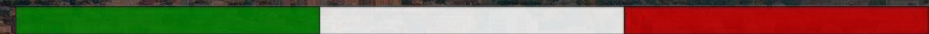


An aerial photograph of a historic Italian city, likely Florence, showing a dense cluster of buildings with terracotta roofs. A river, the Arno, flows through the center of the city. In the foreground, a large, prominent building with a series of arches, possibly a palace or a church, is visible. The sky is a soft, hazy orange, suggesting a sunset or sunrise.

DELPHIDAY



italian conference

MCPConnect

Connect your world with the AI



PAOLO ROSSI

WINTech ITALIA - CTO

SENCHA & EMBA MVP



blog.paolorossi.net



paolo@paolorossi.net



twitter.com/awebguy



github.com/paolo-rossi



linkedin.com/in/paolo-rossi-pc

DELPHIDAY

italian conference

9-10 Giugno 2026
Piacenza



wintech
italia



GITHUB PROJECTS



github.com/paolo-rossi



Delphi JWT

JSON Web Token Library



Delphi Neon

JSON Serialization Library



NATS Delphi

NATS Client Library for Delphi



Linux Daemon

Real Linux daemons



OpenAPI-Delphi

OpenAPI 3.0 Library



GITHUB PROJECTS



github.com/delphi-blocks



WiRL Library

REST Library for Delphi



MCPConnect

MCP Framework for Delphi



Delphi-Blocks

A **real** package manager for Delphi



Logify

Meta Logger for Delphi



AGENDA



1. What is this all about?
2. MCP: the protocol
3. MCP Apps
4. MCP & Delphi
5. Introducing MCPConnect



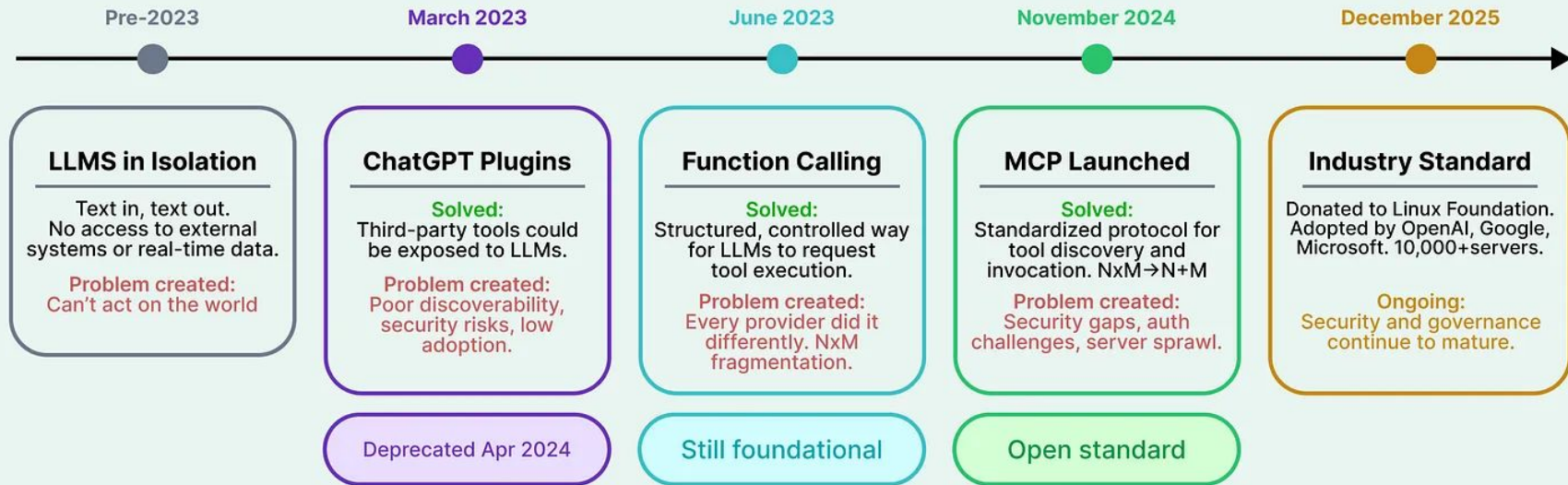
WHAT IS THIS
ALL ABOUT ?

1



CONNECTING LLMs TO THE WORLD

A Timeline of Tool Use in LLMs

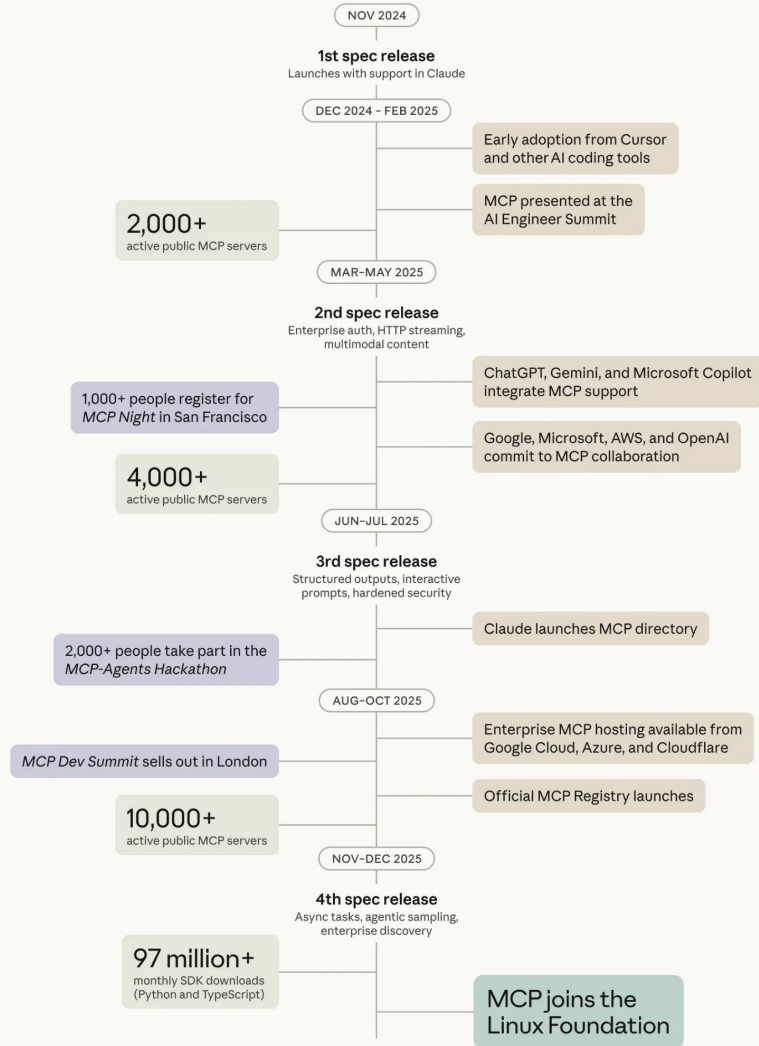


Core constant across every stage: The model reasons about what should happen. The application layer controls whether it actually does.



ONE YEAR OF MCP!!

- Nov 2024
 - First release (Anthropic)
- May 2025
 - Second release, HTTP support
- Jun 2025
 - Third release, better json schema
- Nov 2025
 - Fourth release, mcp apps
- Today
 - 20,000+ active public servers!
 - MCP joins the Linux Foundation





MCP The Protocol

2





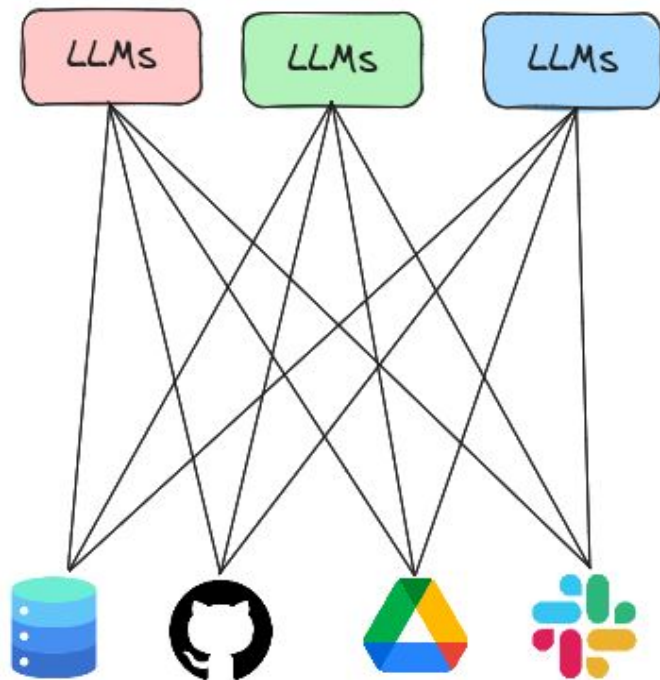
Model Context Protocol

- Open-source standard for connecting AI applications to external systems
- Using MCP, LLM can connect to
 - data sources (e.g. local files, databases)
 - tools (e.g. search engines, calculators)
 - workflows (e.g. specialized prompts)
- This enables LLM to access key information and perform tasks

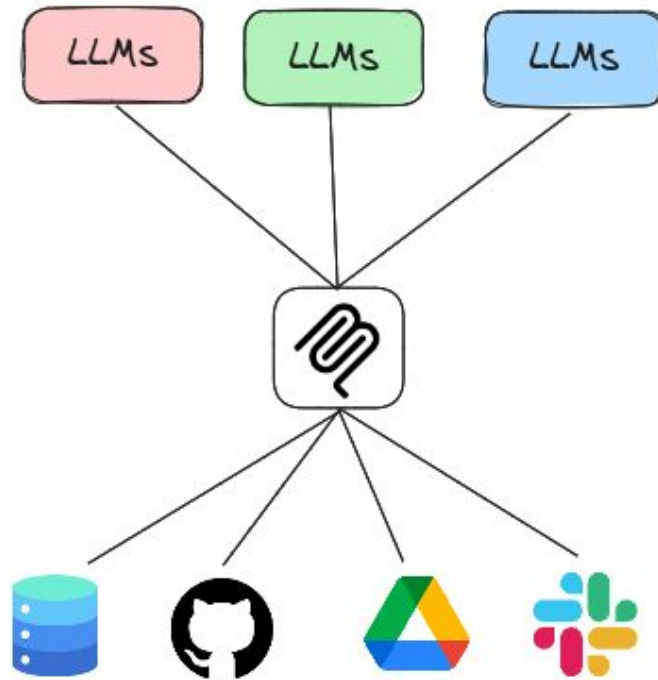


Why MCP?

without MCP



with MCP





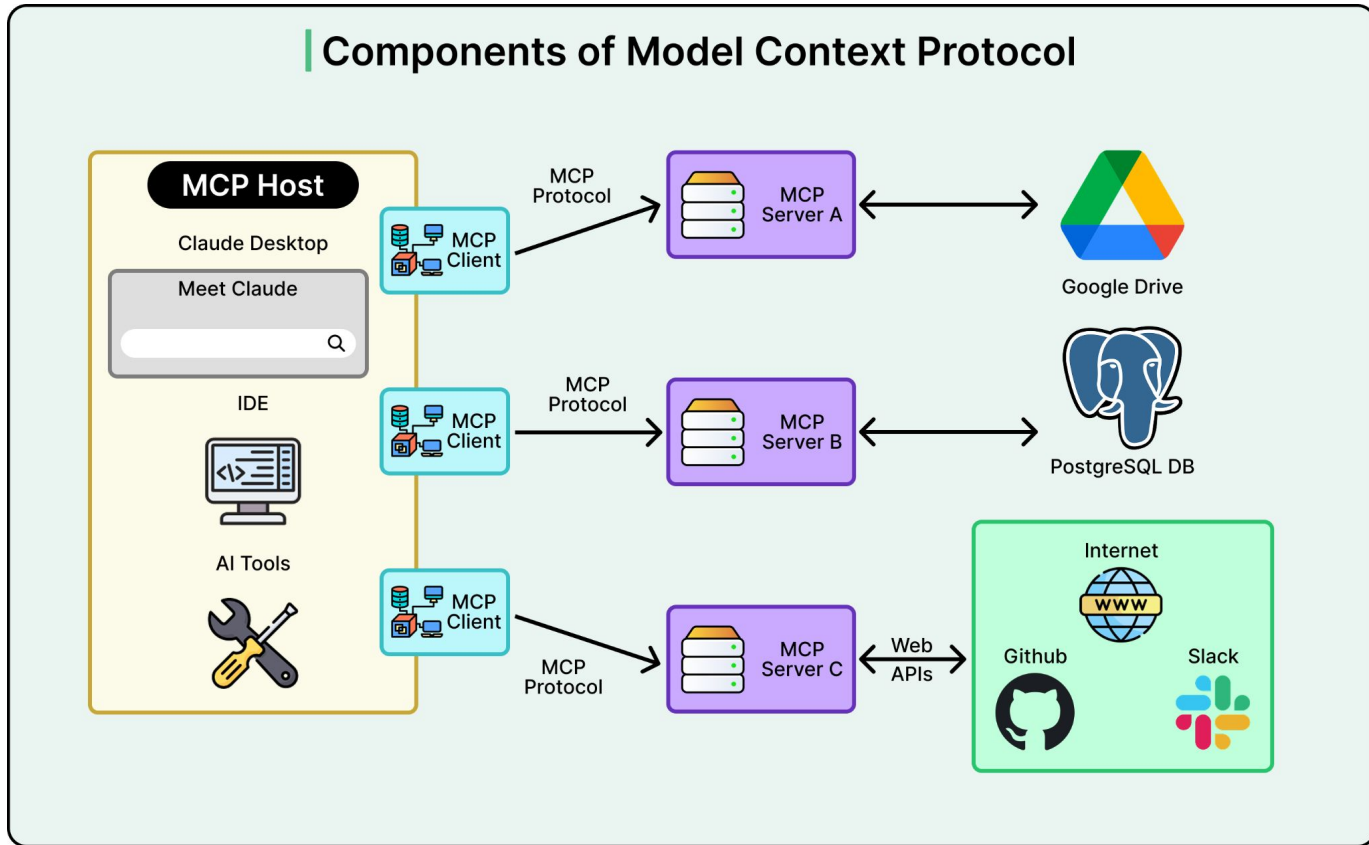
MCP architecture

- **MCP Host**
 - The AI application that coordinates and manages one or multiple MCP clients
- **MCP Client**
 - A component that maintains a connection to an MCP server and obtains context from an MCP server for the MCP host to use
- **MCP Server**
 - A program that provides context to MCP clients





MCP: overview





Layers

- Data layer
 - Defines the JSON-RPC based protocol for client-server communication, including lifecycle management, and core primitives, such as **tools**, **resources**, **prompts** and **notifications**.
- Transport layer
 - Defines the communication mechanisms and channels that enable data exchange between clients and servers, including transport-specific connection establishment, message framing, and authorization



Data Layer

- JSON-RPC 2.0 exchange protocol
- Defines the message structure and semantics:
 - **Lifecycle management:** connection initialization, capabilities negotiation, and connection termination
 - **Server features:** enables servers to provide functionality including tools, resources, and prompts
 - **Client features:** enables servers to ask the client to sample from the host LLM, and log messages to the client
 - **Utility features:** supports additional capabilities like notifications





Basic JSON-RPC example



```
{  
  "jsonrpc": "2.0",  
  "method": "subtract",  
  "params": [42, 23],  
  "id": 1  
}
```



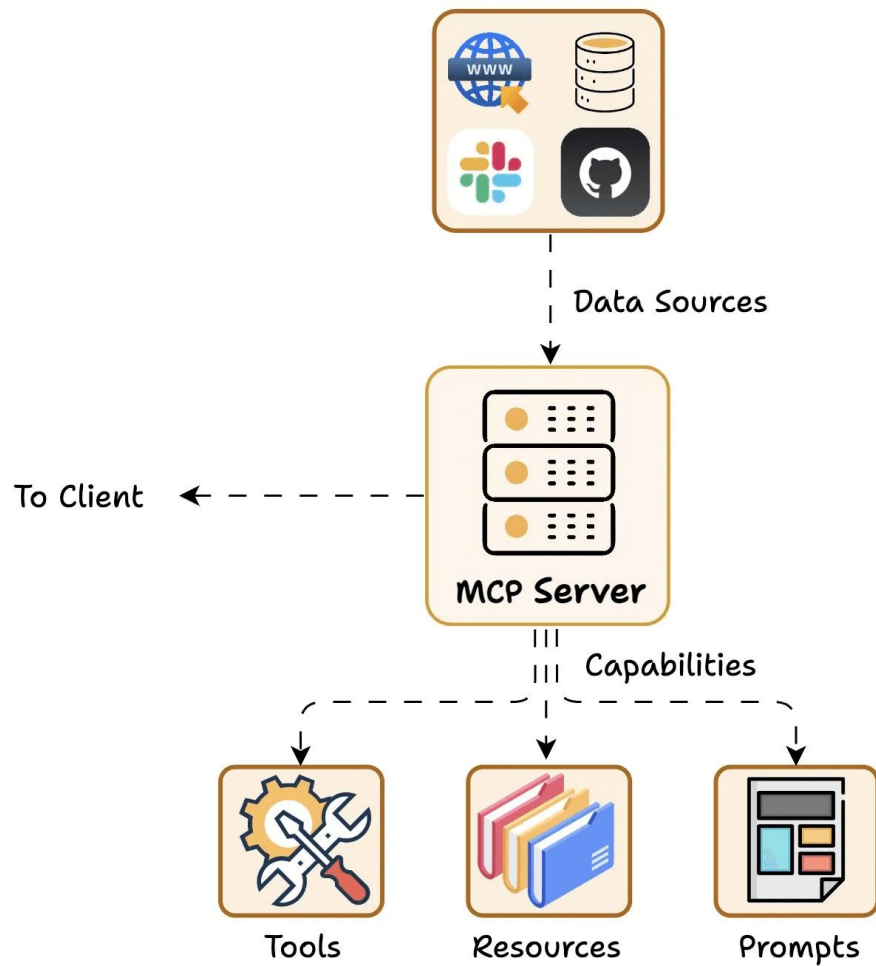
tools/call example

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "tools/call",
  "params": {
    "name": "weather_current",
    "arguments": {
      "location": "San Francisco",
      "units": "imperial"
    }
  }
}
```



Ttransport Layer

- Manages communication channels and authentication between clients and servers MCP supports two transport mechanisms:
 - **Stdio**: Uses standard input/output streams for direct process communication between local processes on the same machine, providing optimal performance with no network overhead
 - **Streamable HTTP**: Uses HTTP POST for client-to-server messages with optional Server-Sent Events (SSE) for streaming capabilities





Primitives (servers)

- The most important concept within MCP
 - They define what clients and servers can offer each other
- MCP defines three core primitives for **servers**:
 - **Tools**: Executable functions that AI applications can invoke to perform actions (e.g., file operations, API calls, database queries)
 - **Resources**: Data sources that provide contextual information to AI applications (e.g., file contents, database records, API responses)
 - **Prompts**: Reusable templates that help structure interactions with language models (e.g., system prompts, few-shot examples)





Primitives (clients)

- Allow MCP server authors to build richer interactions
 - **Sampling:** Allows servers to request language model completions from the client's AI application.
 - This is useful when servers' authors want access to a language model, but want to stay model independent and not include a language model SDK in their MCP server. They can use the sampling/complete method to request a language model completion from the client's AI application



Primitives (clients)

- Allow MCP server authors to build richer interactions
 - **Elicitation:** Allows servers to request additional information from users. This is useful when servers' authors want to get more information from the user, or ask for confirmation of an action. They can use the elicitation/request method to request additional information from the user.
 - **Logging:** Enables servers to send log messages to clients for debugging and monitoring purposes.



MCP: the API

MCP Client

Invokes **Tools**

Queries for **Resources**

Interpolates **Prompts**

MCP Server

Exposes **Tools**

Exposes **Resources**

Exposes **Prompts**

Tools

Model-controlled

Functions invoked
by the model

Retrieve / search

Send a message

Update DB records

Resources

Application-controlled

Data exposed
to the application

Files

Database Records

API Responses

Prompts

User-controlled

Pre-defined templates
for AI interactions

Document Q&A

Transcript Summary

Output as JSON



Notifications

- The protocol supports real-time notifications
 - Enabling dynamic updates between servers and clients
 - For example, when a server's available tools change, the server can send tool update notifications to inform connected clients
- Notifications are sent as JSON-RPC 2.0 notification messages (without expecting a response) and enable MCP servers to provide real-time updates to connected clients



MCP: what can it do for you?

- LLMs (through agents) can (general)
 - Access your Google Calendar and Notion, acting as a more personalized AI assistant.
 - Generate an entire web app using a Figma design.
 - Connect to multiple databases across an organization, empowering users to analyze data using chat.
 - Create 3D designs on Blender and print them out using a 3D printer.



MCP: what can it do for you?

- LLMs (through agents) can (enterprise)
 - Access your files
 - Access information on databases
 - Access in-memory data (applications)
 - Run code in your system
 - ...





MCP vs API (REST)

Feature	MCP	REST APIs
State Management	Stateful: maintains context across interactions	Stateless: each request is independent
Connection Type	Persistent, bidirectional connections	One-way request/response
Communication Style	JSON-RPC based with ongoing sessions	HTTP-based with discrete requests
Context Handling	Context is intrinsic to the protocol	Context must be manually managed
Tool Discovery	Runtime discovery of available tools	Design-time integration (OpenAPI)
Integration Approach	Runtime integration with dynamic capabilities	Design-time integration (code changes)



Why does MCP matter??

- MCP can have a range of benefits:
 - **Developers:** MCP reduces development time and complexity when building, or integrating with, an AI application or agent.
 - **AI applications or agents:** MCP provides access to an ecosystem of data sources, tools and apps which will enhance capabilities and improve the end-user experience.
 - **End-users:** MCP results in more capable AI applications or agents which can access your data and take actions on your behalf when necessary.

- Tasks (optional)
 - Request augmentation
 - Durable state machines
 - Carry information about the underlying execution state of the request
 - Both clients and servers
- Ping
 - Verify the counterpart is still responsive and the connection is alive
 - Both clients and servers
 - **Implemented in MCPConnect**



Version 2025-11-25

- Icons support (Tools, Resources, Prompts)
 - Add visual icons to your servers, tools, resources, and prompts
 - They provide visual representations for your MCP servers and components
 - When displayed in MCP clients, icons help users quickly identify and navigate your server's capabilities
 - **Implemented in MCPConnect**



MCP APPS

3





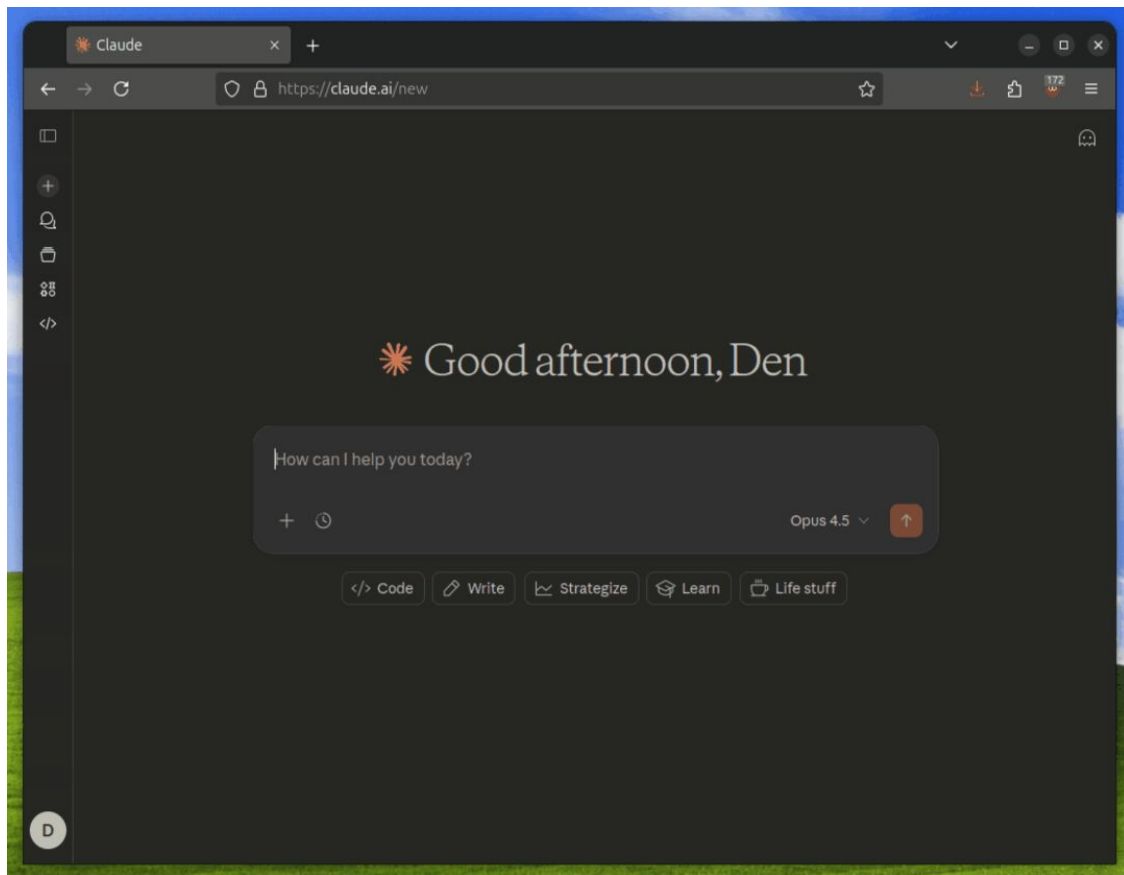
MCP Apps

- Interactive UI applications that render inside MCP hosts
- Text responses can only go so far
- Sometimes users need to interact with data, not just read about it
- MCP Apps let servers return interactive HTML interfaces (data visualizations, forms, dashboards) that render directly in the host





MCP Apps





MCP Apps

- <https://mcpui.dev/>
- <https://github.com/modelcontextprotocol/ext-apps>
- <https://github.com/MCP-UI-Org/mcp-ui>
- <https://modelcontextprotocol.io/docs/extensions/apps>
- <https://modelcontextprotocol.github.io/ext-apps/api/>

[Demo](#)





MCP Apps in MCPConnect

- One of the first MCP libraries to support MCP-Apps
 - The only one in Delphi arena
- Implementation is (almost) complete
 - Expect the final commit in a couple of days
- Remember UI apps are in HTML/CSS/JavaScript
 - React, Vue, HTMX





MCP & DELPHI

4



Tools/list request



```
{  
  "jsonrpc": "2.0",  
  "id": 1,  
  "method": "tools/list"  
}
```



Tools/list response

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "tools": [
      {
        "name": "get_weather",
        "description": "Get current weather information for a location",
        "inputSchema": {
          "type": "object",
          "properties": {
            "location": {
              "type": "string",
              "description": "City name or zip code"
            }
          }
        }
      }
    ]
  }
}
```



DELPHI & MCP

- The MCP server you are building:
- What is for and who is for
 - ◆ You and/or some developer to automate things
 - ◆ You corporate customer that needs reliability, security,



DELPHI & MCP

- A few libraries, some open source, some commercial
 - ◆ Some “ok”, some bad
- A lot of “We support MCP”
 - ◆ Really a bunch of TJSONObject usage to generate requests and responses
 - ◆ Not a **real** MCP protocol support



WHAT TO LOOK FOR

- Complete JSON-RPC v2 support
- Automatic function invocation (through RTTI)
- Complete support for JSON-Schema generation
 - ◆ InputSchema
 - ◆ OutputSchema
 - ◆ StructuredContent
- JSON Serializer/Deserializer for complex structures
- Support for SSE, parallel http channels, notifications, etc...
- Support for sessions, authentication, etc...



MCPConnect

4



wintech *italia*



MCPConnect

- New OpenSource Library
 - <https://github.com/delphi-blocks/MCPConnect>
- MCP Servers (and Clients)
- JSON-RPC 2.0 Library
 - Soon to be an independent library
- Uses Neon
- Uses Logify
- Blocks as dependency manager





Concepts

- Attributes to call class methods
- Automatic tools/list from multiple Delphi classes
- Easy Server configuration
- WebBroker/Indy integration
- Stdio support (local integrations)





Project's structure

- **MCPConnect.JRPC.***
 - JSON-RPC library (independent library in the future)
- **MCPConnect.Configuration.***
 - Configuration classes for JRPC/MCP (Neon, Auth, etc...)
- **MCPConnect.MCP.***
 - MCP core classes
- **MCPConnect.Transport.***
 - Transport mechanisms for MCP servers





MCP Project

- WebBroker application
- On FormCreate
 - FServer.OnParseAuthentication := ParseAuthentication;
 - VHandled := True;
- On WebModule MCPConnect Configuration
 - Like WiRL configuration
 - Fluent style, plugin-based





MCP configuration

```
FJRPCServer
```

```
  .Plugin.Configure<IAuthTokenConfig>
```

```
    .SetToken('my-secret-token')
```

```
    .ApplyConfig
```

```
  .Plugin.Configure<IMCPConfig>
```

```
    .SetServerName('delphi-mcp-server')
```

```
    .SetServerVersion('2.0.0')
```

```
    .RegisterWriter(TMCPImageWriter)
```

```
    .RegisterWriter(TMCPPictureWriter)
```

```
    .RegisterWriter(TMCPStreamWriter)
```

```
    .RegisterWriter(TMCPStringListWriter)
```

```
    .SetToolClass(TDataModule1)
```

```
  .ApplyConfig;
```



MCP Tools

- Attribute [MCPTool] on standard Delphi methods
- Attribute [MCPPParam] on function params
- **That's it, you have your MCPServer up and running!**





Demo:

Basic MCPConnect example

- ★ Server configuration
- ★ Delphi method to MCP tool
- ★ Call from Bruno



JRPC for Delphi

- Complete implementation of the JSON-RPC v2.0 protocol
 - `MCPConnect.JRPC.*`
- Define your APIs using simple Delphi class and attributes
- Automatic Marshaling
 - Conversion of Delphi objects into JSON-RPC requests and responses
- Broad Delphi types support
 - Using **Neon**, JRPC supports virtually every Delphi type





JRPC for Delphi

- Single or Batch requests
 - Important in MCP protocol
- Full Protocol Compliance
 - Full adherence to the JSON-RPC 2.0 specification





Attributes

- Attribute-Driven
 - Simply register classes to automatically discover tools, resources, and prompts
 - Use the **[McpTool]**, **[McpResource]**, **[McpPrompt]** attributes to expose specific methods (even already existing methods)





Session Management

- Thread-safe session support
- Configurable timeout
- Automatic cleanup
- Support for both generic (TJSONObject) and custom typed session data
- Sessions can be automatically injected via **[Context]** attribute





Sessions and Transports

- Session Behavior by Transport:
 - HTTP (WebBroker/Indy): Session ID passed via header or cookie. Server returns Mcp-Session-Id header on first request.
 - STDIO: Implicit session per connection - no session ID needed.





Authentication

- API-Key authentication
 - Bearer Token
 - More to come (OAuth2)
- Only for http servers (not stdio)





Automatic Routing

- MCPConnect automatically scans and registers methods
- Methods must be decorated with the appropriate attributes
- Requests are automatically routed to methods



JSON Schema Generation

- MCPConnect support any Delphi type as parameter or result
- Automatic JSON Schema generation
 - Using the Neon TSchemaGenerator





LLM integrations

- LLM integration through http(s)
 - stdio planned for the future
- Each LLM has a peculiar configuration
- What to use to test/use MCPConnect-based servers?
 - LLMStudio (local LLM)
 - Claude Desktop (remote LLM)
 - Gemini CLI (remote LLM)
 - ...





MCP Servers (base) configuration file

```
{
  "mcpServers": {
    "delphi-mcp": {
      "type": "http",
      "url": "http://localhost:8080/mcp",
      "headers": {
        "Authorization": "Bearer my-secret-token"
      }
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "<YOUR_TOKEN>"
      }
    }
  }
}
```



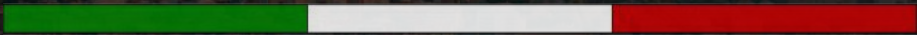
Demo:

LLM integration and use

- ★ Claude Desktop configuration
- ★ Basic interaction through prompts
- ★ Gemini CLI *

An aerial photograph of a city, likely Florence, Italy, showing a dense urban landscape with a river (the Arno) winding through it. A large, prominent square (Piazza della Signoria) is visible in the lower center, featuring a large building with arches. The image is overlaid with a semi-transparent dark brown filter.

DELPHIDAY


italian conference

THANK YOU