

COMPLETED TEST REPORT

Delphi App Translation Studio

Completed Test Report and Remediation Plan

Test cycle	Completed August 9, 2026
Status	Planning baseline - no remediation code implemented
Studio source	C:\Projects\Delphi App Translation
Pilot project	C:\Projects\FMXPilot - Component Test
Pilot branch	codex/component-manager-pilot

Decision requested

Approve Group 0 - Freeze evidence and prove ownership - before any correction code is written.

Documentation refreshed: August 25, 2026

Table of Contents

Appendix A. License Information 13

Table of Contents.....2

1. Executive assessment.....3

2. Test scope and evidence.....4

3. Test-case outcome matrix.....4

4. Detailed defect register.....6

5. Remediation plan.....9

6. Regression acceptance matrix.....11

7. Future feature register.....11

8. Recommended immediate decision.....12

1. Executive assessment

The completed test cycle proves that the central product concept works. The Studio can open and scan a Delphi FMX project, create a development catalog, obtain automatic translations through a configured provider, validate and export JSON runtime packs, generate a non-mutating component kit, support manual Delphi package installation, and drive immediate language changes through a TDATAFMXLanguageManager and bound language selector. The disposable Website Analytics pilot built successfully for Win32 and Win64, discovered deployed English and Spanish packs, persisted the selected language, and reapplied it after restart.

The product is not yet ready for a general release. The test cycle exposed one critical offline failure in the pilot, several high-priority localization lifecycle defects, incomplete string coverage, translation-driven layout problems, and numerous documentation ambiguities. The most important technical lesson is that static designer text, dynamic runtime state, generated messages, owner-drawn text, data values, and help content cannot all be treated as the same class of string.

Release recommendation

Continue development; do not release the current build as version 1.0. Preserve the working component architecture and correct the identified defects in controlled groups. The component approach remains the preferred integration model because the Studio can generate a kit without writing into the target project. The developer deliberately places and configures the manager and optional selector in Delphi.

What has been demonstrated successfully

- Non-mutating generation of a component integration kit.
- Manual installation of the FMX design package through supported Delphi procedures.
- A designer-owned TDATAFMXLanguageManager on the primary form.
- A designer-owned language selector linked to the manager.
- Automatic Google or DeepL translation into JSON catalog data.
- Runtime discovery of English and Spanish language packs.
- Immediate language switching without restarting the application.
- Persistence of the selected language across application restarts.
- Successful Win32 and Win64 Debug builds of the pilot.
- Offline JSON packs that do not themselves require internet access.

Principal release blockers

TC-18 can enter a repeating connection-error dialog loop while the pilot is offline.

Language application can overwrite live runtime state with design-time text.

Startup can produce a mixed-language interface even when Spanish is restored correctly.

The current scan reports 173 entries versus 179 in an earlier test state; the six-key difference has not been reconciled.

Dynamic, generated, owner-drawn, status, chart, and help text is incompletely localized.

Longer translations crowd or truncate controls.

2. Test scope and evidence

The test followed the nineteen cases in the Delphi App Translation Studio Complete Test Guide. The pilot was intentionally disposable and was created from the pristine Website Analytics source. Testing covered Studio navigation, project scanning, provider configuration, catalog creation, automatic translation, validation, runtime export, component kit generation, Delphi package installation, visible component integration, Win32 and Win64 builds, pack deployment, language switching, state preservation, restart persistence, offline operation, and malformed-pack handling.

Evidence consisted of the tester's direct observations, command output, Git status output, build results, and screenshots supplied during the test session. Where no defect was reported for a completed case, this report records the case as passed without an observed defect. That status is not a substitute for automated regression coverage.

3. Test-case outcome matrix

Test	Outcome	Summary
TC-01 Launch and navigation	Pass with usability issue	The Studio launched and workflow navigation operated, but the persistent bottom status panel did not change meaningfully with the selected workflow page.
TC-02 Open the disposable GA4 project	Pass	The correct disposable component-test project was opened. Folder identity and Git cleanliness required clarification during setup.
TC-03 Scan the project	Pass with defect	The scan completed quickly, reporting 173 entries in approximately 50 ms. The earlier 179-entry result remains unexplained, and the results memo can visually overprint/scrunch lines while scrolling.
TC-04 Configure and test a provider	Pass	Provider configuration and the small connection test succeeded. The guide wording made the expected English-to-Italian test sound like a separate action.
TC-05 Create the development catalog	Pass with UI defects	Catalog creation worked. Encoding artifacts such as EspaÃ±ol, a truncated catalog path, and crowded controls were observed. The catalog path should be a complete, clickable folder link.
TC-06 Translate automatically	Pass	Automatic provider translation completed and populated the JSON development catalog. Bulk review and approval operated on the complete set.
TC-07 Focused linguistic and structural review	Pass with usability concern	Review and approval worked, but the workflow and status meanings require clearer explanation for large projects.

Test	Outcome	Summary
TC-08 Validate and export runtime JSON	Pass with documentation defects	Validation and JSON export worked. The guide said Validate Catalog while the UI says Run Validation; warning meanings were unclear, and the referenced output location should be an active link.
TC-09 Generate the non-mutating component kit	Pass after redesign	The component kit was generated without writing target source. Instructions, memo sizing, package locations, and package dependencies caused substantial confusion and were later redesigned around manual installation.
TC-10 Install the FMX design package	Pass with guide corrections needed	The package was ultimately installed manually and the DAT Localization palette displayed the manager and combo box. The guide must describe both components and the exact package build/install order.
TC-11 Perform the visible Delphi integration	Pass after corrections	The manager and combo were placed and configured. An ApplicationId spacing mismatch initially produced an empty selector. The selector was accidentally parented to a paint box and later corrected in the FMX designer resource.
TC-12 Build GA4 Win32 and Win64	Pass	Both platforms compiled. The guide should explicitly say when to save, and explain that the .rsm file beside the executable is a normal compiler-generated file.
TC-13 Deploy language packs	Pass after command correction	Deployment succeeded for Win32 and Win64 when PowerShell was launched with -ExecutionPolicy Bypass. The guide's commands were separated from the steps and the final nested destination was not sufficiently obvious.
TC-14 First launch and source language	Partial / fail	Translation works, but Spanish causes header crowding, untranslated text, truncated text, and English help screens.
TC-15 Immediate language switching	Functional pass; procedure defective	Immediate switching works. The original steps were impractical because secondary forms obscure the main form. The workable sequence is to change the language and then open each secondary form for inspection.
TC-16 Preserve control state and date range	Fail	Changing languages can display Not connected even while the application remains connected. Clicking Update restores the correct status, proving that localization overwrote live state.
TC-17 Preference persistence and restart	Partial / fail	Spanish selection persists after restart, but substantial English text remains because startup and data refresh overwrite or bypass translated values.
TC-18 Offline runtime	Critical fail	While offline, the pilot can enter an effectively infinite loop of connection-error dialogs.

Test	Outcome	Summary
TC-19 Missing and malformed pack behavior	Pass - no defect reported	The test cycle was completed and no separate TC-19 defect was recorded. This behavior still needs automated regression coverage.

4. Detailed defect register

DAT-001 - Repeating offline connection-error dialogs

Priority: P0 / critical for the pilot

Observed in: TC-18

Symptom: With internet connectivity removed, dismissing the server name or address could not be resolved dialog leads to another dialog, preventing normal use.

Important ownership question: Website Analytics requires the internet for GA4 data, whereas the localization runtime does not. The loop may therefore be an existing target-application timer/retry defect rather than a defect in TDATLanguageManager. The first remediation task must reproduce the condition with localization disabled or removed. The localization product must neither trigger nor amplify the loop, but it should not absorb responsibility for an unrelated target networking defect.

Required behavior: One controlled offline notification, automatic refresh suspended or rate-limited, no modal-dialog loop, retained UI state, and a manual retry path after connectivity returns. Any displayed error should follow the selected language where the host application provides localized error resources.

DAT-002 - Localization overwrites live runtime state

Priority: P1 / high

Observed in: TC-16

Symptom: Switching languages changes a genuinely connected application to the design-time status Not connected. Pressing Update recalculates and restores Connected.

Probable cause: The runtime catalog includes a property that is not static UI text. Reapplying the language writes the catalog's design-time value over a value owned by live application state.

Required correction: Introduce explicit runtime classification and exclusion rules. Dynamic-value labels must not be translated as static captions. Where a dynamic status contains human language, the host application should use a stable message key or a resource/template translation API and then inject variable data.

DAT-003 - Mixed-language startup after preference restoration

Priority: P1 / high

Observed in: TC-17

Symptom: Spanish is selected after restart, but status, buttons, grid headings, dashboard cards, chart headings, and generated report text appear partly in English.

Probable causes: The preferred language is applied before all forms and controls finish loading; later startup routines assign English text; generated content does not use translation keys; or the catalog never contained the missing strings.

Required correction: Define a deterministic lifecycle: load preference, load/validate pack, create form, apply static localization after streaming, populate data, and localize generated messages through keyed templates. Add a post-startup reapplication only where ownership rules prevent it from overwriting dynamic state.

DAT-004 - Catalog coverage regression: 179 entries versus 173

Priority: P1 / high

Observed in: TC-03 and subsequent runtime testing

Symptom: An earlier scan reported 179 translatable entries; the current scan reports 173. Several missing runtime translations are now visible.

Required investigation: Locate the earlier catalog or scan artifact and compare stable keys, source text, origin, runtime classification, and checksums. A count comparison is insufficient. Produce an exact six-key added/removed report before modifying scanner rules.

Likely coverage gaps: Pascal assignments, Format templates, runtime status messages, chart headings, canvas-drawn text, grid column headings set at runtime, and strings outside ordinary FMX text properties.

DAT-005 - Untranslated static and generated text

Priority: P1 / high

Observed in: TC-14 through TC-17

Examples: Update, Connected, Views last 30 min, Events/session, Scrolled, chart headings, Top pages and downloads, status-bar messages, and other generated report captions.

Required correction: Expand extraction beyond form properties while avoiding data values and identifiers. Add keyed runtime templates with placeholders for dates, times, counts, property names, and ranges. Define separate rules for owner-drawn canvas text and grid/column captions.

DAT-006 - Translation expansion causes crowding and truncation

Priority: P2 / medium

Observed in: TC-14

Symptom: Spanish header text crowds adjacent controls; card values and Settings captions truncate; translated headings exceed fixed control bounds.

Required correction: Add Studio validation for likely expansion risk and improve the pilot's FMX layouts using designer-authored layouts, anchors, margins, wrapping, and appropriate control widths. Do not add runtime-only UI construction. A component cannot safely redesign an arbitrary target application, so the Studio should report risks and provide guidance rather than silently moving controls.

DAT-007 - Help content remains English

Priority: P2 / medium

Observed in: TC-14

Symptom: The application interface changes to Spanish while help screens remain English.

Required design decision: Treat help as a separate localizable content set. Options include one help package per language, language-specific topics inside one help project, or JSON-driven application help pages. The runtime manager should expose the active locale so the host can select the correct help target.

DAT-008 - Encoding artifacts in language names

Priority: P2 / medium

Observed in: TC-05

Symptom: Names such as Español display as EspaÃ±ol.

Required correction: Enforce UTF-8 consistently at catalog creation, JSON read/write boundaries, source literals, provider responses, CSV interchange, and UI decoding. Add automated round-trip tests with Spanish, French, German, Portuguese, Polish, Greek, Japanese, and right-to-left samples.

DAT-009 - Studio layout and navigation usability defects

Priority: P2 / medium

Observed across: TC-01, TC-03, TC-05, TC-08, and TC-09

Items: Workflow status panel does not change meaningfully by page; scan results can overprint while scrolling; catalog and pack paths truncate; translated-pack status truncates; translation-page buttons need alignment; integration memos allocate space poorly; large code/change review is unreadable in the small memo.

Required correction: Perform a page-by-page FMX designer review at the supported minimum and maximized sizes. Use designer-owned layouts and Object Inspector properties. Make file/folder paths clickable where appropriate.

DAT-010 - Installation and test instructions contain ambiguities

Priority: P2 / medium

Observed across: prerequisites and TC-04, TC-08 through TC-13, TC-15

Items: RAD Studio 13 was stated too narrowly; unnecessary alternate-tool instructions should be removed; button names differ from the UI; package dependency/build order was unclear; package output locations were unclear; Delphi's package installation dialog was not explained; the palette expectation omitted the second component; Save All timing was missing; expected Git changes were missing; .rsm output was unexplained; PowerShell execution-policy requirements were omitted; deployment commands were far from their steps; and secondary-form testing was impractical.

Required correction: Rewrite the guide against the final UI and personally vet every command from a fresh disposable project. Support compatible Delphi 12/12.1/13 editions only after compiling and installing against each supported toolchain.

5. Remediation plan

Group 0 - Freeze evidence and prove ownership

1. Preserve the current Studio catalog, runtime packs, component kit, pilot source, screenshots, build outputs, and test guide.
2. Recover the earlier 179-entry scan/catalog if available.
3. Create a stable-key comparison between the 179-entry and 173-entry states.
4. Reproduce TC-18 with the language manager disabled or absent to determine whether the loop belongs to Website Analytics, the component, or their interaction.
5. Record an owner and acceptance test for every defect before implementation.

Exit criterion: Every release-blocking symptom has a proven owner and reproducible test.

Group 1 - Runtime lifecycle and state ownership

1. Define catalog classifications for static text, dynamic value, runtime template, data value, identifier, and excluded content.
2. Prevent language reapplication from writing design-time values into dynamic state controls.
3. Add a keyed runtime message/template API with safe placeholders.
4. Establish startup ordering for preference restoration, component loading, form localization, data loading, and generated text.
5. Add immediate-switch and restart regression tests for connection state and selected date range.

Exit criterion: Switching and restarting never falsifies connection/date state and produces a consistently selected language.

Group 2 - Scanner and catalog coverage

1. Reconcile the six missing keys before changing rules.
2. Add controlled extraction for Pascal resource strings and approved literal assignments.
3. Detect common Format and concatenation patterns that should become keyed templates.
4. Add explicit support or documented integration points for grid columns, chart headings, canvas text, and generated status messages.

5. Keep program identifiers, analytics values, property names, URLs, and user data out of translation.
6. Add scan-diff reporting so developers see new, changed, unchanged, obsolete, excluded, and runtime-template entries.

Exit criterion: The pilot's known human-language strings are either translated, deliberately excluded with a reason, or registered as runtime templates.

Group 3 - Offline behavior and resilience

1. Correct the owning application's repeating error path identified in Group 0.
2. Stop or back off auto-update after a network failure.
3. Suppress duplicate modal dialogs and provide a manual retry.
4. Confirm that pack loading and language switching remain fully operational without internet access.
5. Test missing, malformed, mismatched-ApplicationId, and unreadable packs without crashing.

Exit criterion: A disconnected machine produces at most one controlled notification per user action or defined retry window and remains usable.

Group 4 - FMX layout and localization quality

1. Review every Studio page and pilot form at minimum supported size and maximized size.
2. Correct clipping, overprinting, crowding, alignment, and path truncation in the FMX designer.
3. Add catalog validation warnings for likely text expansion and non-wrapping controls.
4. Correct all UTF-8 language-name and translation round trips.
5. Define and implement localized help selection.

Exit criterion: Spanish and at least one longer-language stress pack render without clipped essential text, and language names display correctly.

Group 5 - Documentation and clean-room regression

1. Rewrite the Complete Test Guide after the UI and behavior stabilize.
2. Confirm supported Delphi versions by actual build/install testing.
3. Replace ambiguous labels with exact current button/menu names.
4. Put each command directly under the step that uses it, including -ExecutionPolicy Bypass where required.
5. Explain package dependency order, output locations, .rsm files, Save All, expected Git modifications, and selector placement.
6. Repeat TC-01 through TC-19 from a newly created disposable project copied from the pristine baseline.
7. Update the User Guide and Engineering Guide only after the clean-room run passes.

Exit criterion: A developer unfamiliar with the product can complete the guide without undocumented intervention, and all regression tests pass on Win32 and Win64.

6. Regression acceptance matrix

The correction cycle is complete only when all of the following are true:

- No repeating offline dialogs.
- English and Spanish packs load without internet access.
- The selected language survives restart.
- Immediate switching preserves connection state, date range, selected website, and other live values.
- Static captions, registered dynamic templates, charts, grids, status messages, and secondary forms use the selected language.
- Help opens in the selected language or clearly documents that a localized help package is unavailable.
- The scan-count discrepancy is explained by an exact key-level report.
- No mojibake appears in language names or translated strings.
- Essential text is not truncated at the supported minimum window size or when maximized.
- Win32 and Win64 Debug and Release builds pass.
- Runtime packs deploy to the documented folder beside every tested executable.
- A clean Git comparison shows only the developer-approved component integration and language assets.
- The complete test guide can be followed without oral correction.

7. Future feature register

These items are intentionally deferred until the release blockers are corrected.

FUT-001 - Complete built-in language registry

Add a comprehensive, maintained language registry to the Translation Studio and include it in exported/distributed Studio builds. Each entry should provide at least the BCP 47 language tag, English display name, native display name, writing direction, provider mapping, locale defaults, and support status. The registry should populate lists wherever a language can be selected, eliminating free-form language entry where a controlled selection is possible.

This feature does not mean generating or shipping translated target-application packs without developer choice and provider usage. It means the Studio ships with the complete language definitions needed to select, configure, and export supported languages consistently.

FUT-002 - Developer-selectable language presentation

Support both designer-owned selector styles over the same manager and JSON packs:

- A Language menu for target applications that already have a menu bar.
- A language combo box for applications without a menu bar.

The developer chooses the presentation. The Studio should generate precise setup instructions for the selected framework and selector style without silently modifying the target UI.

FUT-003 - Localized help packages

Allow the Studio and runtime manager to associate locale tags with language-specific help files or help topic roots.

FUT-004 - Translation expansion analysis

Estimate expansion risk before export and report controls likely to clip or crowd in the target language.

FUT-005 - Catalog comparison and provenance report

Provide first-class stable-key comparison between scans and catalogs, including added, removed, changed, excluded, obsolete, source origin, provider, and approval state.

8. Recommended immediate decision

Approve Group 0 - Freeze evidence and prove ownership as the next implementation activity. No correction code should be written until the 173-versus-179 catalog difference and the ownership of the TC-18 retry loop have been established. This avoids repairing the wrong subsystem and gives the later runtime/scanner work measurable acceptance criteria.

Document control

Prepared for the Delphi App Translation Studio project.

Last changed: August 25, 2026.

Appendix A. License Information

Unless a particular file or third-party component states otherwise, Delphi App Translation Studio and its project documentation are licensed under the Apache License, Version 2.0. The license permits broad use while preserving attribution, notices, and the stated warranty limitations.

License grant

Subject to the complete license terms, you may use, reproduce, modify, prepare derivative works from, publicly display, publicly perform, sublicense, and distribute the licensed material. The Apache License 2.0 also includes an express patent license from contributors for their applicable contributions.

Important conditions

- Give recipients a copy of the Apache License 2.0 when distributing covered material.
- State significant changes made to modified files.
- Retain applicable copyright, patent, trademark, attribution, and NOTICE information.
- Do not use project or contributor names and trademarks as an endorsement except as the license permits.

Warranty and liability

The licensed material is provided on an AS IS basis, without warranties or conditions of any kind. The complete Apache License 2.0 controls the warranty, liability, contribution, redistribution, and patent provisions.

Your applications and generated output

Using the Studio does not transfer ownership of the Delphi application being translated. The application developer remains responsible for the license and distribution terms of the target application, translated content, generated language packs, and any third-party components. Studio runtime or component source included with a project remains subject to the Apache License 2.0 unless its file states different terms. Third-party software retains its own license.

Full legal text

The complete controlling license is the LICENSE file in the repository and source distribution root. An official reference copy is available at <https://www.apache.org/licenses/LICENSE-2.0>. If this summary and the complete license differ, the complete Apache License 2.0 text controls.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this work except in compliance with the License. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an AS IS BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.