Local AI Reply Envelope Schema

The uniform JSON envelope that every Local AI call returns — both the `ChatRequest` Display action and the `TK.AIExecute` script API — including success, error, disabled, and truncated cases.

AI Integration Local AI Reply Envelope Schema

Why a single envelope

Both Local AI consumer paths return the same JSON shape on every call. The shape is the same for success, for errors, for the master kill-switch being off, and for wall-clock truncation. Customers parse one structure and dispatch on the `status` field — never on a thrown exception, never on a missing field, never on a special-cased error string.

Schema

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text":      "<the LLM answer text, or empty string on non-ok status>",
  "status":    "ok | error | disabled | truncated",
  "toolTrace": [ ... ],
  "latencyMs": 480,
  "warnings":  [ ... ]
}
```

Field reference

Field	Type	Always present?	Notes
`text`	string	Yes	The LLM's terminal answer. Empty string (`""`) on every non-ok status. Callers can rely on the field being present and string-typed; check `status` before consuming the text.
`status`	enum string	Yes	Exactly one of `ok` / `error` / `disabled` / `truncated`. No other values.
`toolTrace`	array	Yes	Sequence of tool dispatches the LLM made during the turn. Empty array (`[]`) when no tools fired. Always empty for atomic `TK.AIExecute` calls (the atomic path has no tool surface). Each entry has the shape documented under **Tool trace entry** below.
`latencyMs`	integer (long)	Yes — including on errors	Wall-clock milliseconds from entry to envelope-build. Reports `0` for pre-condition errors caught before any work (master kill-switch off, query missing required fields). Reports actual elapsed time for post-condition errors (HTTP failure, timeout). Never omitted.
`warnings`	array of string	Yes	Empty array (`[]`) on success-with-no-issues. Populated for hook-handler errors, wall-clock truncation, tool-dispatch cap reached, HTTP errors, malformed query, and anything the platform notes but does not promote to a status flip. `status="ok"` with non-empty `warnings` is valid (e.g., a chat hook threw but the LLM call still succeeded).

No optional fields. Every field present on every envelope, every code path. Callers can parse with confidence:

```
var reply  = JObject.Parse(replyJson);
string status = reply.Value<string>("status");        // never null
long   ms     = reply.Value<long>("latencyMs");       // always parseable
string text   = reply.Value<string>("text") ?? "";    // empty on non-ok
```

Status values

`ok`

The LLM POST completed and returned a terminal reply. `text` contains the model's answer. `warnings` may still be non-empty if a hook handler threw mid-turn (the chat continues; warnings list the failure).

`error`

Pre-POST validation failed, or the HTTP call failed, or response parsing failed, or an exception was caught. `text` is empty. `warnings` carries the message(s) describing the failure. `latencyMs` reflects elapsed time to the error point.

`disabled`

A master gate is off:

- `SolutionSettings.ModelEnabled = false` — applies to BOTH paths.

- `SolutionSettings.ModelOptions` bit `0x02` (`EnableRuntimeMCP`) is OFF — applies to `ChatRequest` only. `TK.AIExecute` does not consult this bit.

`text` is empty. `warnings` carries the gate identifier so the caller can log which gate blocked the call. `latencyMs` is `0` (early-exit before any work).

`truncated`

One of the budget caps tripped mid-turn:

- Wall-clock budget exceeded (default 60 seconds for the full turn).
- Tool-dispatch cap reached (`ChatRequest` only; the LLM kept asking for more tool calls than the platform permits in one turn).
- Tool-dispatch loop hit the iteration limit before the LLM produced a terminal reply (`ChatRequest` only).

`text` carries the partial reply if the LLM produced one before truncation; otherwise empty. `warnings` describes which cap tripped.

Status decision matrix

Situation	`status`	`text`	`warnings`	`latencyMs`
Normal successful chat turn	`ok`	LLM answer	`[]`	elapsed
Successful turn, hook handler threw	`ok`	LLM answer	handler error message(s)	elapsed
`ModelEnabled = false`	`disabled`	`""`	kill-switch identifier	`0`
`ModelOptions` master bit OFF (`ChatRequest` only)	`disabled`	`""`	master-bit identifier	`0`
Query JSON malformed	`error`	`""`	parse error message	elapsed
LLM endpoint unreachable	`error`	`""`	HTTP error message	elapsed
LLM endpoint returned non-2xx	`error`	`""`	HTTP status code + reason	elapsed
Wall-clock budget (60 s) exceeded	`truncated`	partial reply, if any	budget-exceeded message	60000
Tool-dispatch cap reached (`ChatRequest`)	`truncated`	partial reply, if any	cap-reached message	elapsed

Tool trace entry

Each entry in `toolTrace[]` records one tool dispatch (cached `ChatRequest` path only — atomic `TK.AIExecute` calls always return an empty `toolTrace`):

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "name":      "runtime_get_value",
  "args":      { "tag": "Pump1.MotorCurrent" },
  "result":    { "value": 12.4, "quality": "Good" },
  "status":    "ok",
  "timestamp": "2026-04-26T14:32:15.123Z",
  "elapsedMs": 3
}
```

Field	Type	Notes
`name`	string	Tool method the LLM invoked, e.g. `runtime_get_value`, `runtime_browse_uns`, `runtime_get_active_alarms`.
`args`	parsed JSON or string	Arguments the LLM supplied to the tool. Parsed JSON object when the tool's parameters are structured; raw string when the LLM passed an unstructured argument.
`result`	parsed JSON or string	Tool's return value. Parsed JSON when structured, raw string otherwise.
`status`	enum string	`ok` when the tool succeeded; `error` when the tool dispatch threw. Per-tool errors do NOT promote the envelope's top-level `status` away from `ok` — the LLM may still recover from the failed tool call and produce a terminal reply.
`timestamp`	ISO 8601 UTC string	When the tool was dispatched.
`elapsedMs`	integer	Wall-clock milliseconds the tool dispatch took.

JSON examples by status

Successful turn (atomic `TK.AIExecute`)

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text": "The likely cause is a partially blocked impeller given the high motor current with low flow rate.",
  "status": "ok",
  "toolTrace": [],
  "latencyMs": 832,
  "warnings": []
}
```

Successful chat turn with one tool dispatch

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text": "Pump1.MotorCurrent is currently 12.4 A.",
  "status": "ok",
  "toolTrace": [
    {
      "name": "runtime_get_value",
      "args": { "tag": "Pump1.MotorCurrent" },
      "result": { "value": 12.4, "quality": "Good" },
      "status": "ok",
      "timestamp": "2026-04-26T14:32:15.123Z",
      "elapsedMs": 3
    }
  ],
  "latencyMs": 1247,
  "warnings": []
}
```

Master kill-switch is off

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text": "",
  "status": "disabled",
  "toolTrace": [],
  "latencyMs": 0,
  "warnings": [ "Local AI master kill-switch (ModelEnabled) is off." ]
}
```

LLM endpoint unreachable

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text": "",
  "status": "error",
  "toolTrace": [],
  "latencyMs": 1043,
  "warnings": [ "LLM endpoint HTTP error: Connection refused (http://localhost:11434/v1/chat/completions)" ]
}
```

Wall-clock truncation with partial reply

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "text": "Based on the readings I have so far, the early indicators suggest...",
  "status": "truncated",
  "toolTrace": [ /* dispatched tools up to truncation */ ],
  "latencyMs": 60012,
  "warnings": [ "LLM POST wall-clock budget (60s) exceeded." ]
}
```

Consuming the envelope

From a Server.Class script

```
string replyJson = TK.AIExecute(query);
JObject reply    = JObject.Parse(replyJson);

switch (reply.Value<string>("status"))
{
    case "ok":
        @Tag.Last.Answer = reply.Value<string>("text");
        break;
    case "disabled":
        @Tag.Last.Answer = "Local AI is currently disabled.";
        break;
    case "truncated":
        @Tag.Last.Answer = reply.Value<string>("text");
        // log warnings for diagnostics
        break;
    default: // "error"
        @Tag.Last.Answer = "Could not generate a reply.";
        // warnings array carries the cause
        break;
}
```

From a Display Action with no scripting

If the reply tag is typed **JSON**, the platform's tag-method surface lets a Display Expression extract fields directly:

- `@Tag.Chat.ReplyJson.JsonString("text")` — the answer text.
- `@Tag.Chat.ReplyJson.JsonString("status")` — the envelope status.
- `@Tag.Chat.ReplyJson.JsonValue<long>("latencyMs")` — the wall-clock latency.

Wire these into Result expressions on the `ChatRequest` Action; see ChatRequest Action Reference.

Guarantees

- **Always parseable.** Every envelope is well-formed JSON. `JObject.Parse(reply)` never throws on a Local AI reply.
- **Field-level stability.** Adding new top-level fields in future versions is non-breaking; existing five fields stay present and stay typed as documented.
- **No exceptions ever surface to the caller.** Network failures, parser failures, gate-off conditions, exceptions in the platform's own LLM-handling code — all become envelopes with `status="error"` and a populated `warnings` array.

In this section...