Complete Test Guide

Delphi App Translation Studio

Version 1.0

Last changed: September 1, 2026

User acceptance, component integration, regression, and release testing

Windows - Delphi VCL and FireMonkey - Win32 and Win64

A printable, start-to-finish procedure based on the current source, designer forms, build metadata, package graph, and verified test harness.

Table of Contents

1. Purpose and Test Philosophy 1

2. Exact Folders and Files 2

3. Prerequisites and Test Data 3

4. Build the Studio Manually 4

5. Automated Release Gate 5

6. Manual Studio Acceptance 7

7. Component Integration Acceptance 14

8. Runtime Acceptance 19

9. VCL Acceptance Delta 23

10. Studio Self-Translation 24

11. Starting Over and Dependency Recovery 24

12. Negative and Recovery Tests 25

13. Performance and Large-Project Checks 26

14. Defect Recording 27

15. Final Acceptance Checklist 27

16. Appendix A - Exact Test Scripts 28

17. Appendix B - What Each JSON File Is 29

18. Appendix C - Current Verified Baseline 29

1. Purpose and Test Philosophy

This guide provides the complete acceptance path for Delphi App Translation Studio: safeguard the source, build the Studio and packages, run the automated release gate, scan and automatically translate a real FMX application, generate offline JSON packs, integrate one manager component without Studio-authored target changes, deploy the packs, and verify language switching on Win32 and Win64.

The recommended production path is Component Integration. The Studio generates a kit beneath its own export folder and never opens the selected target project, DPR, DPROJ, PAS, DFM, or FMX files for writing. The developer then performs a small, normal Delphi Form Designer change in a disposable test copy: one manager on the primary form and, when desired, one typed language selector. This is visible, reviewable, and reversible in Git.

The automated matrix is the engineering release gate. The manual cases are user acceptance tests. Both are required for a complete sign-off because automation can prove invariants while a human must confirm visual layout, translated meaning, normal application behavior, and operational clarity.

> **Critical source-protection rule.** Do not place test components in the pristine GA4 repository. Copy it to a disposable working folder first. The pristine repository remains the known-good comparison point.

1.1 Scope

- Studio host: Windows Win32 and Win64.
- Target frameworks: Delphi VCL and FireMonkey (FMX).
- Target architectures: Win32 and Win64, Debug and Release where specified.
- Translation providers: Google Cloud Translation Basic v2 and DeepL API Free/API Pro.
- Runtime operation: local JSON packs with no Internet connection and no provider key.
- Out of scope: macOS, iOS, Android, Linux, C++Builder, automatic control resizing, and runtime cloud translation.

1.2 Stop conditions

- Stop if the selected target is the pristine GA4 folder rather than a disposable copy.
- Stop if Git reports unexplained modifications before integration begins.
- Stop if a provider response alters placeholders, format specifiers, or accelerator semantics and validation reports an error.
- Stop if Component Integration changes any target file before the developer manually opens the target in Delphi.

- Stop if the release validation script reports any failure; preserve the complete console output for diagnosis.
- Stop if a build or launch error dialog appears. Record its full text and the exact executable used.

2. Exact Folders and Files

Purpose	Exact path	Handling
Studio source repository	C:\DelphiProjects\Delphi App Translation	This is the active product folder used by this guide.
Pristine GA4 reference	C:\DelphiProjects\FMXPilot - Pristine	Read-only reference. It was clean when this guide was generated. Do not integrate directly into it.
Recommended disposable GA4 test copy	C:\DelphiProjects\FMXPilot - Component Test	Create from the pristine reference. Perform all manual component integration here.
RAD Studio environment	C:\Program Files (x86)\Embarcadero\Studio\37.0\bin\rsvars.bat	Run builds elevated so MSBuild can read the RAD Studio environment.
Studio project	C:\DelphiProjects\Delphi App Translation\DelphiAppTranslationStudio.dproj	Open this in RAD Studio 13 Florence.
Studio form	C:\DelphiProjects\Delphi App Translation\source\studio\DAT.Studio.MainForm.fmx	Designer-authored FMX user interface.
Complete release gate	C:\DelphiProjects\Delphi App Translation\tools\tests\RunPhase10ReleaseValidation.ps1	Builds and tests the supported matrix.
Component-kit output root	C:\DelphiProjects\Delphi App Translation\export\component-integration	Generated by the recommended Integration mode.
Project-local development catalogs	<Target>\Localization\Development	Lossless editable JSON catalogs; not runtime deployment files.
Project-local runtime packs	<Target>\Localization\Languages	Compact offline JSON packs produced by Export.
Per-user language preference	%LOCALAPPDATA%\<ApplicationId>\language.ini	Stores the last selected language; does not alter the application folder.

2.1 Studio executable matrix

Build	Exact executable
Win32 Debug	C:\DelphiProjects\Delphi App Translation\bin\Win32\Debug\DelphiAppTranslationStudio.exe
Win32 Release	C:\DelphiProjects\Delphi App Translation\bin\Win32\Release\DelphiAppTranslationStudio.exe

Build	Exact executable
Win64 Debug	C:\DelphiProjects\Delphi App Translation\bin\Win64\Debug\DelphiAppTranslationStudio.exe
Win64 Release	C:\DelphiProjects\Delphi App Translation\bin\Win64\Release\DelphiAppTranslationStudio.exe

> **Recommended first manual run.** Use the Win32 Debug Studio executable. After the workflow succeeds, repeat the launch and essential scan checks with Win64 Debug and both Release builds.

2.2 Package source and build outputs

Package	Source DPK	Built output
Core runtime	packages\runtime\DATLanguageManagerCoreRuntime.dpk	bin\packages\<Platform>\<Config>\DATLanguageManagerCoreRuntime.bpl/.dcp
VCL runtime	packages\runtime\DATLanguageManagerVCLRuntime.dpk	bin\packages\<Platform>\<Config>\DATLanguageManagerVCLRuntime.bpl/.dcp
FMX runtime	packages\runtime\DATLanguageManagerFMXRuntime.dpk	bin\packages\<Platform>\<Config>\DATLanguageManagerFMXRuntime.bpl/.dcp
VCL design	packages\design\DATLanguageManagerVCLDesign.dpk	bin\packages\Win32\<Config>\DATLanguageManagerVCLDesign.bpl/.dcp
FMX design	packages\design\DATLanguageManagerFMXDesign.dpk	bin\packages\Win32\<Config>\DATLanguageManagerFMXDesign.bpl/.dcp

Runtime packages build for Win32 and Win64. The Delphi IDE loads the self-contained Win32 design package. Installation is an explicit developer action performed only through RAD Studio's Component > Install Packages > Add command. The Studio does not copy, register, or automatically install BPLs.

3. Prerequisites and Test Data

- Windows account able to run RAD Studio and an elevated PowerShell window for builds.
- RAD Studio 13 Florence / toolchain 37.0 installed at the paths in Section 2.
- Git installed at C:\Program Files\Git and available for status/diff verification.
- A working Google Cloud Translation Basic v2 API key or DeepL API key for the live automatic-translation case.
- Internet access for Provider Settings, Test Connection, and Translate Automatically only.
- The pristine GA4 reference folder and enough free space for a disposable copy and four builds.
- A second target language that can be visually distinguished from English. Spanish (Spain) [es-ES] or Italian (Italy) [it-IT] is suitable.

Credential safety. Never paste an API key into a source file, catalog, runtime pack, test report, screenshot, Git commit, or command line. Enter it only in the Studio's masked Provider Settings field. Remembered keys are stored as Windows Generic Credentials for the signed-in user.

3.1 Create the disposable GA4 test copy

1. Close FMXPilot.exe and close the GA4 project in RAD Studio.
2. Confirm C:\DelphiProjects\FMXPilot - Pristine is clean with: git -C "C:\DelphiProjects\FMXPilot - Pristine" status --short.
3. Copy the complete pristine folder to C:\DelphiProjects\FMXPilot - Component Test. Preserve its .git directory so Git can show every manual integration change.
4. Run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short. The result must be empty before the test begins.
5. Record the starting commit with: git -C "C:\DelphiProjects\FMXPilot - Component Test" rev-parse HEAD.

Existing older test folder. C:\DelphiProjects\FMXPilot - Test already contains earlier automatic-source-integration changes and Localization files. Do not use it as the clean component-first acceptance target.

3.2 Establish a clean Delphi package baseline

1. Start RAD Studio without opening the disposable target project or any target form.
2. Choose Component > Install Packages.
3. If DAT Language Manager FireMonkey design-time package is listed, select it, choose Remove, confirm the removal, and choose OK. If it is not listed, choose Cancel and continue.
4. Close and restart RAD Studio. Confirm that no package-load error appears. Do not manually delete BPLs and do not edit the BDS registry.
5. Keep target files saved and leave the target application closed until TC-10 installs the freshly generated design package. The RAD Studio project may remain open.

Use Delphi's package manager only. Never use Component > Install Component for this product, never select a .dpk, and never copy package files into Delphi system folders. The accepted workflow uses the compiled Win32 design .bpl through Component > Install Packages > Add.

4. Build the Studio Manually

Run builds from an elevated PowerShell or Developer Command Prompt. The project file sends executables to bin\<Platform>\<Configuration> and DCUs to dcu\<Platform>\<Configuration>.

4.1 RAD Studio method

1. Open C:\DelphiProjects\Delphi App Translation\DelphiAppTranslationStudio.dproj.
2. Select Win32 / Debug and choose Project > Build DelphiAppTranslationStudio.
3. Repeat for Win64 / Debug, Win32 / Release, and Win64 / Release.
4. Confirm the four exact executables listed in Section 2.1 have current timestamps and nonzero sizes.
5. Return the active platform to Win32 after the matrix unless another immediate test requires Win64.

4.2 Command-line method

```
cd "C:\DelphiProjects\Delphi App Translation"

cmd /d /c "call ""C:\Program Files (x86)\Embarcadero\Studio\37.0\bin\rsvars.bat"" && msbuild
DelphiAppTranslationStudio.dproj /t:Build /p:Config=Debug /p:Platform=Win32 /v:minimal"

cmd /d /c "call ""C:\Program Files (x86)\Embarcadero\Studio\37.0\bin\rsvars.bat"" && msbuild
DelphiAppTranslationStudio.dproj /t:Build /p:Config=Debug /p:Platform=Win64 /v:minimal"

cmd /d /c "call ""C:\Program Files (x86)\Embarcadero\Studio\37.0\bin\rsvars.bat"" && msbuild
DelphiAppTranslationStudio.dproj /t:Build /p:Config=Release /p:Platform=Win32 /v:minimal"

cmd /d /c "call ""C:\Program Files (x86)\Embarcadero\Studio\37.0\bin\rsvars.bat"" && msbuild
DelphiAppTranslationStudio.dproj /t:Build /p:Config=Release /p:Platform=Win64 /v:minimal"
```

5. Automated Release Gate

Run this before manual acceptance. It performs one uninterrupted Debug/Release and Win32/Win64 matrix. It builds core, VCL, FMX, runtime, design, Studio, streaming, lifecycle, selector, launch, self-localization, foundation, runtime, integration, and non-mutation tests.

```
cd "C:\DelphiProjects\Delphi App Translation"

powershell.exe -ExecutionPolicy Bypass -File .\tools\tests\RunPhase10ReleaseValidation.ps1
```

- Expected final line: Phase 10 complete release validation passed.
- Expected behavior: test processes may open briefly and close automatically.
- The script restores the Studio's self-localization pack and preference after its Italian launch checks.
- Any nonzero exit or exception is a failed release gate. Do not continue to sign-off until repaired and rerun from the beginning.

5.1 What the gate proves

Area	Evidence
Package graph	Debug and Release runtime packages for Win32/Win64; Win32 VCL/FMX design packages; DFM/FMX streaming.
Managers	Initialization, lifecycle discovery, stable form identity, switching generations, state preservation, cleanup, and architecture parity.

Area	Evidence
Studio	Four builds, direct FMX form streaming, ordinary launch, maximized layout contracts, and Italian self-localization.
Translation core	Detection, scanning, schema/provenance, catalog merge, provider contracts, validation, pack generation, and preference handling.
Safety	Component-kit generation writes only below the export root and preserves target project/form SHA-256 hashes.
Project-file safety	Dependency preparation, repeated-run idempotence, temporary build Search Path, direct deployment, and byte-for-byte target-file preservation.

6. Manual Studio Acceptance

TC-01 Launch and navigation

Objective. Verify the designer-authored Studio shell opens maximized and every workflow page is reachable.

Procedure

1. Run C:\DelphiProjects\Delphi App Translation\bin\Win32\Debug\DelphiAppTranslationStudio.exe.
2. Confirm there is no startup error dialog and the title is Delphi App Translation Studio.
3. Select Project, Scan, Translate, Validation, Export, Integration, and Provider Settings in order.
4. Read the bottom status card after every selection.
5. Resize the maximized window and inspect the top, bottom, left, and right edges of each page.

Expected result

- The window starts maximized.
- The blue workflow selection follows the active page.
- The bottom status card changes to page-specific guidance and never retains a message from the previous page.
- Every control remains visible, aligned, and usable; nothing bleeds below or beyond the client area.
- Component Integration (Recommended) is the default Integration method.
- Provider Settings contains only DeepL and Google Cloud Translation, with no CLI-agent controls.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-02 Open the disposable GA4 project

Objective. Verify correct FMX and platform detection without changing the target.

Procedure

1. Select Project > Open Project.
2. Open C:\DelphiProjects\FMXPilot - Component Test\FMXPilot.dproj.
3. Read the project summary before scanning.
4. In a separate console, run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short.

Expected result

- Project: FMXPilot.
- Framework: FireMonkey.
- Windows targets: Win32 and Win64.
- The Scan Project button is enabled.
- Git status remains empty; opening performs no target write.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-03 Scan the project

> **Objective.** Verify fast read-only extraction of FMX designer properties and resourcestrings.

Procedure

1. Select Scan.
2. Choose Scan Project and wait for Scan complete in the status panel.
3. Record total entries, form properties, resourcestrings, files scanned, and elapsed milliseconds.
4. Inspect representative labels, buttons, headings, memo text, list content, and any resourcestring entries.
5. Scroll rapidly from the top to the bottom and back through the scan-result list.
6. Run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short again.

Expected result

- The scan completes without an exception.
- The result list contains stable form/component/property keys and readable source text.
- Rows remain separated while scrolling; text never paints over adjacent rows.
- Elapsed time is reported in milliseconds. Performance is recorded, not hard-coded to a particular machine.
- No Localization folder is created merely by scanning.
- Git status remains empty.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-04 Configure and test a provider

Objective. Verify secure Google or DeepL configuration and a live connection without exposing the key.

Procedure

1. Select Provider Settings.
2. Choose Google Cloud Translation for a Google Basic v2 API key, or choose DeepL and the correct API Free/API Pro plan.
3. Leave timeout at 30 seconds and strings per request at 40 for the first test.
4. Paste the key into the masked API key field.
5. For a persistent test, check Remember securely on this computer. For a temporary test, clear it.
6. Choose Replace / Save Key, then Test Connection.
7. Confirm no API key appears in the status panel, project files, or generated JSON.

Expected result

- The masked field does not display the clear key.
- Test Connection succeeds with a small English-to-Italian request.
- Remembered mode reports Windows Credential Manager storage; session-only mode reports that the key will be forgotten when the Studio closes.
- The target project remains unchanged.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-05 Create the development catalog

Objective. Verify target-language lists, locale defaults, and project-local JSON catalog creation.

Procedure

1. Select Translate.
2. Confirm source language English (United States) [en-US].
3. Select Spanish (Spain) [es-ES], Italian (Italy) [it-IT], or another intended language from the list.
4. Review native language name, left-to-right/right-to-left direction, short/long date, short/long time, decimal, thousands, and currency values.

5. Choose Save.
6. Click the blue displayed catalog path. Confirm File Explorer opens with the JSON catalog selected, then open it in a text editor and confirm it is JSON, not CSV.

Expected result

- Catalog path: C:\DelphiProjects\FMXPilot - Component Test\Localization\Development\FMXPilot.<locale>.translation-project.json.
- The file is UTF-8 JSON and includes application, framework, locale, entries, status, provenance, source checksum, and runtime classification data.
- CSV Out and CSV In remain optional interchange tools; they are not the normal automatic-translation path and are not runtime packs.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-06 Translate automatically

> **Objective.** Verify the Studio sends eligible catalog entries to the configured provider, records returned values one by one, and auto-saves the JSON catalog.

Procedure

1. On Translate, choose Translate Automatically.
2. Read the confirmation: it must name the provider, unresolved count, protected reviewed/approved behavior, and automatic saving.
3. Choose Yes and allow every bounded batch to finish.
4. Inspect entries from the top, middle, and bottom of the catalog.
5. Close and reopen the catalog or Studio, then confirm the translations remain present.

Expected result

- Eligible Needs Translation, Source Changed, and Error entries receive translated text.
- Returned entries are marked Machine translated with Google or DeepL origin; they are not silently approved.
- Reviewed, Approved, Excluded, and Obsolete entries are not overwritten.
- The development JSON catalog is saved automatically after a successful provider result.
- Provider credentials do not appear in the catalog.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-07 Focused linguistic and structural review

Objective. Verify review actions remain available without requiring every entry to be individually approved merely to create a draft.

Procedure

1. Review visible UI strings, product names, short buttons, multiline instructions, placeholders, and accelerators.
2. Correct a deliberately awkward machine result in Translated text and choose Apply Translation.
3. Mark that entry Reviewed, then Approve it.
4. For the remaining translated drafts, choose Review All, verify the count in the confirmation, and accept. Then choose Approve All, verify its Reviewed count, and accept.
5. If the catalog has a Pascal resourcestring, confirm its runtime classification and leave Manual TranslateText wiring confirmed clear unless the target code has actually been wired.

Expected result

- An edited entry records human origin and Edited status.
- Reviewed and Approved are separate deliberate steps.
- Each bulk action operates on its stated eligible set, saves once, and leaves Excluded, Obsolete, and already Approved entries unchanged.
- Manual resourcestring wiring readiness remains separate from designer-property coverage.
- The catalog is saved after edits when a catalog filename exists.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-08 Validate and export runtime JSON

Objective. Verify structural defects block export and a valid catalog produces a compact offline pack.

Procedure

1. Select Validation and choose Run Validation.
2. Read the plain-language summary. Double-click an entry-specific warning and confirm Translate opens with that entry selected.
3. Resolve all errors involving missing text, duplicate keys, changed source, placeholders, sequential/indexed Format arguments, or accelerator keys. Review warnings; informational runtime placeholders require no action.
4. Treat manual resourcestring wiring as a separate readiness warning and resolve it before claiming complete runtime coverage.
5. Select Export and choose Export Runtime Pack.
6. Open the displayed output file and confirm it is JSON.

Expected result

- Export is blocked while validation errors exist.
- A valid catalog reports that the runtime pack is ready.
- Runtime path: C:\DelphiProjects\FMXPilot - Component Test\Localization\Languages\<locale>.json.
- The runtime pack is smaller than the development catalog and contains no API key or development history.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-08A Incremental rescan after a saved UI change

Objective. Prove that a normal Delphi form change enters the existing catalog without discarding completed translations, and that stale scan results cannot be exported.

Procedure

1. In the disposable target project, add a clearly named test label such as lblLocalizationRetest with the source text Localization update test:, or deliberately revise an existing test caption.
2. Choose File > Save All in RAD Studio. The scanner reads the saved DFM/FMX and Pascal files; an edit that exists only in the IDE buffer cannot be scanned.
3. Return to the same Studio project and target-language catalog, then choose Scan Project again.
4. Confirm the summary reports the new or changed entry. If a component was renamed, confirm its previous stable key is retained as Obsolete and its new key is added.

5. Choose Translate Automatically. Confirm that the prompt counts only eligible new, changed, or otherwise unresolved entries—not every entry in the catalog.
6. Run Validation, then export the runtime pack. Confirm the new stable key is present in the runtime JSON.
7. For the stale-scan guard, save one more harmless target text change after the scan and attempt validation/export without rescanning. Confirm final processing is blocked because target source was saved after the last scan.
8. Rescan, translate the newly unresolved entry, validate, export, rebuild the target, deploy the packs, and verify the changed control in both source and target languages.

Expected result

- Matching Reviewed and Approved translations remain unchanged.
- Only new, changed, or unresolved eligible text is sent to the provider.
- Removed or renamed component keys remain traceable as Obsolete rather than silently disappearing.
- Saving target source after the scan invalidates that scan and prevents an outdated runtime pack from being exported.
- The Studio does not change the target form; the only target modification is the deliberate Form Designer change made by the tester.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

7. Component Integration Acceptance

TC-09 Prepare the verified project-local dependencies

Objective. Verify the recommended mode creates the component kit, atomically installs one managed dependency tree, performs the baseline build/deployment, and leaves every target project/source/form file unchanged.

Procedure

1. Before preparation, run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short and record the expected Localization files. Also record the SHA-256 hash of FMXPilot.dproj.
2. Select Integration and leave Component Integration (Recommended) selected.
3. Choose Build Integration Plan.
4. Read the setup plan and choose Prepare / Update Dependencies.
5. Select generated filenames in the left pane and inspect their text in the right pane.
6. Confirm dependencies\DelphiAppTranslation contains source, deployment\Languages, integration-manifest.json, and integrity-sha256.json.
7. Confirm the status reports a baseline Release build using a temporary DAT dependency Search Path and direct pack deployment, or records a specific build error without undoing the valid dependency preparation.
8. Run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short again. Confirm the only new integration area is dependencies\DelphiAppTranslation, plus expected generated Localization data.
9. Recalculate the FMXPilot.dproj hash and confirm it is identical to the pre-preparation value.
10. Choose Prepare / Update Dependencies a second time. Confirm it refreshes the same managed folder and does not create a second dependency tree or duplicate Search Path entry.

Expected result

- Kit root: C:\DelphiProjects\Delphi App Translation\export\component-integration\FMXPilot.
- The kit contains README.txt, component-integration.json, Deploy-LanguagePacks.ps1, ComponentSource, and Localization\Languages.
- The replaceable per-project kit contains no DelphiPackages folder, no BPL, and no automatic installer. Delphi's registered package comes from the Studio's stable bin\packages\Win32\Release folder.
- The language folder contains the validated target pack plus an automatically generated en-US.json source pack.
- Managed dependency root: <Target>\dependencies\DelphiAppTranslation. It contains the complete verified unit closure and canonical language packs.
- Target Pascal, DPR, DPROJ, DFM, and FMX hashes are unchanged. No .targets file or project import is created.

- The second run replaces the one managed dependency set atomically rather than nesting or duplicating it.
- Apply, Restore, Complete Reset, and source-authorization controls remain unavailable in recommended mode.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

7.1 Expected component-kit contents

Path below FMXPilot kit	Purpose
README.txt	Ordered Object Inspector, Search Path, deployment, and build instructions.
component-integration.json	Application identity, FMX framework, manager/selector classes, language metadata, and scanner form roots.
Deploy-LanguagePacks.ps1	Copies only JSON packs to a selected executable directory.
Localization\Languages\en-US.json	Generated source-language pack for deterministic return to English.
Localization\Languages\<locale>.json	Validated translated runtime pack.
ComponentSource\DAT.Components.Core.pas	Framework-neutral manager core.
ComponentSource\DAT.Components.FMX.pas	FMX lifecycle adapter and manager.
ComponentSource\DAT.Components.FMX.LanguageSelector.pas	Optional typed FMX language combo box.
ComponentSource\DAT.Runtime.*.pas	Pack discovery, preferences, runtime manager, and FMX property application.

TC-10 Install the FMX design package through Delphi

> **Objective.** Use RAD Studio's standard package manager to make the manager and selector available without changing the target.

Procedure

1. In the Translation Studio Integration page, choose Show Design BPL. Confirm File Explorer selects C:\DelphiProjects\Delphi App Translation\bin\packages\Win32\Release\DATLanguageManagerFMXDesign.bpl.

2. Start or activate RAD Studio without opening the disposable target project or a target form.
3. Choose Component > Install Packages, then choose Add.
4. Browse to the exact BPL selected by Show Design BPL, select DATLanguageManagerFMXDesign.bpl, and choose Open. Never use Component > Install Component and never select DATLanguageManagerFMXDesign.dpk.
5. Confirm DAT Language Manager FireMonkey design-time package is listed and checked. Confirm the normal Embarcadero package list remains present and Embarcadero FMX Components is listed and checked. Choose OK.
6. Open an FMX form and confirm DAT Localization appears in the Tool Palette.

Expected result

- TDATFMXLanguageManager and TDATFMXLanguageComboBox are available.
- The stable Studio bin\packages path remains the registered package location; the Studio has not copied any BPL into a Delphi system or public Bpl folder.
- The design BPL has no custom DAT runtime-BPL import, so it loads without a missing-module error.
- The complete standard package list remains available; standard controls such as TTabControl stream normally.
- Closing and reopening RAD Studio produces no package-load error.
- No package is installed for VCL unless a VCL target is being tested.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

Do not register a generated-kit path. Per-project component kits are replaceable output. Installing a BPL from a kit would make Delphi depend on a volatile path and could prevent clean kit regeneration. Always use Show Design BPL and the stable Studio bin\packages\Win32\Release file.

Exact compiler Search Path for the target project. In RAD Studio, open Project > Options > Building > Delphi Compiler, select All configurations and All platforms, and append $(PROJECTDIR)\dependencies\DelphiAppTranslation\source to Search path exactly once. Preserve every existing entry and use a semicolon between entries when necessary.

TC-11 Perform the visible Delphi integration

Objective. Add one manager and one optional selector through normal Form Designer operations in the disposable target copy.

Procedure

1. Open C:\DelphiProjects\FMXPilot - Component Test\FMXPilot.dproj in RAD Studio.
2. Verify that Project Options contains $(PROJECTDIR)\dependencies\DelphiAppTranslation\source exactly once in the Delphi Compiler Search path for all configurations and both Windows platforms. Retain every existing entry.
3. Open FMXPilot.MainForm.fmx (frmMainDashboard) in the FMX Form Designer.
4. Drop one TDATFMXLanguageManager on the primary form.
5. Set ApplicationId to FMXPilot, LanguagesFolder to Localization\Languages, and SourceLanguage to en-US.
6. Leave AutoLoadPreferred, AutoTranslateOwner, AutoTranslateNewForms, ReapplyOpenForms, and PreserveControlState True for the acceptance test.
7. Place one TDATFMXLanguageComboBox in an appropriate visible header/menu area.
8. Set its LanguageManager property to the manager. Leave AutoPopulate and ShowLanguageCode True.
9. Choose File > Save All. Do not rely on Build to preserve unsaved Form Designer changes.
10. Close and reopen FMXPilot.MainForm.fmx in the Form Designer. Confirm the manager, selector, label, and all Object Inspector properties are still present.
11. Run git -C "C:\DelphiProjects\FMXPilot - Component Test" status --short and inspect the exact diff before building.

Expected result

- Only normal developer-authored Form Designer changes appear in the target diff. The Studio leaves target Pascal, DPR, DPROJ, DFM, and FMX files unchanged.
- FMXPilot.MainForm.fmx contains the developer-placed TDATFMXLanguageManager and TDATFMXLanguageComboBox. The target Pascal and DPROJ files contain no Studio-written integration changes.
- No ordinary secondary form receives a manager component.
- No Studio-generated DPR startup call or translation unit is added in Component Integration mode.
- Every change is readable in Git and reversible before commit.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

Ninety-form projects. One manager on the primary form covers ordinary FMX forms through the additive before-show lifecycle adapter. The developer does not place a component on every form. Use FormIdentityMappings only for inherited or unusually renamed forms whose runtime identity differs from the scanner's form root.

Mandatory save checkpoint. Do not continue to TC-12 until File > Save All has completed and reopening the form proves that the manager and selector persisted. A successful compile of an older saved form is not evidence that unsaved designer components were included.

TC-12 Build GA4 Win32 and Win64

Objective. Verify the component-first disposable target builds normally for both supported architectures.

Procedure

1. Choose File > Save All once more before changing platforms or building.
2. In RAD Studio select Win32 / Debug and build FMXPilot.
3. Select Win64 / Debug and build FMXPilot.
4. Confirm C:\DelphiProjects\FMXPilot - Component Test\bin\Win32\Debug\FMXPilot.exe exists.
5. Confirm C:\DelphiProjects\FMXPilot - Component Test\bin\Win64\Debug\FMXPilot.exe exists.
6. Resolve any search-path problem by checking the ComponentSource path for the affected platform/configuration; do not copy random DCUs into the target.

Expected result

- Both builds finish with no compiler or linker errors.
- The target contains the manager and selector through normal compiled source/designer resources.
- No provider credential is linked into either executable.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-13 Deploy language packs

Objective. Place identical local JSON assets beside each executable without copying development catalogs or credentials.

Procedure

1. Open PowerShell. Use the explicit ExecutionPolicy Bypass commands printed immediately below this test case.
2. Run the bypass command for C:\DelphiProjects\FMXPilot - Component Test\bin\Win32\Debug.
3. Run the bypass command for C:\DelphiProjects\FMXPilot - Component Test\bin\Win64\Debug.
4. Inspect both Localization\Languages folders.

Expected result

- Each executable directory contains Localization\Languages\en-US.json and the translated <locale>.json.
- No Localization\Development folder is deployed beside the executable.
- No API key, provider settings JSON, or credential artifact is deployed.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

Use these exact commands, substituting the generated kit only if the project name changes:

```
& "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -ExecutionPolicy
Bypass -File "C:\DelphiProjects\Delphi App Translation\export\component-
integration\FMXPilot\Deploy-LanguagePacks.ps1" -ApplicationDirectory "C:
\DelphiProjects\FMXPilot - Component Test\bin\Win32\Debug"
```

```
& "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -ExecutionPolicy
Bypass -File "C:\DelphiProjects\Delphi App Translation\export\component-
integration\FMXPilot\Deploy-LanguagePacks.ps1" -ApplicationDirectory "C:
\DelphiProjects\FMXPilot - Component Test\bin\Win64\Debug"
```

8. Runtime Acceptance

TC-14 First launch and source language

Objective. Verify a fresh profile opens normally and the English source pack is available.

Procedure

1. Remove only the disposable test preference file, if present: %LOCALAPPDATA% \FMXPilot\language.ini.
2. Run C:\DelphiProjects\FMXPilot - Component Test\bin\Win32\Debug\FMXPilot.exe.
3. Inspect the title, primary controls, language selector, date-range control, and application data.
4. Open representative secondary forms such as Settings, Property Manager, and Diagnostics.

Expected result

- The application opens without EArgumentNilException or 'An FMX form is required.'
- The source language is English and the selector lists canonical native names and locale codes.
- No malformed or mojibake language names appear.
- Secondary forms open normally and remain in the source language until another selection is made.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-15 Immediate language switching

> **Objective.** Verify a selection applies immediately to every open form without restarting.

Procedure

1. Keep the primary form and at least one secondary form open.
2. Select the translated language in TDATFMXLanguageComboBox.
3. Observe the application title, headings, labels, buttons, menu/list text, hints where practical, and the already-open secondary form.
4. Open another secondary form after the language change.
5. Switch back to en-US.

Expected result

- All currently open eligible forms refresh immediately.
- A form opened after the change appears translated before first paint.
- Switching to en-US restores deterministic English from en-US.json.
- No application restart is required for either direction.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-16 Preserve control state and date range

Objective. Verify translation refresh does not clear user input or reset the GA4 date-range selection.

Procedure

1. Choose a date range other than the default, for example Yesterday or Last 30 days.
2. Enter representative editable text in any safe test field and select part of it if available.
3. Select the translated language, then switch back to English.
4. Refresh or run a normal report action if it is safe in the test environment.

Expected result

- The date-range combo remains on the same logical selection and does not become blank.
- The reporting period does not silently change to This year or another default.
- Writable edit/memo content, focus, selections, and list/combo ItemIndex remain intact.
- Read-only instructional content is translated where cataloged.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

TC-17 Preference persistence and restart

Objective. Verify the selected language is saved per user and safely applied on the next launch.

Procedure

1. Select the translated language and close Website Analytics normally.
2. Open %LOCALAPPDATA%\FMXPilot\language.ini and confirm Selected=<locale> without editing it.
3. Restart the same Win32 executable.
4. Repeat the translated selection, close, and launch the Win64 executable.

Expected result

- The next launch opens in the saved language without an exception.
- Win32 and Win64 share the ApplicationId preference as intended.
- The preference file contains only language selection data and no provider key.
- Selecting English and closing changes the saved value back to en-US.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-18 Offline runtime

Objective. Prove the finished application needs no network, provider account, or Studio process.

Procedure

1. Close Delphi App Translation Studio.
2. Disconnect the test computer from the Internet or block the disposable FMXPilot executable through the test firewall policy.
3. Launch FMXPilot, switch between English and the translated language, open secondary forms, and restart.
4. Restore the network after the test.

Expected result

- Both languages load from local JSON packs.
- Switching and persistence continue to work offline.
- No provider dialog, API-key prompt, network delay, or cloud dependency appears in the target.

Result	PASS / FAIL / N/A	Tester	
Date		Defect / notes	

TC-19 Missing and malformed pack behavior

Objective. Verify a deployment fault fails safely and does not corrupt the application.

Procedure

1. Close the target.
2. Move the translated JSON file to a temporary folder outside Localization\Languages; do not delete it.
3. Launch the target and inspect the selector and source-language behavior.

4. Restore the file, then make a disposable copy with invalid JSON and place that copy under an unrelated filename in the language folder.
5. Launch again, confirm the invalid unrelated pack is ignored, then remove the disposable invalid file.

Expected result

- The unavailable language is not offered as a valid selection.
- The application remains usable in the source language according to MissingPackBehavior.
- A malformed/unrelated pack does not crash startup or replace valid packs.
- Restoring the valid file and relaunching restores the language choice.

Result	PASS / FAIL / N/A	**Tester**	
Date		**Defect / notes**	

8.1 Repeat the runtime matrix

Executable	Required abbreviated checks	Result
bin\Win32\Debug\FMXPilot.exe	TC-14 through TC-19 in full.	PASS / FAIL
bin\Win64\Debug\FMXPilot.exe	Launch, switch both directions, date range/state, restart, offline.	PASS / FAIL
bin\Win32\Release\FMXPilot.exe	After Release build/deploy: launch, switch, restart, offline.	PASS / FAIL / N/A
bin\Win64\Release\FMXPilot.exe	After Release build/deploy: launch, switch, restart, offline.	PASS / FAIL / N/A

9. VCL Acceptance Delta

Repeat the component workflow with a disposable VCL project when formal VCL acceptance is required. Install DATLanguageManagerVCLDesign.bpl, generate a VCL component kit, place one TDATVCLLanguageManager on the primary VCL form, and optionally place TDATVCLLanguageComboBox. The component-source path and JSON deployment pattern are otherwise the same.

VCL scenario	Expected behavior
Primary and existing forms	Manager applies the active pack and discovers ordinary forms through its private TApplicationEvents lifecycle.

VCL scenario	Expected behavior
Modal form	Manager handles the modal display boundary before normal interaction.
Dynamically created modeless form	VCL has no public additive before-show notification for every such form. If first-paint translation must be flicker-free, call LanguageManager.ApplyToForm(NewForm) after construction and before NewForm.Show.
State protection	Writable edits/memos, focus, selection, and list/combo indexes remain intact when switching.

10. Studio Self-Translation

The Studio can scan and translate its own project. The automated self-localization test uses samples\StudioSelfLocalization\it-IT.json, temporarily deploys it under the Studio project Localization\Languages folder, writes %LOCALAPPDATA%\DelphiAppTranslationStudio\language.ini, launches all four Studio executables, expects the Italian title, and restores any previous files in a finally block.

```
cd "C:\DelphiProjects\Delphi App Translation"
powershell.exe -ExecutionPolicy Bypass -File .
\tools\tests\RunStudioSelfLocalizationSmokeTest.ps1
```

- Expected title: Studio di traduzione app Delphi.
- Expected matrix: Win32/Win64 Debug/Release all Passed=True.
- Expected final line: Studio self-localization smoke tests passed.
- The script must restore the prior Italian pack and language preference or remove its temporary files when none existed.

11. Starting Over and Dependency Recovery

11.1 Recommended component path

Component Integration writes only generated localization data and the managed dependencies folder, so its clean restart procedure is deliberately ordinary and transparent. In the disposable target, close the application, use Git or the Form Designer to inspect/revert the manually added manager and selector, remove the one developer-owned Project Options Search Path entry when it is no longer needed, and remove only generated Localization and dependency data after confirming the exact paths.

Regenerating a component kit replaces generated files under the Studio export root. Preparing dependencies atomically refreshes the single project-local dependencies\DelphiAppTranslation tree,

and direct deployment replaces the canonical JSON set beside each selected executable so retired locale packs are not left stale.

> **Target-source reset is intentionally unavailable.** Apply, Restore, and Complete Reset remain disabled because the Studio does not own target Pascal, form, DPR, or DPROJ files. Revert developer-authored IDE changes through Git or the Form Designer. Refresh or remove only the clearly named managed dependency and generated Localization areas after reviewing their exact paths.

11.2 Managed dependency recovery test

1. Use only a disposable target that has completed Prepare / Update Dependencies.
2. Record the SHA-256 hash of the target DPROJ and one DAT unit under dependencies\DelphiAppTranslation\source.
3. Make a clearly disposable change to that dependency copy so it no longer matches the generated kit. Do not edit target source or project files.
4. Choose Prepare / Update Dependencies again.
5. Confirm the Studio reports the previous dependency snapshot and atomically restores the canonical DAT unit from the current kit.
6. Confirm the target DPROJ hash is unchanged, the dependency tree is present exactly once, and no DelphiAppTranslation.targets file exists.
7. Build and run the disposable target, then compare Git status/diff with the expected dependency-only refresh.
   - The stale dependency copy is replaced by the verified current kit version.
   - The one managed dependency location is refreshed rather than duplicated or nested.
   - The prior dependency snapshot is retained in the reported backup location.
   - Target Pascal, DPR, DPROJ, DFM, and FMX files remain byte-for-byte unchanged.
   - A failed staged copy or promotion restores the previous managed dependency tree and reports the failure.

12. Negative and Recovery Tests

Condition	Action	Expected result
No API key	Choose Translate Automatically.	Studio moves to Provider Settings and explains which provider key is missing; catalog remains usable.
Wrong DeepL plan	Use Free key with Pro endpoint or vice versa, then Test Connection.	Connection fails clearly without exposing the key; correct plan can be selected and retried.
Provider offline/timeout	Disconnect during a disposable provider test.	Bulk translation reports failure; existing catalog data is retained and can be retried.

Condition	Action	Expected result
Validation error	Blank a translation or break a placeholder.	Validation identifies the entry and Export remains blocked until corrected.
Wrong application pack	Place another application's JSON beside the target.	Discovery rejects it by applicationId and does not offer it as a valid language.
Missing component source path	Build a disposable target with the path omitted.	Compiler identifies missing DAT units; adding the exact generated ComponentSource path resolves the problem.
Missing deployed packs	Launch with no Localization\Languages beside the executable.	Application remains usable in source text according to configured fallback; no cloud call occurs.
Corrupt preference	Use a disposable malformed language.ini.	Application must not crash; source/default behavior remains recoverable. Restore or remove the test file afterward.

13. Performance and Large-Project Checks

Do not impose one fixed scan-time pass threshold across all computers. Record form count, source count, entry count, files scanned, elapsed milliseconds, processor, storage type, and whether antivirus or indexing was active. Compare repeat runs on the same machine and investigate substantial regressions.

For a 90-form project, the acceptance goal is one manager on the primary form, not 90 components. Scan and kit generation should remain read-only. Open a representative set of startup, modal, modeless, inherited, settings, data-entry, and report forms; verify stable identities and translations. Use FormIdentityMappings only where the runtime class/instance does not match the scanner root.

Metric	Run 1	Run 2	Run 3	Notes
Forms / source files				
Translatable entries				
Scan milliseconds				
Provider strings / batches				
Provider elapsed time				
Kit generation time				

14. Defect Recording

- Record the exact Studio and target executable path, platform, configuration, and timestamp.
- Record the selected project path, framework, locale, provider, and DeepL plan when applicable. Never record the API key.
- Capture the complete status-panel message and any Windows/RAD Studio dialog text.
- Attach the stable catalog key, source text, translated text, status, origin, and validation message for translation defects.
- Attach Git status and the smallest relevant diff for integration defects.
- For runtime state defects, record the control name, state before switching, selected languages, state after switching, and whether restart was involved.
- State whether the defect reproduces on Win32, Win64, Debug, Release, VCL, and/or FMX.

Field	Record
Defect ID / title	
Date / tester	
Studio executable	
Target executable	
Project / framework	
Locale / provider	
Steps to reproduce	
Expected result	
Actual result	
Attachments / logs	

15. Final Acceptance Checklist

Gate	Acceptance requirement	Result
Backup	Pre-change product backup exists and was verified.	PASS / FAIL
Clean target	Disposable GA4 copy began clean; pristine reference remained untouched.	PASS / FAIL
Release matrix	RunPhase10ReleaseValidation.ps1 passed uninterrupted.	PASS / FAIL

Gate	Acceptance requirement	Result
UI	All seven pages fit, navigate, and display professionally at tested resolutions.	PASS / FAIL
Scan	Project detection and scan are correct, fast, and read-only.	PASS / FAIL
Provider	Connection and automatic translation succeed; key remains secret.	PASS / FAIL
Catalog	Development JSON auto-saves and preserves status/provenance.	PASS / FAIL
Incremental update	Saved UI changes rescan correctly; completed translations survive; stale scans cannot export.	PASS / FAIL
Validation	Errors block export; corrected catalog passes.	PASS / FAIL
Runtime pack	Compact JSON exists under target Localization\Languages.	PASS / FAIL
Kit safety	Component kit is complete and target hashes/status do not change during generation.	PASS / FAIL
IDE integration	One manager and optional selector are designer-authored and reviewable.	PASS / FAIL
Builds	GA4 builds Win32 and Win64; Release variants tested as required.	PASS / FAIL
Instant switching	Open and new forms switch immediately and return to English.	PASS / FAIL
State	Date range, editable values, selection, focus, and indexes survive switching.	PASS / FAIL
Persistence	Selected locale survives a restart without startup exception.	PASS / FAIL
Offline	Target switches languages and restarts with no Internet or Studio.	PASS / FAIL
VCL	VCL delta and modeless-form boundary are accepted/documented.	PASS / FAIL / N/A
Documentation	Guide results and defects are complete enough to reproduce.	PASS / FAIL

Approval	Name	Signature	Date
Test lead			
Developer			
Product owner			

16. Appendix A - Exact Test Scripts

Script	Purpose
tools\tests\RunPhase10ReleaseValidation.ps1	Master supported release matrix.

Script	Purpose
tools\tests\RunLanguageManagerPackageTests.ps1	Debug/Release runtime/design packages and streaming/selector tests.
tools\tests\RunLanguageManagerCoreTests.ps1	Framework-neutral manager invariants.
tools\tests\RunFMXLanguageManagerTests.ps1	FMX lifecycle, switching, identities, and state.
tools\tests\RunVCLLanguageManagerTests.ps1	VCL lifecycle, modal/modeless boundary, switching, identities, and state.
tools\tests\RunRuntimeSmokeTests.ps1	Foundation, runtime, integration, build/deploy, and launch workflow.
tools\tests\RunStudioLaunchSmokeTests.ps1	Real Studio startup in all four configurations.
tools\tests\RunStudioSelfLocalizationSmokeTest.ps1	Italian Studio launch with preference/pack restoration.

17. Appendix B - What Each JSON File Is

File	Role	Deployment
Localization\Development\FMXPilot.<locale>.translation-project.json	Lossless working catalog with source text, translated text, keys, checksums, statuses, origins, locale formats, runtime classification, and review data.	Developer source workspace only.
Localization\Languages\<locale>.json	Compact, validated runtime pack with application identity, language metadata, locale settings, strings, and source catalog checksum.	Copy beside each executable.
export\component-integration\FMXPilot\component-integration.json	Generated kit manifest for application/framework/component/language/form-root setup.	Developer reference; not required by runtime manager.
%LOCALAPPDATA%\FMXPilot\language.ini	Per-user last-selected locale.	Created at runtime; never package as an application asset.

> **JSON is the product format.** CSV is optional translator interchange only. Automatic provider translation reads and updates the in-memory development catalog and saves the JSON catalog in place; the runtime consumes exported JSON packs.

18. Appendix C - Current Verified Baseline

As of August 11, 2026, commit 6ced725 had passed the complete Phase 10 release validation without Delphi running: current Studio executables in all four supported build folders, package outputs for Win32/Win64 Debug/Release, design streaming, component/lifecycle/runtime/integration tests, all four

Studio form-streaming and launch tests, and Studio self-localization. The clean pristine GA4 reference repository remained the comparison source; older automatic-source-integration test folders remained excluded from the clean component acceptance path.

This baseline is evidence, not a substitute for rerunning the tests. Record new executable timestamps, release-gate output, provider results, and target Git diffs for each acceptance session.