Server Namespace Reference

Access server state and operations via scripts.

The **Server Namespace** exposes runtime server state to .NET scripts. Use **`@Server`** to reference the namespace, followed by the accessors listed below.

Child Namespaces
Server Members
@Server.Retentive Members

Child Namespaces

Sub-namespaces exposing collections and grouped configuration. Reach each via its qualified path.

Path	Description
`@Server.dateTimeInfo` (17 members)	Gets the server date/time subsystem view (ServerDateTimeInfo), exposing Now, Date, Time, and every individual component (Year, Month, Day, Hour, Minute, Second, DayOfWeek, DayOfYear, Ticks). Use this when scripts need server time rather than client time. Accessed as @Server.DateTimeInfo.
`@Server.redundancy` (19 members)	Gets the redundancy-pair subsystem view (ServerRedundancy), exposing primary/secondary role, standby status, last-switch reason, and manual failover commands. Accessed as @Server.Redundancy.
`@Server.retentive` (3 members)	Gets the retentive-storage subsystem view (ServerRetentive), exposing the backlog of pending retentive writes, the background-writer block size, and the on-demand `Snapshot()` flush method. Accessed as @Server.Retentive. See the **@Server. Retentive Members** section below.
`@Server.systemMonitor` (21 members)	Gets the system-monitor subsystem view (ServerSystemMonitor), exposing CPU, memory, disk, tag-throughput, uptime, and the tag-count / tag-property-count counters. Accessed as @Server.SystemMonitor.

Server Members

Direct properties and methods on `@Server`.

Member	Kind	Type	Description
`computerIP`	Property	`String`	Gets the IP address of the server computer hosting the runtime. When multiple network interfaces are present, returns the address selected for solution communication.
`computerName`	Property	`String`	Gets the host name of the computer running the runtime server, as reported by the OS.
`getAllConnectionsAsync`	Method	`Threading.Tasks.Task`1[[System.Data.DataTable, System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]]`	*(no description)*
`getClientConnectionsAsync`	Method	`Threading.Tasks.Task`1[[System.Data.DataTable, System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089]]`	*(no description)*
`getComputerIP`	Method	`String`	*(no description)*
`httpAddress`	Property	`String`	Gets the base HTTP URL at which the runtime server publishes its web endpoints (HTML5 runtime, REST API, MCP endpoints). Empty when no HTTP listener is configured.
`isRunningAsService`	Property	`Boolean`	Gets whether the runtime server is executing under a Windows service host rather than as an interactive process. Useful to branch behavior that depends on console or UI availability.
`isRunningOnDocker`	Property	`Boolean`	Gets whether the runtime server is executing inside a Docker container (detected via the container environment at startup). Useful to branch behavior that depends on filesystem persistence or host-access assumptions.
`loadSolutionVersion`	Method	`Boolean`	*(no description)*

`osVersion`	Property	`String`	Gets the operating-system description of the computer running the solution (e.g., "Microsoft Windows NT 10.0.22621.0"). Populated once at runtime startup.
`resetStatistics`	Property	`Boolean`	Gets or sets the connection-statistics reset trigger. Writing any new value clears the per-client tag-read, tag-write and message counters tracked for GetClientConnectionsAsync and GetAllConnectionsAsync.
`runAndSaveDiagnostics`	Method	`Int32`	*(no description)*
`saveDiagnostics`	Property	`Boolean`	Gets or sets the diagnostics-capture trigger. Writing a new value causes the runtime to dump current trace-setup counters to a timestamped CSV under the product trace-logs folder (TServer_Diagnostics_YYYY_MM_DD_HH_mm_ss.csv). Read-back reports the last-written value.
`shutdownCommand`	Method	`Int32`	*(no description)*

@Server.Retentive Members

The retentive-storage subsystem persists retentive tag values to disk through a background writer. Beyond inspecting the backlog and block-size counters, scripts can force an immediate flush via `Snapshot()`.

Member	Kind	Type	Description
`backlogCount`	Property	`Int32`	Exposes the retentive subsystem state property reporting the count of retentive writes currently queued in memory awaiting persistence by the background writer. A persistently non-zero value indicates the writer is not keeping up with the incoming rate. See runtime documentation for current behavior.
`blockSize`	Property	`Int32`	Exposes the retentive subsystem state property reporting the background-writer block size — the number of pending retentive writes the writer groups together per disk flush cycle. See runtime documentation for current behavior.
`snapshot`	Method	`Boolean`	Forces an immediate flush of the in-memory retentive buffer to persistent storage on demand, rather than waiting for the background writer's next scheduled cycle. Use before a planned shutdown, before a critical checkpoint, or after a batch of retentive writes that must be durable before the next operation. Returns `true` when the flush completed successfully, `false` on error.

Signature: `Boolean @Server.Retentive.Snapshot()`

Example:

```
// Force a retentive flush before a planned shutdown
bool ok = @Server.Retentive.Snapshot();
if (!ok) @Info.Trace("Retentive snapshot failed");
```

The existing retentive-storage subsystem (backlog of pending writes and background-writer block size) continues to operate unchanged; `Snapshot()` is an on-demand override of the writer's normal schedule.

Member set generated from FrameworX 10.1.5. For the full .NET API surface see the external Server Namespace .NET API Reference.

In this section...
