Progress®
OpenEdge®

What's New in OpenEdge MCP Server

Table of Contents

Preface....7

Learn about OpenEdge MCP Server....9

What's New in OpenEdge MCP Server 1.1....11

Cryptographic User Authentication with JWKS Validation....12
File-Based Tool Arguments....12
JSON Output Mode for CLI....12
Request Timeout Adjustment....13

Copyright

Visit the following page online to see Progress Software Corporation’s current Product Documentation Copyright Notice/Trademark Legend: https://www.progress.com/legal/documentation-copyright.

October 2026

Product version: Progress OpenEdge MCP Server 1.1

1

Learn about OpenEdge MCP Server

The OpenEdge MCP Server addresses the challenges in AI integration by transforming an OpenAPI specifications and optional prompt assets into AI-accessible tools. This process makes each API operation discoverable and usable by AI workflows, without requiring manual coding, enabling secure and efficient interaction with enterprise APIs.

The OpenEdge MCP Server acts as a bridge between AI agents and OpenEdge services, exposing APIs as secure, discoverable tools. The OpenEdge MCP Server connects to PAS for OpenEdge services through REST and WEB interfaces, enabling seamless communication between OpenEdge applications and MCP components.

2

What's New in OpenEdge MCP Server 1.1



OpenEdge MCP Server 1.1 delivers the following enhancements:

- Cryptographic User Authentication with JWKS Validation on page 10
- File-Based Tool Arguments on page 10
- JSON Output Mode for CLI on page 10
- Request Timeout Adjustment on page 11

For details, see the following topics:

- Cryptographic User Authentication with JWKS Validation
- File-Based Tool Arguments
- JSON Output Mode for CLI
- Request Timeout Adjustment

Cryptographic User Authentication with JWKS Validation

OpenEdge MCP Server now supports cryptographic validation of user JSON Web Tokens (JWTs) against a remote JSON Web Key Set (JWKS) endpoint, enabling OAuth2/OIDC-compliant authentication. This enhancement replaces structural-only checks with full RSA signature verification and works with identity providers like Keycloak, Auth0, Azure AD, and Okta.

When `user_auth.jwks_uri` is configured, user JWTs are validated for signature verification, expiration and timing claims, issuer and audience validation against your configured values, and algorithm verification against your explicit allowlist, with scope extraction for fine-grained authorization. Tokens failing validation are rejected outright, with no fallback.

Existing deployments without JWKS configuration continue in passthrough mode—no changes required unless you opt in. Without `user_auth.jwks_uri`, `user_auth.mode: "jwt"` performs structural validation only, without establishing trusted claims or scopes.

For more information, see "Configure user authentication" in *Use the OpenEdge MCP Server*.

File-Based Tool Arguments

The `mcpgen client` command now accepts tool arguments from a file, avoiding fragile shell quoting for large or complex JSON payloads. You can use either `--args-file` or `--args` with the `@` prefix to pass JSON arguments from a file instead of inline on the command line. For example, use:

```
./mcpgen client my-profile --tool getCustomer --args-file request.json
```

or

```
./mcpgen client my-profile --tool getCustomer --args @request.json
```

to invoke tools with complex JSON payloads. Inline JSON and a file argument source cannot be combined, and relative argument file paths resolve from the directory in which `mcpgen` was invoked.

For more information, see "Client" in *Use the OpenEdge MCP Server*.

JSON Output Mode for CLI

The `mcpgen` client now provides clean JSON-only output when using the `--json` flag, with all non-JSON progress messages suppressed. You can use `mcpgen --json` commands in CI/CD pipelines and automation scripts with confidence that the output contains only valid JSON, eliminating the need for complex output filtering. This enhancement provides reliable machine-readable output for automation pipelines without extraneous output from progress messages or warnings, simplifying integration with JSON parsing tools like `jq`.

For more information, see "Client" in *Use the OpenEdge MCP Server*.

Request Timeout Adjustment

The default request timeout has been increased from 500ms to 30 seconds to better accommodate real-world API workload patterns. This change prevents healthy calls to remote OpenEdge services from failing under normal network latency. Deployments experiencing frequent timeout errors should see improved reliability without configuration changes. However, this changes the maximum time an unconfigured request can occupy server resources. You can also set an explicit value using `runtime.middleware.request_timeout.timeout_ms` in your configuration. Progress recommends that you set a value based on the latency of your downstream service and your caller's timeout requirements.

For more information, see "Configure middleware settings" in *Use the OpenEdge MCP Server*.