prd01 core translator engine

PRD 01 — Core Translator Engine (JPE Engine)

1. Product Vision
The Core Translator Engine is the local "brain" of the system. It understands Sims 4 mod files, explains them in Just Plain English (JPE), and converts JPE and JPE-XML back into valid Sims 4 XML tuning. It must be fast, deterministic, safe, and deeply integrated with diagnostics.

2. Users
- Mod authors who want to work in English instead of raw XML.
- Players and hobbyists who want to read and understand what a mod does.
- Power users and tool authors who want a stable API and data model.

3. Scope (v1)
In Scope:
- Read and parse XML tuning files.
- Extract relevant resources from .package files.
- Parse JPE and JPE-XML into a unified Intermediate Representation (IR).
- Generate Sims 4-compatible XML tuning from IR.
- Perform structural and basic semantic validation.
- Emit structured diagnostics for all failures.

Out of Scope (v1):
- Live patching of the running game.
- 3D assets, meshes, or CAS part editing.
- Fully automatic decompilation of game executables.

4. Functional Requirements
4.1 File Intake
- R1.1: Load XML tuning files from directories and individual paths.
- R1.2: Load .package files and extract XML tuning and STBL resources.
- R1.3: Optionally detect and surface Python files in .ts4script archives as references.

4.2 Intermediate Representation (IR)
- R2.1: Define IR entities for Interactions, Buffs, Traits, Statistics, LootActions, TestSets, Enums, and Localization.
- R2.2: Preserve tuning identity fields (module, class, instance IDs, names) in IR.
- R2.3: Represent nested tunables consistently, including lists, enums, primitive values, and references.

4.3 Parsing XML to IR
- R3.1: Map top-level tuning nodes into IR objects.
- R3.2: Convert tuning tags and attributes into typed fields: primitives, lists, variants, and references.
- R3.3: Extract references to localization keys and link them to STBL entries when available.

4.4 JPE and JPE-XML to IR
- R4.1: Parse JPE DSL text into IR using a formal grammar.
- R4.2: Parse JPE-XML using a schema into the same IR structure.
- R4.3: Provide clear and structured error information for malformed JPE and JPE-XML.

4.5 IR to XML
- R5.1: Generate valid Sims 4 XML tuning from IR, including correct attributes and structure.
- R5.2: Allow configurable strategies for generating or preserving instance IDs.
- R5.3: Produce deterministic output given the same IR and configuration.

4.6 Validation

- R6.1: Structural validation: ensure required fields are present and types match expectations.
- R6.2: Semantic validation: check references, enums, and ranges where rules are known.
- R6.3: Per-version validation: load optional rule packs for different game patches.

5. Diagnostics and Error Handling

- R7.1: All parser and generator failures must emit structured EngineError objects.
- R7.2: Errors must include file path, resource identifiers, severity, and optional suggestions.
- R7.3: The engine must be capable of generating error reports for builds, in JSON form and optionally as human-readable text.
- R7.4: The engine must support translation of low-level exceptions into human-friendly messages using error codes and templates.

6. Non-Functional Requirements

- Performance: A medium-sized mod should round-trip through the engine in under a few seconds on a typical PC.
- Reliability: Parser and generator should handle malformed input robustly and never crash the host applications.
- Testability: Maintain a high level of automated tests for parsing, generation, validation, and diagnostics.
- Extensibility: IR and engine APIs should be able to grow with new tuning types and versions.