

DELPHI APP TRANSLATION STUDIO

Group 1 Runtime Lifecycle and State Ownership

Implementation and Validation Report

Status Complete - automated validation passed

Last changed August 9, 2026

Safety boundary No target-application source was changed

Static text is localized automatically. Live state remains owned by the application.

1. Outcome

Group 1 establishes an explicit contract between text extracted by the Studio and text that the installed language manager may write back to a running application. The runtime no longer assumes that every scanned `Text` or `Caption` property is a static interface caption.

The implementation separates static interface text from live application state, data values, identifiers, and runtime-generated messages. Only text classified as static is included in the automatic form-application dictionary. Runtime messages are placed in a separate keyed-template dictionary. Data and identifiers are omitted from runtime application entirely.

The FMX lifecycle was also made deterministic at first display. A form can receive its normal startup assignments and the manager then performs one safe forced static-text pass at `TFormBeforeShownMessage`. The pass preserves editable values, selection, focus, combo-box selection, and event behavior. Because dynamic and data entries are no longer present in the automatic dictionary, the pass cannot restore stale design-time values into those controls.

2. Runtime ownership model

Every development-catalog entry now has a `runtimeTextRole` in addition to its existing `runtimeApplication` value.

Runtime text role	Meaning	Runtime disposition
<code>staticText</code>	Designer-authored interface wording such as headings, button text, menu text, and prompts	Exported to <code>strings</code> ; applied automatically to forms
<code>dynamicValue</code>	Human-readable live state such as connection, refresh, progress, or result text	Exported to <code>templates</code> ; host code requests it by key
<code>runtimeTemplate</code>	A keyed message containing replaceable values or a Pascal resourcestring	Exported to <code>templates</code> ; host code uses <code>Translate</code> or <code>FormatTemplate</code>
<code>dataValue</code>	Application, user, metric, date, count, or service data	Not exported and never written by localization
<code>identifier</code>	URL, email address, path, code, or other technical identity	Not exported and never written by localization
<code>excluded</code>	Placeholder or deliberately non-translatable content	Not exported and never written by localization

The corresponding runtime application modes are:

- `automatic` for `staticText`;
- `manualTranslateText` for `dynamicValue` and `runtimeTemplate`;
- `notApplied` for `dataValue`, `identifier`, and `excluded`.

The development catalog schema is now version 4. Older catalogs load through a compatibility migration. A subsequent scan applies the new classifications to current form entries while preserving intentional exclusions wherever the prior role was already translatable.

3. Scanner policy

The form scanner applies conservative ownership rules based on property type, component identity, and source value.

Static labels, button captions, menu items, hints, prompts, and list items remain automatic. Known live-value naming patterns such as status, activity, message, result, progress, and last-updated controls are classified as dynamic. Value, count, total, data-value, and metric-value controls are protected as data. URLs and email-like values are protected as identifiers. Common runtime placeholders such as `-`, `--`, `---`, and `N/A` are excluded.

Pascal resourcestring entries are classified as runtime templates. This preserves their translation while making the ownership rule honest: the component cannot replace a Pascal expression automatically, so the application requests the translated text through the manager API.

These rules are intentionally conservative. Group 2 can expand extraction coverage after this ownership boundary has been exercised in the pilot.

4. Runtime-pack contract

The runtime pack remains compatible with schema version 1 and retains the existing `strings` object. A new optional `templates` object has been added.

- `strings` contains only automatically applicable static interface text.
- `templates` contains translated dynamic messages and runtime templates.
- `data`, `identifiers`, and `excluded` content are absent.

Existing runtime packs without `templates` continue to load. Existing applications using automatic form application remain compatible. The general `Translate` API can read either static text or templates; automatic FMX and VCL applicators continue to read only `strings`.

For variable messages, the component API now provides:

```
StatusLabel.Text := DATFMXLanguageManager1.FormatTemplate(
  'MainForm.StatusMessage',
  'Updated at %s',
  [FormatDateTime('t', Now, DATFMXLanguageManager1.CurrentFormatSettings)]);
```

This pattern keeps the dynamic value under application ownership while moving the human-language sentence into the language pack.

5. FMX lifecycle correction

The FMX manager still subscribes to FireMonkey lifecycle messages without taking ownership of the form's event handlers. The before-show handler now invokes a deliberate forced reapplication for that form.

The sequence is:

1. Load and validate the saved language preference.
2. Stream and create the FMX form normally.
3. Allow application startup code to initialize controls and data.
4. At `TFormBeforeShownMessage`, reapply only static pack entries.
5. Display the form with localized static text and intact runtime state.

Ordinary generation tracking remains in place for normal application passes. The forced path is limited to the FMX before-show boundary and does not advance the language generation.

6. Studio behavior

The Studio now records and displays the runtime text role with the runtime application mode and translation origin. Automatic provider translation and readiness counts operate only on roles that require translation. Validation does not demand translations for data, identifiers, or excluded content.

Manual runtime entries produce a non-blocking warning until their host application wiring is confirmed. The warning now identifies both supported approaches: `Translate` and `FormatTemplate`.

CSV remains an optional interchange format rather than a runtime language pack. It now carries both protected runtime fields so an import cannot silently alter the ownership classification. JSON remains the authoritative development-catalog and runtime-pack format.

7. Files changed

The implementation changed the following source areas:

- catalog types, JSON persistence, CSV protection, and schemas;
- scan types, scan rules, form scanning, resourcestring scanning, and catalog merge;
- validation and automatic-translation eligibility;
- runtime-pack generation and runtime-pack loading;
- shared runtime and component translation/template APIs;

- FMX before-show lifecycle handling;
- Studio catalog-status presentation;
- foundation and FMX regression tests.

No `.dpr`, `.dproj`, `.pas`, `.fmx`, or localization file in the FireMonkey pilot was modified.

8. Automated validation evidence

The following validation completed successfully on August 9, 2026:

Validation	Win32	Win64
Foundation scanner/catalog/runtime tests	Pass	Pass
Shared language-manager core tests	Pass	Pass
FMX lifecycle and control-state tests	Pass	Pass
VCL compatibility and control-state tests	Pass	Pass
Component package and design-streaming tests, Debug	Pass	Pass
Component package and design-streaming tests, Release	Pass	Pass
Studio Debug build and launch	Pass	Pass
Studio Release build and launch	Pass	Pass

The FMX suite now explicitly proves all of the following:

- a form translated early can receive later startup assignments and still be correctly localized before first display;
- instant language switching preserves combo-box selection;
- writable edit and memo data are preserved;
- selection and focus are preserved;
- localization does not fire protected control change events;
- hidden forms update at their next before-show notification;
- released forms are removed from manager tracking.

9. Developer validation for the the FireMonkey pilot pilot

Group 1 did not mutate the pilot. The developer's next controlled test begins from the existing clean component-test folder.

1. Open the newly built Studio executable, preferably `bin\Win32\Debug\DelphiAppTranslationStudio.exe` for diagnostic testing.
2. Open `C:\Projects\FMXPilot - Component Test\FMXPilot.dproj` in the Studio.
3. Scan the project again so the development catalog is upgraded and the new ownership roles are assigned.
4. Open the Spanish catalog and inspect representative entries:
 - ordinary captions should show `staticText` and `automatic`;
 - live status labels should show `dynamicValue` and `manualTranslateText`;
 - metric/value labels and placeholders should show `dataValue` or `excluded` and `notApplied`.
5. Translate any newly eligible entries, save, validate, and export the JSON runtime pack.
6. Generate a fresh component integration kit. Do not reuse the earlier exported kit because it contains the previous runtime sources.
7. Replace only the previously installed test package/source deployment using the documented manual package procedure.
8. Deploy the new language packs beside both Debug executables.
9. Build the FireMonkey pilot for Win32 and Win64.
10. Select Spanish and verify that live connection state, dates, counts, metrics, and user data do not revert to designer placeholders.
11. Close and restart while Spanish is saved, then verify that static text is Spanish on first display and live state remains accurate.
12. Record any human-language runtime messages that remain English. Those entries belong to Group 2 template coverage and should not be forced into automatic form application.

10. Exit decision

Group 1 meets its implementation exit criteria in automated testing:

- runtime ownership classifications are executable and persisted;
- automatic reapplication cannot consume entries classified as dynamic, data, identifiers, or excluded content;
- keyed runtime templates are supported and backward compatible;
- FMX first-display localization is deterministic;
- Win32 and Win64 builds and component packages pass;
- VCL compatibility remains intact;
- no target application was changed.

The final Group 1 acceptance decision should be made after the developer completes the the FireMonkey pilot pilot steps above. If the pilot confirms that dynamic values remain intact, work can proceed to Group 2 extraction coverage and keyed-message registration.

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.