# Runtime Build and Publish Reference

Build and publish your solution.



---

Runtime Build and Publish (Reference): Commands for compiling solutions and creating deployment packages. Build validates scripts and displays, while Publish creates read-only production versions.

**Access:** Runtime  Build and Publish

Build Operations
When to Build
Publish Operations
Build Status
Build vs Publish
Validation Results
Best Practices Checklist
File Types
Troubleshooting

---

## Build Operations

The **Build** command compiles the solution's script classes, script tasks, script expressions and displays, records the result of every object in the grid, and increments the build number. Four check boxes on the Build panel change what the build actually does.

All four options are stored with the solution, so the state you leave them in is the state you find in the next Designer session. By default all four are cleared. Every build also records the options it ran with in its **Solution History** entry, so you can tell later how a given build number was produced.

| Option | What it does | When to use |
|---|---|---|
| **Save Backup** | Copies the solution file before compilation starts and registers the build as a restore point in Solution History | Before upgrades, mass imports, or any build you may need to undo |
| **Rebuild All** | Ignores previous compile results and recompiles every object in the solution | Before publishing, after a version upgrade, or when results look inconsistent |
| **Validate Displays** | Opens and renders every display, then saves it back, so load-time problems become build errors | Before publishing, after a version upgrade, after mass symbol or template edits |
| **Debug Information** | Emits debug symbols and generated sources so the Script Debugger and breakpoints work | Development and troubleshooting only — leave cleared for production builds |

### Default build (no options selected)

Designer compiles each script and display in the background as you edit and save it, so most objects already carry a compile result before you press Build. With all four boxes cleared, the build recompiles only:

- objects that have never been compiled (no line yet in the build grid), and
- objects whose last compile reported errors or warnings.

Everything else is left untouched and counted in the `Up-to-date` figure of the build summary. This is the fast, everyday build.

### Save Backup

Saves a backup copy of the solution file *before* the first object is compiled, so the backup always reflects the solution as it was going into the build, and marks the build entry in Solution History as having a backup. That entry is what you select to roll the solution back to this build.

- Selecting **Rebuild All** turns **Save Backup** on automatically. You can clear it again, but the pairing is intentional: a full rebuild rewrites the compiled output of every object in the solution.
- Recommended before a version upgrade, a mass import, or any build you may need to undo.

### Rebuild All

Rebuilds the solution from scratch instead of only recompiling what is out of date. Besides compiling every script class, script task, script expression and display, it:

- clears the build result grid and re-checks the solution database schema;
- refreshes all solution types (tags, templates, user-defined types) before compiling, and reports circular template definitions as warnings;
- runs extra dependency passes over the script classes, so a class that depends on another class compiled later in the same pass is retried instead of being reported as an error.

The first **Rebuild All** in a solution asks for confirmation, because it compiles everything and takes considerably longer than the default build. Use it before publishing, after upgrading a solution from a previous product version, after restoring or importing a solution, or whenever the reported build results look inconsistent with the configuration.

### Validate Displays

Goes beyond compiling the display code: the build opens each display, renders it off-screen, and saves it back. For every display this means:

- the display is instantiated with all of its controls, symbols and dynamics, so problems that only appear when a display is loaded — a missing symbol, an invalid control property, a broken dynamic — are reported at build time instead of failing in front of the operator;
- the generated XAML and the display expressions are regenerated and saved;
- the thumbnail shown in the Displays list is refreshed.

This is the slowest option, since displays are loaded one at a time. Use it before publishing, after a version upgrade, and after mass edits to symbols or templates. The option is not available on EdgeGateway, which has no display runtime.

### Debug Information

Makes the compiler emit debug symbols and the generated source files alongside the compiled assemblies in the build temp folder. This is what allows the Script Debugger to work:

- breakpoints can only be set while Debug Information is enabled — otherwise Designer reports that debug information is disabled;
- the attach-debugger options for Startup and Test runs are shown only while it is enabled;
- clearing the check box deletes the generated source and symbol files from the temp folder.

Symbols are emitted anyway while a debugger is attached, whatever the check box says. Leave the option cleared for production builds: it adds time to every compile and leaves generated sources and symbols on disk. The check box is shown only where the Script Debugger is available.

### Temp Files

Selects where the compiler writes its working files — generated sources, compiled assemblies and, when **Debug Information** is enabled, debug symbols.

| Option | Location |
|---|---|
| **DesignerData** | `SolutionPath\DesignerSettings\` |
| **\ProgramData** | `\ProgramData\FrameworX\` |

---

## When to Build

Build is automatic during development but manual build is useful for:

| Scenario | Purpose |
|---|---|
| Many changes made | Full validation |
| Assign build number | Version tracking |
| Pack database | Optimize storage |
| Before publishing | Final verification |

### Build Process Steps

1. Navigate to Runtime  Build and Publish
2. Select Rebuild All and Validate Displays
3. Click Build
4. Review build results
5. Fix any errors shown

---

## Publish Operations

### Publish Process

Creates read-only .dbrun file for production:

| Setting | Function | Result |
|---|---|---|
| **Publish** | Create deployment file | SolutionName_Version.dbrun |
| **Version** | Current version number | Auto-incremented |

| | | |
|---|---|---|
| **Next Increment** | Version strategy | Major/Minor |

### Version Management

| Increment Type | Example | Use Case |
|---|---|---|
| **Major** | 1.0  2.0 | New features, breaking changes |
| **Minor** | 1.0  1.1 | Bug fixes, small updates |

---

## Build Status

Background compilation provides continuous validation:

| Status | Indicator | Action Required |
|---|---|---|
| **Green** | Success | None |
| **Warning** | Warning | Review recommended |
| **Error** | Error | Fix required |

Double-click any row with issues to navigate to source.

---

## Build vs Publish

### Build Only

Suitable when:

- Continuous development expected
- Frequent field changes
- Development environment
- Testing iterations

### Publish Required

Necessary for:

- Regulated industries
- Locked production systems
- Version control
- Read-only deployment

---

## Validation Results

### Warning Types

Warnings are informational:

- Won't stop execution
- May indicate potential issues
- Should be reviewed

### Error Types

Errors prevent execution:

- Script won't run
- Display won't load
- Must be fixed

---

## Best Practices Checklist

### Development Phase

- Use automatic background compilation
- Fix errors as they appear
- Don't build unnecessarily

### Pre-Production

- Run full build with all options
- Fix all errors
- Review warnings
- Create backup

### Production Deployment

- Final build and validation
- Set version number
- Publish solution
- Archive .dbrun file

---

## File Types

| Extension | Type | Purpose |
|---|---|---|
| **.dbsln** | Solution file | Development version |
| **.dbbak** | Backup file | Build backup |
| **.dbrun** | Published file | Production deployment |

---

## Troubleshooting

| Issue | Cause | Solution |
|---|---|---|
| Build fails | Script errors | Check error list |
| Publish disabled | Build errors exist | Fix all errors first |
| Large file size | No compression | Enable database packing |
| Version conflict | Manual edit | Use version controls |

---

**In this section...**

---