MCP Server Architecture

An overview of the Stack Overflow MCP server architecture, authentication, security, and more.

Document generated 10/09/2025

PDF VERSION

Tags | **AI** | **LLM** | **MCP Server** |

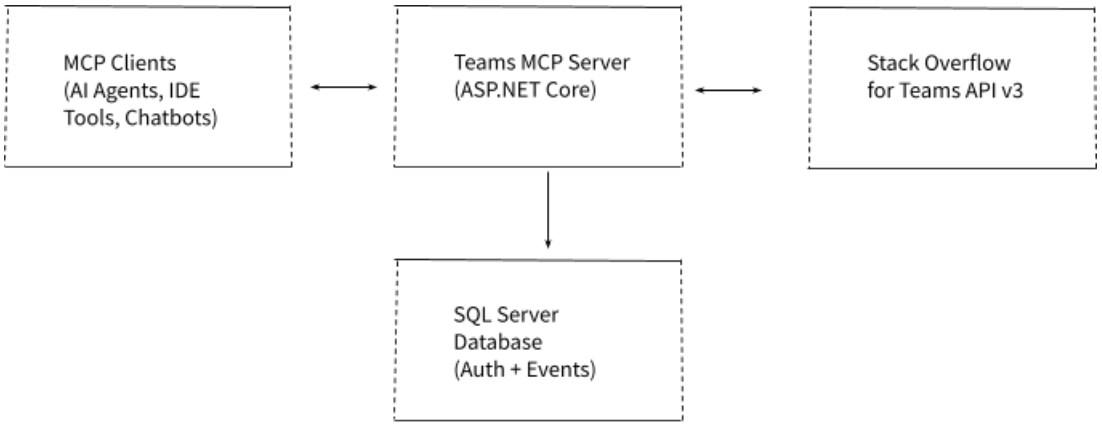
Applies to: Free Basic Business **Enterprise**

*This documentation is for **Stack Overflow for Teams Enterprise**. Free, Basic, and Business users can access their documentation here. Find your plan.*

Overview

The StackOverflow Teams MCP Server is a Model Context Protocol (MCP) server that provides access to private content from an organization's Stack Overflow for Teams instance. It enables AI agents, coding assistants, and enterprise chat interfaces to retrieve and interact with internal technical Q&A, best practices, team processes, onboarding documentation, and enterprise knowledge.

High-level system architecture



OAuth 2.0 dynamic client registration & metadata discovery

The Teams MCP Server is an OAuth 2.1-compliant authorization server that supports dynamic client registration (RFC 7591), metadata discovery (RFC 8414), and the Third-Party Authorization Flow from the Model Context Protocol. It acts as an OAuth client to Stack Overflow for Teams APIv3 and as an authorization server to MCP clients, securely mapping and managing tokens between systems.

OAuth metadata discovery

The server provides OAuth 2.0 authorization server metadata through the `/.well-known/oauth-authorization-server` endpoint, accessible via GET with CORS enabled for all origins. This endpoint returns comprehensive OAuth configuration, including issuer, authorization and token endpoints, supported response types, grant types, and authentication methods.

Metadata response

```
{
  "issuer": "https://base-url",
  "authorization_endpoint": "https://base-url/mcp/authorize",
  "token_endpoint": "https://base-url/mcp/token",
  "registration_endpoint": "https://base-url/mcp/register",
  "response_types_supported": ["code"],
  "response_modes_supported": ["code"],
  "grant_types_supported": ["authorization_code"],
  "token_endpoint_auth_methods_supported": ["none"],
  "code_challenge_methods_supported": ["S256"]
}
```

Dynamic client registration

The server supports dynamic client registration through the `/register` endpoint, accessible via POST without authentication. This public endpoint allows new OAuth clients to register dynamically by providing client name and redirect URIs.

Client registrations are stored long-term in the database and can only be updated if the client has the registration token. This ensures that only authorized clients can modify their registration details.

Registration request

```
{
  "client_name": "My MCP Client",
  "redirect_uris": [
    "https://client.example.com/callback",
    "http://localhost:3000/callback"
  ]
}
```

Registration response

```
{
  "client_id": "12345678-1234-1234-1234-123456789012",
  "client_name": "My MCP Client",
  "registration_client_uri": "https://base-url/registration/12345678-1234-1234-1234-123456789012",
  "registration_access_token": "reg-abcdef1234567890...",
  "grant_types": ["authorization_code"],
  "token_endpoint_auth_method": "none",
  "redirect_uris": [
    "https://client.example.com/callback",
    "http://localhost:3000/callback"
```

```
  ]
}
```

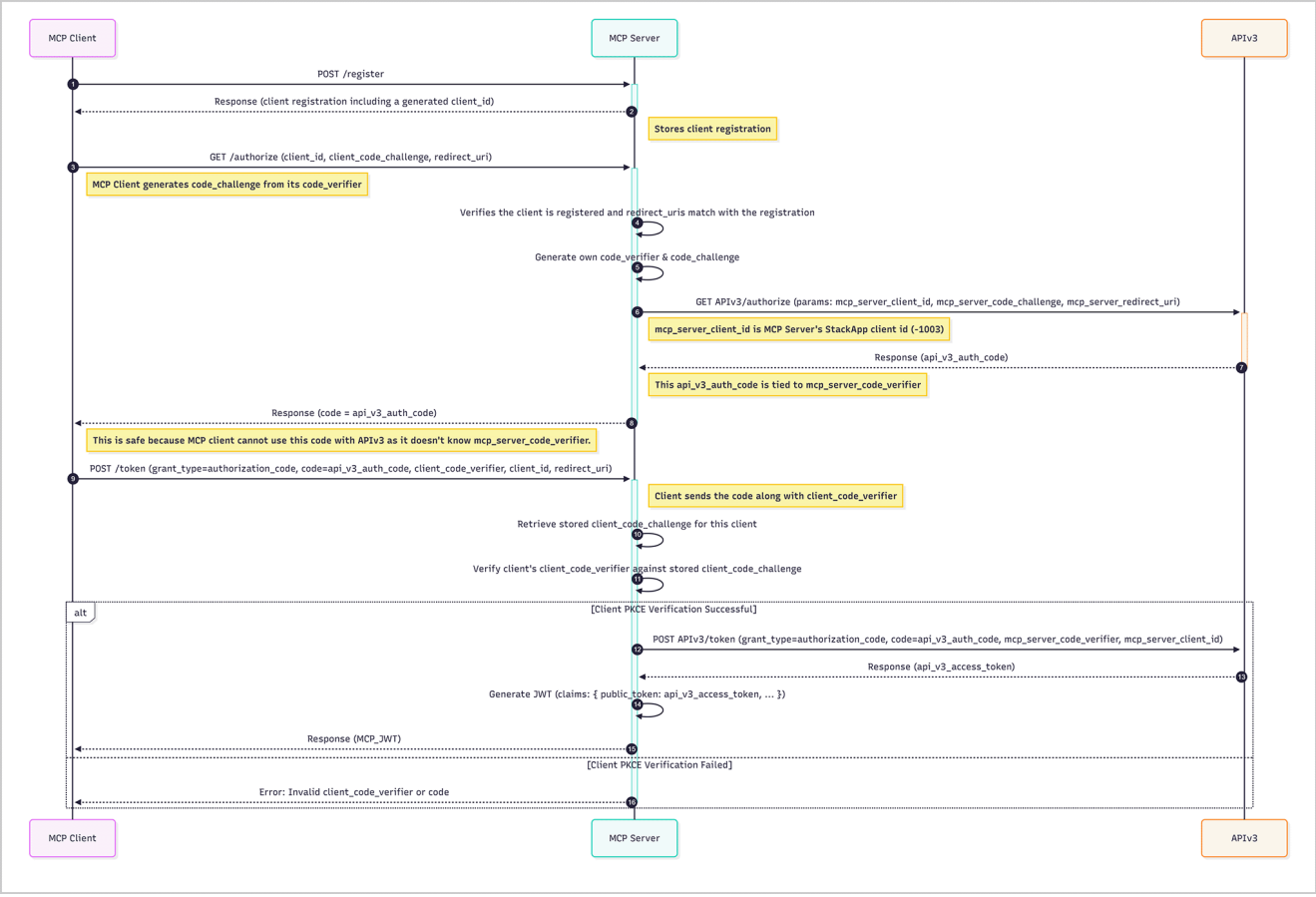
Client management endpoints

The server provides comprehensive client management through three endpoints under `/registration/{client_id}`. The GET endpoint retrieves client registration details, the PUT endpoint updates client name and redirect URIs, and the DELETE endpoint removes client registration. All operations require authentication via Registration Access Token (Bearer).

OAuth 2.0 authorization flow

The authorization flow is initiated through the `/authorize` endpoint via GET, requiring `client_id`, `code_challenge` (SHA256), `code_challenge_method` ("S256"), `redirect_uri`, and an optional `state` parameter. The complete flow involves client registration, metadata discovery, authorization request, server PKCE flow with APIv3, code exchange, token request with PKCE verification, and JWT generation with `public_token` claim (continues below).

Detailed OAuth flow sequence



Token endpoint

The `/token` endpoint (POST) exchanges authorization codes for access tokens, requiring `grant_type` ("authorization_code"), code from callback, `code_verifier`, `redirect_uri` matching the authorization request, and

registered `client_id`.

Security features

The system implements PKCE with SHA256 code challenge method and cryptographically random code verifiers to prevent authorization code interception attacks. Client validation includes strict redirect URI validation against registered URIs, client ID validation, and CSRF protection via state parameters. Registration access tokens use the `reg-{guid}` prefix format and are limited to client registration management operations.

Bearer token authentication

MCP clients must provide bearer tokens with a `public_token` claim, which the server extracts and uses for APIv3 authentication. The server returns `401 Unauthorized` for missing or invalid tokens. The token flow involves client providing the bearer token, server extracting the `public_token` claim, using it for Stack Overflow for Teams APIv3 authentication, and forwarding all API calls with the extracted token.

If you need further support or have questions, contact your site administrator.