

Wizard-to-Runtime Complete Procedure

Delphi App Translation Studio



Current pre-release test workflow

Last changed: August 12, 2026

Windows - Delphi VCL and FireMonkey - Win32 and Win64

Printable, start-to-finish operator procedure

Every required transition is included: clean test copy, Wizard, in-Wizard localization review, automatic finalization, optional Studio verification, RAD Studio component placement, builds, deployment, and runtime testing.

Table of Contents

1. Read This Before Starting	1
2. Exact Product Files and Generated Locations	2
3. Prepare a Completely Clean Test	3
4. Start the Setup Wizard	4
5. Single Wizard Pass - Create, Review, and Finalize	4
6. Complete the In-Wizard Localization Review	7
7. Automatic Resume - Finalize Reviewed Output	8
8. Optional: Return to the Main Translation Studio	9
9. Install the Design Package in RAD Studio	11
10. Place and Configure the Components	11
11. Verify Search Path and Build Configuration	12
12. Build Win32 and Win64	13
13. Runtime Test - First Launch and Switching	13
14. Runtime Layout and Content Acceptance	14
15. Diagnose Missing, Random, or Runaway Text	15
16. Procedure After Later UI Changes	15
17. Adding More Languages	16
18. Final Completion Checklist	17
19. What to Send Back After Testing	18
20. Important Stop Conditions	18
21. Appendix A - License Information	20

1. Read This Before Starting

The Setup Wizard performs the first project scan, automatic provider translation, validation, JSON export, component-kit generation, Search Path configuration, and language-pack deployment configuration. It does not place components on a Delphi form; that remains a normal RAD Studio Form Designer operation.

The current workflow uses one protected Wizard processing pass. It creates the translated development catalog, automatically opens Localization Review, waits while the developer records terminology and layout decisions, and resumes automatically when the Review Center closes. The resumed pass applies those decisions before final validation, runtime JSON export, component-kit generation, Search Path configuration, and deployment configuration.

After the single Wizard pass, the main Studio may be used for optional detailed inspection or manual correction. RAD Studio is then used to install the design package, place the manager and selector, build the target, and test runtime behavior.

Do not use an original project. Create a new disposable test folder from the pristine copy. Keep the pristine folder and the original application untouched. The Wizard creates its own ZIP and DPROJ transaction backups, but those are additional safeguards, not substitutes for the pristine copy.

Current automatic-layout boundary. The runtime can apply accepted, checksum-backed Width, Height, WordWrap, and AutoSize rules from a language pack. It does not automatically move neighboring controls, redesign a form, change fonts, or guarantee that a wider control will not overlap another control. Any proposal left Pending, Rejected, or Manual is not applied automatically.

Current glossary boundary. An empty Project Glossary is normal before terms are approved. Built-in computer-interface terminology still operates. Catalog-derived suggestions appear after translations exist; only approved project terms become authoritative for that project and language.

1.1 What this test should prove

- [] The Wizard completes without an access violation or a stopped-processing message.
- [] The Wizard translates unresolved entries automatically with the selected API provider.
- [] The protected Wizard pass pauses for Localization Review before its definitive export.
- [] Closing Localization Review resumes the same Wizard pass and embeds accepted glossary/layout decisions.

- [] The main Studio opens the same catalog, validates it, and exports the final JSON pack.
- [] The target project contains only intentional designer changes plus the marked DPROJ configuration block.
- [] Win32 and Win64 builds receive the correct Localization\Languages folder.
- [] The target starts in the expected language, switches immediately, restores layout on switching back, and remembers the user's choice after restart.
- [] Dynamic application text does not produce repeated characters.

2. Exact Product Files and Generated Locations

Purpose	Path or pattern
Recommended Studio build	C:\DelphiProjects\Delphi App Translation\bin\Win32\Debug\DelphiAppTranslationStudio.exe
Win64 Studio build	C:\DelphiProjects\Delphi App Translation\bin\Win64\Debug\DelphiAppTranslationStudio.exe
FMX design package	C:\DelphiProjects\Delphi App Translation\bin\packages\Win32\Release\DATLanguageManagerFMXDesign.bpl
VCL design package	C:\DelphiProjects\Delphi App Translation\bin\packages\Win32\Release\DATLanguageManagerVCLDesign.bpl
Development catalog	<Test Project>\Localization\Development\<ApplicationId>.<language>.translation-project.json
Project glossary	<Test Project>\Localization\Glossaries\<ApplicationId>.<language>.glossary.json
Runtime language pack	<Test Project>\Localization\Languages\<language>.json
Component kit	C:\DelphiProjects\Delphi App Translation\export\component-integration\<ApplicationId>
Localization review	C:\DelphiProjects\Delphi App Translation\export\localization-review\<ApplicationId>\<language>
Saved language preference	%LOCALAPPDATA%\<ApplicationId>\language.ini
Wizard safety backup	%USERPROFILE%\Documents\Delphi App Translation Backups\<ApplicationId>\<timestamp>.zip

Application ID means project name, not folder name. For C:\...\ExampleApp.dproj the detected ApplicationId is ExampleApp. Always use the exact Application ID displayed by the Wizard; do not type the full path and do not include .dproj or .exe.

3. Prepare a Completely Clean Test

1. Close the target application executable if it is running.
2. Close the target project in RAD Studio. Saving is allowed before closing, but do not leave unsaved form changes.
3. Locate the pristine project folder. Confirm that it is the known-good unlocalized source.
4. Create a new sibling test folder by copying the entire pristine folder. Use a clear name such as <Application Name> - Translation Test.
5. Do not copy an earlier test folder. Earlier Localization folders, DPROJ Search Paths, components, and language preference files would invalidate the test.
6. Open the new test folder in File Explorer. Confirm the expected .dproj or .dpr file is present.
7. If the pristine copy contains old bin or dcu output folders, remove those only from the new disposable test copy or use Delphi Clean before the baseline build. Never clean the pristine source as part of this test.
8. Open the new test project's .dproj in RAD Studio, perform a baseline Win32 Debug build, and run it once in its original language.
9. Open every important form or page and confirm that the pristine application itself does not already contain mixed-language or runaway dynamic text.
10. Close the baseline executable. Choose File > Close All in RAD Studio so the target project is closed before Wizard final processing.

No Git requirement. Git is useful evidence but is not required. A pristine copy plus a fresh disposable test copy is an accepted safety baseline. If Git is available, record git status --short now; it should be empty.

3.1 Important: Clear a Previous Saved-Language Preference

A new project folder does not clear the per-user language preference because that file lives under Local AppData. If the same ApplicationId was tested previously, the new executable may immediately restore Spanish or another language even though the project folder is new.

1. Press Windows+R.
2. Enter %LOCALAPPDATA% and press Enter.
3. Open the folder whose name exactly matches the Wizard's Application ID, if it exists.
4. If language.ini exists, rename it to language.ini.pretest. Do not rename unrelated application settings.
5. If the Application ID folder does not exist, no preference cleanup is required.

4. Start the Setup Wizard

1. Run C:\DelphiProjects\Delphi App Translation\bin\Win32\Debug\DelphiAppTranslationStudio.exe.
2. Wait for the Delphi App Translation Studio main form to appear.
3. Click Start Setup Wizard on the Project page. Do not click Open Project for this initial path.
4. Confirm the dimmed Studio background is blocked by the modal Translation Setup Wizard.
5. Do not run a second Studio instance during this test.

5. Single Wizard Pass - Create, Review, and Finalize

5.1 Step 1 - Welcome

1. Read the safety statement. It explains that final processing creates a backup and writes localization/configuration files but does not rewrite Pascal or form source.
2. Click Next.

5.2 Step 2 - Delphi Project

1. Click Browse.
2. Navigate to the new disposable test folder, not the pristine folder and not the original project folder.
3. Select the target .dproj. Use the .dpr only when no .dproj exists.
4. Confirm the displayed project path is the disposable test folder.
5. Confirm the detected framework is FireMonkey or VCL as expected.
6. Confirm Win32 and/or Win64 platforms and the form-resource count are plausible.
7. Write down the detected Application ID. You will use this exact value in the Object Inspector later.
8. Leave the workflow on Automatic. For a new folder it should report Create a new translation.
9. Click Next.

5.3 Step 3 - Deployment Destinations

The Wizard detects ordinary Win32 and Win64 build-output folders automatically. Use this page only for separate installed, portable, network, or USB application folders. The choices remain staged until authorized final processing.

1. Leave the list empty when only normal Delphi build-output folders are needed.
2. For each separate application copy, click Add Application Folder and select the full folder containing—or intended to contain—the executable. Include the correct drive letter for USB or other non-C drives.
3. Do not select the Localization or Languages subfolder.
4. Remove any obsolete or incorrect destination before continuing.

5. Click Next. Cancel remains safe because the staged destinations are not saved until authorized final processing.

Automatic deployment. Final processing deploys immediately to every configured destination that is currently available. Future builds retry the remembered destinations automatically. An unplugged drive is skipped with a warning and does not fail the build.

5.4 Step 4 - Languages

1. Set Source language to the language in which the Delphi application was authored, normally English (United States) [en-US].
2. Select exactly one Target language for this pass, for example Spanish (Spain) [es-ES].
3. Confirm Native language name is correct and uses proper characters, for example Español.
4. Confirm Direction is Left-to-right unless the language requires right-to-left text.
5. Review the date, time, decimal, thousands, and currency values. These locale settings are stored in the JSON pack; correct only values you deliberately want the target application to use.
6. Confirm Automatic still reports Create a new translation, then click Next.

One target language per Wizard run. To add more languages, repeat the create/update workflow for each target language. The runtime selector later lists the source-language pack and every valid target-language JSON pack deployed beside that executable.

5.5 Step 5 - Translation Service

1. Choose Google Cloud Translation or DeepL.
2. For DeepL, choose API Free or API Pro to match the account. Google does not use the DeepL plan field.
3. If the provider key is already stored in Windows Credential Manager, leave the API key field blank and verify that the status reports a stored key.
4. If no key is stored, paste the API key into the masked field.
5. Leave Remember securely on this computer checked to store the key in Windows Credential Manager. Uncheck it only for a session-only key.
6. Click Save / Replace Key when a new key was entered.
7. Click Test Connection.
8. Do not continue until the status explicitly reports that the connection test passed.
9. Click Next.

The target application remains offline. Only the developer's Studio uses the provider and API key. The deployed target application reads local JSON packs and does not need the Internet or the API key.

5.6 Step 6 - Scan Project

1. Click Scan Project.
2. Wait for Scan complete. Do not click Next while the scan is running.
3. Record the total translatable entries and the new/changed/unchanged/obsolete counts.
4. Scroll through representative items. Confirm form text and resourcestring entries belong to the selected project.
5. Look specifically for suspicious repeated characters, data values, paths, filenames, URLs, IDs, or logging output. These should be protected or reviewed rather than translated as normal interface text.

Record	Value observed	Notes
Scan date and time		
Disposable test-project path		
Application ID		
Source and target languages		
Total translatable entries		
New / changed / unchanged / obsolete		
Suspicious or excluded text noted		

Rescan after source changes. If you edit or save any target PAS, FMX, DFM, DPR, or DPROJ file after this scan, return to this step and scan again before final processing.

6. Localization Review is intentionally unavailable at this point because layout analysis requires translated text. Continue with Next. During final processing, the Wizard translates the unresolved entries and then opens Localization Review automatically with terminology suggestions and applicable layout proposals.

5.7 Step 7 - Delphi Component

1. Read the project-specific instructions from top to bottom.
2. Confirm the instructions show the correct target project path and exact Application ID.

3. Click Show Design BPL. File Explorer should select DATLanguageManagerFMXDesign.bpl for an FMX target or DATLanguageManagerVCLDesign.bpl for a VCL target under bin\packages\Win32\Release.
4. Do not use Delphi's Install Component wizard. Do not select a .dpk. The later approved procedure is Component > Install Packages > Add and selection of the compiled design BPL.
5. Close File Explorer and return to the Setup Wizard.
6. Check Required: I understand the remaining manual RAD Studio phase.
7. Confirm the orange reminder changes to a confirmation message.
8. Click Next.

5.8 Step 8 - Review and Authorize

1. Read the project, Application ID, framework, target language, workflow, provider, scan count, unresolved count, integration method, deployment statement, and backup statement.
2. Confirm the required backup box is checked. It is intentionally mandatory.
3. Confirm RAD Studio has no target project open. If it does, close the project now and return to the Wizard.
4. Check Required: the target project is closed in RAD Studio.
5. Check I reviewed these choices and authorize final processing.
6. Click Begin Final Processing only after both confirmations are checked.
7. After processing begins, do not try to close the Wizard or Studio. Back, Cancel, and the step rail remain disabled. The Wizard will translate, open Localization Review, and automatically resume finalization after Review closes.

5.9 Step 9 - Processing and Completion

1. Watch the progress log until it stops changing.
2. Confirm a timestamped ZIP backup was created.
3. Confirm unresolved entries were translated by the selected provider or resolved by terminology/translation memory.
4. Confirm Development catalog saved appears.
5. Confirm the log reports the number of separate application destinations saved, and confirms deployment to each destination currently available.
6. Confirm the progress log reports that the required in-Wizard localization review is opening. The Translation Studio and Wizard remain protected while the modal Review Center is active.

6. Complete the In-Wizard Localization Review

1. Wait for the Localization Review Center to open automatically. Do not click Finish and do not start another Wizard.

2. On Audit & Confidence, read the summary and inspect all High risk findings and representative Warning findings.
3. Click Open Visual Review. The review package is generated automatically when the Review Center opens and is refreshed whenever a layout decision is saved. The browser shows the current translated geometry beside the proposed geometry; proposed changes are outlined in green. This preview does not alter a Delphi form.
4. Return to the Localization Review Center.
5. Open Terminology Suggestions. Select suggestions and read Source term, Suggested target, Context, Semantic concept, Provenance, Confidence, and Why suggested.
6. Use Approve Selected only for a term you understand and want enforced for this project and language.
7. Use Approve High-confidence All only after inspecting the proposed group. Provider output is not promoted automatically merely because it exists.
8. Open Project Glossary. Confirm approved terms now appear. If a required term is absent, click New, enter Source term and Preferred translation, optionally enter Semantic concept and Developer note, leave Approved terminology checked, click Add / Update Term, and click Save Project Glossary.
9. Open Layout Proposals. Select a proposal and read its form, control, property, current value, proposed value, reason, decision, and What happens explanation. Use Open Visual Review before accepting a group so the proposed result can be judged as a screen, not merely as numbers.
10. For the ordinary test, click Accept All Safe Proposals only after understanding that Width, Height, WordWrap, and AutoSize rules will be applied at runtime for this language.
11. For any proposal that would collide with another control or exceed its parent, choose Manual or Rejected and click Save.
12. Click Close. This records the end of the review phase and returns control to the protected Wizard processing pass.

Do not start a second Wizard pass. Closing Localization Review now resumes the original processing pass. The Wizard applies the saved glossary and accepted layout rules before it performs the definitive validation, runtime-pack export, and component-kit generation.

7. Automatic Resume - Finalize Reviewed Output

1. After the Review Center closes, do not press anything while the progress log resumes automatically.
2. Confirm the log reports Localization review closed and reviewed decisions being applied.
3. Confirm Reviewed development catalog saved appears.
4. Confirm the progress log lists the final localization review, layout proposal, and multilingual layout envelope paths. In File Explorer, open C:\DelphiProjects\Delphi App Translation\export\localization-

- review\<ApplicationId>\<language> and verify that localization-review.html, layout-proposal.json, and multilingual-layout-envelope.json exist and are not zero bytes.
5. Confirm validation passed. Warnings may remain; blocking errors must not remain.
 6. Confirm Runtime JSON pack exported appears with a path under the disposable test project's Localization\Languages folder.
 7. Confirm Component integration kit generated appears.
 8. Confirm Project Search Path and automatic post-build deployment configured appears.
 9. Confirm the footer says Setup Wizard completed successfully. If it says stopped, do not continue to RAD Studio. Copy or photograph the entire progress log and footer message, close the Wizard only after processing has stopped safely, preserve the Wizard ZIP backup and disposable test folder, and send the exact STOPPED text and screenshots to the development team. Resume only after the cause is corrected and new test instructions are provided.
 10. Click the underlined blue component-kit path or Open Kit Folder. Confirm ComponentSource, Localization, component-integration.json, Deploy-LanguagePacks.ps1, README.txt, and Wizard-Completion-Report.txt exist.
 11. Read the repeat/troubleshooting explanation. No command or deployment button is required during a normal successful run. Final processing has already deployed detected build outputs and every available application destination entered earlier.
 12. Confirm the completion report states that review was completed inside the same Wizard processing pass before final export.
 13. Click Finish once. The Wizard closes and returns to the Studio main form.

8. Optional: Return to the Main Translation Studio

The single Wizard pass has already translated, reviewed, validated, exported, and refreshed the project. If it completed successfully, validation reported no errors, and no additional translation, glossary, locale, or layout correction is needed, Section 8 is optional. You may stop using the Translation Studio here and proceed directly to Section 9 for the required RAD Studio installation and component-placement work.

Use Section 8 when you want detailed catalog inspection, a manual translation correction, another validation report, or a final runtime-pack and component-kit refresh. Do not start a second translation from scratch. Any correction made here must be saved, validated, exported, and followed by the component-kit refresh in Section 8.4 before continuing to RAD Studio.

8.1 Open the matching project and catalog

1. On the Studio left rail, click 1 Project.
2. Click Open Project and select the same disposable test .dproj.
3. Click 3 Translate.

4. Click Open.
5. Open the development catalog from <Test Project>\Localization\Development. Its name is <ApplicationId>.<language>.translation-project.json.
6. Confirm the target language, native language name, locale formats, entry count, translated count, status, and origin fields are populated.
7. Click 2 Scan, then click Scan Project. This proves that the catalog still matches the saved Delphi sources.
8. Return to 3 Translate. Confirm reviewed/approved translations and existing provider results were preserved.

8.2 Inspect and correct translations

1. Select entries with short or ambiguous interface text, unknown context, provider-basic confidence, or suspicious ownership.
2. Read Source text, the context hint, runtime role, origin, and Translated text.
3. Correct an inaccurate translation in Translated text and click Apply Translation. This changes the development JSON catalog, not Delphi source.
4. Use Mark Reviewed only after a human has actually reviewed that entry. Use Approve only after it is already Reviewed.
5. Review All and Approve All are catalog-wide decisions. Do not use them merely to remove warnings; use them only when one qualified review decision truly applies to the entire displayed group.
6. Click Save after completing edits. Confirm Development catalog saved.
7. Click Translate Automatically only if the readiness line or confirmation dialog reports unresolved entries. The command sends only eligible unresolved entries; reviewed and approved entries remain unchanged.

8.3 Validate and export the final runtime pack

1. Click 4 Validation.
2. Click Run Validation.
3. If errors are reported, double-click each error, correct the referenced translation or setting, save, and run validation again. Do not export with errors.
4. Warnings do not block export. Review warnings about identical source/translation, placeholders, accelerator counts, runtime Translate calls, and suspicious text; document why any accepted warning is safe.
5. When the summary reports 0 errors, click 5 Export.
6. Click Export Runtime Pack.
7. Confirm the runtime entry count and click the blue output path.

8. In File Explorer, confirm the final <language>.json exists under the disposable test project's Localization\Languages folder.
9. Open the JSON in a text-safe viewer. Confirm schemaVersion is 3 and confirm a layout array is present when safe layout proposals were accepted.

8.4 Refresh the component kit after the final export

1. Click 6 Integration.
2. Confirm Integration method is Component Integration (Recommended).
3. Click Build Integration Plan.
4. Confirm the target framework, translated pack count, generated English pack statement, manager class, and selector class are correct.
5. Click Generate Component Kit.
6. Click Open Kit Folder and confirm the kit was refreshed. This step is important because the kit must contain the same final runtime JSON pack that you just exported.
7. Click Show Design BPL and leave File Explorer open with the exact Win32 Release design BPL selected.

9. Install the Design Package in RAD Studio

Manual installation is intentional. The product does not automatically register a design-time BPL. Manual installation through RAD Studio's approved package dialog avoids changing the IDE behind the developer's back.

1. Start RAD Studio without opening the target project or target form.
2. Choose Component > Install Packages.
3. If DAT Language Manager FireMonkey design-time package or the VCL equivalent is already listed and checked, do not add a duplicate. Click OK and continue to the next section.
4. Otherwise click Add.
5. Browse to the exact BPL selected by Show Design BPL: DATLanguageManagerFMXDesign.bpl for FMX or DATLanguageManagerVCLDesign.bpl for VCL.
6. Select the .bpl and click Open. Do not choose a .dpk and do not use Component > Install Component.
7. Confirm the DAT design-time package appears in the Design packages list and is checked.
8. Click OK.
9. Create or open a harmless blank form if necessary and confirm the Tool Palette contains DAT Localization with both the language manager and language combo box.

10. Place and Configure the Components

1. Open the disposable target project's .dproj in RAD Studio.
2. Open the primary form in the Form Designer. If Delphi opened it automatically, simply confirm that the primary form is the active designer.
3. From DAT Localization, place one TDATEFMXLanguageManager for FMX or TDATEVCLLanguageManager for VCL on the primary form.
4. Select the manager in the Object Inspector.
5. Set ApplicationId to the exact value displayed by the Wizard.
6. Leave LanguagesFolder as Localization\Languages.
7. Set SourceLanguage to the source language code used by the Wizard, normally en-US.
8. For a clean first-launch test, set AutoLoadPreferred to False. This forces the source language initially. After first-launch behavior passes, set it back to True to test saved-user preference behavior.
9. Leave AutoTranslateOwner, AutoTranslateNewForms, ReapplyOpenForms, and PreserveControlState enabled unless this test intentionally exercises another setting.
10. Place one TDATEFMXLanguageComboBox or TDATEVCLLanguageComboBox on the visible primary form. The selector is required unless the application supplies a connected Language menu.
11. Select the combo box. In the Object Inspector, set LanguageManager to the manager component you just placed. Do not leave it blank.
12. Position and size the combo box where it is visible and does not cover existing controls. Add a normal designer-authored label such as Language: if the application needs one.
13. Choose File > Save All. This save is essential; it writes the two designer components and their properties to the form resource and updates the project metadata as required by Delphi.

One manager, not one per form. Place the manager on the primary form only. It observes and translates other open or newly created forms. Ordinary secondary forms do not need another manager component.

11. Verify Search Path and Build Configuration

1. Choose Project > Options.
2. Select Building > Delphi Compiler > Search path.
3. Inspect Value from All configurations. It should contain the generated kit's ComponentSource folder under C:\DelphiProjects\Delphi App Translation\export\component-integration\<ApplicationId>\ComponentSource.
4. Use the Target selector to inspect Debug/Release and Windows 32-bit/64-bit as applicable. The Wizard-created DPROJ block is intended to cover all configurations and platforms.

5. If the path is absent, stop. Do not manually invent a different path during this acceptance test; record the missing configuration because the Wizard was supposed to add it.
6. Click Cancel in Project Options when no manual change is necessary.

12. Build Win32 and Win64

1. Select Win32 and Debug in Project Manager, then choose Project > Build <ApplicationId>.
2. Confirm the build completes with zero fatal errors.
3. Locate the Win32 Debug executable using Project > Options > Building > Delphi Compiler > Output directory if the project uses a nonstandard path.
4. Beside the executable, confirm Localization\Languages exists and contains en-US.json plus the target-language JSON file.
5. Repeat for Win64 Debug.
6. If release testing is required, repeat for Win32 Release and Win64 Release.
7. After every build, verify the executable's own folder contains Localization\Languages. The project-root Localization folder alone is not sufficient for runtime discovery.

The .rsm file is normal. A Delphi build may create both <ApplicationId>.exe and <ApplicationId>.rsm. The .exe is the application. The .rsm contains debug symbol information and is not a language pack.

12.1 Manual deployment fallback

Use this only if automatic post-build deployment did not create the language folder. Substitute the actual kit and executable folder paths shown by the Wizard. The command explicitly uses ExecutionPolicy Bypass.

```
& "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -ExecutionPolicy
Bypass -File "C:\DelphiProjects\Delphi App Translation\export\component-
integration\<ApplicationId>\Deploy-LanguagePacks.ps1" -ApplicationDirectory "<Folder
containing ApplicationId.exe>"
```

A successful command reports Language packs deployed to <application folder>\Localization\Languages. The PowerShell process should exit when the command finishes.

12.2 Portable or USB executable

1. Before final processing, enter the desired portable folder or USB drive on the Wizard's Deployment Destinations page, including the drive letter such as F:\PortableApp.
2. Copy or build the final executable into that configured folder. Final processing and future builds deploy its packs automatically whenever the destination is available.
3. Confirm <Portable Folder>\Localization\Languages contains en-US.json and every target-language JSON file.

4. Use Deploy New App Folder or the fallback command only for a new, temporary, or troubleshooting destination that was not configured earlier.
5. Keep the manager's LanguagesFolder property relative as Localization\Languages. Do not put F: or another drive letter in the component property.

13. Runtime Test - First Launch and Switching

1. Confirm the target application is closed.
2. Confirm AutoLoadPreferred is False for this first-launch test, or confirm the prior language.ini was renamed as described in Section 3.1.
3. Run the Win32 Debug executable directly from its output folder.
4. Confirm the application initially displays its source language and the selector shows the source language.
5. Open the selector. Confirm it contains the source language plus one item for every deployed valid target pack. Fifteen target packs should produce sixteen choices when the source pack is included.
6. Select the target language.
7. Confirm visible designer text changes immediately without restarting the application.
8. Open each secondary form and visit each page or tab. Confirm forms created after the selection also appear in the active language.
9. Return to the primary form and select the source language.
10. Confirm source text and original layout values are restored.
11. Repeat source-to-target-to-source switching at least ten times. Confirm controls do not grow cumulatively and the application does not lose dates, selections, focus, connection state, or editable values.
12. Close the executable.
13. Set AutoLoadPreferred back to True in the Object Inspector, Save All, and rebuild the configuration.
14. Run the rebuilt executable, select the target language, close it, and run it again.
15. Confirm it restores the user's last selected language. This is expected saved-preference behavior, not corruption of the source application.
16. Repeat representative switching and restart checks with Win64 Debug and required Release builds.

14. Runtime Layout and Content Acceptance

14.1 What accepted layout rules should do

- [] Accepted Width or Height rules apply only when their target language is active.
- [] Accepted WordWrap or AutoSize rules apply only to the identified control and language.

- [] Long label and check-box text may use WordWrap with a calculated Height instead of uncontrolled horizontal growth.
- [] Grid column proposals enlarge translated headings conservatively without resizing the entire grid beyond its parent.
- [] Switching away restores the original property value before another language is applied.
- [] Repeated language changes do not compound Width or Height.
- [] No target PAS, FMX, DFM, DPR, or DPROJ form layout is rewritten by a runtime layout rule.

14.2 What still requires human inspection

- [] A wider translated control does not overlap its neighbor or leave its parent container.
- [] Wrapped labels have enough height and remain aligned with related controls.
- [] Buttons, tabs, menu items, column headers, charts, grids, and status areas remain readable.
- [] Dynamic text produced by application code uses Translate or FormatTemplate where required.
- [] The translation is correct in application context, not merely a valid dictionary translation.
- [] Form titles, secondary forms, help text, and runtime status messages have each been considered separately.

15. Diagnose Missing, Random, or Runaway Text

1. Record the form, page, control, active language, exact displayed text, and a screenshot.
2. Switch to the source language. Determine whether the same bad text exists in the pristine baseline application.
3. If the problem exists in the pristine baseline, classify it as a target-application defect rather than a translation defect.
4. If the source language is correct but the target language is wrong, search the development catalog for the control key or source text.
5. If the entry exists, inspect its source ownership, runtime role, context, translation, and status. Correct the translation or runtime classification as appropriate.
6. If the entry does not exist, save all target files, close the target project, reopen it in the Studio, scan again, and record whether the scan count increases.
7. For runtime-generated text, verify that the target application uses the manager's Translate or FormatTemplate API with the catalog key. Static form scanning cannot translate arbitrary text assembled later by application code.
8. For repeated o characters or other actively growing output, stop the executable. Record whether the source-language run also reproduces it. A JSON translation pack supplies a fixed string;

unbounded growth usually indicates an application timer, refresh, append, or data-writing loop and must be isolated separately.

9. After any catalog correction, Save, Run Validation, Export Runtime Pack, regenerate the component kit, rebuild/deploy, and retest.

16. Procedure After Later UI Changes

Batch changes when practical. Add or revise labels, buttons, menus, headings, hints, and resourcestrings in one saved batch when possible. This reduces repeated scans and provider calls, but the same procedure works for one change.

1. In RAD Studio, make the designer or source changes in the target application.
2. Choose File > Save All.
3. Close the running target executable.
4. Close the target project in RAD Studio before any Wizard final-processing pass.
5. Open the project in the Translation Studio.
6. Open the existing development catalog before scanning.
7. Run Scan Project.
8. Confirm new entries are New, revised source text is Source Changed, unchanged completed translations remain preserved, and removed items become Obsolete.
9. Run Translate Automatically. Confirm only eligible unresolved/new/changed entries are offered to the provider.
10. Review the new translation and any new terminology/layout proposal.
11. Save the catalog, run validation, and resolve all errors.
12. Export Runtime Pack.
13. Regenerate the Component Kit so its Localization folder contains the final pack.
14. Build the target; confirm post-build deployment refreshes Localization\Languages beside the executable.
15. Run the target and retest the changed form in the source and target language.

17. Adding More Languages

Use this procedure when the application already has working localization and you want to add one new target language. Do not recreate the project from its pristine copy, remove existing language packs, reinstall the design package, or place additional localization components. One Wizard run adds one target language while preserving the languages already present.

1. Open the existing approved working project in the Translation Studio. Keep its current Localization folder, language packs, project glossary, language manager, and connected selector intact.
2. Start the Setup Wizard and select the same project file used for the existing translation. Confirm that the detected Application ID exactly matches the ApplicationId already assigned to the language manager.
3. Choose the same source language used by the existing packs. In Target language, choose only the new language being added.
4. Leave Workflow set to Automatic (Recommended). Because this target language does not yet have a development catalog, the Wizard should report that it will create a new translation. Existing catalogs and runtime packs for other languages remain unchanged.
5. Confirm the translation provider and test its connection. Scan the project and review the scan total before continuing.
6. Complete the component-information and authorization steps, then click Begin Final Processing.
7. When Localization Review opens automatically, review terminology and layout proposals for the new language. Save the desired decisions and click Close.
8. Wait while the same Wizard pass resumes automatically. Confirm that final validation passes, the new runtime JSON pack is exported, the component kit is refreshed, deployment is configured, and the footer reports successful completion.
9. Click Finish. A separate Studio validation/export or second Wizard pass is not required when this single-pass completion succeeds. Use the main Studio only if a manual correction or detailed inspection is needed.
10. Do not reinstall the design package and do not place another manager or selector. The components already in the target application serve every language pack that shares the same Application ID.
11. Build the required Win32 and Win64 configurations. Confirm the new language JSON file appears under Localization\Languages beside each executable and in every available application destination configured on the Wizard Deployment page.
12. Run the application. Open the existing language selector and confirm the new language appears alongside the source language and all earlier target languages.
13. Select the new language and test its translation, layout, secondary forms, source-language restoration, repeated switching, and saved preference independently. Then verify that previously installed languages still work.

18. Final Completion Checklist

- [] Disposable test folder was copied from pristine and the pristine/original folders remained untouched.
- [] The single Wizard pass translated unresolved entries and opened Localization Review before final export.

- [] Localization audit, terminology suggestions, glossary, and layout proposals were reviewed.
- [] The same Wizard pass resumed automatically and applied saved review decisions.
- [] If an optional Main Studio correction was made, the matching catalog was saved, validated with zero errors, exported, and followed by a component-kit refresh.
- [] If no optional Studio correction was needed, the single-pass Wizard's final runtime pack and refreshed component kit were used unchanged.
- [] Correct Win32 Release design BPL was installed through Component > Install Packages > Add.
- [] One manager and one connected visible selector were placed and saved on the primary form.
- [] ApplicationId, LanguagesFolder, SourceLanguage, and LanguageManager properties were verified in Object Inspector.
- [] Search Path contains the generated ComponentSource folder for all required configurations/platforms.
- [] Win32 and Win64 builds completed.
- [] Each executable folder contains Localization\Languages with en-US.json and all target packs.
- [] First launch used the source language under the controlled test condition.
- [] Immediate switching, secondary forms, source restoration, repeated switching, and restart preference passed.
- [] Accepted safe layout rules worked without cumulative growth or unacceptable overlap.
- [] No missing/random translation or runaway dynamic-text defect remains unexplained.

19. What to Send Back After Testing

- Disposable test-folder path and selected .dproj path.
- Application ID, framework, source language, target language, and provider.
- Single-pass scan total, unresolved count, and automatic-resume result.
- Wizard completion log or the exact STOPPED message.
- Validation error/warning/information counts.
- Number of glossary terms approved and layout proposals accepted/manual/rejected.
- Whether the final runtime JSON has schemaVersion 3 and a layout array.
- Win32/Win64 build results and exact executable/output folders.
- Contents of each executable's Localization\Languages folder.
- First-launch language, selector choices, restart language, and ten-cycle switching result.

- Screenshots and exact control/form names for every untranslated, mistranslated, truncated, overlapping, or runaway item.

20. Important Stop Conditions

- Stop if the selected project path is the pristine or original application.
- Stop if Wizard final processing reports STOPPED or validation reports blocking errors.
- Stop if the design package fails to load; do not remove unrelated Delphi packages.
- Stop if the Wizard Search Path is absent rather than masking the defect with an unrelated global path.
- Stop if runtime packs beside the executable do not match the Application ID or selected language.
- Stop a target executable that generates unbounded repeated text; preserve evidence before restarting.
- Do not distribute or publish this pre-release test output as a production localization solution.

21. 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.