JPE Sims 4 Mod Translation Suite
Icon System PRD (v1.0)

1. Purpose & Scope

This document defines the icon system for the JPE Sims 4 Mod Translation Suite. It covers:

- App and product icons
- File-type icons
- Diagnostics and state icons
- Action and control icons
- Technical specifications, folder structure, and integration requirements

The goal is to give designers and engineers a clear, implementable map of all icons required by the suite as described in the overall SOP and branding PRD.

2. Goals & Constraints

2.1 Goals

1) Create a consistent, recognizable icon language across desktop, iOS, and any optional cloud UI.
2) Allow users to instantly distinguish between different Sims 4-related file types (XML, STBL, package, scripts, configs, JPE, JPE-XML).
3) Surface diagnostics and problem states clearly and intuitively.
4) Keep icons neutral, tool-focused, and legally distinct from Sims 4/EA graphical assets.

2.2 Constraints

- Icons must render clearly at small sizes (16×16, 20×20) and scale up to 512×512 for app icons.
- All core icons must be vector-based (SVG) at source.
- Icons must not imitate Sims 4 visual assets (plumbobs, key art, specific gradients, etc.).
- Icons should rely on shape plus color; not color alone, especially for diagnostics.

3. Icon Layers Overview

The icon system consists of four primary layers:

1) App & product icons
2) File-type icons
3) Diagnostics & state icons
4) Action & control icons (toolbar, buttons, tabs)

Each icon must be assigned a stable ID (e.g., icon.app.desktop, icon.file.xml, icon.diag.error) used in code and configuration.

4. App & Product Icons

4.1 JPE Studio (Desktop)

Use cases:

- Windows executable icon (ICO)
- Taskbar and Start Menu
- Installer wizard header
- About dialog and splash screen

Requirements:

- Primary mark: core JPE symbol (e.g., stylized document or node cluster with JPE identity).
- Secondary motif: desktop or editor metaphor (window frame, cursor, panels).
- ICO file must include standard Windows icon sizes (16, 24, 32, 48, 64, 128, 256, 512 px as available).

4.2 JPE Mobile (iOS)

Use cases:

- Home screen icon
- App Store listing
- In-app splash and settings

Requirements:

- Uses same JPE core mark as desktop, adapted into rounded square.
- Export full iOS icon set (all required @2x and @3x sizes).
- Respect Apple’s safe-zone margins and rounded-corner mask guidelines.

4.3 Engine, Language, and Cloud Badges

Small badges used in settings, system status, and docs:

- JPE Engine: gear + node or small document-plus-gear motif.
- JPE Language (JPE plain text): text bubble or sheet icon.
- JPE-XML Language: angle bracket + text sheet.
- JPE Cloud (if used): cloud with JPE mark.

These badges should be designed to work 16–24 px and be legible next to labels in settings and status areas.

5. File-Type Icon Set

5.1 Requirements

The system must visually distinguish at least the following file types:

- Sims XML tuning files (.xml)
- JPE-XML files (.jpxml or equivalent)
- JPE plain English source files (.jpe or equivalent)
- STBL string tables (.stbl)
- Package containers (.package)
- Script files (.ts4script, .py)
- JSON / config / ini files

All file-type icons share:

- A common document silhouette for consistency.
- A distinguishing glyph and accent color to identify the specific type.

5.2 Shared Document Base

Base icon features:

- Document shape with slight dog-ear in top-right corner.
- Neutral outline, minimal shading to keep icons clear at small sizes.

- Space reserved in bottom-right or center for type glyph.

5.3 Mandatory File-Type Icons

1) Sims XML Tuning (.xml)

- Base document + angle-bracket glyph (“<>” style shape) to represent XML.
- Accent color A (distinct from JPE and JPE-XML colors).

2) JPE-XML (.jpxml or similar)

- Base document + combined angle bracket and “A-Z” hint (e.g., bracket overlapped with a letter).
- Accent color B tied to JPE-XML language badge.
- Must not be easily confused with raw XML; distinct color and glyph variation required.

3) JPE Plain English Source (.jpe or similar)

- Base document + lines-of-text or speech bubble glyph.
- Accent color C tied to the JPE language identity.
- Emphasis on “readable text” rather than code brackets.

4) STBL String Table (.stbl)

- Base document + speech bubble or chat glyph, representing localized strings.
- Accent color D (can share role with “localization” or “text” semantics).

5) Package Containers (.package)

- Box/container glyph (e.g., small crate or closed box) as the main shape.
- Can use a modified document base (box silhouette instead of document) if visually helpful.
- Accent color E and subtle outlines to keep legible at 16–24 px.

6) Script Files (.ts4script, .py)

- Base document + code brackets or curly braces glyph.
- Could share a core icon with small variant (e.g., tiny Python-like snake hint vs generic script), but this is optional.
- Accent color F that clearly signals “code”.

7) JSON / Config / INI

- Base document + gear, sliders, or braces glyph (“{}”).
- Accent color G that aligns with “configuration” semantics.

5.4 In-UI Usage

- Project explorer: each node shows the correct file icon before the file name.
- Recent files lists: icons precede names and help visually group types.
- Mobile file list: 24–32 px icons used in leading column, with file name and description to the right.

6. Diagnostics & State Icons

6.1 Severity States

The icon set must represent at least four core severities:

- Error
- Warning
- Info
- Success/Resolved

Shapes and ideas:

- Error: solid circle or diamond with an “X” or exclamation, high-contrast danger color.
- Warning: triangle with exclamation, caution color.
- Info: circle with “i”, neutral or cool color.
- Success: circle or badge with checkmark, “go” color.

6.2 Contexts

Diagnostics icons are used in:

- Desktop problems pane: one icon per row, aligned with text severity.
- Editor gutter: per-line icons where issues exist.
- Project tree: aggregated state icons as overlays or next to folders/files that contain issues.
- Mobile diagnostics list: leading icon next to each item; optionally repeated as a small overlay on file icons.

6.3 Metadata & Mapping

Each diagnostic message from the engine must contain:

- severity: error | warning | info | success
- code: stable string identifier
- icon_id: string referencing icon asset

The UI maps icon_id to an SVG asset or sprite. The engine does not hardcode SVG paths; it only emits IDs.

7. Action & Control Icons

7.1 Core Desktop Actions

Icons needed for desktop toolbar, menus, and context menus:

- Translate to JPE
- Translate to JPE-XML
- Roundtrip (XML $\rightarrow$ JPE $\rightarrow$ XML)
- Validate / Run checks
- View diagnostics
- Open original XML
- Open in external editor
- Sync with cloud (if enabled)
- Search, filter, refresh
- Create new file (various types)
- Save, save all
- Undo, redo

Design constraints:

- Simple geometric icons that remain legible at 16–20 px in toolbars.
- Use a limited color palette; rely mostly on shape and a small accent region.
- Use labeling in tooltips and menus to ensure clarity.

7.2 Mobile Actions

Icons needed for iOS navigation and controls:

- Tab bar icons:
  - Projects
  - Files
  - Diagnostics
  - Settings
- Action icons:
  - Edit JPE / JPE-XML
  - View error details
  - Pull from / sync with desktop or cloud

Icons must comply with typical mobile icon guidelines: simplified silhouettes, no tiny type.

8. Technical Specifications

8.1 Formats & Sizes

Source format: SVG for all icons.

Export formats:

- Desktop:
  - ICO for main application icon (multi-size stack)
  - PNG sprite sheets or individual icons at 1x/2x for in-app usage
- iOS:
  - PNG app icon set in asset catalog structure
  - PNG vector-backed UI icons (or SF Symbol mapping where possible)
- Web/docs (if any):
  - SVG and 2x PNG variants

Standard size targets:

- 16×16, 20×20, 24×24, 32×32, 48×48, 64×64 for in-app icons.
- 128×128, 256×256, 512×512 for app and launcher icons.

8.2 Folder Structure

Recommended structure within the repository:

- docs/branding/
  - icons_app/
  - icons_filetypes/
  - icons_diagnostics/
  - icons_actions/
  - icon_map.json
- desktop/assets/icons/
- mobile/ios/JPEAssets.xcassets/
- cloud/assets/icons/ (if a web UI is implemented)

8.3 Icon Map Configuration

Create an icon_map.json (or similar) describing:

- icon_id

- description
- category (app, filetype, diagnostic, action)
- file path(s) to SVG and PNG exports
- fallback icon_id (if any)

Example entry (conceptual):

```
{
  "icon_id": "file.xml",
  "description": "Sims XML tuning file",
  "category": "filetype",
  "svg": "docs/branding/icons_filetypes/xml.svg",
  "png_1x": "desktop/assets/icons/xml_16.png",
  "png_2x": "desktop/assets/icons/xml_32.png",
  "fallback": "file.generic"
}
```

9. Integration Requirements

9.1 Desktop Integration

- Project tree must show file-type icons based on extensions and/or metadata.
- Issues/problems pane must show diagnostic icons based on severity and icon_id.
- Editor gutter must display per-line severity icons with tooltips.
- Menu and toolbar actions must use the defined action icons, not ad hoc glyphs.

9.2 Mobile Integration

- File lists and diagnostics lists must display the correct icons for file types and severity.
- Tab bar and common actions use the standard action icons set for visual consistency with desktop.
- Where screen space is limited, ensure icons are still recognizable and supplemented with text labels.

9.3 Engine & Diagnostics Integration

- Engine emits `icon_id` and `severity` for diagnostic items.
- UI is responsible for mapping these to actual graphics.
- New diagnostic types must be registered with an icon_id and added to icon_map.json; undefined icon_ids must fall back to a generic “info” or “unknown” icon.

10. Accessibility

- Icons representing important actions or states must always be accompanied by text labels in at least one place in the workflow (tooltips, list text, etc.).
- Diagnostics states must not rely on color alone: shapes and labels like ERROR/WARNING/INFO must be present.
- Ensure icon shapes are distinguishable for users with low vision or color vision deficiency.

11. Legal & Compliance

- Do not use any Sims 4 assets, plumbobs, or key art in icons.
- Do not design icons that look like or imply official EA/Sims approval.
- Include a standard disclaimer in docs and About pages that clarifies the tool’s independence.

12. Acceptance Criteria

The icon system is considered implemented when:

1) All required icons (app, filetype, diagnostics, actions) are designed in SVG and exported to the necessary formats and sizes.
2) icon_map.json (or equivalent) exists and is used by desktop and mobile UIs to resolve icon_id to assets.
3) Desktop project explorer, problems pane, and editor gutter display the correct icons for files and diagnostics.
4) Mobile app file lists and diagnostics lists show the correct icons for files and diagnostics.
5) Accessibility rules are satisfied: shape + label usage, not color only.
6) Any new file type or diagnostic introduced into the engine is blocked from release unless an icon_id has been mapped to a valid icon or generic fallback.