ARC XT

A governance layer for AI-assisted coding

ARC XT — Product Manual

v0.1.13 · Clean Baseline Release · April 5, 2026

User-facing guide for installing, configuring, and operating ARC XT in VS Code.

github.com/habrahgithub/arc-extension · Apache-2.0 License
This document corresponds to ARC XT v0.1.13.

> ***Version 0.1.13*** *· Clean Baseline Release A governance layer for AI-assisted coding in VS Code*

Table of Contents

---

1. What Is ARC XT

ARC XT is a VS Code extension that introduces a **deterministic, tamper-evident audit layer** inside your IDE. It sits between code generation and version control:

```
AI Generator (Copilot / Cursor / Claude)
          ↓
   ARC XT (Audit Layer)
          ↓
   Git / PR / CI
```

When you save a file, ARC XT classifies the change by risk, makes a decision, and records the event in an append-only, hash-chained audit log — all locally, with no external dependencies required.

What it is

- A **governance layer** that enforces accountability for every code save
- A **local-first** system — no cloud services, no telemetry, no external data transmission

- A **proof system** that links high-risk changes to documented governance directives
- A **review surface** with multiple webview panels for audit inspection and decision context

What it is NOT

- A linter, formatter, or static analyzer
- A replacement for code review or CI pipelines
- A telemetry or monitoring tool
- A productivity helper or assistant

---

2. Quick Start

Install

1. Download the latest `.vsix` from the Releases page
2. In VS Code, open the Extensions view (`Ctrl+Shift+X`)
3. Click `···` → **Install from VSIX...**
4. Select the downloaded `.vsix` file
5. Reload VS Code

Verify It's Working

1. Open any workspace
2. Save a file — ARC XT activates automatically on startup
3. Check the status bar indicator for enforcement state
4. Run `ARC XT: Show Welcome Guide` for orientation

First Governed Save

Make a change to `package.json` and save. ARC XT will detect it as a high-risk configuration change and prompt you to acknowledge the risk before the save proceeds.

---

3. How It Works

The Decision Pipeline

Every time you press `Ctrl+S` :

1. **Intercept** — ARC XT hooks the save event before it reaches disk
2. **Classify** — Risk rules match the file path and change context against known patterns

3. **Decide** — A decision floor is applied: `ALLOW`, `WARN`, `REQUIRE_PLAN`, or `BLOCK`
4. **Enforce** — If the decision requires acknowledgment, ARC XT prompts the operator
5. **Log** — The event is recorded in an append-only, SHA-256 hash-chained audit log
6. **Proceed** — The save completes only after the decision is satisfied

Risk Classification Rules

Rule	Severity	Triggers	Decision
Auth/Session Changes	HIGH	Files in `auth/`, `session/` paths	Require blueprint
Config Files	HIGH	`package.json`, `tsconfig.json`, `.env`	Require blueprint
Schema Changes	MEDIUM	`schema.sql`, `.prisma` files	Risk acknowledgment
General Files	LOW	All other files	Allow

Decision Tiers

Decision	What It Means	Operator Action
ALLOW	Low-risk change	Save proceeds automatically
WARN	Medium-risk change	Acknowledge risk, then save
REQUIRE_PLAN	High-risk change	Link to a governance blueprint, then save
BLOCK	Critical violation	Save blocked until risk is resolved

4. Core Concepts

Directive

A **directive** is a governance identifier — a unique code like `ARC-101` that labels a specific change intent. Directives are the primary unit of accountability.

Blueprint

A **blueprint** is a markdown file in `.arc/blueprints/` that documents the plan, scope, constraints, and acceptance criteria for a directive. It serves as a **proof of intent** — evidence that the change was deliberate and authorized.

Route Policy

A **route policy** (`.arc/router.json`) controls how ARC XT evaluates saves. Three modes:

Mode	Behavior
`RULE_ONLY`	Local rules only (default, fail-closed)
`LOCAL_PREFERRED`	Local model evaluation, falls back to rules
`CLOUD_ASSISTED`	Cloud model evaluation with local fallback

Both lanes are **disabled by default** — ARC XT runs in `RULE_ONLY` mode out of the box.

Audit Log

Stored at `.arc/audit.jsonl` as JSON Lines. Each entry is linked to the previous via SHA-256 hash chain. **Tamper-evident**: any modification to a single entry breaks the chain.

Governed Root

The workspace root where `.arc/` lives. ARC XT detects this automatically and applies governance rules relative to it.

---

5. Commands & Navigation

All commands are accessible via the Command Palette (`Ctrl+Shift+P`):

Core Commands

Command	Description
`ARC XT: Show Welcome Guide`	First-use orientation and onboarding
`ARC XT: Task Board`	Milestone pipeline — see active, queued, and completed tasks
`ARC XT: Liquid Shell`	**Preview** — Full governance shell with Runtime, Review, Architect, and Tasks views
`ARC XT: Review Audit Log`	Browse the tamper-evident audit trail
`ARC XT: Review Blueprint Proofs`	Check which blueprints satisfy which saves
`ARC XT: Review False-Positive Candidates`	Review changes that were flagged but may be safe
`ARC XT: Guided Proof Workflow`	Step-by-step wizard for creating and linking blueprints

Review Surfaces

Command	Description
`ARC XT: Review Home`	Entry point for all review surfaces
`ARC XT: Decision Feed`	Chronological stream of all save decisions
`ARC XT: Audit Timeline`	Visual timeline of audit events
`ARC XT: Why Panel`	Detailed explanation of why a specific decision was made
`ARC XT: Runtime Status`	Current workspace governance state

Task Management

Command	Description
`ARC XT: Select Active Task`	Set the current working directive
`ARC XT: Clear Active Task`	Remove the active directive context

Configuration

Command	Description
`ARC XT: Create Minimal Route Policy`	Generate `.arc/router.json` with safe defaults
`ARC XT: Create Minimal Workspace Mapping`	Generate `.arc/workspace-map.json`

Legacy Compatibility

Command	Description
`Lintel: Show Welcome`	→ `ARC XT: Show Welcome Guide` (deprecated)
`Lintel: Review Audit`	→ `ARC XT: Review Audit Log` (deprecated)

Legacy commands are retained for backward compatibility with existing keybindings.

---

6. The Save Flow

Normal Save (ALLOW)

1. You press `Ctrl+S`
2. ARC XT classifies the file — no rules match or all are LOW risk
3. Save proceeds normally
4. Event logged to `.arc/audit.jsonl`

Risk-Acknowledged Save (WARN)

1. You press `Ctrl+S`
2. ARC XT detects a medium-risk pattern (e.g., config file)
3. A dialog appears asking you to acknowledge the risk
4. You confirm → save proceeds
5. Event logged with `WARN` decision

Blueprint-Linked Save (REQUIRE_PLAN)

1. You press `Ctrl+S`
2. ARC XT detects a high-risk pattern (e.g., auth module change)
3. A dialog prompts you to enter a **Change ID** (e.g., `ARC-101`)
4. ARC XT validates that a corresponding blueprint exists at `.arc/blueprints/ARC-101.md`
5. If valid → save proceeds; if invalid → save blocked
6. Event logged with `directive_id` and `blueprint_id`

Blocked Save (BLOCK)

1. You press `Ctrl+S`
2. ARC XT detects a critical violation
3. Save is blocked with an error message
4. You must resolve the violation before the save can proceed

---

7. Blueprint Proofs

What Is a Blueprint?

A blueprint is a **governance artifact** — a markdown file that documents the rationale, scope, and constraints for a specific change. It serves as **proof of intent**.

Blueprint Format

```
# ARC XT Blueprint: ARC-101
**Directive ID:** ARC-101

> Status: COMPLETE

## Objective
Describe the specific change intent.

## Scope
List the files or surfaces this directive covers.

## Constraints
Record risk bounds and governance constraints.

## Acceptance Criteria
Define how this change will be validated.

## Rollback Note
Describe how to revert if needed.
```

Creating a Blueprint

1. Create `.arc/blueprints/ARC-101.md`
2. Fill in all sections — **no placeholders allowed**
3. The blueprint must be complete before it can authorize saves

Linking a Blueprint to a Save

When prompted during a governed save, enter the directive ID. ARC XT will:

1. Verify the blueprint file exists
2. Validate the directive ID format
3. Confirm the blueprint is complete (no `[REQUIRED]` placeholders)
4. If valid → allow the save; if invalid → block

---

8. Review Surfaces

Decision Feed

A chronological stream of every save decision. Each entry shows:

- Timestamp
- File path
- Decision tier (ALLOW / WARN / REQUIRE_PLAN / BLOCK)
- Risk level
- Matched rules

Audit Timeline

A visual timeline of audit events with filtering by risk level, decision, and time range.

Why Panel

When you need to understand **why** a specific decision was made, the Why Panel shows:

- Which rules matched
- Why those rules fired
- What the risk assessment was
- What the next action should be

False-Positive Review

Review changes that were flagged by risk rules but may actually be safe. You can review and mark these for future reference.

---

9. Liquid Shell (Preview)

The Liquid Shell is a **unified governance interface** that replaces the scattered panel approach with a single, coherent workspace.

Views

View	Primary Question	Hero Surface
Runtime	What is happening now?	Posture banner (safe/degraded/blocked)
Tasks	What is progressing?	Phase-grouped task pipeline
Review	What requires judgment?	Structured deviation rows
Architect	What defines the shape?	System topology map

Navigation

Three synchronized navigation surfaces:

- **Icon Rail** (64px) — Muscle-memory navigation
- **Sidebar** (260px) — Contextual control, active directive, EXECUTE_RUN
- **Top Bar** — Route tabs and utility icons

EXECUTE_RUN

A ceremonial execution control that reflects system readiness:

- **Dormant** — No active directive
- **Ready** — Directive is live, system is safe (subtle glow)
- **Executing** — Locked with pulse animation
- **Warning** — Amber tone when risk detected

Access: `ARC XT: Liquid Shell`

---

10. Configuration

Out-of-the-Box Defaults

ARC XT works immediately after installation with no configuration:

Setting	Default	Meaning
Route Mode	`RULE_ONLY`	Local rules only
Local Lane	Disabled	No model evaluation
Cloud Lane	Disabled	No external data transmission
Data Class	`LOCAL_ONLY`	All data stays local
Governance Mode	`ENFORCE`	Rules are enforced, not observed

Optional: Route Policy

Create a route policy to enable model evaluation:

```
# Via command palette:
ARC XT: Create Minimal Route Policy
```

This generates `.arc/router.json`:

```
{
  "mode": "RULE_ONLY",
  "local_lane_enabled": false,
  "cloud_lane_enabled": false,
  "cloud_data_class": "LOCAL_ONLY"
}
```

Optional: Workspace Mapping

```
# Via command palette:
ARC XT: Create Minimal Workspace Mapping
```

This generates `.arc/workspace-map.json`:

```
{
  "mode": "LOCAL_ONLY",
  "rules": [],
  "ui_segments": []
}
```

Adding Custom Rules

Edit `rules/default.json` to add or modify risk classification rules. Each rule has:

```
{
  "id": "rule-custom",
  "risk_flag": "CUSTOM_CHANGE",
  "severity": "MEDIUM",
  "decision_floor": "WARN",
  "patterns": {
    "file_names": ["custom.config"],
    "extensions": [".custom"],
    "path_segments": ["custom/"]
  }
}
```

11. Troubleshooting

"No workspace folder open"

ARC XT requires an open VS Code workspace. Open a folder or workspace first.

"Change ID is not valid"

The directive ID must be uppercase, hyphenated, and match the format `ARC-XXX`. Example: `ARC-101`, `ARCXT-UI-001`.

Blueprint validation failed

Ensure your blueprint:

- Has no `[REQUIRED]` placeholder text
- Contains a valid `Directive ID` matching the file name
- Has all sections filled in (Objective, Scope, Constraints, Acceptance Criteria, Rollback Note)

Save is blocked

Check the error message for the specific rule that triggered the block. Common causes:

- Missing blueprint for a high-risk file
- Invalid directive ID format
- Blueprint file does not exist

Verify Audit Log Integrity

```
cd your-workspace
npx arc-audit verify .arc/audit.jsonl
```

If the hash chain is intact, all entries are verified. If broken, the log has been tampered with.

Extension Not Activating

1. Check VS Code version — requires **1.90.0+**
2. Check the Output panel (`View → Output`) and select "ARC XT" from the dropdown
3. Try `ARC XT: Show Welcome Guide` — if the command is not found, the extension failed to activate

12. Building From Source

```
git clone https://github.com/habrahgithub/arc-extension.git
cd arc-extension
npm install
npm run build        # Compile TypeScript
npm run test         # Run test suite
npm run pack         # Create .vsix
```

The VSIX will be created in the project root directory.

Build Commands

Command	Description
`npm run build`	Compile TypeScript to `dist/`
`npm run typecheck`	Type-check without emitting
`npm run test`	Run all tests
`npm run lint`	ESLint check
`npm run pack`	Create VSIX package

13. Support

- **Repository:** github.com/habrahgithub/arc-extension
- **Issues:** Report a bug
- **License:** Apache-2.0

This manual corresponds to ARC XT v0.1.13. For changes between versions, see the CHANGELOG.