SGCWEBSOCKETS · TECHNICAL DOCUMENT

MCP Server

Model Context Protocol (MCP) server — expose your Delphi resources, tools and prompts to any MCP client (Claude, ChatGPT, IDEs).

eSeGeCe

WWW.ESEGECE.COM

Overview

TsgcWSAPIServer_MCP exposes the Model Context Protocol (MCP) over an sgcWebSockets HTTP server endpoint. The component bridges incoming HTTP requests with the TsgcAI_MCP_Server engine so that MCP-compatible clients can negotiate sessions, enumerate prompts/resources/tools, and invoke tools through JSON-RPC style calls.

At a glance

COMPONENT CLASS

`TsgcWSAPIServer_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	`TsgcWSAPIServer_MCP` (unit `sgcAI_MCP_Server`)
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.

`EndpointOptions`	HTTP endpoint (path) that receives MCP JSON-RPC traffic; defaults to /mcp.
`TransportOptions`	Enable/disable the HTTP and HTTP Streamable transports exposed by the server.
`Server`	HTTP server instance that hosts the MCP endpoint (TsgcWebSocketHTTP_Server or TsgcWebSocketServer_HTTPAPI).
`MCPOptions`	Core MCP server settings: ServerInfo, SessionTimeout and authentication policies.
`Version`	Returns the current version string of the sgcWebSockets library.

Main methods

The principal public methods exposed by the component.

`RequestRootsList()`	Sends a roots/list request to the MCP client identified by the session.
`RequestSamplingCreateMessage()`	Sends a sampling/createMessage request so the client's LLM can produce a completion.
`RequestElicitationCreate()`	Sends an elicitation/create request asking the client to collect structured input from the user.

`SendNotificationToolsListChanged()`	Broadcasts notifications/tools/list_changed so clients refresh their tool catalogue.
`SendNotificationPromptsListChanged()`	Broadcasts notifications/prompts/list_changed so clients refresh their prompt catalogue.
`SendNotificationResourcesListChanged()`	Broadcasts notifications/resources/list_changed so clients refresh their resource catalogue.
`SendNotificationResourcesUpdated()`	Notifies subscribed clients that a specific resource URI has been updated.
`SendLogMessage()`	Sends a notifications/message log entry to every session whose logging level allows it.
`KeepAlive()`	Sends a keep-alive frame on the supplied connection to prevent idle disconnection.

Public events

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

`OnMCPException`	Invoked when an unhandled exception reaches the server; allows overriding the HTTP response code.
`OnMCPHTTPRequest`	Fires when an HTTP request reaches the MCP endpoint; set Handled := True to supply a custom response.
`OnMCPHTTPResponse`	Fires after the HTTP response has been serialized; use it to add headers or log the exchange.
`OnMCPInitialize`	Fired during the MCP handshake so server capabilities and ServerInfo can be customised per session.
`OnMCPRequestPrompt`	Raised when a client issues prompts/get; fill the messages returned for that prompt template.
`OnMCPRequestResource`	Raised when a client issues resources/read; stream the resource content back to the caller.
`OnMCPRequestTool`	Raised when a client issues tools/call; populate the response with the tool's result.

`OnMCPResponseElicitationCreate`	TsgcWSAPIServer_MCP › Events › OnMCPResponseElicitationCreate
`OnMCPResponseRootsList`	Receives the client's reply to RequestRootsList, exposing the shared roots.
`OnMCPResponseSamplingCreateMessage`	TsgcWSAPIServer_MCP › Events › OnMCPResponseSamplingCreateMessage
`OnMCPSessionEnd`	Triggered when a session is closed or expires through SessionTimeout.
`OnMCPSessionNew`	property OnMCPSessionNew: TsgcAI_MCP_Server_OnSessionNewEvent; // TsgcAI_MCP_Server_OnSessionNewEvent = procedure(Sender: TObject; const aSession: TsgcAI_MCP_Session) of object __property TsgcAI_MCP_S...

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 snippets below request contact information, wait for the reply, and log the resulting action plus payload.

Delphi (VCL / FireMonkey)

```
procedure TFRMMCPServer.FormCreate(Sender: TObject);
begin
  MCPServer.OnMCPResponseElicitationCreate := MCPServerElicitationResponse;
end;
procedure TFRMMCPServer.RequestContactDetails(
  const aSession: TsgcAI_MCP_Session);
var
  LRequest: TsgcAI_MCP_Request_ElicitationCreate;
  LField: TsgcAI_MCP_Elicitation_Schema_Property;
begin
  if not aSession.ClientCapabilities.Elicitation.Enabled then
    Exit;
  LRequest := TsgcAI_MCP_Request_ElicitationCreate.Create;
  try
    LRequest.Params._Message := 'Please confirm your contact information.';
    LRequest.Params.RequestedSchema.Title := 'Contact information';
    LRequest.Params.RequestedSchema.Description :=
      'Used to keep you updated about this ticket.';
    LField := LRequest.Params.RequestedSchema.Properties.AddProperty('name');
    LField._Type := 'string';
    LField.Title := 'Full name';
    LField.Required := True;
    LField := LRequest.Params.RequestedSchema.Properties.AddProperty('email');
    LField._Type := 'string';
    LField.Title := 'Email address';
    LField.Format := 'email';
    LField.Required := True;
    LField := LRequest.Params.RequestedSchema.Properties.AddProperty('updates');
    LField._Type := 'boolean';
    LField.Title := 'Receive status updates';
    LField.Description := 'Tick to receive notifications when the ticket changes.';
    LField.DefaultBoolean := False;
    MemoLog.Lines.Add('elicitation/create id ' +
      MCPServer.RequestElicitationCreate(aSession, LRequest));
  finally
    LRequest.Free;
  end;
end;
procedure TFRMMCPServer.MCPServerElicitationResponse(Sender: TObject;
  const aSession: TsgcAI_MCP_Session;
  const aRequest: TsgcAI_MCP_Request_ElicitationCreate;
  const aResponse: TsgcAI_MCP_Response_ElicitationCreate);
begin
  MemoLog.Lines.Add('elicitation action: ' + aResponse.Action);
  if SameText(aResponse.Action, 'accept') and Assigned(aResponse.Content) then
    MemoLog.Lines.Add('payload: ' + aResponse.Content.Text);
end;
```

C++ Builder

```
void __fastcall TFRMMCPServer::FormCreate(TObject *Sender)
{
    MCPServer->OnMCPResponseElicitationCreate = MCPServerElicitationResponse;
}
void __fastcall TFRMMCPServer::RequestContactDetails(
    const TsgcAI_MCP_Session *ASession)
{
    if (!ASession->ClientCapabilities->Elicitation->Enabled)
        return;
    std::unique_ptr <TsgcAI_MCP_Request_ElicitationCreate> request(
        new TsgcAI_MCP_Request_ElicitationCreate());
    request->Params->_Message = "Please confirm your contact information.";
    request->Params->RequestedSchema->Title = "Contact information";
    request->Params->RequestedSchema->Description =
        "Used to keep you updated about this ticket.";
    auto *name = request->Params->RequestedSchema->Properties->AddProperty("name");
    name->_Type = "string";
    name->Title = "Full name";
    name->Required = true;
    auto *email = request->Params->RequestedSchema->Properties->AddProperty("email");
    email->_Type = "string";
    email->Title = "Email address";
    email->Format = "email";
    email->Required = true;
    auto *updates = request->Params->RequestedSchema->Properties->AddProperty("updates");
    updates->_Type = "boolean";
    updates->Title = "Receive status updates";
    updates->Description = "Tick to receive notifications when the ticket changes.";
    updates->DefaultBoolean = false;
    MemoLog->Lines->Add(
        "elicitation/create id " +
        MCPServer->RequestElicitationCreate(ASession, request.get())
    );
}
void __fastcall TFRMMCPServer::MCPServerElicitationResponse(
    TObject *Sender,
    const TsgcAI_MCP_Session *ASession,
    const TsgcAI_MCP_Request_ElicitationCreate *ARequest,
    const TsgcAI_MCP_Response_ElicitationCreate *AResponse)
{
    MemoLog->Lines->Add("elicitation action: " + AResponse->Action);
    if (AResponse->Action.LowerCase() == "accept" && AResponse->Content != nullptr)
        MemoLog->Lines->Add("payload: " + AResponse->Content->Text);
}
```

.NET (C#)

```
public partial class FRMMCPServer : Form
{
    public FRMMCPServer()
    {
        InitializeComponent();
        MCPServer.OnMCPResponseElicitationCreate += MCPServer_ElicitationResponse;
    }
    private void RequestContactDetails(TsgcAI_MCP_Session session)
    {
        if (!session.ClientCapabilities.Elicitation.Enabled)
            return;
        var request = new TsgcAI_MCP_Request_ElicitationCreate();
        request.Params._Message = "Please confirm your contact information.";
        request.Params.RequestedSchema.Title = "Contact information";
        request.Params.RequestedSchema.Description =
            "Used to keep you updated about this ticket.";
        var name = request.Params.RequestedSchema.Properties.AddProperty("name");
        name._Type = "string";
        name.Title = "Full name";
        name.Required = true;
        var email = request.Params.RequestedSchema.Properties.AddProperty("email");
        email._Type = "string";
        email.Title = "Email address";
        email.Format = "email";
        email.Required = true;
        var updates = request.Params.RequestedSchema.Properties.AddProperty("updates");
        updates._Type = "boolean";
        updates.Title = "Receive status updates";
        updates.Description = "Tick to receive notifications when the ticket changes.";
        updates.DefaultBoolean = false;
        var requestId = MCPServer.RequestElicitationCreate(session, request);
        memoLog.AppendText($"elicitation/create id {requestId}\r\n");
    }
    private void MCPServer_ElicitationResponse(
        object sender,
        TsgcAI_MCP_Session session,
        TsgcAI_MCP_Request_ElicitationCreate request,
        TsgcAI_MCP_Response_ElicitationCreate response)
    {
        memoLog.AppendText($"elicitation action: {response.Action}\r\n");
        if (string.Equals(response.Action, "accept", StringComparison.OrdinalIgnoreCase)
            && response.Content != null)
        {
            memoLog.AppendText("payload: " + response.Content.Text + "\r\n");
        }
    }
}
```

Common scenarios

The following scenarios are lifted verbatim from the online help. Each shows the configuration and method calls needed to drive the component through a specific real-world flow.

1 · MCP Server | Sessions

After a successful initialization between the server and the client, a new session is created. The session is always sent in the request and response between the client and the server to identify which session is being used in a request/response flow. When using the HTTP transport, the lifetime of a session is a single HTTP request and response, meaning that once the response has been sent, the session is closed.

Delphi (VCL / FireMonkey)

```
procedure OnMCPSessionEnd(Sender: TObject; const aSession:
    TsgcAI_MCP_Session);
begin
  WriteLn('#session_end: ' + aSession.Id);
end;
procedure OnMCPSessionNew(Sender: TObject; const aSession:
    TsgcAI_MCP_Session);
begin
  WriteLn('#session_new: ' + aSession.Id);
end;
```

C++ Builder

```
void __fastcall TForm1::OnMCPSessionEnd(TObject *Sender, const TsgcAI_MCP_Session *aSession)
{
    printf("#session_end: %s\n", aSession->Id.c_str());
}
void __fastcall TForm1::OnMCPSessionNew(TObject *Sender, const TsgcAI_MCP_Session *aSession)
{
    printf("#session_new: " + aSession->Id);
}
```

.NET (C#)

```
public void OnMCPSessionEnd(object sender, TsgcAI_MCP_Session aSession)
{
    Console.WriteLine("#session_end: " + aSession.Id);
}
public void OnMCPSessionNew(object sender, TsgcAI_MCP_Session aSession)
{
    Console.WriteLine("#session_new: " + aSession.Id);
}
```

2 · Tools List

The server keeps an in-memory catalogue of tools in TsgcAI_MCP_ToolsList. Tools are guaranteed to be unique by name, expose descriptions and provide a JSON-Schema-like input description that is emitted in the response to the specification's tools.list method. When a client invokes tools.list, TsgcWSAPIServer_MCP loads the request, serializes the current catalogue and sends it back with a 200 HTTP status code.

```
Delphi (VCL / FireMonkey)
```

```
procedure TMainForm.FormCreate(Sender: TObject);
var
  oTool: TsgcAI_MCP_Tool;
begin
  oTool := MCPServer.Tools.AddTool('math.add', 'Adds two numbers');
  oTool.InputSchema.Properties.AddProperty('a', True, aimcpjtNumber, 'Left operand');
  oTool.InputSchema.Properties.AddProperty('b', True, aimcpjtNumber, 'Right operand');
end;
```

```
C++ Builder
```

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Tool* oTool;
    // Add a tool named "math.add"
    oTool = MCPServer->Tools->AddTool("math.add", "Adds two numbers");
    // Define input schema properties
    oTool->InputSchema->Properties->AddProperty("a", true, aimcpjtNumber, "Left operand");
    oTool->InputSchema->Properties->AddProperty("b", true, aimcpjtNumber, "Right operand");
}
```

```
.NET (C#)
```

```
public partial class MainForm : Form
{
    private void MainForm_Load(object sender, EventArgs e)
    {
        var oTool = MCPServer.Tools.AddTool("math.add", "Adds two numbers");
        // Define input schema properties
        oTool.InputSchema.Properties.AddProperty("a", true, AIMCPJsonType.Number, "Left operand"
        oTool.InputSchema.Properties.AddProperty("b", true, AIMCPJsonType.Number, "Right operand
    }
}
```

3 · Prompts List

The server keeps an in-memory catalogue of prompts in TsgcAI_MCP_PromptsList. Prompts are guaranteed to be unique by name, expose descriptions and provide a JSON-Schema-like input description that is emitted in the response to the specification's prompts.list method. When a client invokes prompts.list, TsgcWSAPIServer_MCP loads the request, serializes the current catalogue and sends it back with a 200 HTTP status code.

Delphi (VCL / FireMonkey)

```
procedure TMainForm.FormCreate(Sender: TObject);
var
  oPrompt: TsgcAI_MCP_Prompt;
begin
  MCPServer.Prompts.Clear;
  oPrompt := MCPServer.Prompts.AddPrompt('CodeReview',
    'Asks the LLM to analyze code quality and suggest improvements');
  oPrompt.Arguments.AddArgument('code', 'The code to review', True);
end;
```

C++ Builder

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Prompt* oPrompt;
    // Clear all existing prompts
    MCPServer->Prompts->Clear();
    // Create a new prompt named "CodeReview"
    oPrompt = MCPServer->Prompts->AddPrompt(
        "CodeReview",
        "Asks the LLM to analyze code quality and suggest improvements"
    );
    // Add argument 'code'
    oPrompt->Arguments->AddArgument(
        "code",
        "The code to review",
        true  // required
    );
}
```

```
.NET (C#)
```

```
public partial class MainForm : Form
{
    private void MainForm_Load(object sender, EventArgs e)
    {
        // Clear existing prompts
        MCPServer.Prompts.Clear();
        // Create a new prompt named "CodeReview"
        var oPrompt = MCPServer.Prompts.AddPrompt(
            "CodeReview",
            "Asks the LLM to analyze code quality and suggest improvements"
        );
        // Add argument 'code'
        oPrompt.Arguments.AddArgument(
            "code",
            "The code to review",
            true // required
        );
    }
}
```

4 · Resources List

The server keeps an in-memory catalogue of resources in TsgcAI_MCP_ResourcesList. Resources are guaranteed to be unique by uri, expose descriptions and provide a JSON-Schema-like input URI that is emitted in the response to the specification's resources.list method. When a client invokes resources.list, TsgcWSAPIServer_MCP loads the request, serializes the current catalogue and sends it back with a 200 HTTP status code.

Delphi (VCL / FireMonkey)

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  MCPServer.Resources.Clear;
  MCPServer.Resources.AddResource(
        'file:///project/src/main.rs', // URI
    'main.rs',  // Name
        'Rust Software Application Main File', // Title
    'Primary application entry point', // Description
        'text/x-rust');
end;
```

C++ Builder

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    // Clear any previously registered resources
    MCPServer->Resources->Clear();
    // Register a Rust source file resource
    MCPServer->Resources->AddResource(
        "file:///project/src/main.rs",        // URI
        "main.rs",                            // Name
        "Rust Software Application Main File",// Title
        "Primary application entry point",    // Description
        "text/x-rust"                         // MIME type
    );
}
```

.NET (C#)

```
public partial class MainForm : Form
{
    private void MainForm_Load(object sender, EventArgs e)
    {
        // Clear existing resources
        MCPServer.Resources.Clear();
        // Register a Rust source file resource
        MCPServer.Resources.AddResource(
            "file:///project/src/main.rs",       // URI
            "main.rs",                           // Name
            "Rust Software Application Main File", // Title
            "Primary application entry point",     // Description
            "text/x-rust"                          // MIME type
        );
    }
}
```

5 · MCP Server Prompts — Advanced example

The next example publishes a prompt that guides an LLM through a triage workflow. It highlights optional arguments and how to return multi-step instructions in the handler.

Delphi (VCL / FireMonkey)

```
procedure TMainForm.FormCreate(Sender: TObject);
var
  LPrompt: TsgcAI_MCP_Prompt;
begin
  MCPServer.Prompts.Clear;
  LPrompt := MCPServer.Prompts.AddPrompt('IssueTriage', 'Collect details before escalating suppo
  LPrompt.Arguments.Clear;
  LPrompt.Arguments.AddArgument('summary', 'One-line description supplied by the user', True);
  LPrompt.Arguments.AddArgument('customerPriority', 'Optional priority label', False);
  LPrompt.Messages.AddText('system', 'You are a support assistant that triages technical issues.
end;
procedure TMainForm.MCPServerMCPRequestPrompt(Sender: TObject;
  const aSession: TsgcAI_MCP_Session; const aRequest: TsgcAI_MCP_Request_PromptsGet;
  const aResponse: TsgcAI_MCP_Response_PromptsGet);
begin
  if not SameText(aRequest.Params.Name, 'IssueTriage') then
    Exit;
  aResponse.Result.Description := 'Guided checklist for support analysts.';
  aResponse.Result.Messages.Clear;
  aResponse.Result.Messages.AddText('user', 'Review the ticket summary and ask clarifying questi
  aResponse.Result.Messages.AddText('assistant', 'Acknowledge the request and confirm the escala
  if aRequest.Params.Arguments.Node['customerPriority'].AsString <> '' then
    aResponse.Result.Messages.AddText('assistant', 'Adjust SLA checks based on the provided cust
end;
```

C++ Builder

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    TsgcAI_MCP_Prompt *prompt;
    MCPServer->Prompts->Clear();
    prompt = MCPServer->Prompts->AddPrompt("IssueTriage", "Collect details before escalating sup
    prompt->Arguments->Clear();
    prompt->Arguments->AddArgument("summary", "One-line description supplied by the user", true)
    prompt->Arguments->AddArgument("customerPriority", "Optional priority label", false);
    prompt->Messages->AddText("system", "You are a support assistant that triages technical issu
}
void __fastcall TMainForm::MCPServerMCPRequestPrompt(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_PromptsGet *aRequest,
    const TsgcAI_MCP_Response_PromptsGet *aResponse)
{
    if (!SameText(aRequest->Params->Name, "IssueTriage"))
        return;
    aResponse->Result->Description = "Guided checklist for support analysts.";
    aResponse->Result->Messages->Clear();
    aResponse->Result->Messages->AddText("user", "Review the ticket summary and ask clarifying q
    aResponse->Result->Messages->AddText("assistant", "Acknowledge the request and confirm the e
    if (aRequest->Params->Arguments->Node["customerPriority"]->AsString() <> "")
        aResponse->Result->Messages->AddText("assistant", "Adjust SLA checks based on the provid
}
```

.NET (C#)

```
private void MainForm_Load(object sender, EventArgs e)
{
    MCPServer.Prompts.Clear();
    var prompt = MCPServer.Prompts.AddPrompt("IssueTriage", "Collect details before escalating s
    prompt.Arguments.Clear();
    prompt.Arguments.AddArgument("summary", "One-line description supplied by the user", true);
    prompt.Arguments.AddArgument("customerPriority", "Optional priority label", false);
    prompt.Messages.AddText("system", "You are a support assistant that triages technical issues
}
private void MCPServer_MCPRequestPrompt(
    object sender,
    TsgcAI_MCP_Session session,
    TsgcAI_MCP_Request_PromptsGet request,
    TsgcAI_MCP_Response_PromptsGet response)
{
    if (!string.Equals(request.Params.Name, "IssueTriage", StringComparison.OrdinalIgnoreCase))
        return;
    response.Result.Description = "Guided checklist for support analysts.";
    response.Result.Messages.Clear();
    response.Result.Messages.AddText("user", "Review the ticket summary and ask clarifying quest
    response.Result.Messages.AddText("assistant", "Acknowledge the request and confirm the escal
    if (!string.IsNullOrWhiteSpace(request.Params.Arguments.Node["customerPriority"].AsString())
        response.Result.Messages.AddText("assistant", "Adjust SLA checks based on the provided c
}
```

6 · MCP Server Resources — Advanced example

The example below publishes a JSON resource that returns aggregated metrics. It demonstrates how to validate the incoming URI and craft different responses depending on the request.

```
Delphi (VCL / FireMonkey)
```

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  MCPServer.Resources.Clear;
  MCPServer.Resources.AddResource('metrics://daily', 'DailyMetrics', 'Daily KPI snapshot', 'Summ
end;
procedure TMainForm.MCPServerMCPRequestResource(Sender: TObject;
  const aSession: TsgcAI_MCP_Session; const aRequest: TsgcAI_MCP_Request_ResourcesRead;
  const aResponse: TsgcAI_MCP_Response_ResourcesRead);
var
  LTargetUri: string;
begin
  LTargetUri := aRequest.Params.Uri.ToLower;
  aResponse.Result.Contents.Clear;
  if LTargetUri = 'metrics://daily' then
  begin
    aResponse.Result.Contents.AddContentText('metrics://daily', 'DailyMetrics', 'Daily KPI snaps
    Exit;
  end;
  aResponse.Result.IsError := True;
  aResponse.Result.Contents.AddContentText(aRequest.Params.Uri, 'UnknownResource', 'Not found',
end;
```

C++ Builder

```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
    MCPServer->Resources->Clear();
    MCPServer->Resources->AddResource(
        "metrics://daily",
        "DailyMetrics",
        "Daily KPI snapshot",
        "Summaries generated each night",
        "application/json");
}
void __fastcall TMainForm::MCPServerMCPRequestResource(
    TObject *Sender,
    const TsgcAI_MCP_Session *aSession,
    const TsgcAI_MCP_Request_ResourcesRead *aRequest,
    const TsgcAI_MCP_Response_ResourcesRead *aResponse)
{
    UnicodeString targetUri = aRequest->Params->Uri.LowerCase();
    aResponse->Result->Contents->Clear();
    if (targetUri == "metrics://daily")
    {
        aResponse->Result->Contents->AddContentText(
            "metrics://daily",
            "DailyMetrics",
            "Daily KPI snapshot",
            "application/json",
            BuildMetricsPayload(Date()));
        return;
    }
    aResponse->Result->IsError = true;
    aResponse->Result->Contents->AddContentText(
        aRequest->Params->Uri,
        "UnknownResource",
        "Not found",
        "text/plain",
        "Resource is not published by this server.");
}
```

```
.NET (C#)
```

```
private void MainForm_Load(object sender, EventArgs e)
{
    MCPServer.Resources.Clear();
    MCPServer.Resources.AddResource(
        "metrics://daily",
        "DailyMetrics",
        "Daily KPI snapshot",
        "Summaries generated each night",
        "application/json");
}
private void MCPServer_MCPRequestResource(
    object sender,
    TsgcAI_MCP_Session session,
    TsgcAI_MCP_Request_ResourcesRead request,
    TsgcAI_MCP_Response_ResourcesRead response)
{
    response.Result.Contents.Clear();
    if (string.Equals(request.Params.Uri, "metrics://daily", StringComparison.OrdinalIgnoreCase))
    {
        response.Result.Contents.AddContentText(
            "metrics://daily",
            "DailyMetrics",
            "Daily KPI snapshot",
            "application/json",
            BuildMetricsPayload(DateTime.Today));
        return;
    }
    response.Result.IsError = true;
    response.Result.Contents.AddContentText(
        request.Params.Uri,
        "UnknownResource",
        "Not found",
        "text/plain",
        "Resource is not published by this server.");
}
```

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/Server/TsgcWSAPIServer_MCP.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\15.AI\03.MCP\01.MCP_Server`
.NET demo project (in the sgcWebSockets package)	`.net\demos\15.AI\03.MCP\01.MCP_Server`
Component page	www.esegece.com/products/websockets/ai/mcp-server/
Product page	www.esegece.com/products/websockets/

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