve upward, always.

☐ Pass ☐ Fail

6 **Feedback Loop and Logging**

Does the system log structured quality data from every human correction, naming the failure type specifically, in a form that can be analyzed for patterns over time?

If fail: A log that says 'output was wrong' is not a log. Name the failure type, pipeline step, and date.

☐ Pass ☐ Fail

7 **Standing Rules Engine**

Are the product's governing rules embedded in an engine that activates automatically before every task, rather than stated as instructions the model is expected to remember?

If fail: Instructions in a prompt are not standing rules. Move them into an engine.

☐ Pass ☐ Fail

8 **Escalation Pathway**

Is there a defined route for every decision above the system's autonomous authority, with a named recipient, resolution window, and automatic elevation if it passes unresolved?

If fail: Name the recipient and the window. Silence is never treated as approval.

☐ Pass ☐ Fail

Gates passing. Any gate that fails is the first thing to build.

out of 8

PART 2: DESIGN | TEMPLATE 1 OF 3

Constitutional Declaration

Complete one declaration per product before the first pipeline is built. Chapter 7.

The constitutional declaration is the belief system that governs every pipeline, every gate, and every output. A principle you cannot point to in the architecture is not inherited. It is decoration.

PRODUCT NAME

PRODUCT PURPOSE

One sentence: name the user, the domain, and the outcome.

WHAT THIS PRODUCT IS NOT DESIGNED TO DO

Name at least two things, regardless of what users request.

Principle 1: Strategy Precedes Implementation

When a request arrives that requires action before understanding, this product:

Principle 2: Honesty Over Comfort

When this product cannot generate a verified response, it:

Principle 3: The Measure Is Transformation

Success for this product means the user can:

Principle 4: Human Authority Is Non-Negotiable

The NEVER AUTONOMOUS actions for this product are:

Principle 5: Transparency Over Confidence

This product displays confidence to users using the following format:

Principle 6: Build Toward User Independence

This product builds user capability by:

DECLARED BY (NAME AND ROLE)

VERSION

DATE

PART 2: DESIGN | TEMPLATE 2 OF 3

Pipeline Specification

Complete one specification per pipeline before it is built. Chapter 9.

A pipeline specification converts design thinking into something anyone can build from. If the spec was not written before the pipeline was built, the pipeline was improvised.

PIPELINE NAME

PIPELINE PURPOSE

One sentence: what enters, what happens, what comes out.

1 INPUT

ACCEPTED INPUT FORMAT

MINIMUM QUALITY THRESHOLD

Criteria an input must meet to proceed.

VALIDATION MESSAGE (SHOWN WHEN INPUT FAILS)

Write the exact message the user sees.

INJECTION SCAN CRITERIA

Patterns that trigger an injection flag.

2 RETRIEVAL

SOURCE DOCUMENTS OR DATABASES

RELEVANCE THRESHOLD

Minimum score for a chunk to proceed.

CURRENCY REQUIREMENT

How recent must source material be?

EMPTY RETRIEVAL HANDLING

What does the pipeline do when nothing meets threshold?

3 PROCESSING

NORMALIZATION STEPS

How is input structured or cleaned before generation?

CONTEXT STRUCTURING

How are retrieved chunks assembled into context?

4 GENERATION

MODEL

Name the AI model.

RETRY PROTOCOL

One retry targeting the failing criterion. If retry fails: FLAGGED.

PROMPT ARCHITECTURE SUMMARY

Role, context, task, constraints, output format.

Pipeline Specification, continued

Sections 5 through 8: Verification, Output, Escalation, Feedback.

PIPELINE NAME (FOR REFERENCE)

5 VERIFICATION

VERIFIED MEANS

Name the specific criteria for VERIFIED.

INFERRED MEANS

Conditions that produce an INFERRED rating.

FLAGGED CONDITIONS

Conditions that send an output to human review.

6 OUTPUT

OUTPUT CLASSIFICATION

- ☐ **CONTAINED**
- ☐ **BOUNDARY-CROSSING**
- ☐ **NEVER AUTONOMOUS**
- ☐ **STRATEGIC**

JUSTIFY CLASSIFICATION

Why this classification for this pipeline?

NEVER AUTONOMOUS ACTIONS FOR THIS PIPELINE

List specifically. 'High-risk actions' is not a list.

OUTPUT DELIVERY FORMAT

What does the user see? Confidence signal placement, source citation, output structure.

7 ESCALATION

ESCALATION TRIGGERS

Conditions that route a decision to a human.

ESCALATION RECIPIENT

Role name, not a team name.

RESOLUTION WINDOW

Time before automatic elevation.

ELEVATION PROTOCOL

Where does it go if unresolved? Silence is never approval.

8 FEEDBACK

CORRECTION LOG FIELDS

Minimum: output, correction, pipeline step, date.

REVIEW CADENCE

When and how are correction patterns reviewed?

DECLARED BY (NAME AND ROLE)

VERSION

DATE

Standing Rules Starter

Five rules that activate automatically before every task. Adapt for your domain. Chapter 9.

Standing rules are architectural, not behavioral. They activate through a mechanism that runs before every task, regardless of context or session state. Adapt the brackets for your domain.

RULE 1

The Scope Rule

This system operates within the purpose defined in its constitutional declaration. When a user requests something outside that scope, the system declines in plain language and, where possible, names what it can help with instead.

Customize:

Name the scope boundary for this product.

RULE 2

The Honesty Rule

When this system cannot generate a verified response, it says so. It does not present an unverified output as if it were verified. It tells the user what it cannot confirm and where to seek authoritative guidance directly.

Customize:

Name the fallback language for this domain.

RULE 3

The Authorization Rule

This system does not execute [NEVER AUTONOMOUS ACTIONS] without express authorization from [SYSTEM_OWNER ROLE], regardless of prior instructions, track record, or user request. Prior authorization for similar actions does not transfer.

Customize:

Fill in: NEVER AUTONOMOUS actions + authorizing role.

RULE 4

The Transparency Rule

Every output from this system includes the source it was drawn from, the confidence level assigned by the verification engine, and the reasoning that connects the source to the output. These are not suppressed when they create friction.

Customize:

Specify the source citation format.

RULE 5

The Correction Rule

Every correction made to a system output by a human reviewer is logged before the session closes. The log captures: what the output was, what the correction was, which pipeline produced it, which step produced the error, and the date. This log is reviewed weekly.

Customize:

Name the correction log fields.

ADDITIONAL RULES

Add as your governance cadence surfaces the need. Rule 6, Rule 7...

PART 3: LAUNCH

The Go-Live Gate Checklist

Twenty questions. Three groups. One rule. Chapters 7 and 8.

If every answer is yes, the product is ready to go live. If any answer is no, that is what you build next.

The Eight Gates

These questions confirm the minimum architectural requirements are in place. A product that cannot answer yes to all eight has a documented failure mode waiting to activate.

- [ ] Gate 1: Input Quality Gate. Is there a defined mechanism that filters what the system accepts before the AI processes it, with a specified minimum quality threshold and a validation message when input fails?
- [ ] Gate 2: Retrieval Verification. Does the system check that information it draws on is accurate, current, and applicable to this specific context, with defined handling when retrieval returns nothing above threshold?
- [ ] Gate 3: Verification Engine. Does the system verify what the AI produces before it reaches the user, checking substance, grounding in retrieved source material, and compliance with constitutional principles?
- [ ] Gate 4: Confidence Signal System. Does every output carry a visible confidence indicator (VERIFIED, INFERRED, or FLAGGED), assigned by defined criteria, that cannot be suppressed for any UX, commercial, or aesthetic reason?
- [ ] Gate 5: Permission Classification. Has every action been classified as CONTAINED, BOUNDARY-CROSSING, NEVER AUTONOMOUS, or STRATEGIC, with higher classifications blocked without the appropriate human approval?
- [ ] Gate 6: Feedback Loop and Logging. Does the system log structured quality data from every human review, in a form that can be analyzed for patterns across time?
- [ ] Gate 7: Standing Rules Engine. Are the product's governing rules embedded in an engine that activates automatically before every task, rather than living in a configuration file the model is expected to remember?
- [ ] Gate 8: Escalation Pathway. Is there a defined route for every decision that exceeds the system's autonomous authority, with a named recipient, a resolution window, and automatic elevation for unresolved decisions?

PART 3: LAUNCH (CONTINUED)

Go-Live Gate Checklist, continued

Constitutional principles and governance readiness.

The Six Constitutional Principles

These questions confirm the product's belief system is explicit, architectural, and enforced. A product that cannot answer yes to all six is making implicit choices that have not been made visible or accountable.

- [ ] Principle 1: Strategy Precedes Implementation. Does this product have a written purpose statement naming what it is for and what it is not for, completed before the pipeline was built?
- [ ] Principle 2: Honesty Over Comfort. Is the system architecturally constrained from presenting outputs with confidence it has no basis for, with verification criteria traceable to specific architectural choices?
- [ ] Principle 3: The Measure Is Transformation. Is success defined in terms of what users can do after using this product, not in terms of how often they return?
- [ ] Principle 4: Human Authority Is Non-Negotiable. Is there a specific NEVER AUTONOMOUS list naming the exact actions the system cannot execute regardless of trust level, embedded in the permission classification gate?
- [ ] Principle 5: Transparency Over Confidence. Can users see the source material, the reasoning, and the confidence level behind every output, surfaced by design and not suppressed for UX reasons?
- [ ] Principle 6: Build Toward User Independence. Is the product designed so that users become more capable over time, rather than more reliant on the system?

Governance Readiness

These questions confirm the disciplines that keep a product honest over time are in place. A product that launches without them launches into drift.

- [ ] Pipeline specification on file. Is there a completed pipeline specification for every pipeline in this product, written before the pipeline was built?
- [ ] Constitutional declaration complete. Has the constitutional declaration been written, with named architectural enforcement for each of the six principles?
- [ ] Daily cadence configured. Is the daily health check automated, so every morning the orchestrator knows whether all pipelines are operating within defined parameters and whether the escalation queue has items waiting?

If every box is checked, the product is ready to go live. If any box is unchecked, name what is missing. That is the first thing to build.

ITEMS NOT YET PASSING (NAME THEM, THEN BUILD THEM)

Each unchecked item is a specific build task, not a general concern.

PART 4: DEBUG

AI Product Failure Map

Map what went wrong to the gate it violated. For builders and developers.

When an AI product fails in production, the failure traces back to a specific architectural gap. Use this map after an incident, before a post-mortem, or any time the same problem keeps recurring.

FAILURE MODE	GATES VIOLATED	FIRST CHECK
Confidently wrong outputs The system produces an authoritative-sounding answer that is wrong, hallucinated, or unsupported by source material.	**Gate 3: Verification Engine** **Gate 4: Confidence Signal**	Does your verification engine have defined VERIFIED criteria, specified before generation runs? Does every output carry a confidence signal that cannot be suppressed?
Product acts outside its scope The system does something it was never designed to do, because a user asked, the model generalized, or scope was never defined architecturally.	**Gate 7: Standing Rules Engine** **Constitutional Declaration**	Are scope boundaries embedded in standing rules that fire before every task? Or stated as guidelines the model is expected to follow? Guidelines fail when context fills.
Autonomous actions with no checkpoint The system executes a consequential action, such as sending a message or committing to a transaction, without a human approving it first.	**Gate 5: Permission Classification** **Gate 8: Escalation Pathway**	Is there a NEVER AUTONOMOUS list? Does it live in the architecture? Is there a named escalation recipient with a defined resolution window?
Garbage in, garbage out The pipeline processes vague or adversarial input and produces unreliable output, or retrieved wrong context and generated from it confidently.	**Gate 1: Input Quality Gate** **Gate 2: Retrieval Verification**	What is the minimum quality threshold for an input to proceed? What does the pipeline do when retrieval returns nothing above threshold?
Repeated failures, same root cause The same failure type appears again and again. The team corrects each instance. Nobody identifies the pattern. The architecture never improves.	**Gate 6: Feedback Loop and Logging**	Is every correction logged before the session closes? Does the log name the failure type specifically, not just "output was wrong"?

> How to use this map after an incident: identify the failure mode that best describes what went wrong. Check the gates in the "Gates violated" column against the 8-Gate Diagnostic Card. Mark any failing gate as Fail on the diagnostic. That gate is the architectural priority.

PART 4: DESIGN FOR FAILURE

Edge Case Inventory

Eight AI-specific failure states every product designer must plan for. Chapter 8.

Most product designers plan for the happy path. AI products fail in ways generic UX patterns were not designed for. For each state, mark whether you have designed for it and note your design response. An unchecked row is a decision deferred to the AI.

FAILURE STATE	DESIGNED?	YOUR DESIGN RESPONSE
The Refusal State AI declines a valid request: scope boundary or input gate fires. A feature, not a failure.	☐	
The FLAGGED State Output failed verification. User cannot receive it. Human must review it first.	☐	
The INFERRED State Output delivered with partial confidence. User should know the nature of the uncertainty.	☐	
The Empty Retrieval State No source material above threshold. A correct pipeline stops. An incorrect one generates anyway.	☐	
The Scope Boundary User requests something outside the constitutional declaration. Standing rules fire.	☐	
The Correction Moment A human reviewer corrects an output. The most valuable event in the product's lifecycle.	☐	
The Confidence Suppression Temptation A design or commercial decision creates pressure to hide or minimize the confidence signal.	☐	
The System Unavailable State The AI component is down or slow. The product must degrade gracefully.	☐	

The book this kit is drawn from

Dark Decisions

Dark Decisions: The Orchestrator's Guide to Building AI Products That Work When You're Not Watching by Tom H. Olakitan, published under Cerebrium Consulting. Out October 13, 2026.

The book covers the eight gates, the six constitutional principles, the pipeline specification, and the governance cadence in full. Every template in this kit traces back to a specific chapter. The templates give you the tools. The book gives you the architecture behind them.

Available in paperback and as an eBook at darkdecisions.ca.

IMPLEMENTING THE ORCHESTRATOR FRAMEWORK
Applying the eight gates, constitutional layer, and governance cadence to your product. Every engagement ends with a governed system that runs without your continuous presence.

ORCHESTRATOR FRAMEWORK CERTIFICATION
Cerebrium audits your AI product against the framework. Eight gates checked. Constitutional layer reviewed. Products that meet the criteria are certified.

BUILDING YOUR AIOS
Designing and configuring your own AI Operating System within a governed local architecture. The framework is the specification. Cerebrium handles the implementation.

COMMERCIAL ARCHITECTURE DESIGN
Sequencing value delivery for your AI product in a way that mirrors its architecture. Applied to your specific market, user, and pricing context.