

PackML in the AI Era

How standards-aware AI can accelerate adoption, compliance, engineering, and troubleshooting

A PLCs.ai White Paper

PLCs.ai | October 2026

Table of Contents

Click any section title to jump directly to that section.

Executive Summary	3
1. Why PackML Matters to Industrial AI	3
2. The PackML Lifecycle in the AI Era	3
3. Automatic Recognition: Making the Standard Part of the Context	4
4. From Recognition to Measurable Compliance	4
5. Turning Findings into Action	5
6. Plain-Language Access to PackML Expertise	6
7. Prioritized Remediation, Not a Flat Checklist	7
8. Human-Guided AI for Engineering Changes	8
9. Generating and Modifying Code in Accordance with PackML	9
10. From Analysis to Organizational Knowledge	11
11. From Engineering Standard to Production Troubleshooting	14
12. Why This Matters for the PackML Community	14
13. A Virtuous Cycle: PackML Makes AI Better, AI Makes PackML Easier to Use	15
Conclusion	15
About PLCs.ai	16
References and Notes	16

Executive Summary

PackML was created to give automated machinery a common operational model and consistent behavior. OMAC describes the standard as a way to make it easier to transfer and retrieve consistent machine data, with benefits that include operational consistency, reusable training, more robust software, and easier troubleshooting.

The arrival of industrial AI creates a new opportunity: not to replace PackML, but to make the standard easier to adopt and more valuable throughout the machine lifecycle. A standards-aware AI system can recognize PackML in an existing PLC project, evaluate how well the implementation follows the standard, explain gaps in plain language, recommend a remediation sequence, and help engineers generate or modify code while preserving the standard as an active engineering constraint.

The same understanding can continue into production. When AI can reason over PackML structure, PLC logic, machine relationships, and live runtime values, a technician or engineer can ask what is preventing a machine or line from progressing and receive a root-cause-oriented answer grounded in both the program and the current operating state.

Industry validation: PLCs.ai's PackML capabilities have been tested in collaboration with a leading U.S. OMAC member, providing real-world validation of the platform's ability to understand and work with PackML-based automation projects.

PackML gives AI standardized machine context. AI can make PackML easier to adopt, implement, maintain, troubleshoot, and use.

1. Why PackML Matters to Industrial AI

AI is most useful in automation when it has context. Generic language capability is not enough to reliably reason about a machine. The system needs to understand how the machine is structured, how it is expected to behave, what states and commands mean, and how engineering decisions are represented in the controller.

PackML is therefore more than a compliance target for AI. It is a structured vocabulary and behavioral framework that can make machine logic more interpretable. A consistent state model, PackTags, modular structures, and common conventions provide the AI with signals it can use to understand intent, compare implementation with expectation, and communicate findings to people.

This creates a reciprocal relationship. PackML improves AI by supplying standardized industrial context, while AI can reduce the practical friction involved in adopting and sustaining PackML.

2. The PackML Lifecycle in the AI Era

The opportunity is best viewed across the lifecycle rather than as a single feature. Standards-aware AI can assist from the first encounter with a PLC project through engineering, commissioning, maintenance, and production troubleshooting.

- **Adopt:** recognize where PackML applies and help an engineering team move an existing project toward a PackML architecture.
- **Comply:** assess implementation against the standard, score key areas, and expose specific gaps.
- **Engineer:** generate and modify PLC code with PackML retained as part of the project's active context.

- Maintain: explain the implementation in plain language and help teams understand how the machine was structured.
- Troubleshoot: reason across PackML states, PLC logic, dependencies, and, when connected, live runtime values.

3. Automatic Recognition: Making the Standard Part of the Context

When a new PLC project is uploaded to PLCs.ai, the platform analyzes the project for applicable industry and standards context. A PackML-structured project can be identified automatically, allowing the platform to treat PackML as part of the project's operating context for subsequent analysis and interaction.

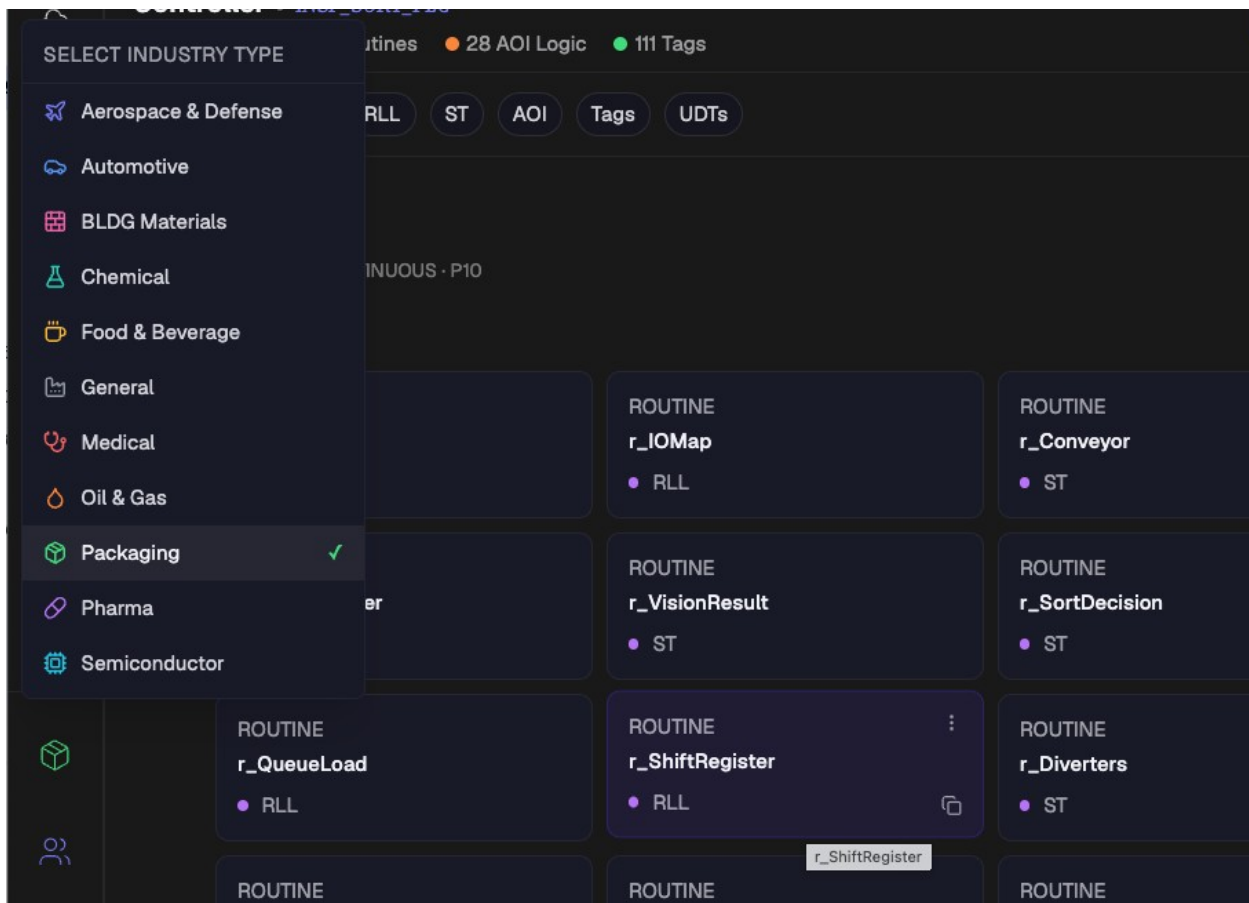


Figure 1. Project classification and standards context. PackML can be identified as part of the uploaded project's automation context.

This distinction matters. A general-purpose AI may recognize the term PackML when it appears in code. A standards-aware industrial AI should instead understand that the project is expected to follow a particular machine model and use that knowledge when interpreting, explaining, reviewing, or changing the program.

4. From Recognition to Measurable Compliance

Recognition alone does not tell an engineering team whether a machine is well implemented. PLCs.ai performs an automated compliance analysis that evaluates the project across multiple PackML-related dimensions and produces an overall score together with category-level findings.

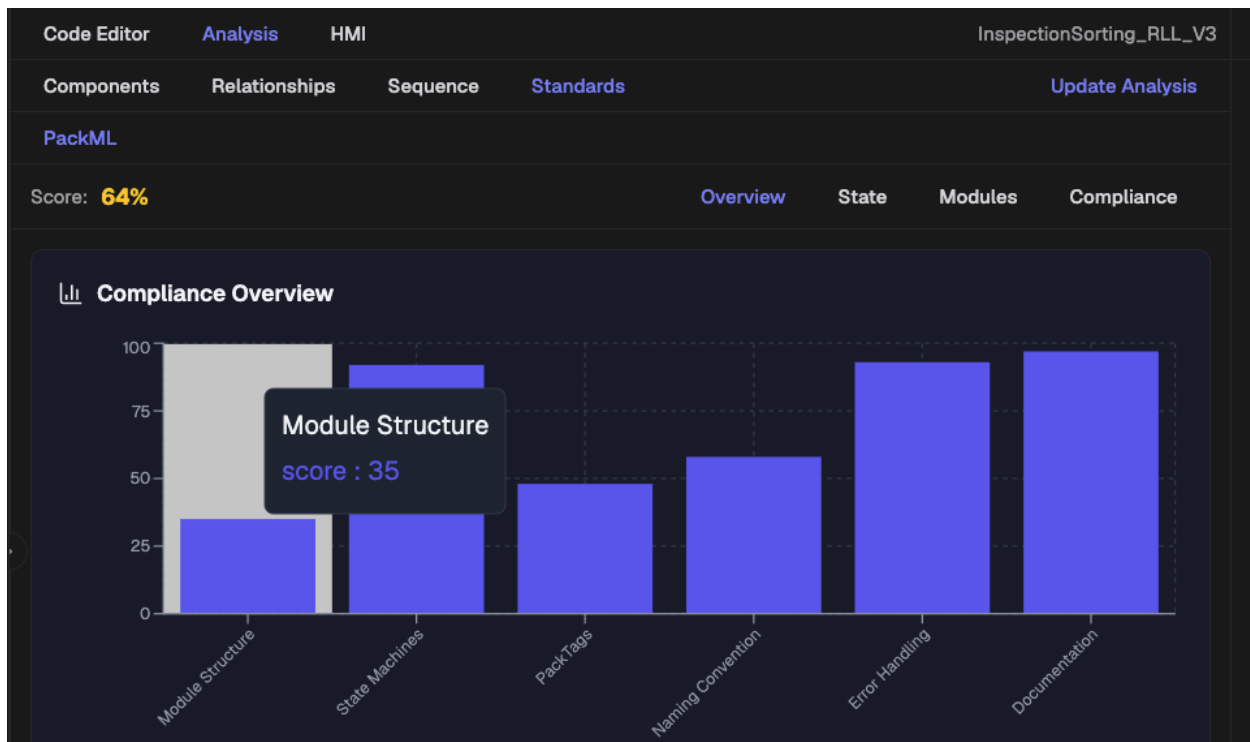


Figure 2. Automated PackML compliance assessment. The example project receives an overall score and category-level ratings, exposing weak areas such as module structure.

A score is useful because it creates a baseline, but the objective is not the number itself. The objective is to turn the standard into an engineering improvement process. A project can be partially aligned, strongly aligned in some areas, and weak in others. AI can make those differences visible without requiring an engineer to manually inspect every routine, tag, and structure against a checklist.

5. Turning Findings into Action

The next step is remediation. For each finding, the AI can explain what it observed in the actual PLC project and recommend a project-specific change. In the example below, the analysis identifies weaknesses in module hierarchy and PackTags, then describes how the project could be restructured or extended to improve alignment.

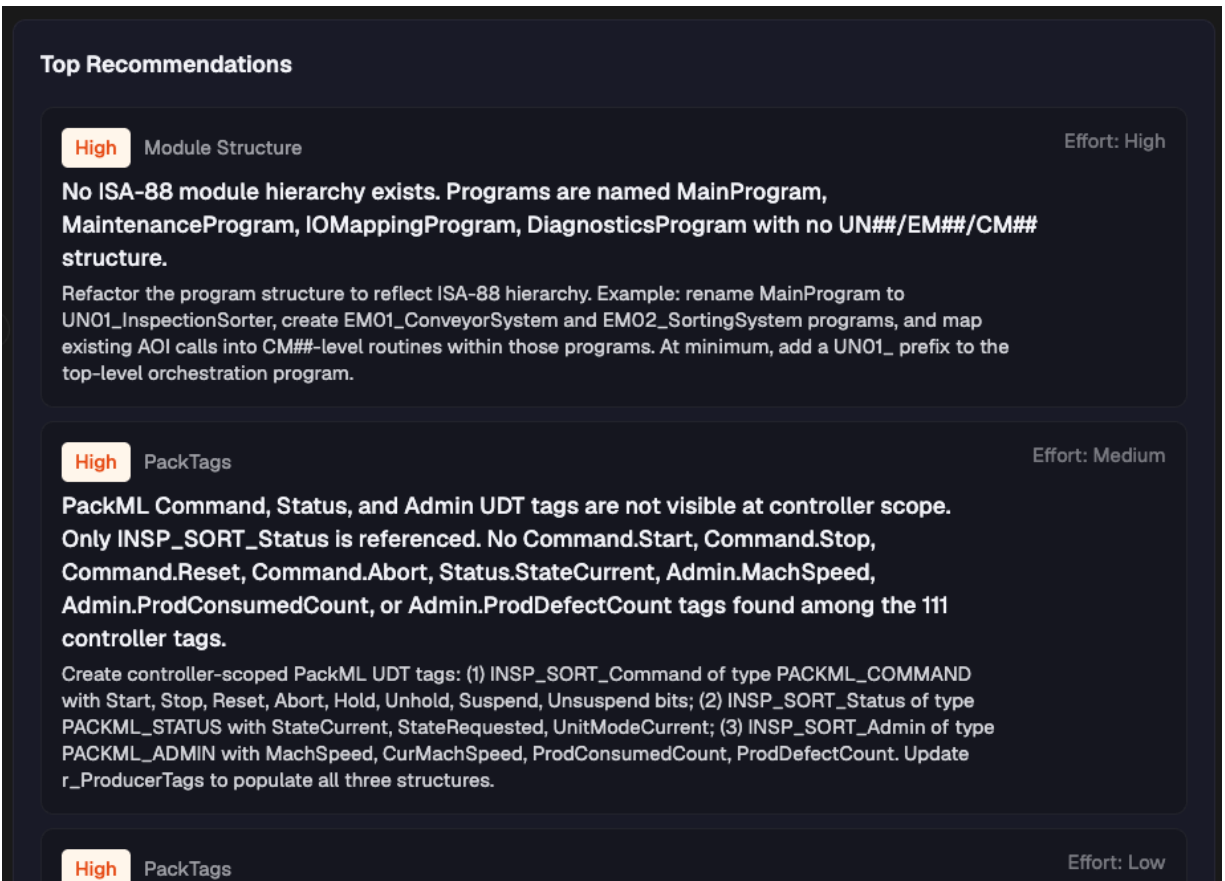


Figure 3. Project-specific recommendations translate compliance findings into concrete engineering actions.

This is a significant shift from static standards documentation. The engineer is no longer moving repeatedly between the standard, a PLC IDE, project documentation, and personal knowledge. The standard can become an active part of the engineering workflow.

6. Plain-Language Access to PackML Expertise

Standards create the greatest value when more people can use them correctly. In a conventional workflow, understanding whether a project is truly PackML-aligned may require knowledge of the standard and detailed familiarity with the PLC program. A conversational interface can make that expertise accessible to a broader engineering and maintenance audience.

An engineer can ask a simple question such as "Is this a PackML project?" The AI can respond with evidence from the project, distinguish between a complete implementation and a partial framework, and identify the gaps that prevent stronger compliance.

Project selector + 🕒 ⬇ ✕

is this a packml project?

Yes — this is a PackML-structured project, though it's a partial/framework implementation rather than a fully wired-up PackML machine.

✓ Clear PackML Evidence

Element	What's Present
AOI	PackML_AOI exists in the AOI library
UDTs	UDT_PackML_Admin , UDT_PackML_Status , UDT_PackML_Command , UDT_PackML_Alarm — directly mirror ISA-TR88.00.02
State tag	Sys (UDT_SystemState) carries LineState and ActiveMode
Alarm management	Alarm_AOI with ack/reset/latch semantics, PackML-compatible
Inter-machine comms	r_ProducerTags publishes INSP_SORT_Status ; consumes MAT_PREP_Status and INJ_MOLD_Status

Ask me about your project. Use / for the skills menu, @ for element names

⚠ Gaps Worth Noting

- [PackML_AOI](#) is defined but never instantiated — no controller-scope tag of type [PackML_AOI](#) exists, so the state machine isn't actively driven by it
- [Sys.ActiveMode](#) is referenced throughout (modes 1 = Auto, 2/3 = Manual) but the PackML state machine transitions (Starting → Execute → Completing, etc.) are not explicitly coded in any routine
- No [UnitMode](#) or [MachSpeed](#) PackTags are wired to HMI

🔍 Bottom Line

The PackML *data model* is fully in place (UDTs, AOI, alarm structure, producer/consumer tags). The PackML *state machine execution* is partially implemented — the framework is there but [PackML_AOI](#) isn't instantiated and state transitions aren't explicitly managed. It's best described as PackML-ready / PackML-influenced, not fully PackML-compliant.

Response generated in 41s

👍 🗨 ⬆

Figure 4. A plain-language PackML question produces an implementation-aware answer, including evidence, gaps, and an assessment of how completely the standard is wired into the machine.

7. Prioritized Remediation, Not a Flat Checklist

Real projects rarely contain a single isolated issue. Some findings are prerequisites for others, and fixing them in the wrong order can create unnecessary work. AI can organize detected gaps by severity and dependency, show where each issue appears, and propose a logical sequence for remediation.

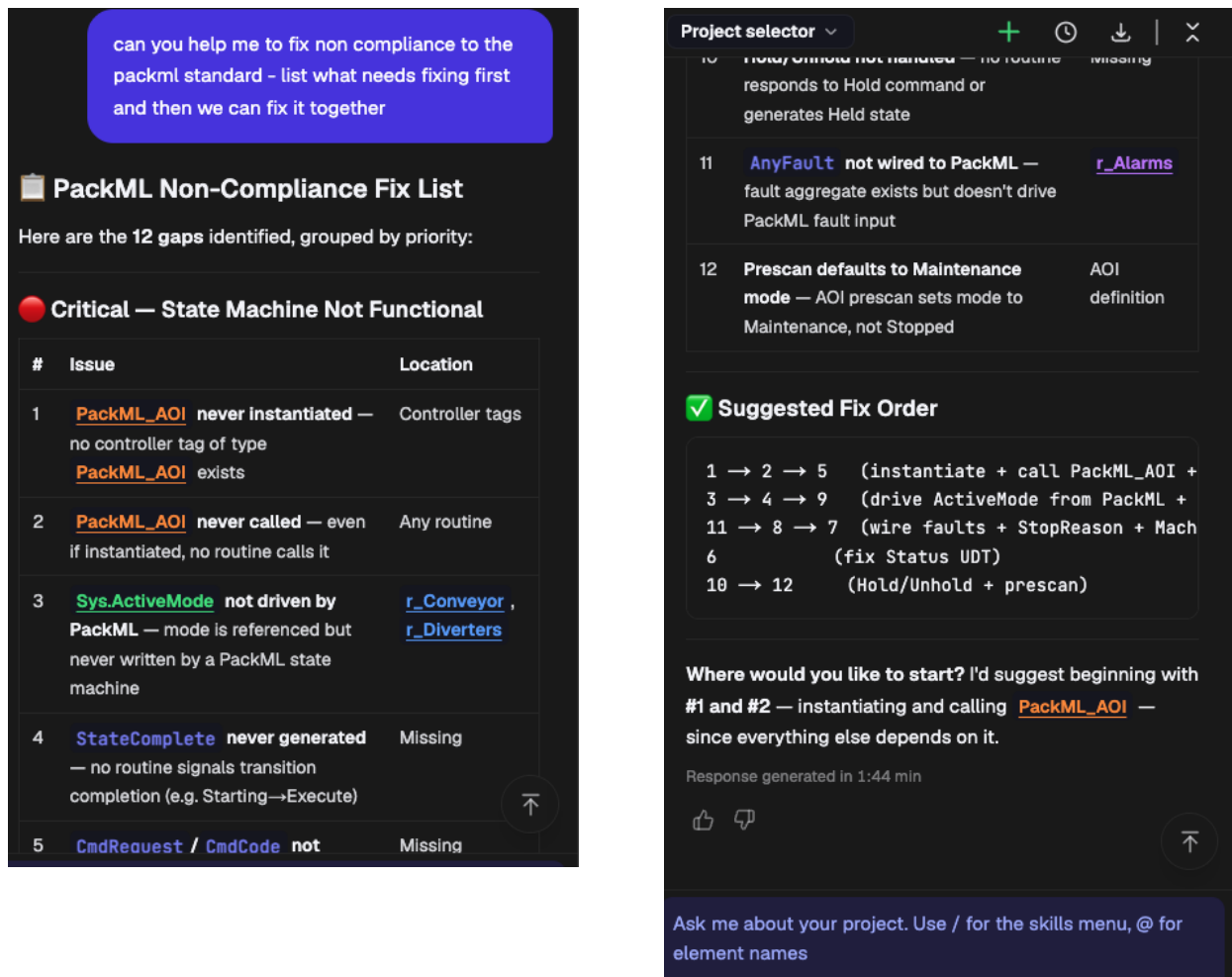


Figure 5. Non-compliance findings are grouped and prioritized, with a suggested fix order based on engineering dependencies.

8. Human-Guided AI for Engineering Changes

A critical requirement for AI in controls engineering is knowing when not to guess. If the available project data does not fully expose a required interface or implementation detail, the system should surface that uncertainty before proposing a change.

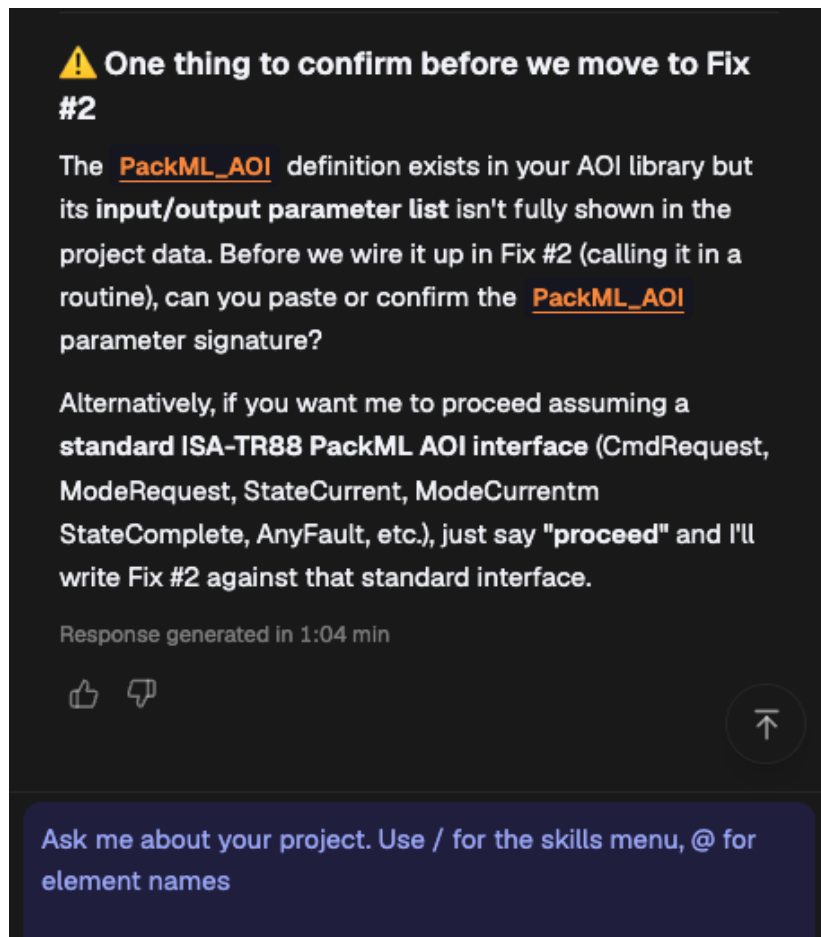


Figure 6. Before continuing a PackML remediation step, the AI identifies missing project context and asks the engineer to confirm the interface or explicitly authorize a standards-based assumption.

This interaction illustrates a practical model for human-AI engineering: the AI carries the project and standards context, explains the next step, and involves the engineer when additional information or judgment is required. The objective is not autonomous modification at any cost. It is to make standards-based engineering faster, more understandable, and more controlled.

9. Generating and Modifying Code in Accordance with PackML

Once PackML is part of the project's active context, AI can help engineers move beyond analysis and into controlled code modification. The important distinction is that understanding a project and changing it are different privileges and different workflows.

Explain and Generate: Separate Modes for Separate Responsibilities

PLCs.ai separates interaction into two modes. Explain is a read-only mode for understanding PLC logic, asking PackML questions, analyzing compliance, and troubleshooting without permitting code changes. Generate is the controlled mode used when an engineer wants the platform to propose and apply modifications to the project.

Access to Generate is role-based. Only users whose assigned permissions allow code generation can initiate a modification workflow. This allows an organization to make PackML knowledge broadly accessible while restricting actual engineering changes to authorized personnel.

what is the most critical finding we need to fix now

Most Critical Finding — Fix Immediately

Mode-change guard is incomplete in [PackML_AOI](#)

The Problem

Inside the `ModeRequest` branch (section 3 of the AOI logic), `InWaitState` is checked using only these states:

```
Stopped (2) | Idle (4) | Aborted (9) | Complete (17)
```

Held (11) and Suspended (5) are missing — both are valid PackML wait states per ISA-TR88.00.02.

What Goes Wrong

When the machine is in **Held** or **Suspended**, an operator mode-change request is silently rejected instead of being accepted. This means:

- Operators can be **locked out** of a valid mode transition during a hold or suspension event
- The machine must be fully stopped before a mode change is possible — violating the standard

The Fix

In [PackML_AOI](#), update the `InWaitState` check inside the `ModeRequest` branch to include all six wait states:

```
InWaitState := (StateCurrent = 2) (* Stopped *)
OR (StateCurrent = 4) (* Idle *)
OR (StateCurrent = 5) (* Suspended *) (* ← ADD THIS *)
OR (StateCurrent = 9) (* Aborted *)
OR (StateCurrent = 11) (* Held *) (* ← ADD THIS *)
OR (StateCurrent = 17); (* Complete *)
```

Note: The `InWaitState` output at the bottom of the AOI (section 8) already correctly includes states 5 and 11 — only the mode-change guard in section 3 is missing them. It's a one-line fix with high operational impact.

READ: 14 LOCATIONS
Response generated in 1:54 min

Ask me about your project. Use / for the skills menu, @ for element names

Explain

can you fix this for me?

Here is the exact change to make in the [PackML_AOI](#) Logic body, section 3:

Find (current code)

```
(* == 3. PROCESS UNIT MODE CHANGE REQUEST == *)
(* Mode changes only allowed in wait states (Stopped, Idle, Aborted, Cc
IF ModeRequest AND NOT LastModeRequest THEN
    InWaitState := (StateCurrent = 2) OR (StateCurrent = 4) OR (StateC
    IF InWaitState AND ModeCode ≥ 1 AND ModeCode ≤ 4 THEN
        UnitModeCurrent := ModeCode;
    END_IF;
    LastModeRequest := ModeRequest;
```

Replace with (fixed code)

```
(* == 3. PROCESS UNIT MODE CHANGE REQUEST == *)
(* Mode changes only allowed in wait states (Stopped, Idle, Held, Suspe
IF ModeRequest AND NOT LastModeRequest THEN
    InWaitState := (StateCurrent = 2) (* Stopped *)
OR (StateCurrent = 4) (* Idle *)
OR (StateCurrent = 5) (* Suspended *)
OR (StateCurrent = 9) (* Aborted *)
OR (StateCurrent = 11) (* Held *)
OR (StateCurrent = 17); (* Complete *)
    IF InWaitState AND ModeCode ≥ 1 AND ModeCode ≤ 4 THEN
        UnitModeCurrent := ModeCode;
    END_IF;
    LastModeRequest := ModeRequest;
```

Change summary

	Before	After
Wait states checked	2, 4, 9, 17	2, 4, 5, 9, 11, 17
States added	—	Suspended (5), Held (11)

Ask me about your project. Use / for the skills menu, @ for element names

Explain

Figure 7. Explain mode can identify and describe a PackML issue in context. When an authorized user moves to Generate, the platform can propose the exact standards-aligned code correction.

A Human-Approved Change Plan

The workflow is deliberately not prompt-to-code. The platform first creates a proposed change plan describing what will be changed, where the change will be made, why the approach is appropriate, and any assumptions involved. The authorized engineer reviews that plan before anything is modified.

The engineer can Approve, Amend, or Dismiss the proposed plan. Only after approval does PLCs.ai apply the change to the project. The result is then accompanied by a clear explanation of what changed, how the revised logic works, why the change addresses the PackML finding, and what should be considered during commissioning.

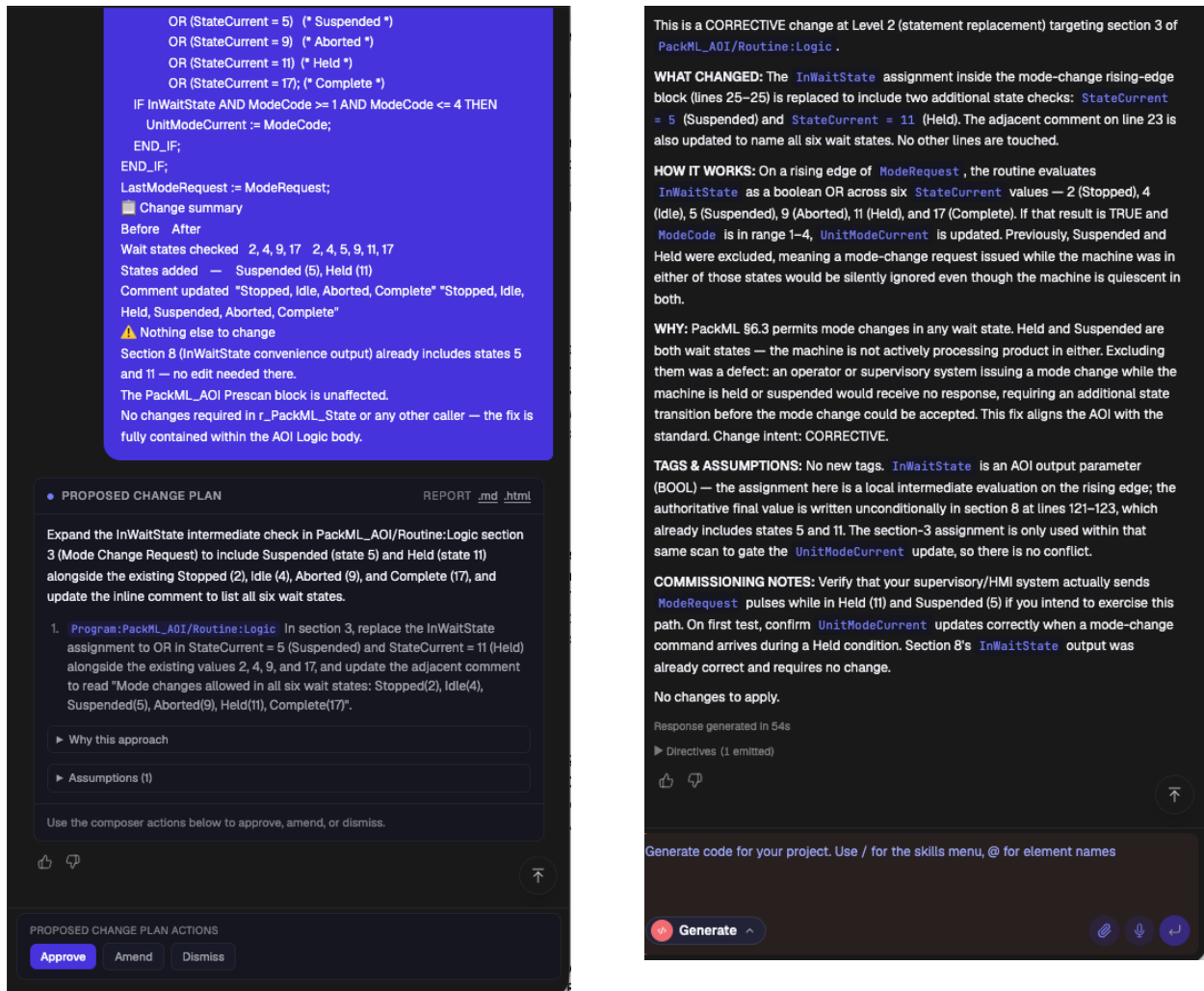


Figure 8. The Generate workflow presents a proposed change plan for review, with Approve, Amend, and Dismiss controls. After approval, the platform applies the change and documents exactly what changed and why.

AI-Assisted Engineering, Separated from PLC Deployment

PLCs.ai does not write changes directly to the running PLC. The platform modifies and saves the engineering project file. The user then downloads the new version and uses the relevant PLC vendor engineering environment to review, commission, and upload the project to the controller.

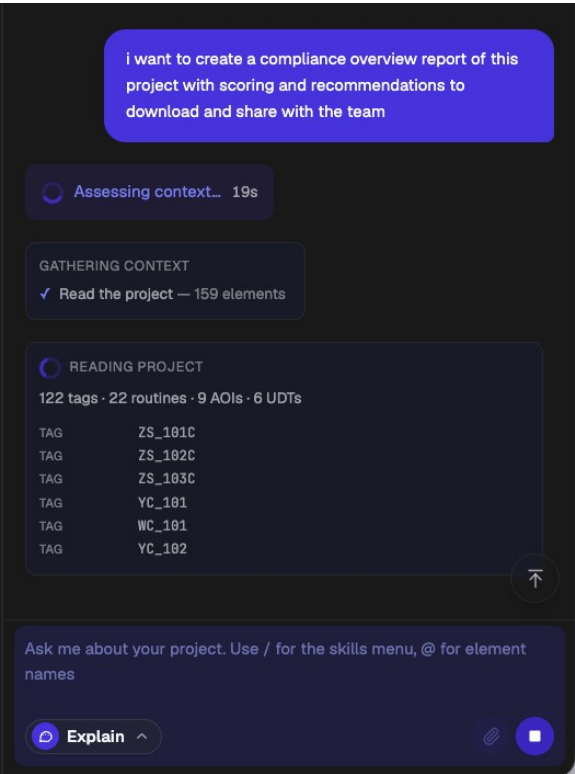
This separation preserves the organization's established PLC deployment and commissioning process while still automating much of the analysis and engineering work that precedes deployment. Allen-Bradley and Siemens are currently fully supported for this code-generation and project-modification workflow.

Analyze comprehensively. Change with permission. Review and approve before modification. Deploy through the established PLC engineering workflow.

10. From Analysis to Organizational Knowledge

PackML compliance should not live only in the head of the engineer who performed the review, or disappear when an individual analysis ends. The findings, scores, recommendations, and improvement status need to be documented in a form that can be shared across engineering teams and management.

PLCs.ai can turn the same project-aware analysis into a structured compliance report. An engineer can request the report in plain language, and the platform reads the underlying project before generating a downloadable artifact that records project scope, standards context, category-level scoring, evidence, gaps, and prioritized recommendations. The completed report can then be downloaded or shared directly by email, making it easy to circulate findings across engineering teams, management, OEMs, and integration partners.



A plain-language request triggers project-wide analysis and report generation.

mat-prep-packml-compliance-report.md
14.9 KB
Download
Share

MAT_PREP_PLC — PackML Compliance Overview Report

Project: MAT_PREP_PLC (ST 01 — MaterialPrep)
Controller: Allen-Bradley CompactLogix 5380 (5069-L306ER)
Standard: ISA-TR88.00.02 (OMAC PackML v3.0) + ISA-18.2 (Alarm Management)
Audit type: Static code analysis — offline project file
Scope: All 4 programs, 23 routines, 9 AOIs, 10 UDTs, 146 controller tags

Disclaimer: Evidence found means the project contains supporting code, tags, or structure. It does NOT constitute certification or validation. Procedural controls, IQ/OQ/PQ, hazard analyses, and real-world runtime behaviour cannot be established by static analysis.

Summary

Area	Score	Status
PackML State Machine	9/10	✓ Strong
PackML Modes	8/10	✓ Strong
PackML PackTags	8/10	✓ Strong
Alarm Management (ISA-18.2)	7/10	⚠ Gaps
OEE / KPI Metrics (ISO 22400)	6/10	⚠ Gaps
Recipe Management	9/10	✓ Strong
Safety & Interlocks	8/10	✓ Strong
Maintenance Mode	8/10	✓ Strong
Diagnostics & Comms Watchdog	7/10	⚠ Gaps
I/O Abstraction	10/10	✓ Strong
Overall	80/100	⚠ Good — address gaps before release

Section 1 — PackML State Machine

Requirement: ISA-TR88.00.02 §6 — 17-state machine with correct transitions, auto-progression, and command handling.

Generated by PLCs.ai

The report continues beyond the visible summary into detailed requirement areas, evidence, gaps, and recommendations.

mat-prep-packml-compliance-report.md
14.9 KB
Download
Share
Email

MAT_PREP_PLC — PackML Compliance Overview Report

Project: MAT_PREP_PLC (ST 01 — MaterialPrep)
Controller: Allen-Bradley CompactLogix 5380 (5069-L306ER)
Standard: ISA-TR88.00.02 (OMAC PackML v3.0) + ISA-18.2 (Alarm Management)

Figure 9. From a plain-language request to a persistent, shareable PackML compliance report. The report captures project scope, standards context, scoring and status, then continues into detailed findings and recommendations. Built-in calls to action allow the completed report to be downloaded or shared directly by email.

This creates an organizational layer around PackML. Reports can preserve what was found, what requires improvement, and the status of the implementation over time. They can support handoffs between OEMs, integrators, plant engineering, and maintenance, while reducing dependence on individual memory and making technical findings more accessible to engineering leadership and management.

For organizations operating many machines or sites, this provides the foundation for a living repository of automation knowledge: a consistent record of PackML adoption, compliance posture, engineering decisions, open improvement points, and remediation progress.

11. From Engineering Standard to Production Troubleshooting

The value of PackML does not end at commissioning. OMAC explicitly identifies easier troubleshooting and reduced mean time to repair among the benefits of PackML. AI can extend that advantage by making the PackML model conversational during production.

A maintenance engineer can ask why a machine is not advancing, what is holding a state transition, which condition is preventing execution, or whether a fault on another machine is contributing to the problem. The AI can reason through the PackML structure and the actual PLC logic instead of requiring the user to manually trace the program.

This becomes more powerful when the platform is connected to a live runtime stream. The AI can combine three forms of context: what PackML says the machine should do, how this particular PLC project implements that behavior, and what the controller values indicate is happening now.

From a PackML Machine to the Entire Production Line

Production problems rarely respect controller boundaries. A machine may be stopped, starved, blocked, held, or unable to transition because of a condition originating elsewhere on the line.

PLCs.ai can extend PackML-aware troubleshooting beyond an individual controller by bringing multiple PLC projects into a shared line context. This allows the AI to reason across machine relationships, PackML states, interlocks, handshakes, faults, and dependencies when investigating a production problem.

An engineer can therefore start with the symptom, rather than first determining which PLC contains the root cause. A question such as “Why is the packaging line not running?” can be investigated across the relevant machines and controllers, with the AI tracing the conditions and dependencies that lead to the problem.

When combined with live runtime streams from the line, this becomes more powerful: PLCs.ai can correlate the PackML model and PLC logic across the line with the values and states occurring in production at that moment, helping engineers move more quickly from symptom to root cause.

The engineer troubleshoots the line, not the PLC.

Standard + implementation + live runtime = a new troubleshooting interface for PackML-enabled production.

12. Why This Matters for the PackML Community

OMAC's mission is centered on collaborative standards and support that simplify automation, save time and resources, and create room for innovation. Standards-aware AI is aligned with that objective because its value increases when the underlying automation is structured and interoperable.

For end users, AI can lower the effort required to assess consistency across machines, understand inherited code, and troubleshoot production. For OEMs, it can support repeatable implementation and reduce the effort required to explain and maintain standards-based machine logic. For system integrators,

it can accelerate assessment and remediation of heterogeneous projects. For technology providers, PackML offers a common machine model around which new AI-assisted workflows can be built.

The broader implication is that AI may become an adoption accelerator for industrial standards. The easier a standard is to understand, implement, validate, and use in daily operations, the lower the barrier to applying it consistently.

13. A Virtuous Cycle: PackML Makes AI Better, AI Makes PackML Easier to Use

Industrial AI needs context to move beyond generic answers. PackML provides exactly the kind of structured machine context that makes reasoning more useful: common states, behaviors, tags, and architectural expectations. At the same time, AI can reduce the specialist effort traditionally required to interpret and apply those structures.

That creates a virtuous cycle. More standardized machines provide better context for AI. Better AI tools make standards easier to adopt and sustain. As both mature, the standard can become more than a design-time reference. It can become a living source of context used throughout engineering and operations.

Conclusion

PackML already provides the manufacturing industry with a common machine model and consistent behavior. The next opportunity is to make that model directly usable by AI and directly accessible to the people responsible for machines.

The emerging workflow is straightforward: automatically recognize PackML, measure implementation quality, explain gaps, prioritize remediation, help engineers make standards-aligned changes, and carry that same understanding into production troubleshooting. With live runtime context, the AI can connect the intended machine behavior to what is actually happening on the line.

The result is not AI replacing PackML expertise. It is AI helping the industry apply that expertise more consistently, across more machines and more people. For a standard whose purpose is consistency and interoperability, that may be one of the most consequential ways AI can contribute.

About PLCs.ai

PLCs.ai is an AI platform built around the PLC code that controls industrial production. The platform is designed to help engineers understand, troubleshoot, and modify automation code through natural-language interaction while preserving the engineering and standards context of the project. Its PackML capabilities include automatic recognition, compliance analysis and scoring, project-specific recommendations, guided remediation, standards-aware code modification, production troubleshooting, and integration with live runtime values.

References and Notes

- OMAC, PackML: the automation standard developed by OMAC and adopted by ISA as TR88.00.02. OMAC describes PackML as enabling consistent machine data, behavior, and a common look and feel.
- OMAC Packaging Workgroup: PackML is positioned as supporting interoperability, easier integration, faster development, and improved troubleshooting.
- This paper was prepared by PLCs.ai and shared with the OMAC community in October 2026. It offers a practical perspective on how AI can help manufacturers adopt PackML, maintain alignment with the standard, improve existing implementations, preserve organizational knowledge, and extend the value of PackML into day-to-day engineering and production operations.
- ISA-TR88.00.02-2015, Machine and Unit States: An Implementation Example of ANSI/ISA-88.00.01 (the PackML state model, modes, and PackTags as published by ISA).

Screenshot usage and rights

PLCs.ai product screenshots shown in this paper are proprietary to PLCs.ai, Inc. and are included for informational purposes. Reproduction, extraction, republication, or other public use of these screenshots outside this paper requires prior written consent from PLCs.ai.