SGCWEBSOCKETS · TECHNICAL DOCUMENT

MCP Client

Model Context Protocol (MCP) client — connect Delphi apps to any MCP server (resources, tools, prompts).

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcWSAPIClient_MCP implements a Model Context Protocol (MCP) consumer on top of sgcWebSockets networking components. It takes care of session negotiation, JSON-RPC request/response marshalling and convenience events so Delphi & Cbuilder applications can discover prompts, resources and tools exposed by any MCP-compliant server.

At a glance

COMPONENT CLASS

`TsgcWSAPIClient_MCP`

STANDARDS / SPEC

Model Context Protocol — specification

TRANSPORTS

TCP, TLS

PLATFORMS

Windows, macOS, Linux, iOS, Android

FRAMEWORKS

VCL, FireMonkey, Lazarus / FPC, .NET

EDITION

Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	Model Context Protocol — specification · MCP — introduction
Component class	`TsgcWSAPIClient_MCP` (unit `sgcAI_MCP_Client`)
Frameworks	VCL, FireMonkey, Lazarus / FPC, .NET
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`MCPOptions`	Centralises every runtime option for the MCP client (endpoint URL, TLS, authentication, client identity, heartbeat, transport).
`Version`	Read-only semantic version of the MCP client component.

Main methods

The principal public methods exposed by the component.

`Ping()`	Issues a JSON-RPC ping request to keep the session alive.
`RequestTool()`	Invokes a named tool on the server (tools/call) with optional JSON arguments.
`RequestPrompt()`	Obtains the rendered content of a prompt template (prompts/get).
`RequestResource()`	Reads the contents of a resource identified by URI (resources/read).
`SubscribeResource()`	Subscribes to change notifications for a specific resource (resources/subscribe).
`UnsubscribeResource()`	Cancels a previous resource subscription (resources/unsubscribe).
`ListTools()`	Requests the catalogue of tools published by the server (tools/list).
`ListPrompts()`	Retrieves the available prompt templates from the server (prompts/list).

`ListResources()`	Enumerates resource descriptors published by the server (resources/list).
`ListResourceTemplates()`	Enumerates resource template descriptors (resources/templates/list).

Public events

The component exposes the following published events; consult the online help for full event-handler signatures.

`OnMCPElicitationCreate`	Fires when the server asks the user for additional input (elicitation/create).
`OnMCPInitialize`	Fires when the server replies to the initialize handshake; inspect capabilities and set Accept := False to abort.
`OnMCPListPrompts`	Receives the prompt templates catalogue returned by prompts/list.
`OnMCPListResources`	property OnMCPListResources: TsgcAI_MCP_Client_OnListResourcesEvent; // TsgcAI_MCP_Client_OnListResourcesEvent = procedure(Sender: TObject; const aRequest: TsgcAI_MCP_Request_ResourcesList; const aRes...
`OnMCPListRoots`	Fires when the server asks the client for its filesystem roots (roots/list).
`OnMCPListTools`	property OnMCPListTools: TsgcAI_MCP_Client_OnListToolsEvent; // TsgcAI_MCP_Client_OnListToolsEvent = procedure(Sender: TObject; const aRequest: TsgcAI_MCP_Request_ToolsList; const aResponse: TsgcAI_MC...
`OnMCPPing`	Fires when the server acknowledges a ping. Useful for round-trip telemetry.
`OnMCPResponsePrompt`	Delivers the rendered prompt content returned by prompts/get.
`OnMCPResponseResource`	Delivers the payload returned by resources/read, including streamed chunks.
`OnMCPResponseTool`	property OnMCPResponseTool: TsgcAI_MCP_Client_OnResponseToolEvent; // TsgcAI_MCP_Client_OnResponseToolEvent = procedure(Sender: TObject; const aRequest: TsgcAI_MCP_Request_ToolsCall; const aResponse: ...
`OnMCPSamplingCreateMessage`	TsgcWSAPIClient_MCP › Events › OnMCPSamplingCreateMessage

`OnMCPStreamMessage`	Fires while streaming responses are being read; inspect the raw JSON fragment and set Cancel := True to abort the stream.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **Sample code** configuration sourced from the online help.

> **About this scenario.** The example below loads the prompt list, invokes a prompt, and prints the returned messages.

Delphi (VCL / FireMonkey)

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 MCP.OnMCPListPrompts := MCPListPrompts;
 MCP.OnMCPResponsePrompt := MCPResponsePrompt;
end;
procedure TForm1.LoadPrompts;
begin
 MCP.ListPrompts;
end;
procedure TForm1.MCPListPrompts(Sender: TObject;
 const ARequest: TsgcAI_MCP_Request_PromptsList;
 const AResponse: TsgcAI_MCP_Response_PromptsList);
var
 LIndex: Integer;
 LPrompt: TsgcAI_MCP_Prompt;
begin
 for LIndex := 0 to AResponse.Prompts.Count - 1 do
 begin
  LPrompt := TsgcAI_MCP_Prompt(AResponse.Prompts.Item[LIndex]);
  Memo1.Lines.Add(Format('%s: %s', [LPrompt.Name, LPrompt.Description]));
 end;
end;
procedure TForm1.ExecutePrompt;
var
 LArgs: TsgcJSON;
begin
 LArgs := TsgcJSON.Create(nil);
 try
  LArgs.AddPair('code', 'ShowMessage(''Hello World'')');
  MCP.RequestPrompt('CodeReview', LArgs);
 finally
  LArgs.Free;
 end;
end;
procedure TForm1.MCPResponsePrompt(Sender: TObject;
 const ARequest: TsgcAI_MCP_Request_PromptsGet;
 const AResponse: TsgcAI_MCP_Response_PromptsGet);
var
 LIndex: Integer;
 LMessage: TsgcAI_MCP_Response_PromptsGet_Result_Message;
begin
 Memo1.Lines.Add('Prompt description: ' + AResponse.Result.Description);
 for LIndex := 0 to AResponse.Result.Messages.Count - 1 do
 begin
  LMessage := TsgcAI_MCP_Response_PromptsGet_Result_Message(AResponse.Result.Messages.Item[LInde
  if LMessage is TsgcAI_MCP_Response_PromptsGet_Result_Message_Text then
   Memo1.Lines.Add(LMessage.Role + ': ' +
    TsgcAI_MCP_Response_PromptsGet_Result_Message_Text(LMessage).Content.Text)
  else
   Memo1.Lines.Add(LMessage.Role + ': ' + LMessage.Write);
```

```
    end;
  end;
```

C++ Builder

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  MCP->OnMCPListPrompts = MCPListPrompts;
  MCP->OnMCPResponsePrompt = MCPResponsePrompt;
}
void __fastcall TForm1::LoadPrompts()
{
  MCP->ListPrompts();
}
void __fastcall TForm1::MCPListPrompts(TObject *Sender,
  const TsgcAI_MCP_Request_PromptsList &ARequest,
  const TsgcAI_MCP_Response_PromptsList &AResponse)
{
  for (int LIndex = 0; LIndex < AResponse.Prompts->Count; LIndex++)
  {
    TsgcAI_MCP_Prompt *LPrompt = (TsgcAI_MCP_Prompt*)AResponse.Prompts->Item[LIndex];
    Memo1->Lines->Add(Format("%s: %s", ARRAYOFCONST((LPrompt->Name, LPrompt->Description))));
  }
}
void __fastcall TForm1::ExecutePrompt()
{
  TsgcJSON *LArgs = new TsgcJSON(nullptr);
  try
  {
    LArgs->AddPair("code", "ShowMessage('Hello World')");
    MCP->RequestPrompt("CodeReview", LArgs);
  }
  __finally
  {
    delete LArgs;
  }
}
void __fastcall TForm1::MCPResponsePrompt(TObject *Sender,
  const TsgcAI_MCP_Request_PromptsGet &ARequest,
  const TsgcAI_MCP_Response_PromptsGet &AResponse)
{
  Memo1->Lines->Add("Prompt description: " + AResponse.Result->Description);
  for (int LIndex = 0; LIndex < AResponse.Result->Messages->Count; LIndex++)
  {
    TsgcAI_MCP_Response_PromptsGet_Result_Message *LMessage =
      (TsgcAI_MCP_Response_PromptsGet_Result_Message*)AResponse.Result->Messages->Item[LIndex];
    if (dynamic_cast<TsgcAI_MCP_Response_PromptsGet_Result_Message_Text*>(LMessage))
    {
      auto *MsgText = (TsgcAI_MCP_Response_PromptsGet_Result_Message_Text*)LMessage;
      Memo1->Lines->Add(LMessage->Role + ": " + MsgText->Content.Text);
    }
    else
    {
      Memo1->Lines->Add(LMessage->Role + ": " + LMessage->Write());
    }
```

```
  }
}
```

.NET (C#)

```
private void Form1_Load(object sender, EventArgs e)
{
  MCP.OnMCPListPrompts += MCP_ListPrompts;
  MCP.OnMCPResponsePrompt += MCP_RequestPrompt;
}
private void LoadPrompts()
{
  MCP.ListPrompts();
}
private void MCP_ListPrompts(object sender, MCPResponsePromptsList request, MCPResponsePromptsLi
{
  foreach (var prompt in response.Prompts)
  {
    memo1.AppendText($"{prompt.Name}: {prompt.Description}\r\n");
  }
}
private void ExecutePrompt()
{
  using (var args = new TsgcJSON())
  {
    args.AddPair("code", "ShowMessage('Hello World')");
    MCP.RequestPrompt("CodeReview", args);
  }
}
private void MCP_RequestPrompt(object sender, MCPResponsePromptsGet request, MCPResponsePromptsG
{
  memo1.AppendText("Prompt description: " + response.Result.Description + "\r\n");
  foreach (var message in response.Result.Messages)
  {
    if (message is MCPResponsePromptsGetResultMessageText msgText)
    {
      memo1.AppendText($"{message.Role}: {msgText.Content.Text}\r\n");
    }
    else
    {
      memo1.AppendText($"{message.Role}: {message.Write()}\r\n");
    }
  }
}
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — Model Context Protocol — specification	modelcontextprotocol.io/specification/2025-06-18
Primary standard / spec — MCP — introduction	modelcontextprotocol.io/introduction
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/AI/MCP/Client/TsgcWSAPIClient_MCP.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\15.AI\03.MCP\02.MCP_Client`
.NET demo project (in the sgcWebSockets package)	`.net\demos\15.AI\03.MCP\02.MCP_Client`
Component page	www.esegece.com/products/websockets/ai/mcp-client/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the MCP Client component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.