@push.rocks/smartagent

Documentation for @push.rocks/smartagent

- readme.md for @push.rocks/smartagent
- changelog.md for @push.rocks/smartagent

readme.md for @push.rocks/smartagent

A lightweight agentic loop built on **Vercel AI SDK v6** via `@push.rocks/smartai`. Register tools, get a model, call `runAgent()` — done. 🧰

Install

```
pnpm install @push.rocks/smartagent
```

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Overview

`@push.rocks/smartagent` wraps the AI SDK's `streamText` with `stopWhen: stepCountIs(n)` for **parallel multi-step tool execution**. No classes to instantiate, no lifecycle to manage — just one async function:

```
import { runAgent, tool, z } from '@push.rocks/smartagent';
import { getModel } from '@push.rocks/smartai';

const model = getModel({
  provider: 'anthropic',
  model: 'claude-sonnet-4-5-20250929',
  apiKey: process.env.ANTHROPIC_TOKEN,
```

```
});

const result = await runAgent({
  model,
  prompt: 'What is 7 + 35?',
  system: 'You are a helpful assistant. Use tools when asked.',
  tools: {
    calculator: tool({
      description: 'Perform arithmetic',
      inputSchema: z.object({
        operation: z.enum(['add', 'subtract', 'multiply', 'divide']),
        a: z.number(),
        b: z.number(),
      }),
      execute: async ({ operation, a, b }) => {
        const ops = { add: a + b, subtract: a - b, multiply: a * b, divide: a / b };
        return String(ops[operation]);
      },
    }),
  },
  maxSteps: 10,
});

console.log(result.text);    // "7 + 35 = 42"
console.log(result.steps);   // number of agentic steps taken
console.log(result.usage);   // { promptTokens, completionTokens, totalTokens }
```

Architecture

```
┌──────────────────────────────────────────────────┐
│  runAgent({ model, prompt, tools, maxSteps })    │
│                                                  │
│  ┌──────────────┐   ┌──────────────┐   ┌──────────────┐ │
│  │ Messages   │──▶│ streamText│──▶│  Tools    │ │
│  │ (history)  │◀──│ (AI SDK)  │◀──│ (ToolSet) │ │
│  └──────────────┘   └──────────────┘   └──────────────┘ │
│                                                  │
│  stopWhen: stepCountIs(maxSteps)                 │
```

```
│  + retry with backoff on 429/529/503            │
│  + context overflow detection & recovery        │
│  + tool call repair (case-insensitive matching) │
└─────────────────────────────────────────────────┘
```

Key features:

- □□ **Multi-step agentic loop** — the model calls tools, sees results, and continues reasoning until done
- ⚡ **Parallel tool execution** — multiple tool calls in a single step are executed concurrently
- □□ **Auto-retry with backoff** — handles 429/529/503 errors with header-aware retry delays
- □□ **Tool call repair** — case-insensitive name matching + invalid tool sink prevents crashes
- □□ **Token streaming** — `onToken` and `onToolCall` callbacks for real-time progress
- □□ **Context overflow handling** — detects overflow and invokes your `onContextOverflow` callback

Core API

`runAgent(options): Promise<IAgentRunResult>`

The single entry point. Options:

Option	Type	Default	Description
`model`	`LanguageModelV3`	*required*	Model from `@push.rocks/smartai`'s `getModel()`
`prompt`	`string`	*required*	The user's task/question
`system`	`string`	`undefined`	System prompt
`tools`	`ToolSet`	`{}`	Tools the agent can call
`maxSteps`	`number`	`20`	Max agentic steps before stopping
`messages`	`ModelMessage[]`	`[]`	Conversation history (for multi-turn)
`maxRetries`	`number`	`5`	Max retries on rate-limit/server errors

Option	Type	Default	Description
`onToken`	`(delta: string) => void`	—	Streaming token callback
`onToolCall`	`(name: string) => void`	—	Called when a tool is invoked
`onContextOverflow`	`(messages) => messages`	—	Handle context overflow (e.g., compact messages)

`IAgentRunResult`

```
interface IAgentRunResult {
  text: string;              // Final response text
  finishReason: string;      // 'stop', 'tool-calls', 'length', etc.
  steps: number;             // Number of agentic steps taken
  messages: ModelMessage[];  // Full conversation for multi-turn
  usage: {
    promptTokens: number;
    completionTokens: number;
    totalTokens: number;
  };
}
```

Defining Tools □□

Tools use Vercel AI SDK's `tool()` helper with Zod schemas:

```
import { tool, z } from '@push.rocks/smartagent';

const myTool = tool({
  description: 'Describe what this tool does',
  inputSchema: z.object({
    param1: z.string().describe('What this parameter is for'),
    param2: z.number().optional(),
  }),
  execute: async ({ param1, param2 }) => {
    // Do work, return a string
    return `Result: ${param1}`;
  },
```

```
});
```

Pass tools as a flat object to `runAgent()`:

```
await runAgent({
  model,
  prompt: 'Do the thing',
  tools: { myTool, anotherTool },
  maxSteps: 10,
});
```

ToolRegistry

A lightweight helper for collecting tools:

```
import { ToolRegistry, tool, z } from '@push.rocks/smartagent';

const registry = new ToolRegistry();

registry.register('random_number', tool({
  description: 'Generate a random integer between min and max',
  inputSchema: z.object({
    min: z.number(),
    max: z.number(),
  }),
  execute: async ({ min, max }) => {
    return String(Math.floor(Math.random() * (max - min + 1)) + min);
  },
}));

registry.register('is_even', tool({
  description: 'Check if a number is even',
  inputSchema: z.object({ number: z.number() }),
  execute: async ({ number: n }) => n % 2 === 0 ? 'Yes' : 'No',
}));

const result = await runAgent({
  model,
```

```
  prompt: 'Generate a random number and tell me if it is even',
  tools: registry.getTools(),
  maxSteps: 10,
});
```

Built-in Tool Factories □□

Import from the `@push.rocks/smartagent/tools` subpath:

```
import { filesystemTool, shellTool, httpTool, jsonTool } from '@push.rocks/smartagent/tools';
```

`filesystemTool(options?)`

Returns: `read_file`, `write_file`, `list_directory`, `delete_file`

```
const tools = filesystemTool({ rootDir: '/home/user/workspace' });

await runAgent({
  model,
  prompt: 'Create a file called hello.txt with "Hello World"',
  tools,
  maxSteps: 5,
});
```

Options:

- `rootDir` — restrict all file operations to this directory. Paths outside it throw `Access denied`.

`shellTool(options?)`

Returns: `run_command`

```
const tools = shellTool({ cwd: '/tmp', allowedCommands: ['ls', 'echo', 'cat'] });

await runAgent({
  model,
  prompt: 'List all files in /tmp',
```

```
  tools,
  maxSteps: 5,
});
```

Options:

- `cwd` — working directory for commands
- `allowedCommands` — whitelist of allowed commands (if set, others are rejected)

`httpTool()`

Returns: `http_get`, `http_post`

```
const tools = httpTool();

await runAgent({
  model,
  prompt: 'Fetch the data from https://api.example.com/status',
  tools,
  maxSteps: 5,
});
```

`jsonTool()`

Returns: `json_validate`, `json_transform`

```
const tools = jsonTool();

// Direct usage:
const result = await tools.json_validate.execute({
  jsonString: '{"name":"test","value":42}',
  requiredFields: ['name', 'value'],
});
// → "Valid JSON: object with 2 keys"
```

Streaming & Callbacks □□

Monitor the agent in real-time:

```
const result = await runAgent({
  model,
  prompt: 'Analyze this data...',
  tools,
  maxSteps: 10,

  // Token-by-token streaming
  onToken: (delta) => process.stdout.write(delta),

  // Tool call notifications
  onToolCall: (toolName) => console.log(`\n□□Calling: ${toolName}`),
});
```

Context Overflow Handling □□

For long-running agents that might exceed the model's context window, use the compaction subpath:

```
import { runAgent } from '@push.rocks/smartagent';
import { compactMessages } from '@push.rocks/smartagent/compaction';

const result = await runAgent({
  model,
  prompt: 'Process all 500 files...',
  tools,
  maxSteps: 100,

  onContextOverflow: async (messages) => {
    // Summarize the conversation to free up context space
    return await compactMessages(model, messages);
  },
});
```

Output Truncation ✂

Prevent large tool outputs from consuming too much context:

```
import { truncateOutput } from '@push.rocks/smartagent';

const { content, truncated, notice } = truncateOutput(hugeOutput, {
  maxLines: 2000,   // default
  maxBytes: 50_000, // default
});
```

The built-in tool factories use `truncateOutput` internally.

Multi-Turn Conversations □□

Pass the returned `messages` back for multi-turn interactions:

```
// First turn
const turn1 = await runAgent({
  model,
  prompt: 'Create a project structure',
  tools,
  maxSteps: 10,
});

// Second turn — continues the conversation
const turn2 = await runAgent({
  model,
  prompt: 'Now add a README to the project',
  tools,
  maxSteps: 10,
  messages: turn1.messages, // pass history
});
```

Exports

Main (`@push.rocks/smartagent`)

Export	Type	Description

`runAgent`	function	Core agentic loop
`ToolRegistry`	class	Tool collection helper
`truncateOutput`	function	Output truncation utility
`ContextOverflowError`	class	Error type for context overflow
`tool`	function	Re-exported from `@push.rocks/smartai`
`z`	object	Re-exported Zod for schema definitions
`stepCountIs`	function	Re-exported from AI SDK
`jsonSchema`	function	Re-exported from `@push.rocks/smartai`

Tools (`@push.rocks/smartagent/tools`)

Export	Type	Description
`filesystemTool`	factory	File operations (read, write, list, delete)
`shellTool`	factory	Shell command execution
`httpTool`	factory	HTTP GET/POST requests
`jsonTool`	factory	JSON validation and transformation

Compaction (`@push.rocks/smartagent/compaction`)

Export	Type	Description
`compactMessages`	function	Summarize message history to free context

Dependencies

- `@push.rocks/smartai` — Provider registry, `getModel()`, re-exports `tool` / `jsonSchema`
- `ai` v6 — Vercel AI SDK (`streamText`, `stepCountIs`, `ModelMessage`)
- `zod` — Tool input schema definitions
- `@push.rocks/smartfs` — Filesystem tool implementation

- `@push.rocks/smartshell` — Shell tool implementation
- `@push.rocks/smartrequest` — HTTP tool implementation

License and Legal Information

Trademarks

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

changelog.md for @push.rocks/smartagent

2026-03-06 - 3.0.2 - fix(agent)

use output parameter when invoking onToolResult instead of toolCall.result

- Replace (toolCall as any).result with the explicit output parameter when calling options.onToolResult.
- Prevents undefined/misread results by aligning the callback with the tool runner's output signature.

2026-03-06 - 3.0.1 - fix(readme)

adjust ASCII art in README to fix box widths and spacing in agent diagram

- Updated readme.md diagram: expanded 'Messages' box width and realigned 'streamText' and 'Tools' columns
- Documentation-only change; no code or behavior affected

2026-03-06 - 3.0.0 - BREAKING CHANGE(api)

Migrate public API to ai-sdk v6 and refactor core agent architecture: replace class-based DualAgent/Driver/Guardian with a single runAgent function; introduce ts_tools factories for tools, a compactMessages compaction subpath, and truncateOutput utility; simplify ToolRegistry to return ToolSet and remove legacy BaseToolWrapper/tool classes; update package exports and dependencies and bump major version.

- Major API break: DualAgent/Driver/Guardian classes and many legacy tool wrapper classes were removed and replaced by runAgent and functional tool factories.

- Tooling refactor: new ts_tools (filesystem, shell, http, json) produce ToolSet-style tools; ToolRegistry now stores ToolSet and exposes getTools().
- Context management: added compactMessages in ts_compaction for onContextOverflow handling and truncateOutput util to limit tool outputs.
- Package changes: package.json bumped to 2.0.0, added exports map, updated devDependencies and dependencies (@push.rocks/smartai -> ^2.0.0, ai -> ^6.0.0, zod added).
- Tests updated/added to reflect new API (unit tests and an end-to-end test added).
- Consumers must update imports/usages (e.g. import runAgent, tool, z, stepCountIs and new subpath exports) — this is a breaking change.

2026-01-20 - 1.8.0 - feat(tools)

add ToolRegistry, ToolSearchTool and ExpertTool to support on-demand tool visibility, discovery, activation, and expert/subagent tooling; extend DualAgentOrchestrator API and interfaces to manage tool lifecycle

- Introduce ToolRegistry to manage tool registration, visibility (initial vs on-demand), activation, initialization, cleanup, and search
- Add ToolSearchTool providing search/list/activate/details actions for discovering and enabling on-demand tools
- Add ExpertTool to wrap a DualAgentOrchestrator as a sub-agent (expert) tool and the IExpertConfig interface
- Extend DualAgentOrchestrator API: registerTool(tool, options?), registerExpert(config), enableToolSearch(), getRegistry(); registerStandardTools/start now initialize visible tools via registry
- Add IToolRegistrationOptions and IToolMetadata/IExpertConfig types to smartagent.interfaces and export ToolRegistry/ToolSearchTool/ExpertTool in public entry points
- Documentation updates (readme) describing tool visibility, tool search, and expert/subagent system

2026-01-20 - 1.7.0 - feat(docs)

document native tool calling support and update README to clarify standard and additional tools

- Add 'Native Tool Calling' section documenting useNativeToolCalling option and behavior for providers (e.g., Ollama).
- Explain tool name mapping when native tool calling is enabled (toolName_actionName) and streaming markers ([THINKING], [OUTPUT]).

- Add example showing enabling useNativeToolCalling and note ollamaToken config option (Ollama endpoint).
- Clarify that registerStandardTools() registers five tools (Filesystem, HTTP, Shell, Browser, Deno) and that JsonValidatorTool must be registered manually as an additional tool.
- Documentation-only changes (README updates) — no code functionality changed in this diff.

2026-01-20 - 1.6.2 - fix(release)

bump version to 1.6.2

- No source changes detected in the diff
- Current package.json version is 1.6.1
- Recommend a patch bump to 1.6.2 for a release

2026-01-20 - 1.6.1 - fix(driveragent)

include full message history for tool results and use a continuation prompt when invoking provider.collectStreamResponse

- When toolName is provided, include the full messageHistory (do not slice off the last message) so tool result messages are preserved.
- Set userMessage to a continuation prompt ('Continue with the task. The tool result has been provided above.') when handling tool results to avoid repeating the tool output.
- Keeps existing maxHistoryMessages trimming and validates provider.collectStreamResponse is available before streaming.

2026-01-20 - 1.6.0 - feat(smartagent)

record native tool results in message history by adding optional toolName to continueWithNativeTools and passing tool identifier from DualAgent

- continueWithNativeTools(message, toolName?) now accepts an optional toolName; when provided the message is stored with role 'tool' and includes a toolName property (cast to ChatMessage)
- DualAgent constructs a toolNameForHistory as `${proposal.toolName}_${proposal.action}` and forwards it to continueWithNativeTools in both normal and error flows
- Preserves tool-origin information in the conversation history to support native tool calling and tracking

2026-01-20 - 1.5.4 - fix(driveragent)

prevent duplicate thinking/output markers during token streaming and mark transitions

- Add isInThinkingMode flag to track thinking vs output state
- Emit "\n[THINKING] " only when transitioning into thinking mode (avoids repeated thinking markers)
- Emit "\n[OUTPUT] " when transitioning out of thinking mode to mark content output
- Reset thinking state after response completes to ensure correct markers for subsequent responses
- Applied the same streaming marker logic to both response handling paths

2026-01-20 - 1.5.3 - fix(driveragent)

prefix thinking tokens with [THINKING] when forwarding streaming chunks to onToken

- Wraps chunk.thinking with '[THINKING] ' before calling onToken to mark thinking tokens
- Forwards chunk.content unchanged
- Change applied in ts/smartagent.classes.driveragent.ts for both initial and subsequent assistant streaming responses
- No API signature changes; only the token payloads sent to onToken are altered

2026-01-20 - 1.5.2 - fix()

no changes in this diff; nothing to release

- No files changed; no release required
- No code or dependency changes detected

2026-01-20 - 1.5.1 - fix(smartagent)

bump patch version to 1.5.1 (no changes in diff)

- No code changes detected in the provided diff
- Current package.json version is 1.5.0
- Recommended semantic version bump: patch -> 1.5.1

2026-01-20 - 1.5.0 - feat(driveragent)

preserve assistant reasoning in message history and update @push.rocks/smartai dependency to ^0.13.0

- Store response.reasoning in messageHistory for assistant responses (two places in driveragent)
- Bump dependency @push.rocks/smartai from ^0.12.0 to ^0.13.0

2026-01-20 - 1.4.2 - fix(repo)

no changes detected in diff

- No files changed in diff; no code or metadata updates were made.
- No version bump required.

2026-01-20 - 1.4.1 - fix()

no changes detected (empty diff)

- No files changed in this commit
- No release required

2026-01-20 - 1.4.0 - feat(docs)

document Dual-Agent Driver/Guardian architecture, new standard tools, streaming/vision support, progress events, and updated API/export docs

- Add DualAgentOrchestrator concept and describe Driver/Guardian agents and BaseToolWrapper
- Document six standard tools including new JsonValidatorTool and expanded descriptions for Filesystem, Http, Shell, Browser, Deno
- Add examples for scoped filesystem with exclusion patterns and line-range reads
- Add token streaming (onToken) and progress events (onProgress) examples and event types
- Document vision support for passing images as base64 and example usage
- Expose additional config options in docs: name, verbose, maxResultChars, maxHistoryMessages, onProgress, onToken, logPrefix
- Document additional result fields: toolCallCount, rejectionCount, toolLog, and error
- Update API signatures in docs: run(task, options?) and registerScopedFilesystemTool(basePath, excludePatterns?)
- Update re-exports to include IFilesystemToolOptions, TDenoPermission, JsonValidatorTool and re-export several smartai types

2026-01-20 - 1.3.0 - feat(smartagent)

add JsonValidatorTool and support passing base64-encoded images with task runs (vision-capable models); bump @push.rocks/smartai to ^0.12.0

- Add JsonValidatorTool (validate/format actions) implemented in ts/smartagent.tools.json.ts
- Export JsonValidatorTool from ts/index.ts
- Add ITaskRunOptions interface (images?: string[]) in smartagent.interfaces.ts
- DualAgent.run and Driver.startTask accept optional images and pass them to provider.chat/provider.chatStreaming; assistant responses added to message history
- Bump dependency @push.rocks/smartai from ^0.11.1 to ^0.12.0 in package.json

2026-01-20 - 1.2.7 - fix(deps(smartai))

bump @push.rocks/smartai to ^0.11.0

- package.json: @push.rocks/smartai updated from ^0.10.1 to ^0.11.0
- Recommend a patch release since this is a dependency update with no breaking API changes: 1.2.7

2026-01-20 - 1.2.6 - fix(deps)

bump @push.rocks/smartai to ^0.10.1

- Updated dependency @push.rocks/smartai from ^0.8.0 to ^0.10.1 in package.json
- No other code changes; dependency-only update

2025-12-15 - 1.1.1 - fix(ci)

Update CI/release config and bump devDependencies; enable verbose tests

- Add @git.zone/cli release configuration with registries (verdaccio.lossless.digital and registry.npmjs.org) and public access in npmextra.json
- Bump devDependencies: @git.zone/tsbuild -> ^4.0.2, @git.zone/tsbundle -> ^2.6.3, @git.zone/tsrun -> ^2.0.1, @types/node -> ^25.0.2
- Change test script to run tstest with --verbose

2025-12-02 - 1.1.0 - feat(deno)

Add Deno tool and smartdeno integration; export and register DenoTool; update docs and tests

- Introduce DenoTool wrapper (ts/smartagent.tools.deno.ts) to run TypeScript/JavaScript in a sandboxed Deno environment with permission controls
- Add @push.rocks/smartdeno dependency to package.json
- Import and re-export smartdeno in ts/plugins.ts
- Export DenoTool and TDenoPermission from ts/index.ts

- Register DenoTool in DualAgentOrchestrator.registerStandardTools() so it's available as a standard tool
- Update README architecture diagram and docs to include Deno as a standard tool
- Add tests for DenoTool in test/test.ts (exports, instantiation, call summary, permission display)

2025-12-02 - 1.0.2 - fix(core)

Bump version to 1.0.2 (patch release)

- Patch release: no source changes detected in the working tree
- Prepare package for publish — version bump from 1.0.1 to 1.0.2

2025-12-02 - 1.0.1 - initial release

Initial commit: project scaffold and first release.

- Repository initialized with initial project structure and baseline files.
- Version set to 1.0.1.