Safire Windows and Controls Reference

End User and Software Developer Guide

Safire Documentation

2026-07-01

Contents

1 Safire Windows and Controls Reference

Document status: Current Safire windows and controls reference for end users, application designers, and software developers.

Audience:

- End users who operate Safire business applications.
- Support staff who help users understand windows, commands, validation, layouts, and reports.
- Software developers who declare Safire windows, bind controls to data, handle events, define validation, and prepare applications for deployment.
- Application designers who use Safire visual tools and source files together.

Scope: This guide describes Safire windows, controls, control properties, property types, events, data bindings, validation, resources, layout personalization, and usage patterns available in the current Safire desktop and runtime component model.

Safire source rule: Safire windows and controls are source-owned. A window should be understandable from its Safire source and related Safire metadata. Visual editing, runtime execution, validation, reports, and user layout overlays all work from the same application truth.

2 How to read this guide

Each window and control reference section uses the same pattern:

- **Purpose** explains what the window or control is for.
- **Source form** shows a compact Safire-style declaration.
- **Property types** list the data types used by the declaration and runtime metadata.
- **Events** list the user actions or runtime notifications that can call developer procedures.
- **Bindings** explain how the control connects to application data.
- **End user usage** explains how users experience the feature.
- **Developer usage** explains how developers should declare and wire the feature.
- **Comments** give design and support guidance.

The examples are intentionally short. Real applications normally split larger behavior into procedures, classes, data sources, queries, dictionaries, validation rules, report definitions, and resource manifests.

3 Core concepts

3.1 Application

A Safire application is the user-facing program. It can contain one main window, many task windows, browse/update forms, report previews, data sources, command handlers, validation rules, resources, and deployment settings.

3.2 Window

A Safire window is a visual surface. It can be a main application window, a task window, a browse/update window, a dialog, a report preview, a wizard page container, or a utility screen. A window can contain child controls, menu structures, toolbars, command strips, panels, forms, browses, grids, reports, labels, buttons, and entry fields.

3.3 Control

A Safire control is a named visual or functional element inside a window or container. Controls have ids, properties, optional child controls, optional events, optional data bindings, optional validation, and optional resources.

3.4 Source-owned control tree

The control tree is the parent/child structure of a window. Parentage is important. A control inside a panel belongs to that panel. A control inside a group belongs to that group. Selection, hit testing, move, resize, save, reload, and layout personalization all use the Safire control tree.

3.5 Events

Events connect user actions and runtime notifications to Safire procedures. Examples include Open, Close, Click, Changed, Command, RowSelected, RowAccepted, CellChanged, BeforeRun, and AfterRun.

3.6 Data bindings

A binding connects a control property to application data. For example, an entry field can bind its value to `Customer.Name`, and a browse can bind its rows to `Customers`. Bindings make a window data-aware without hiding business rules inside the visual layer.

3.7 Validation

Validation rules check user input before a command is accepted. Rules may be attached to controls, fields, data paths, commands, or forms. Typical rules include required values, minimum length, numeric checks, number ranges, boolean checks, and custom business validation.

3.8 Resources

Resources are external or named assets used by controls and reports. Examples include images, icons, string files, and report definitions. A resource reference should be explicit so the application can be packaged and diagnosed reliably.

3.9 User layout overlay

A user layout overlay records allowed user-specific changes, such as moving a button, resizing a browse, or hiding an optional report panel. The original application source remains the application definition. The overlay stores personalization that can be applied when the user opens the window.

4 End user guide

4.1 Opening and closing windows

Safire windows open from application menus, toolbar commands, shortcuts, workflow buttons, or procedure calls. A window usually displays a title, a working area, one or more command areas, and status feedback. Closing a window can be immediate or can first ask the user to save or cancel changes when data is dirty.

4.2 Using menus and toolbars

Menus provide complete command access. Toolbars and command strips provide quick access to frequent commands. The same action may appear in a menu and toolbar. For example, Save may appear in the File menu and on the toolbar, but both should perform the same command.

4.3 Using browse windows

A browse window shows business records in rows. Users can select a row, accept a row to open details, sort or filter where enabled, and use commands such as New, Edit, Delete, Refresh, Clear Filter, Print, or Close. A status line may show row counts, filter status, validation messages, or save results.

4.4 Using edit forms

An edit form shows fields for one record or work item. Some fields may be read-only, some required, and some restricted to numeric or valid range values. When a user changes a field, Safire can mark the form as dirty. Save validates and writes the current values. Cancel restores the previous values.

4.5 Using validation messages

Validation messages explain what must be fixed before a command can continue. A good validation message names the field and explains the rule in user language. For example: `Customer name is required.` or `Credit limit out of range.`

4.6 Using personalized layouts

Where allowed, users may move or resize selected controls, hide optional panels, and save the layout. Required controls remain visible. Some controls can be resized but not hidden. Some report panels may be hidden but not moved. These rules protect the application while still allowing user comfort.

4.7 Using reports

Reports can be opened from menus, toolbars, buttons, browse rows, or report panels. A report can use the current selected record, a query, a table, or a prepared data source. Users should be able to preview a report before printing or exporting when the application enables those actions.

5 Developer quick start

A typical Safire window declares a title, bounds, child controls, bindings, validation, resources, and event handlers. The following example is compact but shows the normal pattern.

```
application CustomerManager
    title "Customer Manager"

    window CustomerWindow
        title "Customers"
        bounds 40, 40, 900, 620

        menu MainMenu
            text "Main"
            on Command call MainMenu_Command
        end

        toolbar MainToolbar
            text "Customer toolbar"
            bounds 0, 0, 900, 36
            on Command call MainToolbar_Command
        end

        panel MainPanel
            bounds 0, 40, 900, 560
        end

        label NameLabel
            parent MainPanel
            text "Name"
            bounds 20, 20, 120, 24
        end

        entryfield NameEntry
            parent MainPanel
            value ""
            bounds 150, 18, 260, 28
            bind Value to Customer.Name
            on Changed call NameEntry_Changed
        end

        button SaveButton
            parent MainPanel
            text "Save"
            bounds 430, 18, 110, 30
            on Click call SaveButton_Click
        end

        browse CustomerBrowse
            parent MainPanel
            bounds 20, 70, 840, 420
            bind Rows to Customers
            property columns = "Code,Name,City,Balance"
            on RowSelected call CustomerBrowse_RowSelected
            on RowAccepted call CustomerBrowse_RowAccepted
        end
    end
end
```

Use stable ids. Do not rename a control casually after it has event handlers, validation rules, saved user layouts, report dependencies, or data bindings. Renaming a control is a source change and should be treated like renaming a field or procedure.

6 Property and metadata types

Safire control metadata uses a small set of predictable property types. The same type names appear in control descriptors, source declarations, visual property editors, validation rules, and runtime metadata.

Type	Description
`String`	Text value. Used for titles, labels, command ids, selected keys, column lists, source paths, and messages.
`Int`	Whole number. Used for x, y, width, height, row, and similar numeric values.
`Bool`	True or false. Used for visible, enabled, readOnly, editable, required, allowMove, and similar settings.
`Enum`	Named value from a fixed list. Used for settings such as startupPosition or validation kind.
`Font`	Named font or font descriptor. Used for text display and editable control appearance.
`Color`	Named color or color descriptor. Used for foreground and background display.
`BindingRef`	Reference to an application data path, query, table, row collection, selected row, or context value.
`ResourceRef`	Reference to a resource such as an image, icon, string file, or report definition.
`ProcedureRef`	Reference to a Safire procedure used as an event handler or command handler.
`Bounds`	Compound rectangle consisting of x, y, width, and height.
`ValidationRule`	Rule that checks a value or control before a command is accepted.
`LayoutPolicy`	Rules that decide which user layout changes are allowed for each control.

7 Common properties

Most visual controls share a common set of properties. Some controls add data, command, report, or validation properties. The common properties below should be understood before reading the individual control reference.

- **`controlId`** (`String`): Required unique control identity. Used by source, events, bindings, validation, resource links, diagnostics, and user layout overlays.
- **`x`** (`Int`): Left position in the parent coordinate space.
- **`y`** (`Int`): Top position in the parent coordinate space.
- **`width`** (`Int`): Control width.
- **`height`** (`Int`): Control height.
- **`visible`** (`Bool`): Determines whether the control is shown.
- **`enabled`** (`Bool`): Determines whether the user can interact with the control.
- **`font`** (`Font`): Text font for controls that display text.
- **`foregroundColor`** (`Color`): Text or foreground drawing color.
- **`backgroundColor`** (`Color`): Background color.
- **`parentId`** (`String`): The containing window, panel, group, or other allowed parent.

7.1 Bounds syntax

```
button SaveButton
    text "Save"
    bounds 430, 58, 110, 30   // x, y, width, height
end
```

7.2 Property write behavior

Safire distinguishes normal runtime property changes from user layout overlay changes.

- A **runtime-writable** property can be changed by the running application.
- A **layout-overlay-writable** property can be changed by permitted user personalization.
- A **source-only** property, such as a control id, should remain stable after the control is defined.

8 Window types and usage

Window type	Description
Main application window	The primary window for a desktop application. It usually owns the main menu, toolbar or command strip, status line, navigation area, and main work panels.
Dialog window	A focused window used to ask for a choice, confirm an operation, edit a small group of fields, or collect settings.
Browse window	A data window centered on a list of records. It usually supports selection, filtering, sorting, row count status, and open/edit commands.
Update window	A data entry window for one record or work item. It usually supports required fields, read-only keys, validation, Save, Cancel, and dirty-state handling.
Browse/update window	A combined data maintenance window with a browse and an edit form on the same surface.
Report preview window	A window that shows report output or report controls before printing, exporting, or saving.
Wizard window	A step-by-step window used to create a project item, report, form, or configuration through guided pages.
Utility window	A tool or support window used for status, diagnostics, logs, import, export, or auxiliary tasks.
Web screen	A browser-delivered screen generated from Safire application definitions where Safire remains the owner of events, validation, bindings, and data rules.

8.1 Main application window

```
window MainWindow
    title "Sales Console"
    bounds 100, 100, 1100, 700
    startupPosition CenterScreen
    on Open call MainWindow_Open
    on Close call MainWindow_Close
end
```

Use a main application window for the user's primary workspace. Add a menu for complete command discovery, a toolbar or command strip for frequent actions, a status line for feedback, and panels for the main work areas.

8.2 Browse/update window

```
window CustomerManagerWindow
    title "Customer Manager"

    browse CustomerBrowse
        source Customers
        columns CustomerNo, Name, City, Balance
        selection single
        on RowSelected call SelectCustomerForEdit
    end

    form CustomerEditForm
        source selected Customer
        field CustomerNo readonly true
        field Name editable true required true
        field City editable true required true
        field Balance editable true validate numeric
        command SaveCustomer validates CustomerEditForm writes Customers
        command CancelEdit reverts CustomerEditForm
    end
end
```

Use a browse/update window when the user must select a record and edit the details without leaving the task. This is the standard shape for customer, inventory, supplier, product, employee, and document maintenance screens.

8.3 Report preview window

```
window StatementPreviewWindow
    title "Customer Statement Preview"

    report StatementReport
        reportSource resources.StatementReport
        bind DataSource to SelectedCustomerStatementRows
        on BeforeRun call StatementReport_BeforeRun
        on AfterRun call StatementReport_AfterRun
    end
end
```

Use a report preview window when a user must review report output before printing, exporting, or sending. Keep the report source and data source explicit.

9 Built-in runtime component reference

The following controls are part of the current Safire runtime component model. They are available as source-owned application elements and can be loaded from Safire form metadata without requiring a visual design environment at runtime.

Control	Category	Can contain children	Allowed parents	Primary events
Window	Containers	Yes	None	Open, Close
Panel	Containers	Yes	Window, Panel	None
Label	Display	No	Window, Panel	None
EntryField	Input	No	Window, Panel	Changed
Button	Commands	No	Window, Panel	Click
Browse	Data	No	Window, Panel	RowSelected, RowAccepted
Grid	Data	No	Window, Panel	CellChanged

Control	Category	Can contain children	Allowed parents	Primary events
Menu	Commands	Yes	Window	Command
Toolbar	Commands	Yes	Window	Command
Report	Reporting	No	Window, Panel	BeforeRun, AfterRun

9.1 Window control

Component id: `Safire.UI.Window`
Category: Containers
Runtime class: `SafireUIRuntime.Window`
Factory: `CreateWindow`
Allowed parents: None - this is a top-level component.
Can contain children: Yes

Purpose: Top-level visual surface for an application screen, dialog, data form, or report shell.

Source form:

```
window CustomerWindow
    title "Customers"
    bounds 40, 40, 900, 620
    startupPosition CenterScreen
    on Open call CustomerWindow_Open
    on Close call CustomerWindow_Close
end
```

End user usage: The user opens the window from a menu, shortcut, workflow, or procedure. The window owns the screen title, contained controls, initial position, and close behavior.

Developer usage: Declare a window as the parent of screen controls. Give it a stable control id, a title, bounds, event handlers, and data context when needed.

Comments: A window is both a visual object and a source-owned application structure. It should be understandable from source without relying on designer memory.

Properties:

- `controlId` (`String`, required): source controlled; not layout overlay writable.
- `x` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `y` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `width` (`Int`, required; default `100`): runtime writable; layout overlay writable.
- `height` (`Int`, required; default `30`): runtime writable; layout overlay writable.
- `visible` (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- `enabled` (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- `font` (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- `foregroundColor` (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- `backgroundColor` (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- `title` (`String`, required; default `Window`): runtime writable; not layout overlay writable.
- `startupPosition` (`Enum`, optional; default `Default`): runtime writable; layout overlay writable.

Events:

- `Open` - `procedure(sender Window)`. Runtime dispatch: yes. Bubbles: no.
- `Close` - `procedure(sender Window)`. Runtime dispatch: yes. Bubbles: no.

Binding targets: `DataContext`

Supported resources: `Icon`, `Image`, `String`

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- allowMove: yes
- allowResize: yes
- allowHide: no
- allowReparent: no
- allowFont: yes
- allowColor: yes

9.2 Panel control

Component id: Safire.UI.Panel
Category: Containers
Runtime class: SafireUIRuntime.Panel
Factory: CreatePanel
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: Yes

Purpose: Container used to group child controls, define logical screen areas, and provide a parent for nested layout.

Source form:

```
panel MainPanel
    parent CustomerWindow
    bounds 0, 40, 900, 560
end
```

End user usage: The user normally does not interact with the panel directly. The panel gives related controls a common visual area and keeps child controls together.

Developer usage: Use a panel when a group of controls must move, resize, hide, or share a data context together. Place child controls inside the panel by setting their parent.

Comments: Panel containment matters. Hit testing, selection, move, resize, and reparent operations are resolved against the Safire control tree.

Properties:

- **controlId** (String, required): source controlled; not layout overlay writable.
- **x** (Int, required; default 0): runtime writable; layout overlay writable.
- **y** (Int, required; default 0): runtime writable; layout overlay writable.
- **width** (Int, required; default 100): runtime writable; layout overlay writable.
- **height** (Int, required; default 30): runtime writable; layout overlay writable.
- **visible** (Bool, required; default True): runtime writable; layout overlay writable.
- **enabled** (Bool, required; default True): runtime writable; not layout overlay writable.
- **font** (Font, optional; default System): runtime writable; layout overlay writable.
- **foregroundColor** (Color, optional; default SystemText): runtime writable; layout overlay writable.
- **backgroundColor** (Color, optional; default SystemWindow): runtime writable; layout overlay writable.

Events:

- No direct events are defined for this control in the current component descriptor.

Binding targets: DataContext

Supported resources: Image

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- allowMove: yes
- allowResize: yes
- allowHide: yes
- allowReparent: no
- allowFont: yes
- allowColor: yes

9.3 Label control

Component id: `Safire.UI.Label`
Category: Display
Runtime class: `SafireUIRuntime.Label`
Factory: `CreateLabel`
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Read-only text used for captions, field names, hints, and static instructions.

Source form:

```
label NameLabel
    parent MainPanel
    text "Name"
    bounds 20, 20, 120, 24
end
```

End user usage: The user reads the label as guidance. Labels should make fields and commands understandable without requiring a manual.

Developer usage: Use labels for captions and instructions. Bind the Text property only when the caption must change from data or language resources.

Comments: Short, direct labels improve data-entry speed. Avoid labels that only repeat internal field names.

Properties:

- **`controlId`** (`String`, required): source controlled; not layout overlay writable.
- **`x`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`y`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`width`** (`Int`, required; default `100`): runtime writable; layout overlay writable.
- **`height`** (`Int`, required; default `30`): runtime writable; layout overlay writable.
- **`visible`** (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- **`enabled`** (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- **`font`** (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- **`foregroundColor`** (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- **`backgroundColor`** (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- **`text`** (`String`, optional): runtime writable; not layout overlay writable.

Events:

- No direct events are defined for this control in the current component descriptor.

Binding targets: `Text`

Supported resources: None

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- `allowMove`: yes
- `allowResize`: yes
- `allowHide`: yes
- `allowReparent`: no
- `allowFont`: yes
- `allowColor`: yes

9.4 EntryField control

Component id: `Safire.UI.EntryField`
Category: Input
Runtime class: `SafireUIRuntime.EntryField`

Factory: `CreateEntryField`
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Single-value input control for text and simple typed values such as names, codes, phone numbers, and search values.

Source form:

```
entryfield NameEntry
    parent MainPanel
    value ""
    watermarkText "Enter customer name"
    bounds 150, 18, 260, 28
    bind Value to Customer.Name
    on Changed call NameEntry_Changed
end
```

End user usage: The user types or edits a value. Validation can run while editing, when focus changes, or before a command such as Save is accepted.

Developer usage: Bind Value or Text to a data field. Add Required, MinLength, numeric, range, or custom validation for values that must be checked before saving.

Comments: Entry fields should not silently accept invalid business data. Bind them to validation rules or apply validation before save commands.

Properties:

- `controlId` (`String`, required): source controlled; not layout overlay writable.
- `x` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `y` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `width` (`Int`, required; default `100`): runtime writable; layout overlay writable.
- `height` (`Int`, required; default `30`): runtime writable; layout overlay writable.
- `visible` (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- `enabled` (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- `font` (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- `foregroundColor` (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- `backgroundColor` (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- `value` (`String`, optional): runtime writable; not layout overlay writable.
- `watermarkText` (`String`, optional): runtime writable; not layout overlay writable.
- `readOnly` (`Bool`, optional; default `False`): runtime writable; not layout overlay writable.

Events:

- `Changed` - `procedure(sender EntryField, value String)`. Runtime dispatch: yes. Bubbles: yes.

Binding targets: `Value`, `Enabled`

Supported resources: None

Validation: Required rules supported: yes. Custom rules supported: yes.

Personal layout permissions:

- `allowMove`: yes
- `allowResize`: yes
- `allowHide`: yes
- `allowReparent`: no
- `allowFont`: yes
- `allowColor`: yes

9.5 Button control

Component id: `Safire.UI.Button`
Category: Commands
Runtime class: `SafireUIRuntime.Button`
Factory: `CreateButton`
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Command control used to run an action such as Save, Cancel, New, Delete, Search, Refresh, or Close.

Source form:

```
button SaveButton
    parent MainPanel
    text "Save"
    bounds 430, 18, 110, 30
    on Click call SaveButton_Click
end
```

End user usage: The user activates the button with the mouse, touch, keyboard, or command shortcut. The button should have one clear action.

Developer usage: Bind Click to a procedure. Keep business rules in the procedure or service it calls, not in display-only state.

Comments: Use buttons for commands, not for storing state. A disabled button should explain itself through nearby text or status feedback when possible.

Properties:

- **`controlId`** (`String`, required): source controlled; not layout overlay writable.
- **`x`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`y`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`width`** (`Int`, required; default `100`): runtime writable; layout overlay writable.
- **`height`** (`Int`, required; default `30`): runtime writable; layout overlay writable.
- **`visible`** (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- **`enabled`** (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- **`font`** (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- **`foregroundColor`** (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- **`backgroundColor`** (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- **`text`** (`String`, optional): runtime writable; not layout overlay writable.

Events:

- **`Click`** - `procedure(sender Button)`. Runtime dispatch: yes. Bubbles: yes.

Binding targets: `Text`, `Enabled`

Supported resources: `Image`, `Icon`

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- `allowMove`: yes
- `allowResize`: yes
- `allowHide`: yes
- `allowReparent`: no
- `allowFont`: yes
- `allowColor`: yes

9.6 Browse control

Component id: `Safire.UI.Browse`
Category: Data
Runtime class: `SafireUIRuntime.Browse`
Factory: `CreateBrowse`
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Business data list that presents rows from a bound data source and lets the user select or accept a row.

Source form:

```
browse CustomerBrowse
    parent MainPanel
    bounds 20, 70, 840, 420
    dataSource Customers
    columns "Code,Name,City,Balance"
    bind Rows to Customers
    bind SelectedKey to Customer.CustomerNo
    on RowSelected call CustomerBrowse_RowSelected
    on RowAccepted call CustomerBrowse_RowAccepted
end
```

End user usage: The user scrolls, selects, filters, sorts, or accepts a row. A browse is usually paired with an edit form, detail panel, or command buttons.

Developer usage: Bind Rows to a data source and SelectedKey to a record key. Use RowSelected for detail refresh and RowAccepted for open/edit workflows.

Comments: A browse is not just a grid. It represents a business row selection workflow and should have clear row identity.

Properties:

- `controlId` (`String`, required): source controlled; not layout overlay writable.
- `x` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `y` (`Int`, required; default `0`): runtime writable; layout overlay writable.
- `width` (`Int`, required; default `100`): runtime writable; layout overlay writable.
- `height` (`Int`, required; default `30`): runtime writable; layout overlay writable.
- `visible` (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- `enabled` (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- `font` (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- `foregroundColor` (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- `backgroundColor` (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- `dataSource` (`BindingRef`, optional): runtime writable; not layout overlay writable.
- `columns` (`String`, optional): runtime writable; layout overlay writable.
- `selectedKey` (`String`, optional): runtime writable; not layout overlay writable.

Events:

- `RowSelected` - `procedure(sender Browse, key String)`. Runtime dispatch: yes. Bubbles: yes.
- `RowAccepted` - `procedure(sender Browse, key String)`. Runtime dispatch: yes. Bubbles: yes.

Binding targets: `Rows`, `SelectedKey`

Supported resources: None

Validation: Required rules supported: no. Custom rules supported: yes.

Personal layout permissions:

- `allowMove`: yes
- `allowResize`: yes
- `allowHide`: yes

- allowReparent: no
- allowFont: yes
- allowColor: yes

9.7 Grid control

Component id: Safire.UI.Grid
Category: Data
Runtime class: SafireUIRuntime.Grid
Factory: CreateGrid
Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Tabular control for row-and-column data where cell-level editing or spreadsheet-like interaction is needed.

Source form:

```
grid LineGrid
    parent MainPanel
    bounds 20, 100, 760, 300
    dataSource InvoiceLines
    columns "ItemCode,Description,Quantity,Price,Total"
    editable true
    bind Rows to InvoiceLines
    bind Columns to InvoiceLineColumns
    on CellChanged call LineGrid_CellChanged
end
```

End user usage: The user works with tabular cells. A grid is best for repeated detail lines, adjustable values, and row/column review.

Developer usage: Bind Rows and Columns. Use CellChanged when each edit must update the working model immediately.

Comments: Use a grid when the user thinks in cells. Use a browse when the user thinks in records.

Properties:

- **controlId** (String, required): source controlled; not layout overlay writable.
- **x** (Int, required; default 0): runtime writable; layout overlay writable.
- **y** (Int, required; default 0): runtime writable; layout overlay writable.
- **width** (Int, required; default 100): runtime writable; layout overlay writable.
- **height** (Int, required; default 30): runtime writable; layout overlay writable.
- **visible** (Bool, required; default True): runtime writable; layout overlay writable.
- **enabled** (Bool, required; default True): runtime writable; not layout overlay writable.
- **font** (Font, optional; default System): runtime writable; layout overlay writable.
- **foregroundColor** (Color, optional; default SystemText): runtime writable; layout overlay writable.
- **backgroundColor** (Color, optional; default SystemWindow): runtime writable; layout overlay writable.
- **dataSource** (BindingRef, optional): runtime writable; not layout overlay writable.
- **columns** (String, optional): runtime writable; layout overlay writable.
- **editable** (Bool, optional; default False): runtime writable; not layout overlay writable.

Events:

- **CellChanged** - procedure(sender Grid, row Int, column String, value String). Runtime dispatch: yes. Bubbles: yes.

Binding targets: Rows, Columns

Supported resources: None

Validation: Required rules supported: no. Custom rules supported: yes.

Personal layout permissions:

- allowMove: yes
- allowResize: yes
- allowHide: yes
- allowReparent: no
- allowFont: yes
- allowColor: yes

9.8 Menu control

Component id: Safire.UI.Menu
Category: Commands
Runtime class: SafireUIRuntime.Menu
Factory: CreateMenu
Allowed parents: Safire.UI.Window
Can contain children: Yes

Purpose: Window-level command structure used to group commands into user-facing command categories.

Source form:

```
menu MainMenu
    text "Main"
    command "File.Save"
    command "File.Close"
    on Command call MainMenu_Command
end
```

End user usage: The user chooses commands from grouped menu headings. Menus should expose complete application capability, including commands not always visible on a toolbar.

Developer usage: Use command identifiers that match application actions. The Command event receives the chosen command id.

Comments: Menus are the safest place for full command discovery. Toolbar and shortcut commands should mirror menu command ids where practical.

Properties:

- controlId (String, required): source controlled; not layout overlay writable.
- text (String, required; default Menu): runtime writable; not layout overlay writable.

Events:

- Command - procedure(sender Menu, commandId String). Runtime dispatch: yes. Bubbles: yes.

Binding targets: Enabled

Supported resources: Icon

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- allowMove: yes
- allowResize: yes
- allowHide: yes
- allowReparent: no
- allowFont: yes
- allowColor: yes

9.9 Toolbar control

Component id: Safire.UI.Toolbar
Category: Commands
Runtime class: SafireUIRuntime.Toolbar

Factory: `CreateToolbar`
Allowed parents: Safire.UI.Window
Can contain children: Yes

Purpose: Window-level command strip with quick access to frequent commands, usually with text, icons, or both.

Source form:

```
toolbar MainToolbar
    text "Customer toolbar"
    bounds 0, 0, 900, 36
    command "Customer.New"
    command "Customer.Save"
    command "Customer.Delete"
    on Command call MainToolbar_Command
end
```

End user usage: The user clicks common commands quickly. Toolbars should not replace menus; they make frequent actions easier to reach.

Developer usage: Use toolbar commands for high-frequency actions. Keep command ids consistent with matching menu commands.

Comments: Toolbars are productivity controls. Keep captions and icons consistent with the same command in menus and forms.

Properties:

- **`controlId`** (`String`, required): source controlled; not layout overlay writable.
- **`x`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`y`** (`Int`, required; default `0`): runtime writable; layout overlay writable.
- **`width`** (`Int`, required; default `100`): runtime writable; layout overlay writable.
- **`height`** (`Int`, required; default `30`): runtime writable; layout overlay writable.
- **`visible`** (`Bool`, required; default `True`): runtime writable; layout overlay writable.
- **`enabled`** (`Bool`, required; default `True`): runtime writable; not layout overlay writable.
- **`font`** (`Font`, optional; default `System`): runtime writable; layout overlay writable.
- **`foregroundColor`** (`Color`, optional; default `SystemText`): runtime writable; layout overlay writable.
- **`backgroundColor`** (`Color`, optional; default `SystemWindow`): runtime writable; layout overlay writable.
- **`text`** (`String`, optional): runtime writable; not layout overlay writable.

Events:

- **`Command`** - `procedure(sender Toolbar, commandId String)`. Runtime dispatch: yes. Bubbles: yes.

Binding targets: `Enabled`

Supported resources: `Icon`, `Image`

Validation: Required rules supported: no. Custom rules supported: no.

Personal layout permissions:

- `allowMove`: yes
- `allowResize`: yes
- `allowHide`: yes
- `allowReparent`: no
- `allowFont`: yes
- `allowColor`: yes

9.10 Report control

Component id: `Safire.Report.Report`
Category: Reporting
Runtime class: `SafireReportRuntime.Report`
Factory: `CreateReport`

Allowed parents: Safire.UI.Window, Safire.UI.Panel
Can contain children: No

Purpose: Report runtime control that runs or previews a report definition inside a Safire window or panel.

Source form:

```
report StatementReport
    parent MainPanel
    reportSource resources.StatementReport
    bind DataSource to CustomerStatementRows
    on BeforeRun call StatementReport_BeforeRun
    on AfterRun call StatementReport_AfterRun
end
```

End user usage: The user previews, prints, exports, or runs a report from a window command or report control.

Developer usage: Bind reportSource to a report definition resource and dataSource to the data set, query, or selected record context.

Comments: Reports should receive data through explicit bindings so the same report can be previewed, printed, or reused from different windows.

Properties:

- `controlId` (`String`, required): source controlled; not layout overlay writable.
- `reportSource` (`ResourceRef`, required): runtime writable; not layout overlay writable.
- `dataSource` (`BindingRef`, optional): runtime writable; not layout overlay writable.

Events:

- `BeforeRun` - `procedure(sender Report)`. Runtime dispatch: yes. Bubbles: no.
- `AfterRun` - `procedure(sender Report)`. Runtime dispatch: yes. Bubbles: no.

Binding targets: `DataSource`

Supported resources: `ReportDefinition`, `Image`, `String`

Validation: Required rules supported: yes. Custom rules supported: yes.

Personal layout permissions:

- `allowMove`: no
- `allowResize`: no
- `allowHide`: yes
- `allowReparent`: no
- `allowFont`: no
- `allowColor`: no

10 Additional window surface elements

Some Safire application screens use compound surface elements that are not only single controls. They are part of the application window pattern and are documented here because end users and developers work with them frequently.

10.1 Form

Purpose: A compound surface for editing one record or business object. A form contains fields, commands, dirty state, validation, and save/cancel behavior.

Usage: Use a form when a user edits a selected record or creates a new record. Bind fields to data paths and validate before saving.

```
form CustomerEditForm
    source selected Customer
    field CustomerNo readonly true
```

```
    field Name editable true required true
    field City editable true required true
    field Balance editable true validate numeric
    dirtyState true
    command SaveCustomer validates CustomerEditForm writes Customers
    command CancelEdit reverts CustomerEditForm
end
```

10.2 Field

Purpose: A named edit/display element inside a form. A field can be editable, read-only, required, numeric, boolean, date, or custom validated.

Usage: Use fields for business data entry. Keep field names aligned with dictionary or data model names where practical.

10.3 CommandStrip

Purpose: A row or group of command buttons usually placed near the top of a window.

Usage: Use a command strip for frequent actions such as New, Save, Delete, Refresh, Clear Filter, and Close.

```
commandStrip MainCommands
    command NewCustomer
    command SaveCustomer
    command DeleteCustomer
    command ClearFilter
    command CloseWindow
end
```

10.4 StatusBar

Purpose: A feedback area that shows current state, row counts, validation errors, save results, or diagnostic hints.

Usage: Use a status bar to keep users informed without interrupting workflow.

```
statusBar MainStatus
    text "Ready"
end
```

10.5 GroupBox

Purpose: A visual grouping container with a caption. It helps users see related fields as a named group.

Usage: Use a group box for related settings, address fields, pricing fields, or grouped options.

```
groupbox AddressGroup
    parent MainPanel
    text "Address"
    bounds 20, 120, 420, 160
end
```

10.6 SearchBox

Purpose: An entry field used as a filter input for a browse or grid.

Usage: Use a search box with clear filter and no-match feedback.

11 Attribute reference

Attribute	Description
`bounds`	Position and size. Written as x, y, width, height. It is the most common layout attribute.
`title`	Window title displayed to the user.
`text`	Displayed caption for labels, buttons, menus, toolbars, and similar controls.
`value`	Editable or current value of an input control.
`visible`	Shows or hides a control. Required controls may not be hideable by user layout rules.
`enabled`	Allows or blocks interaction with a control while still showing it.
`readOnly`	Allows selection or copying while preventing editing.
`editable`	Allows a grid, field, or form area to accept user changes.
`watermarkText`	Hint text displayed in an empty entry field.
`startupPosition`	Initial window placement mode such as default or centered.
`parent`	Container that owns a child control.
`dataSource`	Binding reference to rows, records, query results, or data context.
`columns`	Column list for a browse or grid.
`selectedKey`	Current selected row key for browse-style selection.
`reportSource`	Resource reference to a report definition.
`source`	Data or context source for a form, browse, report, or field group.
`required`	Validation rule that blocks empty values.
`validate numeric`	Validation rule that requires a numeric value.
`dirtyState`	Tracks whether a form has unsaved changes.

12 Event reference

Event	Applies to	Usage
`Open`	Window	Runs when a window is opened. Use it to load data, apply user layout, set initial focus, or refresh status.
`Close`	Window	Runs when a window is closing. Use it to check dirty state, save settings, or ask the user to confirm cancellation.
`Click`	Button	Runs when a button is activated.
`Changed`	EntryField	Runs when an entry field value changes.
`Command`	Menu, Toolbar	Runs when a menu or toolbar command is chosen. The handler receives a command id.
`RowSelected`	Browse	Runs when a row selection changes. Use it to refresh a detail form.
`RowAccepted`	Browse	Runs when a row is accepted, usually by double-click, Enter, or an Open/Edit command.
`CellChanged`	Grid	Runs when a grid cell is edited.
`BeforeRun`	Report	Runs before report execution. Use it to prepare filters, titles, or data context.

Event	Applies to	Usage
`AfterRun`	Report	Runs after report execution. Use it for status messages or cleanup.
`editChanged`	Form	Runs when a field changes and dirty state should be set.
`save`	Form or window	Runs when a save command is accepted.
`cancel`	Form or window	Runs when a cancel command is accepted.

12.1 Event handler examples

```
procedure SaveButton_Click(sender Button)
    result = Validate(CustomerEditForm)
    if result.IsValid
        SaveCustomer(CustomerEditForm)
        MainStatus.Text = "Customer saved"
    else
        MainStatus.Text = result.Message
    end
end

procedure CustomerBrowse_RowSelected(sender Browse, key String)
    CustomerEditForm.LoadByKey(key)
end
```

13 Data binding guide

Data binding connects controls to data without hiding application rules in the visual surface. Bindings should be clear enough that a developer can tell what data a window reads and writes by inspecting the source and metadata.

13.1 Binding modes

Mode	Use
`OneWay`	Data flows from source to control. Use for labels, read-only fields, browses, reports, and status display.
`TwoWay`	Data flows from source to control and from control back to source. Use for editable fields.
`Command`	A user action runs a procedure rather than moving data directly.

13.2 Binding examples

```
entryfield CustomerNameEntry
    bind Value to Customer.Name        // Two-way by default for editable data
end

browse CustomerBrowse
    bind Rows to Customers             // Rows come from a data source
    bind SelectedKey to Customer.Id     // Current selection identity
end

report StatementReport
```

```
    bind DataSource to StatementRows    // Report receives explicit data
end
```

14 Validation guide

Validation should protect business data and help users correct mistakes quickly. Validation can run before Save, before command execution, during field changes, or during form acceptance.

Rule	Description
`Required`	Value must not be empty.
`MinLength`	Text must be at least a specified length.
`NumberRange`	Numeric value must be within a specified range.
`Boolean`	Value must be true or false.
`Numeric`	Value must be a valid number.
`Custom`	Application-defined procedure validates the value or form.

```
validation CustomerRules
    rule Customer.Name Required message "Customer name is required."
    rule Customer.Name MinLength 2 message "Customer name must be at least two characters."
    rule Customer.CreditLimit NumberRange 0 100000 message "Credit limit out of range."
    rule Customer.Active Boolean message "Active must be true or false."
end

command SaveCustomer
    validates CustomerRules
    writes Customers
end
```

Validation messages should be written for the user. Avoid messages that expose internal field codes or procedure names unless the message is intended for support staff.

15 Resources guide

Use resources for reusable assets that controls and reports need. Resources should be declared and packaged with the application so missing files can be diagnosed before users need them.

```
resources CustomerResources
    image saveIcon path "resources/save.png" required true
    report StatementReport path "reports/StatementReport.sfr" required true
    strings CustomerStrings path "resources/customer_strings.json" required false
end

toolbar MainToolbar
    resource image saveIcon
end

report StatementReport
    reportSource resources.StatementReport
end
```

16 User layout and personalization

User layout personalization lets users adjust permitted parts of a window without changing application source. The application defines what may move, resize, hide, reparent, or change style.

16.1 Layout policy example

```
layoutPolicy CustomerRuntimeWindow
    control CustomerNameEntry allowMove true allowResize true allowHide false allowReparent false requiredV
    control SaveButton allowMove true allowResize true allowHide false allowReparent false requiredVisible
    control CustomerBrowse allowMove true allowResize true allowHide true allowReparent false requiredVisib
    control StatementReport allowMove true allowResize true allowHide true allowReparent true requiredVisib
end
```

16.2 Layout commands example

```
layoutCommands CustomerRuntimeWindow for "local-user"
    move SaveButton x 455 y 22
    resize SaveButton width 130 height 34
    hide StatementReport
    resize CustomerBrowse width 540 height 430
    save
end
```

Required controls should generally not be hideable. Optional report panels, preview panels, helper panels, and secondary grids are good candidates for personalization.

17 Designer and runtime interaction rules

Safire editing and runtime behavior uses the same control identity and control tree. These rules are important for designers and developers because they keep visual editing, source save, reload, and runtime execution consistent.

Rule	Meaning
Hit-test stack	When controls overlap, Safire can resolve the full stack of controls under the pointer and cycle through them instead of always selecting only the topmost control.
Parent/child correctness	A control inside a panel or group remains selectable as a child of that container.
Selection cycling	Repeated selection over an overlapping area can cycle through the candidate controls.
Move isolation	Dragging one control should not move unrelated controls.
Resize isolation	Resizing one control should not resize the parent window unless the user explicitly resizes the parent.
Delta property refresh	During drag, only position-related properties need to refresh. During resize, only size-related properties need to refresh. The full property tree should not be refreshed for every mouse movement.
Source round trip	Saving and reopening a window must preserve ids, parentage, events, bindings, validation, and user-visible layout.

18 Business window patterns

18.1 Customer manager pattern

The customer manager pattern uses a menu, command strip, status bar, browse, edit form, validation, data persistence, and user-friendly commands. It is suitable for master file maintenance.

```
window CustomerManagerWindow
    menu File
        command CloseWindow
```

```
    end
    menu Data
        command SaveCustomer
        command ReloadData
        command CancelEdit
    end
    commandStrip
        button SaveCustomer
        button ReloadData
        button CancelEdit
        button CloseWindow
    end
    browse CustomerBrowse
        source CustomerSampleData
        columns CustomerNo, Name, City, Balance
        selection single
        on selectionChanged call RowSelectionChanged
    end
    form CustomerEditForm
        field CustomerNo readonly
        field Name editable required
        field City editable required
        field Balance editable numeric
        field Email editable
        field Phone editable
        dirtyState true
        command SaveCustomer validates saves reloads
        command ReloadData loads workingCopy
        command CancelEdit restores selectedRow
    end
    statusBar StatusLine
end
```

18.2 Inventory manager pattern

The inventory manager pattern uses the same browse/update structure but changes fields, validation, and commands to match inventory work. It is suitable for item master maintenance, stock lookup, simple adjustment workflows, and product reference data.

```
window InventoryManagerWindow
    title "Inventory Manager"
    browse InventoryBrowse
        source InventoryItems
        columns ItemCode, Description, Category, QuantityOnHand
        selection single
        on RowSelected call SelectInventoryItem
    end
    form InventoryEditForm
        source selected InventoryItem
        field ItemCode readonly true
        field Description editable true required true
        field Category editable true
        field QuantityOnHand editable true validate numeric
        command SaveInventoryItem validates InventoryEditForm writes InventoryItems
        command CancelInventoryEdit reverts InventoryEditForm
    end
end
```

19 Developer best practices

- Give every control a stable, meaningful id.
- Keep user-facing text readable and specific.
- Use panels and group boxes to express containment instead of relying only on coordinates.
- Bind controls to data paths explicitly.
- Validate before writing data.
- Use RowSelected for detail refresh and RowAccepted for open/edit workflows.
- Keep command ids consistent across menus, toolbars, command strips, and shortcuts.
- Use status bars for non-blocking feedback.
- Use message boxes only when the user must make a decision or cannot continue.
- Do not store business truth only in visual state; use data models, validation, procedures, and source-owned definitions.
- Keep reports data-bound so they can be run from different windows and data contexts.
- Use user layout policies to permit safe personalization without breaking the application.

20 Troubleshooting

Symptom	Check
A button does nothing	Check that the Click event is bound to the correct procedure and that the button is enabled.
A field does not save	Check the binding path, validation result, dirty state, and save command.
A browse selection does not refresh the form	Check RowSelected binding and selected key mapping.
A required field message appears	Enter a value or adjust the validation rule if the field should not be required.
A control cannot be hidden by the user	Check the layout policy. Required controls may be protected.
A report does not run	Check reportSource, dataSource, and required report resources.
A nested control is hard to select	Use selection cycling or select from the control tree.
A layout change is not saved	Check whether the control allows move, resize, hide, reparent, or style changes.

21 Glossary

Term	Meaning
Application	The complete user-facing Safire program.
Binding	Connection between a control property and application data.
Bounds	The x, y, width, and height of a control.
Command	A named action such as Save, Delete, Refresh, Close, or Print.
Control	A named visual or functional element in a Safire window.
Control tree	The parent/child structure of windows and controls.
Data context	The data object or row context used by a window, panel, form, or control.
Dirty state	A flag indicating unsaved changes.
Event	A user action or runtime notification that calls a procedure.
Form	A compound edit surface for one record or object.

Term	Meaning
Layout overlay	User-specific layout changes applied on top of source-defined layout.
Resource	Named asset such as an image, icon, strings file, or report definition.
Validation	Rules that check values before commands continue.
Window	A top-level or task-level visual surface.

22 Appendix A. Component property summary

22.1 Window

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `title` - type `String`; required; runtime writable; not layout overlay writable. Default: `Window`.
- `startupPosition` - type `Enum`; optional; runtime writable; layout overlay writable. Default: `Default`.

22.2 Panel

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.

22.3 Label

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `text` - type `String`; optional; runtime writable; not layout overlay writable.

22.4 EntryField

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.

- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `value` - type `String`; optional; runtime writable; not layout overlay writable.
- `watermarkText` - type `String`; optional; runtime writable; not layout overlay writable.
- `readOnly` - type `Bool`; optional; runtime writable; not layout overlay writable. Default: `False`.

22.5 Button

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `text` - type `String`; optional; runtime writable; not layout overlay writable.

22.6 Browse

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `dataSource` - type `BindingRef`; optional; runtime writable; not layout overlay writable.
- `columns` - type `String`; optional; runtime writable; layout overlay writable.
- `selectedKey` - type `String`; optional; runtime writable; not layout overlay writable.

22.7 Grid

- `controlId` - type `String`; required; source controlled; not layout overlay writable.
- `x` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `y` - type `Int`; required; runtime writable; layout overlay writable. Default: `0`.
- `width` - type `Int`; required; runtime writable; layout overlay writable. Default: `100`.
- `height` - type `Int`; required; runtime writable; layout overlay writable. Default: `30`.
- `visible` - type `Bool`; required; runtime writable; layout overlay writable. Default: `True`.
- `enabled` - type `Bool`; required; runtime writable; not layout overlay writable. Default: `True`.
- `font` - type `Font`; optional; runtime writable; layout overlay writable. Default: `System`.
- `foregroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemText`.
- `backgroundColor` - type `Color`; optional; runtime writable; layout overlay writable. Default: `SystemWindow`.
- `dataSource` - type `BindingRef`; optional; runtime writable; not layout overlay writable.
- `columns` - type `String`; optional; runtime writable; layout overlay writable.
- `editable` - type `Bool`; optional; runtime writable; not layout overlay writable. Default: `False`.

22.8 Menu

- **controlId** - type String; required; source controlled; not layout overlay writable.
- **text** - type String; required; runtime writable; not layout overlay writable. Default: Menu.

22.9 Toolbar

- **controlId** - type String; required; source controlled; not layout overlay writable.
- **x** - type Int; required; runtime writable; layout overlay writable. Default: 0.
- **y** - type Int; required; runtime writable; layout overlay writable. Default: 0.
- **width** - type Int; required; runtime writable; layout overlay writable. Default: 100.
- **height** - type Int; required; runtime writable; layout overlay writable. Default: 30.
- **visible** - type Bool; required; runtime writable; layout overlay writable. Default: True.
- **enabled** - type Bool; required; runtime writable; not layout overlay writable. Default: True.
- **font** - type Font; optional; runtime writable; layout overlay writable. Default: System.
- **foregroundColor** - type Color; optional; runtime writable; layout overlay writable. Default: SystemText.
- **backgroundColor** - type Color; optional; runtime writable; layout overlay writable. Default: SystemWindow.
- **text** - type String; optional; runtime writable; not layout overlay writable.

22.10 Report

- **controlId** - type String; required; source controlled; not layout overlay writable.
- **reportSource** - type ResourceRef; required; runtime writable; not layout overlay writable.
- **dataSource** - type BindingRef; optional; runtime writable; not layout overlay writable.

23 Appendix B. Example full window

```
application CustomerManager
    title "Customer Manager"

    resources CustomerResources
        image saveIcon path "resources/save.png" required true
        report StatementReport path "reports/StatementReport.sfr" required true
    end

    validation CustomerRules
        rule Customer.Name Required message "Customer name is required."
        rule Customer.Name MinLength 2 message "Customer name must be at least two characters."
        rule Customer.Balance Numeric message "Balance must be a number."
    end

    window CustomerRuntimeWindow
        title "Customer Manager"
        bounds 100, 100, 920, 620
        on Open call CustomerWindow_Open
        on Close call CustomerWindow_Close

        menu MainMenu
            text "Main"
            command "Customer.New"
            command "Customer.Save"
            command "Customer.Delete"
            command "File.Close"
            on Command call MainMenu_Command
        end

        toolbar MainToolbar
```

```
    text "Customer toolbar"
    bounds 0, 0, 900, 30
    resource image saveIcon
    command "Customer.Save"
    on Command call MainToolbar_Command
end

panel MainPanel
    bounds 0, 30, 900, 560
end

label NameLabel
    parent MainPanel
    text "Customer name"
    bounds 20, 20, 120, 24
end

entryfield CustomerNameEntry
    parent MainPanel
    bounds 150, 18, 260, 28
    watermarkText "Enter customer name"
    bind Value to Customer.Name
    on Changed call CustomerNameEntry_Changed
end

button SaveButton
    parent MainPanel
    text "Save"
    bounds 430, 18, 100, 30
    on Click call SaveButton_Click
end

browse CustomerBrowse
    parent MainPanel
    bounds 20, 70, 510, 440
    bind Rows to Customers
    bind SelectedKey to Customer.CustomerNo
    columns "CustomerNo,Name,City,Balance"
    on RowSelected call CustomerBrowse_RowSelected
    on RowAccepted call CustomerBrowse_RowAccepted
end

grid CustomerGrid
    parent MainPanel
    bounds 550, 70, 320, 260
    bind Rows to Customers
    bind Columns to CustomerColumns
end

report StatementReport
    parent MainPanel
    bounds 550, 350, 320, 160
    reportSource resources.StatementReport
    bind DataSource to CustomerStatementRows
    on BeforeRun call StatementReport_BeforeRun
    on AfterRun call StatementReport_AfterRun
end
```

```
        statusBar StatusLine
            text "Ready"
        end
    end
end
```