Client Namespace Reference

Access per-client state, identity, navigation and mobile capture via scripts.

The **Client Namespace** exposes the state of one connected client station to .NET scripts. Use **`@Client`** to reference the namespace, followed by the accessors listed below.

Every connected client has its own instance, so values such as the signed-in user, the culture, the visible display and the device flags describe *that* client. The namespace is available in Display Code Behind, in Dynamic Properties applied to UI elements, and in expressions.

Display objects themselves (`Open`, `OpenModal`, `NewPopup`, `Close`, custom properties, zoom and scroll state) belong to the Display Namespace.

Child Namespaces
Client Members
    Federated sign-in
Client.Session Members
    Session properties
    Session methods
    Mobile capture methods
Relocated Members
Internal and Auxiliary Members

Child Namespaces

Sub-namespaces grouping related client state. Reach each via its qualified path.

Path	Description
`@Client.Session` (37 members)	Connection metadata (computer name and IP, local or remote, web browser or Smart Client), logon operations, password management, geolocation, and the mobile capture methods. See the **Client.Session Members** section below.
`@Client.UI`	User-interface state (current page, layout, navigation history, tooltip options, simulation mode) and the methods to open, close, print and navigate displays.
`@Client.Context`	The currently selected asset (name, path, ISA-95 level names and IDs) plus per-client general-purpose values (`DigitalValue`, `NumericValue`, `TextValue`, `DateTimeStart`, `DateTimeEnd`). Use as the shared bus between the Assets Tree and display controls.
`@Client.DateTimeInfo`	The workstation's current date and time together with year, month, day, hour, minute, second and `TimeOfDay` components for binding directly to displays.
`@Client.AlarmPage`	Per-client alarm-view filters, selected area, date range and the page-level acknowledge-all trigger that drives the default Alarm Viewer.
`@Client.TrendChart`	Per-client trend-chart state and operations (active pen set, time range, cursor position) that bind to TrendChart display controls.
`@Client.DrillingChart`	Per-client drilling-chart state and operations that bind to DrillingChart display controls.
`@Client.QueryParams`	Read-only, case-insensitive view over the parsed query string or command-line arguments delivered to the client at startup. Values are URL-decoded, and the indexer returns null for absent keys.
`@Client.Messaging`	Reference to the client message queue.

Client Members

Direct properties and methods on `@Client`.

Member	Kind	Type	Description
`AlarmBeepOff`	Property	`Boolean`	Gets or sets whether this client's alarm beep is currently silenced. Set to true to stop the local beep for the duration of the unacknowledged alarm condition. A subsequent new alarm re-raises the beep unless it remains suppressed.
`CultureInfo`	Property	`String`	Gets or sets the client's culture, for example `en-US` or `pt-BR`. Controls how numbers, dates and currency are formatted on the client and which localization strings are resolved. Setting this property updates the active culture immediately.
`CurrentUser`	Property	`Reference`	Gets the client's current user as a reference to the Security user object.
`Localization`	Property	`String`	Gets or sets the name of the active localization dictionary used to translate text shown on displays and in messages. Empty selects the solution's default dictionary. Use `Locale` to look up a single string programmatically.

`SelectedPage`	Property	`String`	Gets or sets the page name selected in the built-in SelectPage dialog. Set from script to preselect an entry before the dialog is shown.
`Theme`	Property	`String`	Gets or sets the active visual theme dictionary name for this client, controlling display colors, brushes and fonts. Setting this property applies the theme immediately if the visualization service can resolve the named theme.
`Units`	Property	`String`	Gets or sets the name of the engineering-units dictionary applied when rendering tag values on displays. Empty selects the solution default. Use this to switch between unit systems, for example metric and imperial, at runtime per client.
`UserName`	Property	`String`	Gets the name of the user currently signed in on this client station, as recorded by the Security module on the most recent successful logon. Empty when no user is signed in. The built-in Guest user shows as `Guest`. Read-only from script.
`ChangeUserPasswordAsync`	Method	`Task<Boolean>`	Asynchronously changes a runtime user's password after validating the current password. Prefer `ChangeUserPasswordWithStatusAsync` when callers need the specific failure reason.
`ChangeUserPasswordWithStatusAsync`	Method	`Task<Boolean>`	Asynchronously changes a runtime user's password and returns a detailed status code through the `status` parameter, so the caller can distinguish a wrong password from a policy rejection.
`GetTDisplay`	Method	`Object`	Returns the live TDisplay instance for the named open display on this client. Use to drive ad-hoc XAML manipulation, find named elements, or inspect display state from script. Returns null when the display is not open or the visualization service is unavailable.
`GetTDisplays`	Method	`Object[]`	Returns every TDisplay instance currently open on this client, including the main panel, popups and secondary panels. Returns an empty array when no displays are open.
`Locale`	Method	`String`	Translates a string through the active localization dictionary. Returns the translated text for the client's current culture, or the original string when no entry matches.
`LogOnAsync`	Method	`Task<Int32>`	Asynchronously signs the client in with the supplied credentials, replacing any currently signed-in user. On success, clears the login dialog's input fields and updates the session's logon timestamp. Returns 0 on success, or a non-zero security error code.
`LogOnGuestAsync`	Method	`Task<Int32>`	Asynchronously signs the client in as the built-in Guest user, replacing any currently signed-in user. Updates the session's logon timestamp and resets the inactivity clock. Returns 0 on success.
`LogOnSSOAsync`	Method	`Task<Int32>`	Asynchronously signs the client in through OIDC federated sign-in, delegating authentication and any MFA enforcement to the configured Identity Provider. See **Federated sign-in** below for the per-client return-value contract.
`OpenDisplay`	Method	`Boolean`	Opens the named display in this client's main panel, the simplest navigation entry point. When the name ends in `.sg` the method first switches the layout, then opens the layout's initial display. Use the `@Client.UI.OpenDisplay` overloads when label-list overrides or position arguments are needed.
`OpenLayout`	Method	`Boolean`	Opens a configured layout on the client, replacing the current layout. The layout's defined initial displays are loaded into their respective panels.
`OpenWebDisplays`	Method	`Int32`	Launches this client's web-display front end against the runtime server. Pass `true` to launch with `https://`. Returns 0 on success, or non-zero on launch failure.
`OpenWebUrl`	Method	`Int32`	Opens an arbitrary URL in this client's web view or external browser. The URL must include the scheme. Returns 0 on success, or non-zero on launch failure.

Federated sign-in

`@Client.LogOnSSOAsync` is the supported entry point for OIDC sign-in on every client flavor. On Rich Client and Smart Client it launches the system browser, listens on a loopback port for the Identity Provider redirect, exchanges the PKCE-protected authorization code for an id_token, and binds the runtime session. On the HTML5 Web Client the same call drives the server-side OIDC start handler, preferring a popup and falling back to navigating the current tab.

The return value differs by client, because the two legs differ. On Rich Client and Smart Client, `0` means the sign-in completed. On the HTML5 Web Client, `0` means the sign-in completed when the popup path ran, but only that navigation started when the popup was blocked and the same-tab fallback took over. Do not write `if (rc == 0) { /* logged in */ }` in a Display that also runs on the web client.

The same-tab fallback needs a return leg: the callback redirects back with `?identityToken=<GUID>`, and a Display Code Behind reads `@Client.QueryParams["identityToken"]` and calls `@Security.LogOnWithTokenAsync` to bind the session. Pass that value through unchanged. The popup path needs no return leg.

```
// Rich Client / Smart Client Code Behind
int rc = await @Client.LogOnSSOAsync();           // single Active OIDC provider
int rc = await @Client.LogOnSSOAsync("EntraID");  // explicit provider
if (rc != 0) @Client.UI.ShowMessageBox("SSO failed: " + rc);

// HTML5 Web Client return leg, needed only for the same-tab fallback
string token = @Client.QueryParams["identityToken"];
if (!string.IsNullOrEmpty(token))
    await @Security.LogOnWithTokenAsync("EntraID", token);
```

Client.Session Members

Per-client session state reached as `@Client.Session`. Each connected client has its own instance, so these values describe that client: its address, host platform, connection origin, inactivity timer, and loaded-tag counters. The methods operate against the Security module on that session's behalf.

Session properties

Member	Type	Description
`ComputerIP`	`String`	Gets the IP address of the client workstation as reported to the server at connection time.
`ComputerName`	`String`	Gets the network name of the client workstation as reported to the server at connection time.
`GeoLocationAccuracy`	`Double`	Gets the current location horizontal accuracy, in meters. Available on mobile devices when running in the Mobile App client.
`GeoLocationAdvanced`	`String`	Gets current location advanced information. Available on mobile devices when running in the Mobile App client.
`GeoLocationLatitude`	`Double`	Gets the current location latitude. Available on mobile devices when running in the Mobile App client.
`GeoLocationLongitude`	`Double`	Gets the current location longitude. Available on mobile devices when running in the Mobile App client.
`IsAndroid`	`Boolean`	Flag indicating the client is running on Android.
`IsBackButtonVisibleOnIOS`	`Boolean`	Flag indicating the back button is visible on iOS.
`IsConnected`	`Boolean`	Gets whether this client session is currently connected to the Runtime server. True while the underlying TCP or web channel is alive, false while disconnected or reconnecting.
`IsIOS`	`Boolean`	Flag indicating the client is running on iOS.
`IsIPad`	`Boolean`	Flag indicating the client is running on an iPad.
`IsIPhone`	`Boolean`	Flag indicating the client is running on an iPhone.
`IsLocal`	`Boolean`	Gets whether this client session is running on the same machine as the Runtime server (loopback or local connection). Inverse of `IsRemote`.
`IsRemote`	`Boolean`	Gets whether this client session is running on a different machine than the Runtime server (network connection). Inverse of `IsLocal`.
`IsSmartClient`	`Boolean`	Gets whether this client session is running as a Smart Client, a locally installed client that connects to the Runtime server over TCP.
`IsSmartDevice`	`Boolean`	Flag indicating the client is running on a smart device.
`IsSmartDevicePortrait`	`Boolean`	Flag indicating the smart device is in portrait orientation.
`IsWebBrowser`	`Boolean`	Gets whether this client session is running inside a web browser (HTML5 client) rather than the native Windows or mobile runtime.
`LogonDateTime`	`DateTime`	Gets the date and time of the last logon on this session.
`NumberOfTagPropertiesLoaded`	`Int32`	Gets the count of tag properties (Value, Min, Max, Quality and others) loaded into memory by this client session. Typically a multiple of `NumberOfTagsLoaded`, and useful for diagnosing why a client holds unexpectedly many properties.
`NumberOfTagsLoaded`	`Int32`	Gets the count of tags loaded into memory by this client session. Useful for memory-footprint diagnostics on large solutions.
`Parameters`	`String`	Gets the startup parameters passed to this client session, typically the query-string portion of the launch URL for web clients, or command-line parameters for Smart Client launches.
`ServerHttpAddress`	`String`	Gets the HTTP or HTTPS URL this client uses to reach the Runtime server, for example `http://server:3101/`. Useful for building links or external deep-links back into the running application.
`StatusBarVisibleOnIOS`	`Boolean`	Flag indicating the status bar is visible on iOS.

`UserInactivity`	`TimeSpan`	Gets the elapsed time since the last user input on this client (keyboard, mouse, touch). Used by auto-logoff logic and screen-saver triggers.
`UserName`	`String`	Gets the client's user name.

Session methods

Member	Type	Description
`CapturePhotoAsync`	`Task<Byte[]>`	Asynchronously captures a photo using the device camera and returns the JPEG bytes. See **Mobile capture methods** below.
`ChangeUserPasswordAsync`	`Task<Boolean>`	Asynchronously changes a runtime user's password after validating the current password. Parameters: `username`, `oldPassword`, `newPassword`. Returns true when the password was changed, false when validation or policy rejected the change. Prefer `ChangeUserPasswordWithStatusAsync` when the caller needs the specific failure reason.
`ChangeUserPasswordWithStatusAsync`	`Task<Boolean>`	Asynchronously changes a runtime user's password and returns a detailed status code through the `status` parameter, so the caller can distinguish a wrong password from a policy rejection. Parameters: `username`, `oldPassword`, `newPassword`, `status`.
`GetPasswordHintAsync`	`Task<String>`	Asynchronously retrieves the password hint configured for a runtime user. Parameter: `userName`. Returns an empty string when the user is unknown or has no hint configured.
`LogOn`	`Int32`	Opens the solution's built-in LogOn display, blocking navigation until the user completes or cancels the login. Convenience helper for putting a Log on button on a display. Returns 0 on a successful login, or a non-zero security error code.
`LogOnGuestAsync`	`Task<Int32>`	Asynchronously logs the client session on as the built-in Guest user, replacing any currently signed-in user. Updates `LogonDateTime` and resets the inactivity clock. Returns 0 on success.
`ScanAnyCodeAsync`	`Task<String>`	Asynchronously scans a code of any supported format, optionally restricted by a format hint. See **Mobile capture methods** below.
`ScanBarcodeAsync`	`Task<String>`	Asynchronously scans a 1D barcode using the device camera. See **Mobile capture methods** below.
`ScanQRCodeAsync`	`Task<String>`	Asynchronously scans a QR Code using the device camera. See **Mobile capture methods** below.
`SetBlockedUserAsync`	`Task<Boolean>`	Asynchronously marks a runtime user as blocked or unblocked. Blocked users cannot log on, but their account and audit history are preserved. Parameters: `username`, `flag`. Returns true when the block state was applied, false on failure such as user not found or insufficient privileges.
`SetDeletedUserAsync`	`Task<Boolean>`	Asynchronously marks a runtime user as deleted (soft delete) or restores a previously soft-deleted user. The user row remains in the runtime database and only the visibility flag changes. Parameters: `username`, `flag`.

Mobile capture methods

Four session methods drive the device camera. All four are available **only when the HTML5 client is hosted inside the Mobile App**. In any other context, including a plain browser, the WPF Rich Client, and server-side scripts, they return null. They also return null when the user denies camera permission, cancels the capture, or the device has no camera.

Always null-check the result before using it. A null return is the normal outcome of a cancelled scan, not an error condition.

Signature	Returns	Usage
`@Client.Session.CapturePhotoAsync()`	`Task<Byte[]>`	Captures a photo and returns the JPEG bytes. Use for operator evidence attached to a work order, an inspection record or an alarm acknowledgement. Write the bytes to a Dataset file field or to disk; the method does not persist anything itself.
`@Client.Session.ScanQRCodeAsync()`	`Task<String>`	Scans a QR Code and returns the decoded text. Use for asset tags that carry a URL or a structured payload, and for equipment labels that map to an asset path in the Unified Namespace.
`@Client.Session.ScanBarcodeAsync()`	`Task<String>`	Scans a 1D barcode and returns the decoded text. Supported symbologies are Code128, Code39, Code93, EAN, UPC, ITF and Codabar. Use for inventory, batch and material identifiers already printed as linear barcodes.

`@Client.Session.ScanAnyCodeAsync(string format)`	`Task<String>`	Scans a code of any supported format, 1D or 2D, optionally narrowed by the `format` hint. Use when the operator may present either a barcode or a QR Code and the display should accept both.

The `format` hint on `ScanAnyCodeAsync` is case-insensitive. Pass an empty string or null to accept every supported format. Combine multiple values with `,`, `;` or `|`, for example `"QR,Code128,EAN13"`. Recognized tokens:

- `""`, `"All"`, `"Any"`: every supported code. This is the default.
- `"1D"`, `"Barcode"`, `"OneDimensional"`: every 1D barcode.
- `"2D"`, `"TwoDimensional"`: every 2D code (QR, Data Matrix, Aztec, PDF417, MaxiCode).
- `"QR"`, `"QRCode"`: QR codes only.
- Specific 1D formats: `"Code128"`, `"Code39"`, `"Code93"`, `"EAN13"`, `"EAN8"`, `"UPC-A"`, `"UPC-E"`, `"ITF"`, `"Codabar"`, `"Rss14"`, `"RssExpanded"`.
- Specific 2D formats: `"DataMatrix"`, `"Aztec"`, `"Pdf417"`, `"MaxiCode"`.

Unknown tokens are ignored. If no recognized token resolves to a format, the scanner accepts every supported format.

```
// Scan an asset tag and navigate to the matching display
string code = await @Client.Session.ScanQRCodeAsync();
if (!string.IsNullOrEmpty(code))
    @Client.OpenDisplay(code);

// Accept either a QR Code or a Code128 barcode
string id = await @Client.Session.ScanAnyCodeAsync("QR,Code128");

// Capture operator evidence
byte[] photo = await @Client.Session.CapturePhotoAsync();
if (photo != null)
    System.IO.File.WriteAllBytes("C:\\Evidence\\capture.jpg", photo);
```

Relocated Members

Several members that were once reached directly on `@Client` now live on a child namespace. The old paths still resolve for compatibility, and they are hidden from the Designer's IntelliSense so that new solutions are written against the current path. Use the right-hand column.

Old path	Current path
`@Client.ComputerIP`, `@Client.ComputerName`, `@Client.IsConnected`, `@Client.IsWebBrowser`, `@Client.IsLocal`, `@Client.IsRemote`, `@Client.IsSmartClient`, `@Client.IsSmartDevice`, `@Client.IsSmartDevicePortrait`, `@Client.IsIPhone`, `@Client.IsIPad`, `@Client.LogonDateTime`, `@Client.NumberOfTagsLoaded`, `@Client.NumberOfTagPropertiesLoaded`, `@Client.ServerHttpAddress`, `@Client.RunAlwaysOnTop`, `@Client.GetPasswordHintAsync`, `@Client.LogOnGuest`, `@Client.ChangeUserPassword`, `@Client.SetBlockedUser`, `@Client.SetDeletedUser`	`@Client.Session.*`
`@Client.CurrentPage`, `@Client.PreviousPage`, `@Client.PreviousLayout`, `@Client.LayoutName`, `@Client.BackPage`, `@Client.NextPage`, `@Client.HistoryPages`, `@Client.HistoryPagesIndex`, `@Client.Uid`, `@Client.BlinkSlow`, `@Client.BlinkFast`, `@Client.Simulation`, `@Client.SimulationAnalog`, `@Client.SimulationDouble`, `@Client.SimulationDigital`, `@Client.TooltipOptions`, `@Client.TooltipInitialShowDelay`, `@Client.AutoScaleMargin`, `@Client.MaxCacheDisplays`, `@Client.IsReuseSymbolEnabled`, `@Client.DisableMultiTouch`, `@Client.OnScreenKeyboard`, `@Client.UserInactivity`, `@Client.IsDisplayOpen`, `@Client.IsDisplayOpeningExecuted`, `@Client.CloseDisplay`, `@Client.NewPopup`, `@Client.OpenPreviousPage`, `@Client.OpenDisplayAtPanel`, `@Client.OpenDisplayAtIndex`, `@Client.PrintDisplay`, `@Client.PrintDisplayDefaultPrinter`, `@Client.PrintLayout`, `@Client.PrintLayoutDefaultPrinter`, `@Client.PrintScreenDefaultPrinter`, `@Client.SaveDisplayAsImageFile`, `@Client.SaveDisplayAsPngFile`, `@Client.SaveLayoutAsImageFile`, `@Client.SaveScreenAsImageFile`, `@Client.SetMainWindowSize`, `@Client.OpenQuickNote`	`@Client.UI.*`
`@Client.Now`, `@Client.UtcNow`, `@Client.Date`, `@Client.DateTime`, `@Client.Time`, `@Client.TimeSpan`, `@Client.TimeMs`, `@Client.Ticks`, `@Client.DayOfWeek`, `@Client.DayOfYear`, `@Client.Year`, `@Client.Month`, `@Client.Day`, `@Client.Hour`, `@Client.Minute`, `@Client.Second`, `@Client.Millisecond`, `@Client.Yesterday`, `@Client.Tomorrow`	`@Client.DateTimeInfo.*`
`@Client.LogOn(username, password)`	`@Client.LogOnAsync`
`@Client.SetLocalization`	`@Client.Localization`

`@Client.GetPasswordHint`	`@Client.Session.GetPasswordHintAsync`

`@Client.SwitchToStandby`, `@Client.Session.SwitchToStandby`, `@Client.ControlWithFocus`, `@Client.StatusBarVisibleOnIOS` and `@Client.IsBackButtonVisibleOnIOS` are retired and no longer carry behavior. The iOS status-bar and back-button flags remain available on `@Client.Session`.

Internal and Auxiliary Members

These members exist on `@Client` but back the product's own dialogs and diagnostics. They are hidden from the Designer's IntelliSense and are listed here for completeness. Solution scripts should not depend on them.

Member	Type	Description
`GCCollect`	`Boolean`	Requests a .NET garbage collection. Rich Client only.
`IdentityProviderList`	`String`	The names of the identity providers this client may sign in with, taken from every Active OIDC row in `SecurityIdentityProviders`, separated by `;`. Backs the provider picker in the default logon window.
`InputIdentityProvider`	`String`	The identity provider the operator picked in the logon window, one of the names in `IdentityProviderList`.
`InputMessage`, `InputPassword`, `InputUserName`	`String`	Auxiliary variables used by the system default logon window.
`IsStarted`, `StartCounter`, `Startup`, `Shutdown`	`Boolean`, `Int32`	Client lifecycle state and triggers. `Shutdown` requests a client shutdown.
`Parameters`	`String`	The launch parameters string for this client. Use `@Client.Session.Parameters` or `@Client.QueryParams` from solution scripts.
`PreloadedTags`, `ProjectChanged`, `ReadOnly`	`Boolean`	Client state flags. `ReadOnly` indicates the client is running in read-only mode.
`GetCursorX`, `GetCursorY`	`Int32`	Current cursor coordinates, optionally in screen coordinates.
`GetParameters`, `GetQueryParameters`	`String`	Backing accessors for `Parameters` and `QueryParams`. The two views can legitimately differ on the HTML5 client.
`AddDisplayInCacheList`	`Void`	Adds a display to this client's cache list.
`StartOidcWebFlow`, `StartOidcPopupFlow`, `ReadOidcHandoffVerifier`, `ReadOidcPopupStatus`, `AbortOidcPopupFlow`	various	Platform plumbing for the OIDC sign-in flow. Scripts call `@Client.LogOnSSOAsync`, which routes here.

Member set verified against FrameworX 10.1.5. For the full .NET API surface see the external Client Namespace .NET API Reference.

In this section...
