MCP Server

The Model Context Protocol (MCP) is a standard that enables AI applications to connect to external systems.
In VTENEXT, this standard is used to expose REST APIs, custom methods and processes to AI applications, to enable them to interact with the CRM itself.

Currently VTENEXT has set up a basic MCP server with the main webservices (referred here as *tools*) that ensure a wide extent of interoperability with the CRM; the user can choose to extend this server by registering other webservices to it, or even create *custom tools* to be used with the servers. Be mindful that, with roughly more than 30 tools registered to a MCP server, the accuracy of the server might degrade.

In VTENEXT, the MCP servers are fundamentally handled like any other webservice, but they have their own SDK methods to work with them.

Tools registered in the base MCP server (REST name: `mcp`)

- `create`
- `delete`
- `describe`
- `listtypes`
- `query`
- `relate`
- `retrieve`
- `retrieveInventory`
- `get_currencies`
- `updateRecord`
- `convert_lead`
- `get_current_context`
- `process_text`
- `summarize`
- `tools_manual`
- `translate`

SDK methods for working with MCP servers

Setting up a MCP server

```
SDK::setMcpServer(string $name, ?string $description = null, bool $isActive = true, ?string $operationName = null)
```

- `$name`: the name of the MCP server;
- `$description`: human-readable description of the MCP server;
- `$isActive`: set it to `false` to prevent the Server from accepting requests;
- `$operationName`: the name to give to the corresponding REST Webservice method; defaults to the name of the MCP server prefixed with `mcp.`.

This function returns the id of the MCP server if created successfully, and `false` otherwise.

Registering a tool to a MCP server

```
SDK::registerMcpTool(string $mcp, string $name, string $type, ?string $tool_name = null, ?string $description = null)
```

- `$mcp`: the name of the MCP server;
- `$name`: the name of the webservice method or the custom MCP tool;
- `$type`: the type of the tool to register, can be `operation` for the webservice method or `custom` for the custom tool;
- `$tool_name`: the name to give to the registered tool, defaults to `$name`;
- `$description`: MCP-specific information to give to the registered tool, will take precedence over any information provided by the webservice method/custom tool.

If successful, this function returns an array with the id of the MCP server and the id of the webservice method/custom tool; this function returns `false` on failure.

Defining a custom tool for MCP servers

```
SDK::setMcpTool(string $name, string $handlerFilePath, ?string $handlerMethodName = null, ?string $description = null, ?array $inputSchema = null, ?array $outputSchema = null)
```

- `$name`: the name of the custom tool;
- `$handlerFilePath`: file path where the function is defined;
- `$handlerMethodName`: name of the function to use from the specified handler file;
- `$description`: informative description of the custom tool;
- `$inputSchema`: the definition of the function input;
- `$outputSchema`: the definition of the function output.

`$inputSchema` and `$outputSchema` follow the JSON Schema 2020-12 Specification.

N.B.: the function should always return an array and, if possible, the array should be associative.

Removing a MCP server

```
SDK::unsetMcpServer(string $mcp, bool $force = false)
```

- `$mcp`: the name of the MCP server to delete;
- `$force`: if `true`, force the deletion of the MCP server, even if it has tools registered to it.

This function returns `true` on success and `false` on failure.

Deregistering a tool from a MCP server

```
SDK::unregisterMcpTool(string $mcp, string $name, string $type)
```

- `$mcp`: the name of the MCP server;
- `$name`: the name of the tool to deregister;
- `$type`: the type of the tool to deregister, can be `operation` or `custom`.

This function returns `true` on success and `false` on failure.

Deregistering a custom tool

```
SDK::unsetMcpTool(string $name)
```

- `$name`: name of the custom tool to delete.

This function returns `true` on success and `false` on failure.

Revision #5
Created 2026-06-25 14:59:48 UTC by m.maporti
Updated 2026-07-08 13:27:53 UTC by f.fabris