

VNA-SB-024 | Oracle Data, AI & Industry Accelerators | Automated AST & AI Reverse-Engineering Blueprints | Sage Canyon

Reverse-Engineer Undocumented PL/SQL, PowerBuilder & Oracle Forms Codebases into Specs

The #1 reason legacy modernization projects fail is that nobody alive knows what the existing 20-year-old application actually does. Before you write a single line of replacement code, our Legacy Code X-Ray engine combines deterministic Abstract Syntax Tree (AST) parsers with domain-tuned LLMs to generate a 100% cited functional specification, entity-relationship lineage map, and dead-code inventory in 2 weeks.

Search Queries & Scope Covered: reverse engineer undocumented PL/SQL packages to requirements | extract business rules from legacy PowerBuilder and Oracle Forms | automated dead code analysis Oracle database PL/Scope | convert legacy stored procedures to functional specification BRD | AST call graph and data lineage for legacy modernization

SOURCE LEGACY STACK (STACK X)		MODERNIZED TARGET ARCHITECTURE (STACK Y)	
Undocumented Legacy Codebases (PL/SQL, PowerBuilder, Forms, Pro*C, T-SQL, RPG/COBOL)		AST Dependency Graphs, Business-Rule Catalogs, OpenAPI Specs & Target Code	
DISCOVERY TIMELINE 2-3 Weeks (vs. 6 Months)	DEAD CODE IDENTIFIED 30-45% Avg Eliminated	RULE TRACEABILITY 100% Line-Number Cited	SUPPORTED SOURCE LANGUAGES PB / Forms / PLSQL / T-SQL

1. Problem Space & Architectural Bottlenecks

- Traditional discovery relies on interviewing business users who only know the screens they click daily and miss 40% of the silent validation, pricing, and compliance rules buried in triggers and stored procedures.
- Conversely, dumping raw 10,000-line PL/SQL packages or PowerBuilder `.srw`` files into generic AI chat windows fails due to context-window truncation, missing cross-file call graphs, and hallucinated summaries.
- Without a dead-code audit, engineering teams waste 35%+ of their modernization budget converting obsolete screens, unused tables, and orphaned procedures that have not executed in years.

2. Technical Solution & Reference Architecture

- 2.1 Stage 1 - Deterministic Compiler AST & Call-Graph Construction:** We parse your complete source repository (`.pbl`/`.srw`/`.srd`, .fmb`/`.pll`/`.rdf`, DBA_SOURCE` PL/SQL, T-SQL, Pro*C, ColdFusion, C#) into a deterministic graph database mapping every UI control -> event trigger -> procedure call -> table/column CRUD operation.`
- 2.2 Stage 2 - Runtime Telemetry & Dead-Code Pruning:** By correlating the static AST call graph with database audit/AWR execution counts (`V$SQL` / `DBA_HIST_SQLSTAT` / table modification timestamps), we isolate the 30% to 45% of legacy objects that are completely dead.`
- 2.3 Stage 3 - AST-Grounded AI Specification Synthesis:** Each topologically sorted call subgraph is fed into our private OCI/Anthropic/OpenAI inference pipeline to produce human-readable Functional Requirements Documents (BRDs), state machines, and edge-case tables where every bullet cites the exact legacy file and line range.
- 2.4 Stage 4 - Automated Target Scaffolding & Test Generation:** From the verified AST and specification, we generate target Oracle APEX 24.2 blueprints, PL/SQL package stubs, OpenAPI 3.1 contracts, and `utPLSQL` / Playwright`

regression test suites.

2.5 Query Fan-Out Rationalization & Runtime/Inference Filtering: Instead of migrating cloned reports and DataWindows 1-for-1, our AST SQL parser clusters queries by their canonical core JOIN graph, builds governed Oracle 23ai views only for those core queries, and generates dynamic WHERE clauses, faceted filters, and ad-hoc report variations at runtime and via AI inference (APEX 24.2 Smart Filters & Select AI).

3. Artifact-by-Artifact Conversion & Modernization Matrix

Legacy Artifact / Baseline Workload	Modernized Oracle / Cloud Target	Engineering Method & Outcome
Undocumented 5,000-Line PL/SQL or T-SQL Packages	Line-Cited Functional Specification + Modular Package Design	AST Control-Flow Graph + LLM Summarization
Spaghetti UI Triggers (PowerBuilder / Oracle Forms)	Extracted Business-Rule Catalog + State Transition Matrix	UI-vs-Domain Logic Separation Classifier
Orphaned Tables, Views & Uncalled Procedures	Verified Dead-Code Decommission Manifest	Static Call-Graph + `DBA_HIST_SQLSTAT` Join
Implicit Foreign Keys (Enforced Only in App Code)	Explicit DDL Constraints & Interactive ERD Lineage Diagram	AST Join-Predicate Mining
Cloned Reports / Queries Differing Only by WHERE Clause or Group-By (Query Fan-Out)	Canonical Core SQL View + APEX 24.2 Runtime / AI Inference Filters	AST Join-Graph Clustering (75-90% Report Count Reduction)

4. Oracle APEX & Ancillary Engineering Toolchain

- Viscosity Multi-Language AST Parser (PowerBuilder, Oracle Forms XML, PL/SQL, T-SQL, ColdFusion, .NET)
- Oracle Database 23ai AI Vector Search & Graph for querying codebase dependencies
- Oracle SQLcl & `DBMS\_METADATA` / `PL/Scope` (`DBA\_IDENTIFIERS`, `DBA\_STATEMENTS`)
- utPLSQL & Playwright automated parity test generators

Anonymized Production Proof: Used as the Phase 0 foundation across our PowerBuilder, Oracle Forms, Sybase, and ERP modernization engagements, Legacy Code X-Ray cuts discovery time from 6 months of manual interviews to 2 weeks of deterministic analysis while eliminating up to 45% of dead legacy code before build begins.

5. Principal Architect Q&A

Q: How is Legacy Code X-Ray different from simply using GitHub Copilot or ChatGPT on our source files?

A: LLMs alone cannot resolve cross-file inheritance, global variables, dynamic SQL, or database trigger chains across 2,000 files. We use real compiler parsers (including Oracle `PL/Scope` and our PBL/FMB AST engines) to build the exact mathematical dependency graph first, so the AI never guesses or hallucinates relationships.

Q: Does our source code ever train public AI models?

A: Never. All AST parsing runs locally inside isolated enclaves, and any semantic summarization uses zero-data-retention enterprise API endpoints.

Q: What is included in the downloadable PDF Solution Brief?

A: The PDF Solution Brief includes a sample Legacy Code X-Ray Diagnostic Output (showing an AST dependency map, extracted business-rule table with line citations, and dead-code scorecard) plus instructions for requesting a 2-week Phase 0 Codebase Assessment.