Upgrading Solutions Reference

Upgrade solutions across platform versions.

Upgrading Solutions FactoryStudio Upgrade: Technical documentation for upgrading legacy projects to FrameworX version 10.

Version 10.1 or newer represents a major advancement with simplified concepts and new technologies. While most legacy projects upgrade smoothly, 100% compatibility cannot be guaranteed. All upgraded projects require validation in a lab environment before production deployment.

Upgrade Requirements
Upgrade Process
Automatic backup on every upgrade (10.1.5+)
If the upgrade fails (10.1.5+)
If the backup cannot be written (10.1.6+)
Warnings logged during a successful upgrade (10.1.5+)
Command-line upgrade — UpgradeSolution.exe (10.1.5+)
Manual Corrections Required
Asynchronous Methods
Additional Corrections
Deprecated Features
Validation Checklist
Best Practices Checklist

Upgrade Requirements

Version Compatibility

Source Version	Direct Upgrade	Process
8.1, 9.1, 9.2	Yes	Direct upgrade to 10.1 or newer
Earlier than 8.1	No	First upgrade to 8.1/9.1/9.2, then to 10.1 or newer

Important Notice

All projects must be revalidated in a lab environment before field deployment. This step is essential to confirm optimal functionality in the new version.

Upgrade Process

Steps

1. Place `<project>.tproj` file in SolutionCenter folder
2. Press Refresh in Solution List
3. Legacy project appears with "Project." prefix
4. Select project and click "Upgrade Version"
5. System creates `.dbsln` file with same name
6. Original `.tproj` file remains intact
7. Designer prompts for Build on first opening

File Handling

Action	Result
Original file	Preserved unchanged
New solution	Created as `.dbsln`
Solution List	Shows only new `.dbsln`
Legacy file	Hidden from list

Automatic backup on every upgrade (10.1.5+)

Starting with 10.1.5, every upgrade creates a preserved backup of the original solution file next to the solution itself. You do not need to copy the file manually before running the upgrade. The backup filename encodes the source version.

For example, upgrading `Plant.dbsln` from an earlier 10.1.x release leaves your solution folder containing:

File	Content
`Plant.dbsln`	The upgraded solution. Open this one in the Designer.
`Plant_fs-10.1.dbsln`	Preserved snapshot of the pre-upgrade solution. Safe to delete once the upgrade is validated in your lab environment.

Legacy `.tproj` upgrades continue to behave the same as in earlier releases: the original `.tproj` file stays on disk untouched and acts as its own rollback copy.

If the upgrade fails (10.1.5+)

When an upgrade does not complete successfully, 10.1.5 guarantees:

- Your original solution file is left exactly as it was. The upgrade process never deletes or overwrites the source on failure.
- The backup copy created at the start of the run is also kept on disk, even though the upgrade did not finish. You can re-run the upgrade after fixing whatever condition caused the failure.

The Solution Center displays a failure dialog that includes the full path to the upgrade log file (typically `%PUBLIC%\Documents\FrameworX\TraceLogs\UpgradeLog.txt`). Open the log to see the exception detail that caused the failure.

If the backup cannot be written (10.1.6+)

From FrameworX 10.1.6, opening a solution from an earlier version always saves a backup copy of the file before the upgrade. If FrameworX fails to write the backup copy, for example because the disk is full or the resulting path is too long, it refuses to open the solution instead of upgrading it without a safety copy. The message names the solution, the backup file and the reason.

- Updating a solution file from the command line or from the AI Console reports the same refusal and leaves the file unchanged.
- A solution in a read-only folder, or a write-protected solution file, opens read-only and is not upgraded.
- Opening a solution already on the current version, or opening it again after a successful backup, works as before.

> **Changed in FrameworX 10.1.6**
>
> A solution from an earlier version no longer opens when its backup copy cannot be written. If the message reports a backup failure, free disk space, shorten the folder path, or move the solution to a writable folder, then open it again. A solution opened read-only is not upgraded. To upgrade it, make the file and its folder writable and open it again.

Warnings logged during a successful upgrade (10.1.5+)

An upgrade can complete successfully while still logging individual non-fatal warnings — for example, a single table cell whose legacy value cannot be mapped to the new schema. When this happens, the Solution Center displays a count on the *Upgrade Concluded* dialog:

> N exception(s) logged during upgrade. Open log file

Click **Open log file** to launch the upgrade log in your default text editor. Each logged entry includes:

- The upgrade step that produced the warning (for example, `CopyMembers cell convert [DisplaysListContents.Xaml]`).
- The full exception message and stack trace.

Review the logged warnings as part of your pre-production validation. Most are informational — a row or cell that could not be automatically migrated — and can be resolved by hand in the Designer after the upgrade.

> Solutions that upgraded silently on 10.1.4 and earlier may show a non-zero warning count on 10.1.5. This is a visibility change, not a behavioral one — the same legacy data condition was being silently skipped before. The upgrade still succeeds in the same scenarios it did before.

Command-line upgrade — UpgradeSolution.exe (10.1.5+)

The headless upgrade utility `UpgradeSolution.exe` (shipped in the FrameworX program folder) prints exception details directly to the console instead of surfacing only the aggregate count. Typical output:

```
> UpgradeSolution.exe "C:\Projects\Plant.dbsln"
Upgrading: C:\Projects\Plant.dbsln
Previous version: fs-10.1.4-12345
Backup file: C:\Projects\Plant_fs-10.1.dbsln
[  5%] Preparing table 'UserTypesContents'...
[ 78%] Copying source tables to target...
Upgrade finished.
Exceptions during upgrade: 1

--- UpgradeLog.txt (this run) ---
2026-04-21 10:32:14 ****** Upgrade started: C:\Projects\Plant.dbsln ******
2026-04-21 10:32:16 Exception on CopyMembers cell convert [DisplaysListContents.Xaml]: ...
2026-04-21 10:35:02 ****** Upgrade finished: C:\Projects\Plant.dbsln ******
--- end of log ---
Full log: C:\Users\Public\Documents\FrameworX\TraceLogs\UpgradeLog.txt
```

Exit codes:

Code	Meaning
`0`	Successful upgrade. Non-fatal warnings may have been logged; check the tail output.
`1`	Fatal upgrade failure. Original file is preserved; backup file is preserved.
`2`	Invalid command-line arguments.

Manual Corrections Required

RuntimeUsers Database

Issue: Table name change from `EditSecurityUsers_Runtime` to `SecurityRuntimeUsers`

Required Action:

1. Rename table in target database
2. Update from `EditSecurityUsers_Runtime`
3. Change to `SecurityRuntimeUsers`
4. Complete before production deployment

Advanced Toolkit Applications

Applications using programmatic engineering (EngWrapper API) may lack compatibility due to internal structure changes. Contact support for upgrade assistance.

Themes

Version	Approach	Migration
9.2	Complex customization	Manual review required
10	Simplified interface	Cannot fully automate

Review screens and objects mapped to theme resources after upgrade.

Asynchronous Methods

Version 10.1 or newer uses async programming patterns for improved performance and HTML5 compatibility.

Required Changes

Server Class Calls from Client

C# Example:

```
public async Task DisplayOpening()
{
    await @Script.Class.ServerMain.Method1();
}
```

VB.NET Example:

```
Public Async Function DisplayOpening() As Task
    Await @Script.Class.ServerMain.Method1()
End Function
```

LogOn Method Update

Replace `LogOn()` with `LogOnAsync()`:

JavaScript:

```
this.DialogOnOK = async function()
{
    var result = await @Client.LogOnAsync("user1", "pass1");
    return true;
};
```

C#:

```
public async Task<bool> DialogOnOK()
{
    var result = await @Client.LogOnAsync("user1", "pass1");
    return true;
}
```

VB.NET:

```
Public Async Function DialogOnOK() As Task(Of Boolean)
    Dim result As Integer = Await @Client.LogOnAsync("user1", "pass1")
    Return True
End Function
```

Complete CodeBehind Example

Legacy Code (Synchronous):

```
Public Function DialogOnOK() As Boolean
    Dim log As eSecurityErrors
    log = DirectCast(@Client.LogOn(@Client.InputUserName, @Client.InputPassword), eSecurityErrors)
    If log = eSecurityErrors.OK Then
        @Display.LogOn.Close()
        Return True
    End If
    Return False
End Function
```

Updated Code (Async):

```
Public Async Function DialogOnOK() As Task(Of Boolean)
    Dim log As eSecurityErrors
    log = DirectCast(Await @Client.LogOnAsync(@Client.InputUserName, @Client.InputPassword), eSecurityErrors)
    If log = eSecurityErrors.OK Then
        @Display.LogOn.Close()
        Return True
    End If
    Return False
End Function
```

Additional Corrections

Symbol Mnemonic

All mnemonics require default values:

Version	Format	Example
Legacy	`#HourOfDay:`	Missing default
10.1 or newer	`#HourOfDay:""`	Includes default

Popup Position

Use Drawing Properties for positioning:

- Path: Drawing Display [DisplayName] Drawing Properties
- Settings: Left and Top

Layout System

Feature	Legacy	10.1 or newer
Method	Docking/alignment	New layout system
Multi-level menus	Multiple docked windows	Single header window
Submenus	Separate windows	ChildDisplay controls

Projects from 9.2 or older versions that did not use the last screen in the layout list as the dynamic screen, which automatically switched screens with an open command and therefore needed to use the OpenAtIndex command, must have their layouts reconfigured, because in a migration to 10.1 or newer, the last screen in the layout list will be considered the main switching screen.

Display WPF only no longer automatically converts LayoutTransform (rotation used in 8.1 version that doesn't work for HTML5) to RenderTransform (rotation used in v10 that is used for WPF and HTML5), it keeps as LayoutTransform. However, if the user ungroups symbols, groups, and shape groups that have LayoutTransform, or removes the Transform, the LayoutTransform will be lost.

Deprecated Features

Feature	Status	Alternative
Synchronous SQL calls	Warning issued	Use async methods
Custom theme editing	Not supported	Use standard themes
`<TagProvider>.("Topic Path")` syntax	Removed	Map to AssetTree
XPS documents	Not supported	Use PDF or other formats

Validation Checklist

Pre-Production Testing

- All displays open correctly
- Scripts execute without errors
- Database connections work
- Alarms function properly
- Historical data logs correctly
- Client connections stable
- Performance acceptable

Common Issues

Issue	Solution
Display errors	Update async methods in CodeBehind
Database failures	Rename RuntimeUsers table
Theme problems	Review color mappings
Layout issues	Redesign using new system

Best Practices Checklist

- **Backup** original project before upgrade
- **Test** thoroughly in lab environment
- **Document** all manual changes made
- **Verify** async methods in all scripts
- **Review** deprecated features for alternatives
- **Plan** theme and layout updates
- **Validate** all functionality before production

In this section...
