# Safire Business UI Source Syntax Guide

Version 1.0 - End User Guide and Developer Preview Reference

*SafireIDE, Safire Language, Source-First Forms, Controls, Events, Data Binding, and Business Browse/Update Patterns*

Prepared for the Safire Version 1 project progress baseline
2026-05-30

## Contents


- 1. Purpose and current Version 1 baseline
- 2. Source-first business UI model
- 3. Project and file organization
- 4. Naming conventions and syntax rules
- 5. Application and window source declarations
- 6. Layout, sizing, anchoring, and business form structure
- 7. Common business controls and properties
- 8. Browse/table source syntax and column-level editing
- 9. Data binding and validation in UI source
- 10. Events and command handlers
- 11. Menus, toolbars, status bars, dialogs, and help
- 12. Reports and UI-driven output
- 13. SafireIDE round-trip workflow
- 14. AI-assisted UI source practices
- 15. Build, run, package, and deployment considerations
- 16. Complete source examples
- 17. Reference appendices


## 1. Purpose and current Version 1 baseline

This guide describes the recommended source syntax for business user interfaces in Safire Version 1. It is written for developers who build practical business systems with forms, browses, update windows, reports, validation, and deployment packages.

The focus is not on experimental syntax. The focus is on a durable application source model that SafireIDE can edit visually, that a developer can still read in a text editor, and that the Safire compiler/runtime path can check, plan, preview, build, and package.

**The current Version 1 project progress** establishes the following baseline:

- Safire source files remain the application truth for windows, controls, reports, procedures, classes, dictionary objects, queries, and project settings.
- SafireIDE can load real source through the runtime-plan service, show a project tree, display designer surfaces, synchronize selection with property grids, and preview source write-back before applying it.
- SafireIDE source write-back is designed around preview, diff, approval, backup, reparse checks, audit history, and restore/undo support.
- The control toolbox, event catalog, runtime plan, property application, and layout behavior have reached the Version 1 proof path.
- The build/run/error-output loop, compiler/CLI regression, runtime smoke test, database runtime, browse/update generator, report runtime, sample business app, deployment package, developer package, and product health regression path have reached the first usable V1 package baseline.

This guide therefore uses a V1 public-facing syntax style that is business-oriented and source-first. Some keywords and exact compiler options may still be finalized as the product hardens, but the document records the intended developer experience.

### 1.1 Audience

- Mature business application developers who maintain systems over many years.
- Developers who think in terms of customers, invoices, stock, employees, orders, cashbooks, ledgers, reports, and deployment packages.
- Developers who want visual productivity without losing readable source ownership.
- Teams that need AI help to be reviewable, reversible, and auditable.

### 1.2 What this guide covers

| Area | Covered in this guide |
|---|---|
| Windows and forms | Window declarations, properties, lifecycle, layout, and business form patterns. |
| Controls | Labels, edits, buttons, check boxes, combo boxes, tables, tabs, panels, images, and related properties. |
| Browses and tables | Browse source syntax, table columns, actions, sort/search flags, masks, row selection, and update patterns. |
| Events | Source-defined event handlers for user actions, form lifecycle, validation, and row navigation. |
| Binding and validation | How controls bind to records, variables, queries, and validation rules. |
| SafireIDE workflow | Preview, diff, approved write-back, backup, restore, and audit. |
| AI-assisted source | How to write UI source that is safe for AI analysis and patch proposals. |

## 2. Source-first business UI model

Safire business UI source is not a hidden designer file. The window, controls, layout, bindings, events, browse definitions, and report links must be recoverable from developer-owned source files.

The designer is a productivity surface over the source. It must not become the only place where the application truth exists.

### 2.1 Durable source versus generated artifacts

| Durable source | Disposable/generated artifact |
|---|---|
| Application declarations | Runtime plans and generated preview files |
| Window/form source files | Designer caches and temporary layout snapshots |
| Report source files | Rendered report previews and print/PDF work files |
| Dictionary/table/query declarations | Indexes, analysis caches, and proof output |
| Procedures, classes, validation, events | Build logs and intermediate compiler output |
| Project manifest and deployment-owned config | Generated staging folders and package scratch space |

### 2.2 UI source life cycle

1. Developer opens a project in SafireIDE.
2. SafireIDE reads the project source and asks the runtime-plan service for a UI plan.
3. The designer shows a live design surface from that source-backed plan.
4. The developer changes controls, properties, bindings, or events.
5. SafireIDE previews the proposed source change and shows the diff.
6. The developer approves the apply.
7. SafireIDE writes back source with a backup, reparses it, and records audit evidence.
8. Build/run/check commands use the same source truth.

### 2.3 Business developer rule

A Safire UI source file should read like a business application declaration, not like low-level window plumbing. Developers should see business objects, controls, bindings, actions, validation, and report commands. Lower-level runtime details belong behind the Safire runtime and the SafireIDE designer.

# 3. Project and file organization

A business project should separate user-owned source from generated and package output. The following layout is recommended for Version 1 projects.

```
CustomerManager/
  project.sfproj
  app/
    Application.sf
  windows/
    WIN_CustomerList.sfwin
    WIN_CustomerEdit.sfwin
    WIN_InvoicePost.sfwin
  reports/
    RPT_CustomerList.sfrpt
    RPT_Invoice.sfrpt
  dictionary/
    Customer.sfdict
    Invoice.sfdict
    InvoiceLine.sfdict
  queries/
    Q_CustomerSearch.sfquery
    Q_OpenInvoices.sfquery
  classes/
    CustomerService.sf
    InvoiceService.sf
  procedures/
    GlobalProcedures.sf
  resources/
    icons/
    images/
    help/
  build/
    generated/
    output/
```

## 3.1 Common source file types

| Extension | Purpose |
|---|---|
| .sfproj | Project manifest and project-level build/deploy metadata. |
| .sf | General Safire source module for procedures, functions, classes, services, and application declarations. |
| .sfwin | Source-backed window/form declaration. |
| .sfrpt | Source-backed report declaration. |
| .sfdict | Source-backed table, field, key, relationship, and dictionary declaration. |
| .sfquery | Source-backed query declaration. |
| .sfstyle | Optional style/theme declarations owned by the developer when used. |

## 3.2 Project manifest example

```
Project CustomerManager
  Name = "Customer Manager"
  Version = "1.0.0"
  Application = app/Application.sf
  SourceFolders = app, windows, reports, dictionary, queries, classes, procedures
  ResourceFolder = resources
  OutputFolder = build/output
  GeneratedFolder = build/generated
```

```
  PackageName = "CustomerManager"
End
```

## 4. Naming conventions and syntax rules

Consistent names help the developer, SafireIDE, the compiler, the runtime-plan service, and the AI assistant understand the application. Prefixes are recommended for UI artifacts, but the business name should remain readable.

| Object type | Recommended pattern | Example |
|---|---|---|
| Window | WIN_BusinessPurpose | WIN_CustomerList |
| Report | RPT_BusinessPurpose | RPT_Invoice |
| Query | Q_BusinessPurpose | Q_CustomerSearch |
| Button | BTN_Action | BTN_Save |
| Edit | EDT_FieldName | EDT_CustomerName |
| Label | LBL_FieldName | LBL_CustomerName |
| Table/Browse control | TBL_ObjectPlural | TBL_Customers |
| Combo box | CBO_FieldName | CBO_PaymentTerms |
| Check box | CHK_FieldName | CHK_IsActive |
| Panel | PNL_Purpose | PNL_ButtonBar |
| Tab | TAB_Purpose | TAB_Customer |
| Menu item | MNU_Action | MNU_FileExit |
| Toolbar item | TBI_Action | TBI_Save |

### 4.1 Basic syntax style

- Use named blocks such as Application, Window, Table, Query, Report, Procedure, Function, Class, and Validation.
- Use property assignment with Property = Value.
- Use End to terminate compound blocks.
- Use short event handlers and move business logic into procedures or classes.
- Use comments for business intent, not for obvious syntax.

```
Window WIN_CustomerEdit
  Title = "Customer"
  Size = 640, 420
  Center = True

  Edit EDT_Name
    Caption = "Name"
    Binding = Customer.Name
    Required = True
  End

  Button BTN_Save
    Caption = "Save"
  End

  On BTN_Save.Click
    SaveCustomer()
  End
End
```

# 5. Application and window source declarations

An application declaration identifies the top-level business application, its main window, default database, resource settings, and deployment-facing metadata.

## 5.1 Application declaration

```
Application CustomerManager
  Title = "Customer Manager"
  Version = "1.0.0"
  Company = "Example Trading"
  MainWindow = WIN_CustomerList
  DefaultDatabase = DB_Main
  Icon = Resource("icons/customer-manager.ico")
  Culture = "en-ZA"
End

Procedure Main()
  OpenWindow(WIN_CustomerList)
End
```

| Property | Purpose |
|---|---|
| Title | User-facing title of the application. |
| Version | Application version used for packaging and support. |
| Company | Publisher or business owner. |
| MainWindow | First window opened for an interactive desktop app. |
| DefaultDatabase | Default database connection or dictionary root. |
| Icon | Application icon resource. |
| Culture | Default locale/culture for formats and captions. |

## 5.2 Window declaration

```
Window WIN_CustomerList
  Title = "Customers"
  Size = 900, 600
  Center = True
  Resizable = True
  MinSize = 720, 440
  DataSource = Customer

  // Controls and events are declared here.
End
```

| Window property | Purpose |
|---|---|
| Title | Window caption. |
| Size | Initial width and height. |
| Position | Initial screen position. |
| Center | Center on screen or parent. |
| Resizable | Whether the user can resize the window. |
| MinSize | Minimum width and height. |
| Icon | Window icon. |
| DataSource | Default source for child bindings. |

| | |
|---|---|
| Layout | Named layout manager or inline layout declaration. |
| HelpTopic | Context help topic for this window. |

## 6. Layout, sizing, anchoring, and business form structure

Safire supports coordinate-based placement for RAD speed and layout-based declarations for forms that must resize well. Business developers should be able to use the simpler coordinate model first, then improve with layout rules where necessary.

### 6.1 Coordinate placement

```
Label LBL_Name
  Caption = "Customer name:"
  At = 20, 22, 120, 20
End

Edit EDT_Name
  Binding = Customer.Name
  At = 150, 20, 260, 24
End
```

### 6.2 Anchoring

```
Table TBL_Customers
  Source = Q_CustomerSearch(SearchText = SearchText)
  At = 10, 45, 860, 460
  Anchor = Left, Top, Right, Bottom
End

Panel PNL_ButtonBar
  At = 10, 520, 860, 36
  Anchor = Left, Right, Bottom
End
```

| Anchor flag | Result |
|---|---|
| Left | Maintain distance from left edge. |
| Top | Maintain distance from top edge. |
| Right | Maintain distance from right edge. |
| Bottom | Maintain distance from bottom edge. |
| Left, Right | Control stretches horizontally when the window resizes. |
| Top, Bottom | Control stretches vertically when the window resizes. |

### 6.3 Layout blocks

```
Layout Vertical MainLayout
  Margin = 10
  Gap = 8

  Add PNL_Search Height = 34
  Add TBL_Customers Expand = True
  Add PNL_ButtonBar Height = 38
End
```

### 6.4 Recommended business form zones

| Zone | Typical contents |
|---|---|
| Search/filter zone | Search fields, date range, status filters, refresh button. |
| Browse/list zone | Table or browse control with business columns. |
| Detail/edit zone | Data-bound edits, combos, check boxes, tabs, and validation messages. |
| Action zone | New, Edit, Delete, Save, Cancel, Print, Export, Close. |
| Status zone | Current record count, validation state, active user, last action. |

## 7. Common business controls and properties

Controls are declared inside a window, panel, group, page, toolbar, menu, or report band. A control should be named, bindable where appropriate, and understandable from source.

### 7.1 Common control properties

| Property | Purpose |
|---|---|
| Caption | User-facing label or button text. |
| Text | Displayed text value where different from Caption. |
| Binding | Field, variable, object property, or row value to bind to. |
| Source | Query, list, dataset, or table backing a list/table control. |
| At | Coordinate and size: x, y, width, height. |
| Visible | Whether the control is visible. |
| Enabled | Whether the user can interact with the control. |
| ReadOnly | Display value but do not allow edits. |
| Required | Marks required data entry. |
| Picture | Input/display format mask. |
| Format | Display format, for example money or date. |
| TabOrder | Keyboard tab order. |
| ToolTip | Short hover help. |
| HelpTopic | Detailed help topic. |
| Note | Developer note visible in SafireIDE. |
| Style | Named visual style. |

### 7.2 Labels and edit controls

```
Label LBL_CustomerCode
  Caption = "Code:"
  At = 20, 24, 90, 20
End

Edit EDT_CustomerCode
  Caption = "Code"
  Binding = Customer.CustomerCode
  Required = True
  Picture = "@S20"
  At = 120, 20, 120, 24
End
```

```
Edit EDT_CreditLimit
  Caption = "Credit limit"
  Binding = Customer.CreditLimit
  Format = "money"
  At = 120, 84, 120, 24
End
```

## 7.3 Buttons

```
Button BTN_Save
  Caption = "Save"
  At = 20, 340, 90, 28
  Default = True
  ToolTip = "Save this customer"
End

On BTN_Save.Click
  SaveCustomer()
End
```

## 7.4 CheckBox, RadioButton, and ComboBox

```
CheckBox CHK_Active
  Caption = "Active customer"
  Binding = Customer.IsActive
  At = 120, 122, 160, 22
End

ComboBox CBO_Terms
  Caption = "Payment terms"
  Binding = Customer.PaymentTermsId
  Lookup = PaymentTerms.PaymentTermsId, PaymentTerms.Description
  At = 120, 156, 240, 24
End
```

## 7.5 Tabs, pages, panels, and groups

```
Tab TAB_Customer
  At = 10, 10, 600, 300

  Page General
    Edit EDT_Name Binding = Customer.Name
    Edit EDT_Email Binding = Customer.EmailAddress
  End

  Page Accounts
    Edit EDT_CreditLimit Binding = Customer.CreditLimit Format = "money"
    CheckBox CHK_StopSupply Binding = Customer.StopSupply
  End
End
```

## 7.6 Images and HTML/help views

```
Image IMG_Logo
  Source = Resource("images/logo.png")
  ScaleMode = Fit
  At = 20, 20, 140, 60
End

HtmlView HLP_CustomerNotes
  Source = Resource("help/customer-screen.html")
```

```
  At = 20, 300, 560, 180
End
```

## 8. Browse/table source syntax and column-level editing

Browses and tables are central to business systems. Safire separates table-level behavior from column-level behavior so the IDE can show the right properties depending on whether the table itself or one column is selected.

### 8.1 Table control declaration

```
Table TBL_Customers
  Source = Q_CustomerSearch(SearchText = SearchText)
  At = 10, 45, 860, 460
  Anchor = Left, Top, Right, Bottom
  AllowRowSelect = True
  AllowSearch = True
  AllowSort = True
  EmptyText = "No customers found."

  Column CustomerCode
    Title = "Code"
    Width = 90
    Binding = Customer.CustomerCode
    Sortable = True
    Searchable = True
  End

  Column Name
    Title = "Customer"
    Width = 260
    Binding = Customer.Name
    Sortable = True
    Searchable = True
  End

  Column Balance
    Title = "Balance"
    Width = 100
    Binding = Customer.Balance
    Align = Right
    Format = "money"
  End
End
```

### 8.2 Table-level properties

| Property | Purpose |
|---|---|
| Source | Query, dataset, table, or collection displayed by the browse. |
| AllowRowSelect | Allows current-row selection. |
| AllowSearch | Enables table-level search where supported. |
| AllowSort | Enables column sorting where supported. |
| MultiSelect | Allows multiple selected rows. |
| RowHeight | Default row height. |
| EmptyText | Message when no rows are available. |
| DefaultSort | Initial sort expression. |
| CurrentRow | Runtime row object used by events and bindings. |

| | |
|---|---|
| Style | Named table style. |

### 8.3 Column-level properties

| Property | Purpose |
|---|---|
| Title | Column heading text. |
| Binding | Field or expression displayed by the column. |
| Width | Column width. |
| MinWidth | Minimum column width. |
| Resizable | User may resize the column. |
| Movable | User may reorder the column. |
| Sortable | Column can sort the table. |
| Searchable | Column participates in search. |
| Visible | Column is visible. |
| Align | Left, Center, or Right. |
| Format | Display format such as money, date, percent. |
| Picture | Input/display mask. |
| ReturnedValue | Value returned when this row/column is selected. |
| HelpTopic | Help for the column. |
| Note | Developer note. |
| Style | Column style. |

### 8.4 Browse declaration with actions

```
Browse BRW_Customers For Customer
  Source = Q_CustomerSearch(SearchText = SearchText)
  DefaultSort = Customer.Name

  Column CustomerCode Title "Code" Width 90
  Column Name Title "Customer" Width 260
  Column Balance Title "Balance" Width 100 Format "money" Align Right

  Action New Opens WIN_CustomerEdit Mode Insert
  Action Edit Opens WIN_CustomerEdit Mode Change
  Action Delete Calls DeleteCustomer
  Action Print Calls PrintCustomerList
  Action Export Calls ExportCustomerList
End
```

## 9. Data binding and validation in UI source

A business UI is useful only when it is connected to data and business rules. Safire source should make bindings and validation visible so they can be checked, generated, reported, and reviewed.

### 9.1 Binding to a table/record field

```
Window WIN_CustomerEdit
  DataSource = Customer

  Edit EDT_Name
    Caption = "Name"
```

```
    Binding = Customer.Name
    Required = True
  End
End
```

## 9.2 Binding to a window variable

```
Window WIN_CustomerList
  Var SearchText is String(100) = ""

  Edit EDT_Search
    Caption = "Search"
    Binding = SearchText
  End

  Button BTN_Search
    Caption = "Search"
  End

  On BTN_Search.Click
    Refresh(TBL_Customers)
  End
End
```

## 9.3 Binding to current browse row

```
Edit EDT_SelectedName
  Caption = "Selected customer"
  Binding = TBL_Customers.CurrentRow.Name
  ReadOnly = True
End
```

## 9.4 Validation declarations

```
Validation CustomerRules For Customer
  Rule NameRequired
    When IsBlank(Customer.Name)
    Message = "Customer name is required."
  End

  Rule CreditLimitValid
    When Customer.CreditLimit < 0
    Message = "Credit limit may not be negative."
  End
End
```

## 9.5 User-facing validation pattern

```
Procedure SaveCustomer()
  If Not Validate(Customer) Then
    ShowValidationMessages()
    Return
  End

  Save(Customer)
  CloseWindow()
End
```

# 10. Events and command handlers

Events connect user actions and runtime lifecycle events to Safire code. Event handlers should remain short and should delegate business work to procedures or classes.

## 10.1 Event handler syntax

```
On Object.Event
  statements
End

On BTN_Save.Click
  SaveCustomer()
End

On Window.Opened
  LoadCustomer()
End
```

## 10.2 Common event catalog

| Event | Typical applies to | Purpose |
|---|---|---|
| Created | Window/control | Object has been created. |
| Opening | Window | Before the window opens. |
| Opened | Window | After the window opens. |
| Closing | Window | Before the window closes. |
| Closed | Window | After the window closes. |
| Click | Button/control | User clicked a control. |
| DoubleClick | Table/list/control | User double-clicked. |
| Changed | Entry controls | Value changed. |
| ValueChanged | Entry controls | Value changed and should be handled. |
| SelectionChanged | Lists/tables | Selection changed. |
| RowSelected | Table/browse | Current row changed. |
| Validating | Entry controls/window | Validate before accepting changes. |
| KeyDown | Window/control | Key pressed. |
| Timer | Timer | Timed event fired. |
| Drop | Drop target | Drag-and-drop data arrived. |

## 10.3 Browse row event pattern

```
On TBL_Customers.DoubleClick
  OpenCurrentCustomer()
End

Procedure OpenCurrentCustomer()
  If TBL_Customers.CurrentRow.IsEmpty Then
    ShowMessage("Select a customer first.")
    Return
  End

  OpenWindow(WIN_CustomerEdit,
    Mode = Change,
    CustomerId = TBL_Customers.CurrentRow.CustomerId)
```

```
  Refresh(TBL_Customers)
End
```

### 10.4 Keyboard shortcuts

```
On Window.KeyDown
  Case Key
    When F5
      Refresh(TBL_Customers)
    When Escape
      CloseWindow()
  End
End
```

## 11. Menus, toolbars, status bars, dialogs, and help

Business applications need predictable command surfaces. Menus, toolbars, status bars, dialogs, and help should also be source-defined where they are part of the application behavior.

### 11.1 MenuBar syntax

```
MenuBar MainMenu
  Menu "File"
    Item MNU_FileNew Caption "New customer" Shortcut "Ctrl+N"
    Item MNU_FilePrint Caption "Print customer list" Shortcut "Ctrl+P"
    Separator
    Item MNU_FileExit Caption "Exit"
  End

  Menu "Help"
    Item MNU_HelpContents Caption "Help contents"
    Item MNU_HelpAbout Caption "About"
  End
End

On MNU_FileExit.Click
  CloseWindow()
End
```

### 11.2 ToolBar syntax

```
ToolBar MainToolBar
  Button TBI_New Caption "New" Icon Resource("icons/new.png")
  Button TBI_Edit Caption "Edit" Icon Resource("icons/edit.png")
  Button TBI_Print Caption "Print" Icon Resource("icons/print.png")
End

On TBI_New.Click
  OpenWindow(WIN_CustomerEdit, Mode = Insert)
End
```

### 11.3 StatusBar syntax

```
StatusBar MainStatus
  Panel StatusText Width = 400
  Panel UserName Width = 160
  Panel RecordCount Width = 120
End

Procedure UpdateStatus()
```

```
  MainStatus.StatusText = "Ready"
  MainStatus.UserName = CurrentUserName
  MainStatus.RecordCount = ToString(TBL_Customers.RowCount) + " customers"
End
```

### 11.4 Dialog and message patterns

```
ShowMessage("Customer saved.")
ShowWarning("This customer is over the credit limit.")
ShowError("The record could not be saved.")

If Confirm("Delete this customer?") Then
  DeleteCustomer()
End
```

## 12. Reports and UI-driven output

Reports are first-class Safire source artifacts. A UI action should call a report by name and pass clean parameters rather than embedding report logic directly in the click handler.

```
On BTN_Print.Click
  PreviewReport(RPT_CustomerList,
    SearchText = SearchText,
    ActiveOnly = CHK_ActiveOnly.Value)
End

On BTN_Export.Click
  ExportReport(RPT_CustomerList,
    Format = "PDF",
    FileName = "CustomerList.pdf",
    SearchText = SearchText)
End
```

### 12.1 Report button design rule

- Keep the UI button event short.
- Pass report parameters explicitly.
- Keep report layout and report data source in the .sfrpt and .sfquery source files.
- Use PreviewReport for user review and ExportReport for PDF/export workflows.
- Allow the report designer to round-trip changes back to report source.

## 13. SafireIDE round-trip workflow

Version 1 progress includes source-backed designer and write-back workflow proofs. The intended SafireIDE workflow is designed to preserve developer trust.

### 13.1 Property grid edit workflow

9. Select a control in the designer.
10. SafireIDE shows table-level or control-level properties as appropriate.
11. Change a property in the property grid.
12. SafireIDE creates a proposed source edit rather than silently changing hidden state.
13. Preview the change and review the diff.
14. Approve the apply.
15. SafireIDE writes the source, creates a backup, reparses the source, and updates the runtime plan.
16. The review history records what changed and supports restore where appropriate.

### 13.2 Source write-back example

Changing a button caption in the designer should result in a source edit like this:

```
Button BTN_Save
  Caption = "Save customer"
  At = 20, 340, 110, 28
End
```

The source should remain readable after the IDE writes it back. Generated runtime plans and preview files may change, but they must be reproducible from the source.

### 13.3 Review history and restore

- Every applied source write-back should record an audit entry.
- Backups should be retained for review and restore workflows.
- Restore should preview the effect before applying it.
- Restored source must be reparsed and checked just like any other applied change.
- Build/test evidence should be attached where relevant.

## 14. AI-assisted UI source practices

Safire AI assistance should understand forms, controls, events, data binding, reports, and project structure. It should not directly control files without a Safire-controlled review/apply workflow.

### 14.1 Write source that AI can safely understand

- Use clear object names such as WIN_InvoicePost and BTN_Post, not Window1 and Button3.
- Keep event handlers short and delegate business rules to procedures/classes.
- Declare bindings explicitly rather than relying on hidden designer metadata.
- Keep generated files under generated folders.
- Use validation declarations for reusable business rules.
- Use developer notes for business intent that is not obvious from source.
- Keep report calls parameterized and visible.

### 14.2 AI request examples

| Request | Good AI workflow result |
|---|---|
| Add an ActiveOnly filter to the customer browse. | AI proposes changes to window variable, filter control, query parameter, browse source, and refresh event. |
| Make the invoice posting form safer. | AI reviews validation, Save/Post event handlers, warning messages, and report output path before proposing a diff. |
| Create a browse/update pair for suppliers. | AI proposes dictionary/query/window source files and shows them for review. |
| Find UI controls without help text. | AI scans source and returns a review list without changing files. |
| Rename CustomerName to Name on the edit window. | AI proposes source-aware rename with diff and reparse check. |

### 14.3 AI safety boundary

The preferred workflow is: question/request -> source context -> proposal -> human review -> diff -> approved apply -> backup -> audit -> build/test proof. The model may help reason and propose. Safire-controlled tooling owns file access, patch application, backup, restore, and build/test execution.

# 15. Build, run, package, and deployment considerations

A source-backed UI must compile and package predictably. Safire Version 1 progress includes the build/run/error-output loop, compiler/CLI regression, runtime smoke path, database runtime, browse/update generator, report runtime, sample business app, end-user package, developer package, fresh-machine first-launch, diagnostics, and product health regression baseline.

## 15.1 Typical command-line workflow

```
safire check CustomerManager.sfproj
safire build CustomerManager.sfproj
safire run CustomerManager.sfproj
safire package CustomerManager.sfproj --configuration Release
```

## 15.2 Deployment output

| Deployment item | Purpose |
|---|---|
| Application EXE/DLL | Compiled application artifact or launcher/runtime entry point. |
| Safire runtime components | Runtime libraries and services needed by the application. |
| Database drivers | Database access components required for the selected provider. |
| Report/PDF support | Preview, print, and export support where used. |
| Configuration files | Environment-specific settings. |
| Runtime license keys where applicable | Runtime licensing or entitlement where required. |
| Support bundle tools | Diagnostics and support capture for deployed systems. |

## 15.3 Licensed IDE versus application deployment

The SafireIDE is the professional productivity product used by developers. End users should not need SafireIDE to run a deployed business application. Deployed applications should contain the runtime and application package needed to run the business system, not the licensed development environment.

# 16. Complete source examples

## 16.1 Customer browse window

```
Window WIN_CustomerList
  Title = "Customers"
  Size = 900, 600
  Center = True
  Resizable = True

  Var SearchText is String(100) = ""
  Var ActiveOnly is Boolean = True

  Panel PNL_Search
    At = 10, 10, 860, 30

    Edit EDT_Search
      Caption = "Search"
      Binding = SearchText
      At = 0, 2, 300, 24
    End

    CheckBox CHK_ActiveOnly
      Caption = "Active only"
      Binding = ActiveOnly
      At = 315, 4, 110, 22
```

```
  End

  Button BTN_Search
    Caption = "Search"
    At = 440, 2, 80, 24
  End
End

Table TBL_Customers
  Source = Q_CustomerSearch(SearchText = SearchText, ActiveOnly = ActiveOnly)
  At = 10, 45, 860, 460
  Anchor = Left, Top, Right, Bottom
  EmptyText = "No customers found."

  Column CustomerCode Title "Code" Width 90 Sortable = True Searchable = True
  Column Name Title "Customer" Width 260 Sortable = True Searchable = True
  Column EmailAddress Title "Email" Width 240
  Column Balance Title "Balance" Width 100 Align Right Format "money"
End

Panel PNL_Buttons
  At = 10, 520, 860, 36
  Anchor = Left, Right, Bottom
  Button BTN_New Caption "New" At = 0, 4, 80, 28
  Button BTN_Edit Caption "Edit" At = 90, 4, 80, 28
  Button BTN_Print Caption "Print" At = 180, 4, 80, 28
  Button BTN_Close Caption "Close" At = 780, 4, 80, 28
End

On BTN_Search.Click
  Refresh(TBL_Customers)
End

On BTN_New.Click
  OpenWindow(WIN_CustomerEdit, Mode = Insert)
  Refresh(TBL_Customers)
End

On BTN_Edit.Click
  OpenCurrentCustomer()
End

On TBL_Customers.DoubleClick
  OpenCurrentCustomer()
End

On BTN_Print.Click
  PreviewReport(RPT_CustomerList, SearchText = SearchText, ActiveOnly = ActiveOnly)
End

On BTN_Close.Click
  CloseWindow()
End
End
```

## 16.2 Customer edit window

```
Window WIN_CustomerEdit
  Title = "Customer"
  Size = 560, 360
  Center = True
  DataSource = Customer

  Edit EDT_Code
    Caption = "Code"
    Binding = Customer.CustomerCode
    Required = True
    At = 130, 25, 120, 24
```

```
  End

  Edit EDT_Name
    Caption = "Name"
    Binding = Customer.Name
    Required = True
    At = 130, 60, 300, 24
  End

  Edit EDT_Email
    Caption = "Email"
    Binding = Customer.EmailAddress
    At = 130, 95, 300, 24
  End

  Edit EDT_CreditLimit
    Caption = "Credit limit"
    Binding = Customer.CreditLimit
    Format = "money"
    At = 130, 130, 120, 24
  End

  CheckBox CHK_Active
    Caption = "Active customer"
    Binding = Customer.IsActive
    At = 130, 165, 160, 22
  End

  Button BTN_Save Caption "Save" At = 130, 260, 90, 28 Default = True
  Button BTN_Cancel Caption "Cancel" At = 230, 260, 90, 28 Cancel = True

  On Window.Opened
    LoadCustomer()
  End

  On BTN_Save.Click
    SaveCustomer()
  End

  On BTN_Cancel.Click
    CloseWindow()
  End
End
```

## 16.3 Support procedures for browse/update

```
Procedure OpenCurrentCustomer()
  If TBL_Customers.CurrentRow.IsEmpty Then
    ShowMessage("Select a customer first.")
    Return
  End

  OpenWindow(WIN_CustomerEdit,
    Mode = Change,
    CustomerId = TBL_Customers.CurrentRow.CustomerId)

  Refresh(TBL_Customers)
End

Procedure SaveCustomer()
  If Not Validate(Customer) Then
    ShowValidationMessages()
    Return
  End

  Save(Customer)
  CloseWindow()
End
```

### 16.4 Query and validation source used by the UI

```
Query Q_CustomerSearch
  Parameters
    SearchText is String(100) = ""
    ActiveOnly is Boolean = True
  End

  From Customer
  Where Customer.Name Contains SearchText
    And (ActiveOnly = False Or Customer.IsActive = True)
  Order By Customer.Name
End

Validation CustomerRules For Customer
  Rule CodeRequired
    When IsBlank(Customer.CustomerCode)
    Message = "Customer code is required."
  End

  Rule NameRequired
    When IsBlank(Customer.Name)
    Message = "Customer name is required."
  End
End
```

## 17. Reference appendices

### Appendix A: Control catalog summary

| Control | Typical use |
|---|---|
| Label | Static text and field captions. |
| Edit | Text, numeric, date, or data-bound entry. |
| Button | Command/action. |
| CheckBox | Boolean selection. |
| RadioButton | Exclusive option selection. |
| ComboBox | Lookup/list selection. |
| ListBox | List selection. |
| Table | Rows and columns / browse. |
| Browse | Business browse/update pattern. |
| TreeView | Hierarchical data. |
| Tab | Multi-page layout. |
| Page | Tab page container. |
| Panel | Container and layout grouping. |
| GroupBox | Visual grouping. |
| Image | Display image or data-bound image. |
| HtmlView | HTML help, preview, or rich display. |
| ProgressBar | Progress display. |
| DatePicker | Date selection. |
| MenuBar | Window menu. |

| | |
|---|---|
| ToolBar | Toolbar actions. |
| StatusBar | Status display. |
| Timer | Timed events. |

## Appendix B: Standard UI command names

| Command | Recommended procedure/action |
|---|---|
| New | Open insert window or create a new record. |
| Edit | Open selected record for change. |
| View | Open selected record read-only. |
| Delete | Confirm and delete selected record. |
| Save | Validate and save current record. |
| Cancel | Close without saving or revert pending edit. |
| Search | Refresh browse based on filters. |
| Print | Preview or print report. |
| Export | Export browse/report output. |
| Close | Close current window. |

## Appendix C: Property quick reference

| Property | Applies to | Purpose |
|---|---|---|
| Caption | Most controls | Displayed label or button text. |
| Binding | Entry/list/table controls | Data field, variable, or row property. |
| Source | Table/list/combo | Data provider, query, lookup, or collection. |
| At | Visual controls | Coordinate placement. |
| Anchor | Visual controls | Resize behavior. |
| Layout | Window/panel | Layout manager. |
| Required | Entry controls | Required data entry. |
| ReadOnly | Entry/display controls | Prevent user edits. |
| Visible | All controls | Show/hide control. |
| Enabled | All controls | Enable/disable control. |
| Format | Display controls | Display format. |
| Picture | Entry controls | Input/display mask. |
| ToolTip | All controls | Short hover help. |
| HelpTopic | Windows/controls | Context help. |
| Note | Source/design objects | Developer note. |
| Style | Windows/controls | Named style. |

## Appendix D: SafireIDE source write-back checklist

17. Load the source-backed project.
18. Open the target window/form in the designer.

19. Select the correct control, table, or column.
20. Change only the intended property or layout item.
21. Review the proposed source diff.
22. Approve apply only after the diff is correct.
23. Confirm SafireIDE reparsed the file and refreshed the designer.
24. Run check/build for any behavior-changing edit.
25. Use review history to inspect or restore prior source if needed.

## Appendix E: Closing note

Safire business UI source should remain plain enough to read, structured enough to design visually, strong enough for compiler and runtime checking, and explicit enough for safe AI assistance. The Version 1 direction is to let experienced business software developers build practical systems faster without giving up ownership of the application source.