AI Runtime Connector (10.1.5.1 draft)

DRAFT for 10.1.5.1. This page is a child-preview of the live AI Runtime Connector page. The live page will be updated after Marco's review. Reviewers are invited to check tool names, descriptions, groupings, and parameter defaults against the locked 10.1.5.1 design.

AI Runtime Connector (10.1.5.1 draft) — refreshed tool catalog, renamed gate-bit categories, and the Operator Subset for TServer-embedded chat.

What Changed in 10.1.5.1

Release 10.1.5.1 rebuilds the RuntimeMCP tool surface for LLM-efficacy (atomic-interaction, bounded payloads) and to reflect the broader scope of the AI Runtime connector — it reads every runtime-object root, not just Tag/*. The short summary:

- **10 tools total** (9 platform tools + solution-authored custom tools), up from the 7 documented in v1.1 (Feb 2026).
- Session bootstrap merged: `runtime_get_info` and `runtime_set_target` collapse into a single `runtime_connect(solution?, profile?)`.
- Semantic rename: `runtime_browse_uns` `runtime_browse_object_model` (the tool walks all 12 runtime roots, not just the P1-strict UNS).
- Verb-shift rename: `runtime_get_object_context` `runtime_describe_object` — and it absorbs the former `runtime_find_by_iri` via a new `iri` parameter.
- `runtime_list_by_type` `runtime_list_instances`, now bounded with `max_results` + `name_mask` wildcard filter.
- All list / browse / history / alarm tools gain server-side `max_*` caps with a `truncated` flag on the response.
- `runtime_list_derived_types` dropped — redundant with `runtime_describe_object`'s `derivedTypes` context block.
- Gate-bit category names in Solution Settings shift from historical *Enable Get** labels to scope-accurate *Enable *Tools* labels, and the wire-level gating was extended so every tool inside a category honors its category toggle (no partial coverage).

Gate-Bit Categories (Solution Settings)

The MCP for Runtime GroupBox in Solution Settings Data Servers now shows four category toggles plus the master toggle. **Bit positions in `ModelOptions` are unchanged** — existing 10.1.4 solutions decode identically; only the labels and C# accessor names change.

Checkbox label	Category	Covers	Bit
Enable MCP for Runtime	Master	Gates every RuntimeMCP tool. `runtime_connect` is gated only by this master so AI clients can bootstrap before any sub-category is enabled.	0x0002
Enable UNS Tools	UNS	`runtime_get_value`, `runtime_browse_object_model`, `runtime_search_uns`, `runtime_describe_object`, `runtime_list_instances`. All 5 tools honor this bit uniformly.	0x04
Enable Alarm Tools	Alarm	Dedicated alarm-query tools only: `runtime_get_active_alarms`, `runtime_query_alarm_history`.	0x08
Enable Historian Tools	Historian	`runtime_get_tag_history`.	0x10
Enable Custom Tools	Custom	Solution-authored `[McpServerTool]` methods on MCP Tool script classes.	0x20

Category semantics. Categories are defined by the TOOL being invoked, not by the runtime-object namespace the path happens to address:

- **Reading a value from the `Alarm.*` namespace** (e.g., `runtime_get_value("Alarm.TotalCount")`) stays under *Enable UNS Tools*. Any runtime-object value read is UNS-category regardless of path.
- **The alarm-state-of-a-tag overlay** inside `runtime_describe_object` (reports whether a specific tag is currently in alarm state) is describing the tag — UNS data — and gates on *Enable UNS Tools*, not *Enable Alarm Tools*.
- ***Enable Alarm Tools*** governs *only* the dedicated alarm-query tools that hit the alarm subsystem's state machine and historian: `runtime_get_active_alarms` and `runtime_query_alarm_history`.

Available Tools

AI Runtime provides **9 built-in platform tools + solution-authored custom tools**, organized by function.

Info — 1 tool

Tool	Description

runtime_connect	Connect to a FrameworX runtime and receive the Context Package (solution identity, profile, namespaces, quality codes, tool workflows, objType schemas, custom tools). CALL THIS FIRST at the start of every session. No-args mode: returns the current Context Package for whichever runtime this MCP is already bound to. With-args mode (HTTP retargetable only): rebinds this RuntimeMCPHttp instance to a different (solution, profile) BEFORE returning the Context Package. STDIO mode and HTTP-pinned mode: args are accepted but silently ignored; the response includes a `warnings` field noting this. Port is a consequence of ExecutionProfile — resolved from the running TServer's command line.

UNS Tools — 5 tools (gate: Enable UNS Tools)

Tool	Description
runtime_get_value	Get the current LIVE value of a tag or object from the running solution's Unified Namespace (UNS). Returns: value, quality (0=Bad, 64=Uncertain, 192=Good), timestamp, and metadata (description, units, min/max) when available. Reads live operational data — not solution configuration from Designer. Use for: process values (`Tag.*`), system status (`Server.*`), alarm counts (`Alarm.*`), device status (`Device.*`) . For historical values over a time range, use `runtime_get_tag_history`.
runtime_browse_object_model	Browse the live runtime object-model tree — returns child nodes at the specified path. Works across all runtime roots: `Tag` root tag folders; `Tag.Plant1/Line2` tags inside a folder; `Tag.Pump1` UserType members; `Server` server properties; `Alarm.Group` alarm groups; `Device.Channel` device channels. Bounded by `max_results` (default 50, hard max 200).
runtime_search_uns	Ranked semantic search across the live runtime UNS — UserTypes, Tags, members, Enumerations, and AssetTree folders. Scores across Name, DisplayText, Labels, SourceIri, and (auto-thresholded) Attributes annotation content. Returns ranked matches with `matchReason`, `snippet`, `score` (0.0–1.0). Tag matches additionally include live `runtimeValue`, `quality`, `timestamp`. Scopes: `all`, `udts`, `tags`, `members`, `enums`, `folders`. Bounded by `max_results` (default 20, hard max 100).
runtime_describe_object	Describe a single runtime UNS object — inheritance chain, derived UDTs, and (for Tag paths) live current value, quality, timestamp, and alarm state. Pass EITHER `path` OR `iri` (exactly one). Path form: slash-delimited tag Name (`Plant1/Line2/Motor1`, optionally with `Tag.` prefix), or bare UDT name (`Motor`). IRI form: reverse-lookup by external SourceIri from an ontology import; the object's canonical path is resolved first, then described. IRI ambiguity returns `IRI_AMBIGUOUS` with the first 2 matches so the caller can disambiguate by path.
runtime_list_instances	List all live tags of a given UDT (UserType), with current value/quality/timestamp for each. Returns one row per matching tag, ordered alphabetically by tag name. Optional `include_derived=true` expands to subclass UDTs via the BaseUserType walk. Optional `name_mask` narrows to tags whose Name matches a wildcard pattern (e.g. `Motor*`, `Area1/*`, `*_Pump`) — applied BEFORE `max_results` truncation. Bounded (default 50, hard max 200).

Alarm Tools — 2 tools (gate: Enable Alarm Tools)

Tool	Description
runtime_get_active_alarms	Get currently ACTIVE alarms from the running solution. Returns alarms that are in alarm state right now (not yet normalized), including tag name, message, priority, active time, area, and group. This is LIVE alarm status, not alarm configuration or history. For PAST alarms that have already normalized, use `runtime_query_alarm_history`. Bounded by `max_results` (default 20, hard max 200).
runtime_query_alarm_history	Query HISTORICAL alarm records from the Alarm Historian database. Only SELECT statements are allowed; INSERT, UPDATE, DELETE, DROP, and other modifying statements are blocked. Available columns: TagName, Message, ActiveTime, AckTime, NormTime, UserName, Priority, Area, GroupName. Server-side row cap applies even if the SQL omits TOP — `max_rows` defaults to 100 (hard max 1000).

Historian Tools — 1 tool (gate: Enable Historian Tools)

Tool	Description
runtime_get_tag_history	Query HISTORICAL time-series data for a tag from the Historian database. The tag must be configured for Historian storage in the solution to have historical data available. This retrieves STORED historical data, not the current live value — use `runtime_get_value` for the current value. 10.1.5.1 ships raw-truncate only (server-side cap via `max_points`). Aggregation / downsampling parameters (avg, min, max, bucket_size) are planned for 10.1.6 — narrow the time range or lower log frequency when the truncation flag fires. Bounded (default 500, hard max 2000).

Custom Tools (gate: Enable Custom Tools)

Solution authors expose additional tools by creating Scripts Classes of type **MCP Tool**. Methods decorated with `[McpServerTool]` + `[Description(...)]` attributes become tools on the RuntimeMCP surface for that solution. Recommended naming: `runtime_<toolName>` in snake_case. See the authoring guidance on the live AI Runtime Connector page for code examples and scripts-module patterns.

Restart the AI client (close from Task Manager, not just the window) after adding or modifying custom methods.

Operator Subset — TServer-Embedded Chat

In 10.1.5.1 the RuntimeMCP surface has two callers:

- **Chain A — external AI client.** Claude, GitHub Copilot, or any MCP-capable LLM reaches the runtime through `RuntimeMCP.exe` (STDIO) or `RuntimeMCPHttp.exe` (HTTP). Gets the full 9-tool platform surface plus the solution's custom tools. This is what the Available Tools section above describes.
- **Chain B — TServer-embedded operator chat.** The FrameworX Display client talks to a local LLM (Qwen default) hosted inside `TServer.exe`. Offline-by-design: zero outbound network traffic; the chat runs entirely on the factory-network. Operators don't explore the namespace — the HMI tells them tag and equipment names — so the operator LLM gets a reduced tool set optimized for atomic interaction and small-model reliability.

Chain B exposes **4 platform tools + solution custom tools**:

- `runtime_connect` (bootstrap — always)
- `runtime_get_value`
- `runtime_get_active_alarms`
- `runtime_get_tag_history`
- + Custom tools (the primary operator UX carrier)

The Operator Subset is declared through a marker attribute read by the Chain B dispatcher — Chain A ignores the marker and serves the full surface. Operator-chat authoring and dispatcher details live in the Chat in Displays documentation (separate release track).

Deprecated / Removed Tools

Customers upgrading from 10.1.5 or earlier should update any AI prompts, solution-authored wrappers, or integration scripts that reference the old names:

Old name (10.1.5 and earlier)	10.1.5.1 replacement	Notes
`runtime_get_info`	`runtime_connect`	Same payload on no-args; optional `(solution, profile)` rebind only on HTTP retargetable instances.
`runtime_set_target`	`runtime_connect (solution, profile)`	Merged as the with-args branch.
`runtime_browse_uns`	`runtime_browse_object_model`	Same parameters plus a bounding `max_results`.
`runtime_get_object_context`	`runtime_describe_object`	Absorbs external-IRI lookup via a new `iri` parameter.
`runtime_find_by_iri`	`runtime_describe_object (iri=...)`	Merged into describe_object — same single-object intent, different lookup key. IRI ambiguity returns `IRI_AMBIGUOUS`.
`runtime_list_by_type`	`runtime_list_instances`	Renamed. Adds `max_results` + `name_mask` wildcard filter.
`runtime_list_derived_types`	— (removed)	Redundant: `runtime_describe_object` already returns `derivedTypes` in the context block.
`runtime_search_tags`	`runtime_search_uns`	(Renamed earlier in 10.1.5; retained here for completeness.)

Change Log

Version	Date	Changes
1.2 (draft)	2026-04-24	Refreshed for 10.1.5.1. Tool catalog went to 9 platform tools + custom (was 7). Session bootstrap unified as `runtime_connect`. `runtime_browse_uns runtime_browse_object_model`. `runtime_get_object_context runtime_describe_object` (absorbs `runtime_find_by_iri`). `runtime_list_by_type runtime_list_instances` with wildcard filter. Gate-bit categories relabeled to Enable *Tools and wire-level gating extended so every tool in a category honors its toggle uniformly. Category semantics clarified: alarm-state-of-tag is UNS data; Enable Alarm Tools governs only the dedicated alarm-query tools. Operator Subset section added.
1.1	2026-02	7 built-in tools; updated architecture diagram; added `runtime_` prefix best practices.

In this section...