22.3 MCP Client

The **MCP Clients** section is used to connect vtenext to an external or internal MCP server, allowing it to retrieve and use the available tools within processes and agents. In practice, this section is where you configure how to reach the MCP server, how to authenticate, and how to keep the list of available tools up to date.

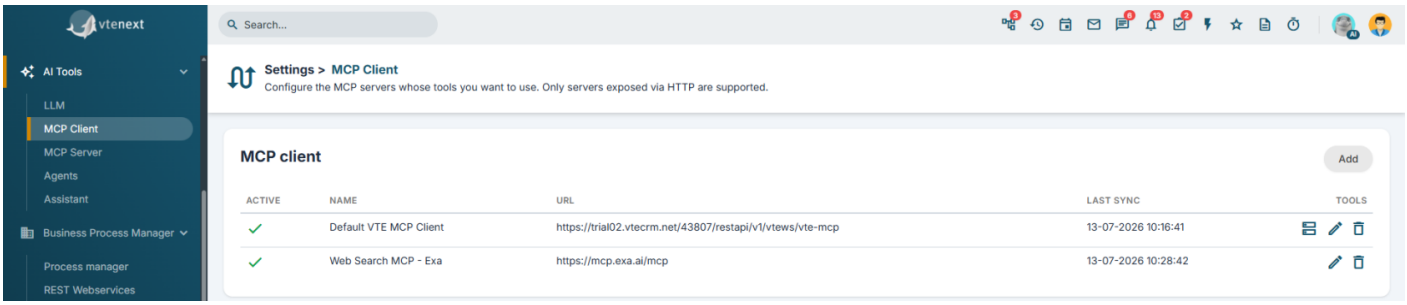
> **In summary:** the **MCP client** does not publish functions externally. It performs the opposite operation: it connects to an **MCP server**, retrieves its tools, and makes them available within vtenext.

When to use it

- To connect vtenext to an already available MCP server.
- To use the tools exposed by an MCP server within processes and agents (see AI Actions).
- To keep the list of available tools synchronized over time.
- To quickly connect an MCP server created in the MCP Servers section.

The List View

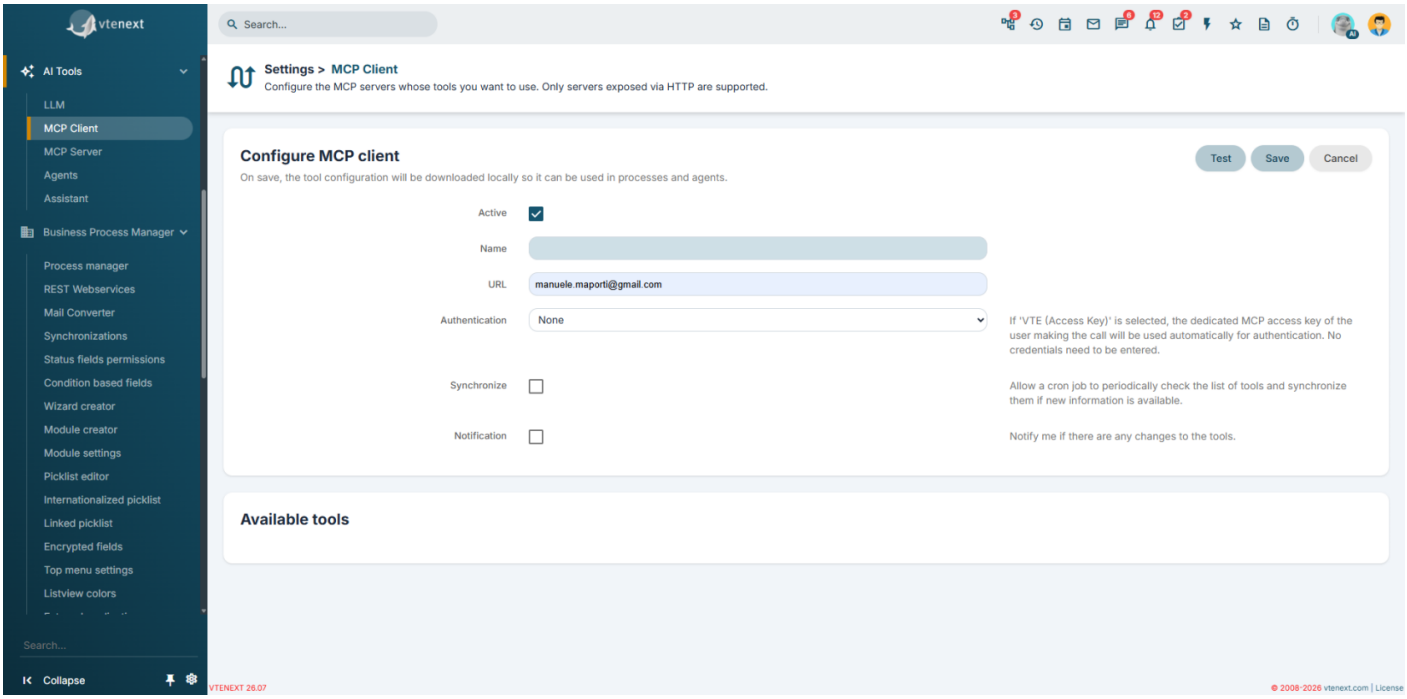
The list displays all previously configured **MCP clients**.



Active	indicates whether the client is enabled
Name	internal name used to identify the connection
URL	address of the MCP server to which the client connects
Last Synchronization	shows when the tools were last synchronized, or indicates that the client has never been synchronized

From the list, you can create a new client, edit it, or delete it. If the client is connected to an internal **MCP Server**, you can also directly open that server from the list.

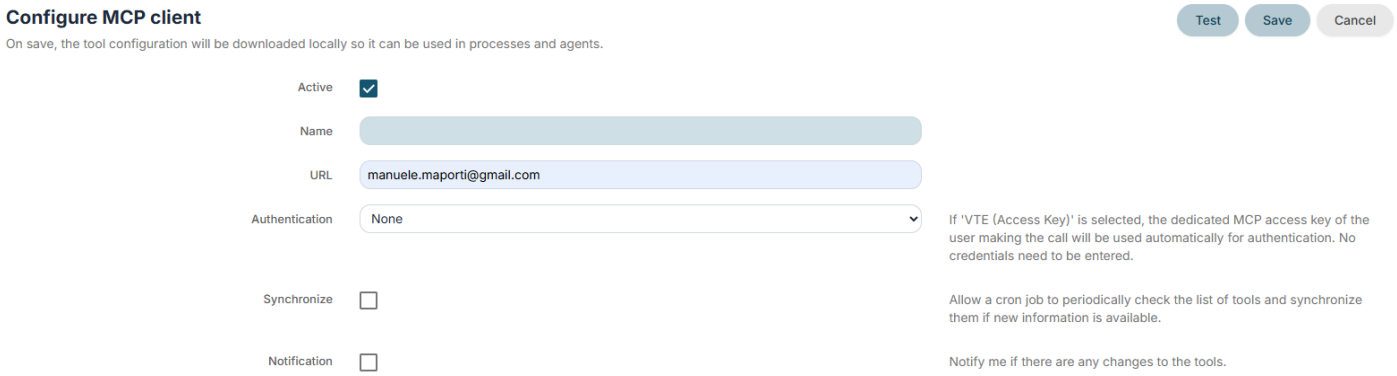
Creating a New MCP Client



1. Open the **MCP Clients** section.
2. Click **Add**.
3. Enter an easily recognizable **Name**.
4. Enter the **URL** of the MCP server.
5. Choose the **Authentication** type.
6. Fill in the required credentials, if applicable.
7. Decide whether to enable **Periodic Synchronization**.
8. Enable **Notifications** if you want to be notified when tools change.
9. Save the configuration.

> **Important:** when saving, vtenext actually attempts to connect to the MCP server. If the connection fails, the configuration is not accepted.

Configuration Parameters



Active	enables or disables the client
Name	descriptive name of the connection
URL	full address of the MCP server to connect to
Authentication	defines how the client authenticates with the MCP server
Username and Password	displayed when **Basic Authentication** is selected
API Key	displayed when **Bearer** or **X-API-Key** authentication is selected
Synchronize	allows a cron job to periodically check the available tools and update them when new ones are detected
Notification	sends a notification when synchronization detects changes in the server's tools

How authentication works

The selected authentication method changes the fields displayed in the configuration form.

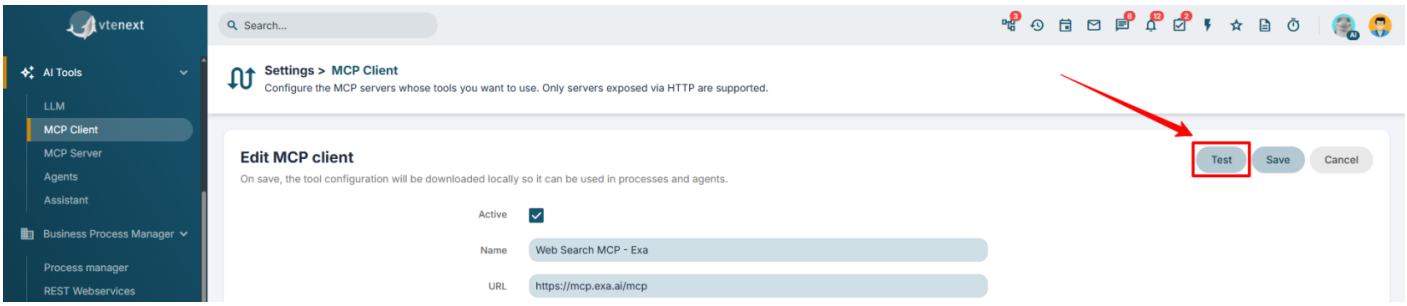
No Authentication	no additional credentials are required
Basic	you must enter a username and password
Bearer	you must enter a token or API key
X-API-Key	you must enter the API key in the dedicated field
VTE (Access Key)	vtenext automatically uses the dedicated MCP access key of the user performing the call. No credentials need to be entered manually

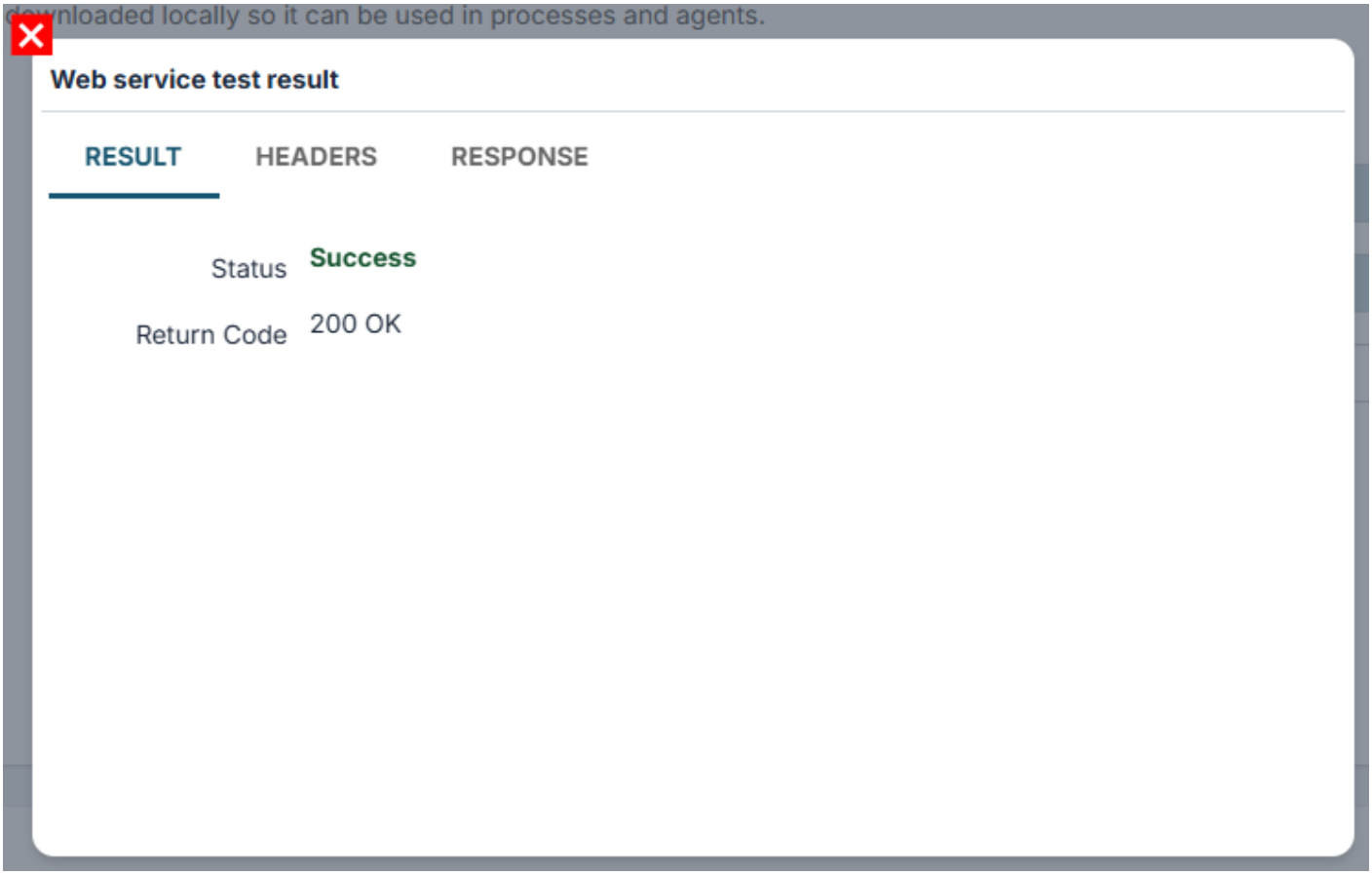
> **When to use VTE (Access Key):** this option is particularly useful when the client points to an MCP Server exposed by vtenext, because it allows the system to manage authentication

automatically.

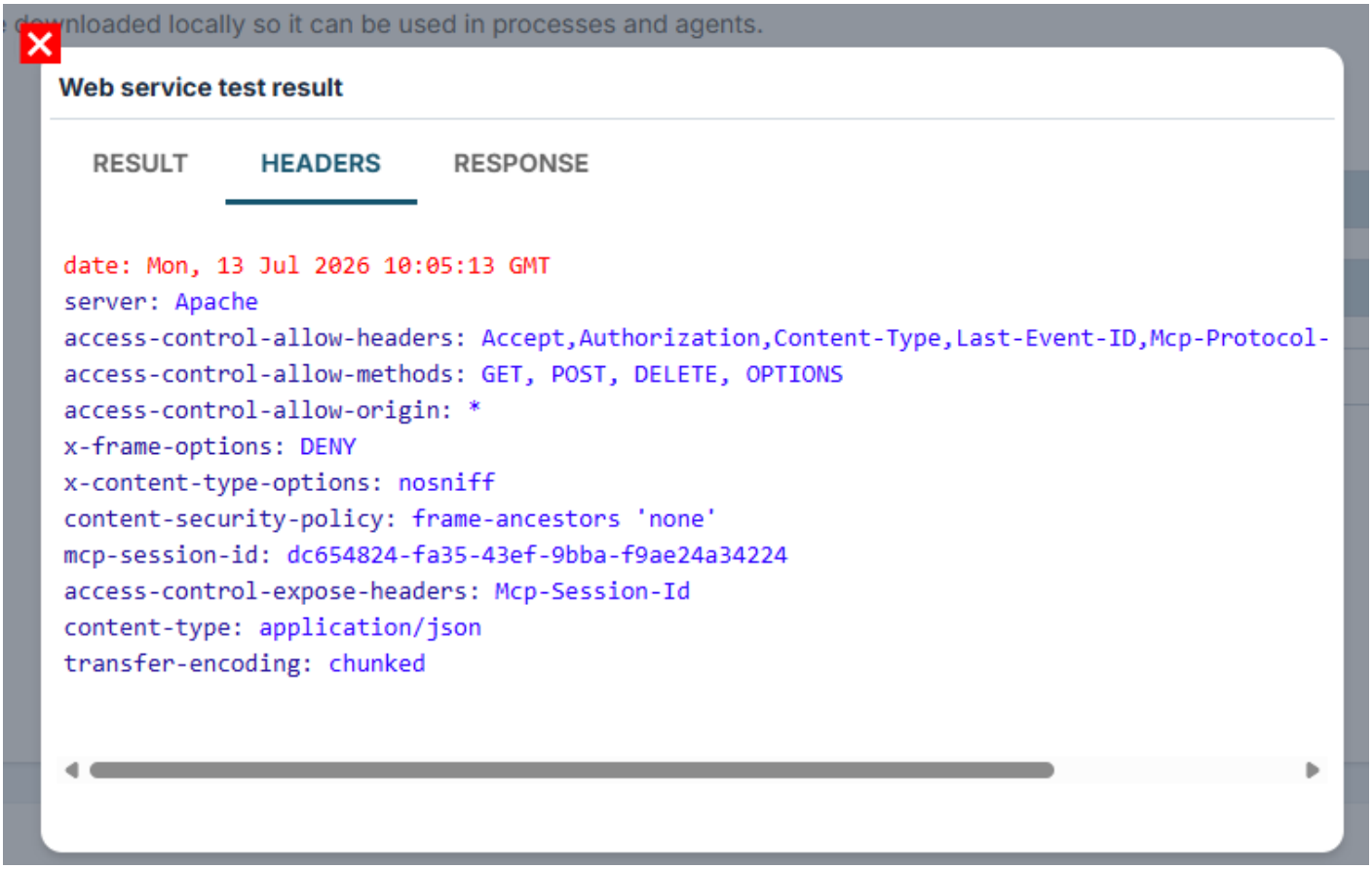
Connection Verification and Tool Mapping

The client screen also provides a verification function.

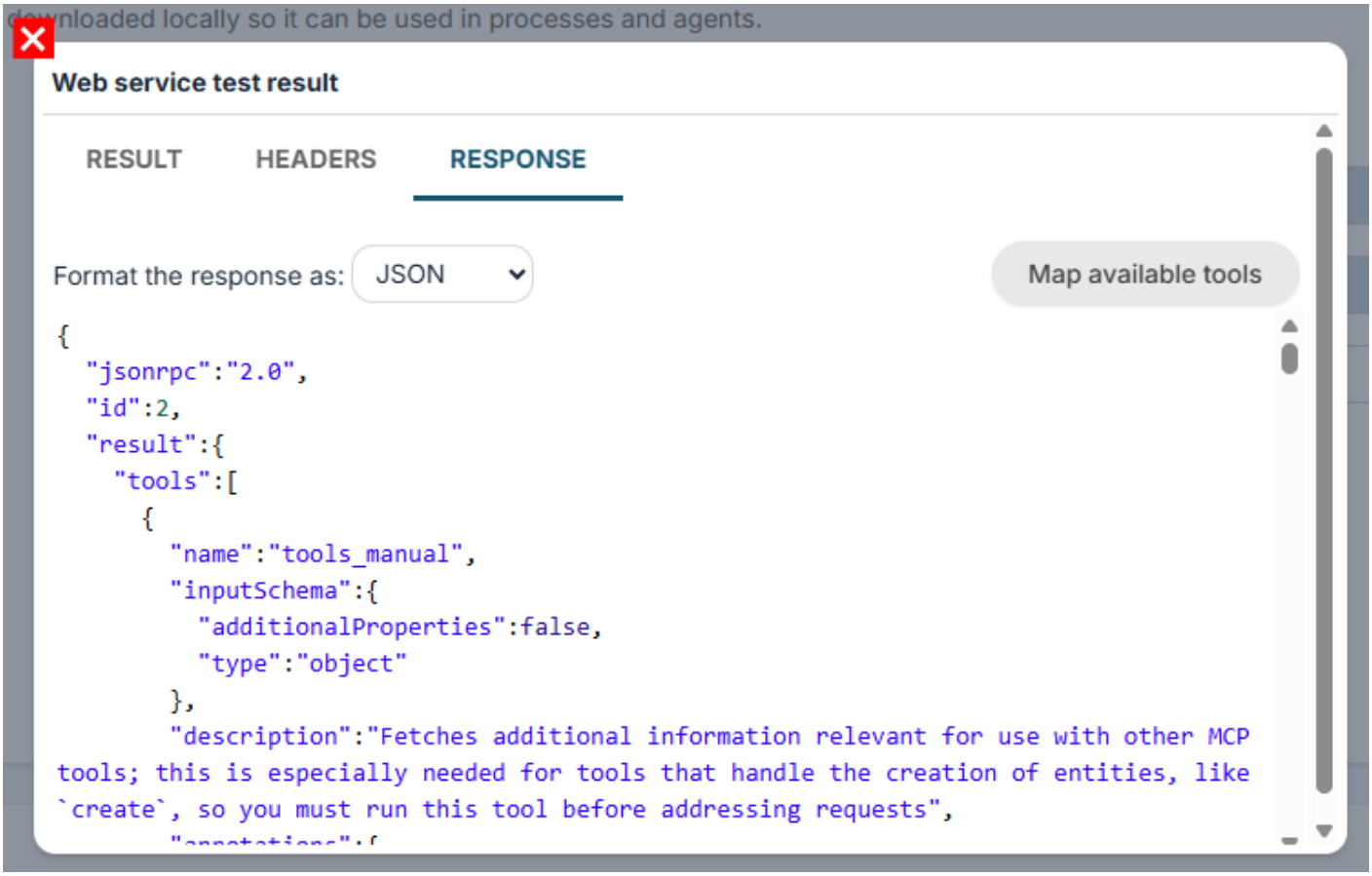




Call Result: RESULT



Call Result: HEADERS



Call Result: RESPONSE

- The test button checks whether the MCP server responds correctly.
- If the result is successful, vtenext can retrieve the list of available tools.
- From the verification window, you can update the available tools section using the **Map Available Tools** command.

This verification is especially useful when configuring a new server or when you want to check that authentication is working correctly before the final save.

Available Tools

After saving, or after mapping from the verification window, vtenext displays the list of tools retrieved from the MCP server.

Available tools

convert_lead

Starts the process for lead conversion

Input schema

leadid	integer	REQUIRED	The ID of the lead to convert

Output schema

info	string	OPTIONAL	The information if the process was delegated succesfully.
error	string	OPTIONAL	Error message if the lead conversion could not be delegated to a process.

create

Create a record of the specified module. To understand the fields of the specified module, you should use the `describe` tool; mind that related id fields (eg. assigned_user_id) are in the format `module_id` + `x` + `record_id`. You can use, if available, the tool `tools_manual` to get the list of modules with their 'module_id', and the tool `query` to get the 'record_id'.

Input schema

elementType	string	REQUIRED	The type of the element (module) examples: ["Accounts"]
element	object	REQUIRED	The encoded JSON with the record information examples: ["{\"accountname\":\"account1\", \"assigned_user_id\":\"19×1\"}"]

delete

Delete the specified record

Input schema

id	string	REQUIRED	The id of the module's record, with the format `module_id` + `x` + `record_id` examples: ["3×306"]

describe

Describe the specified module and their relative fields; you must run this tool before attempting any task, to understand the structure of the CRM's modules

Input schema

elementType	string	REQUIRED	The type of the element (module). In most cases, this corresponds to the CRM module name. Special handling applies to calendar records: Use `Calendar` for tasks, to-dos, reminders, and other non-event activities. Use `Events` for appointments, meetings, calls, and all event-type records, even though they are physically stored in the Calendar module. examples: ["Accounts"]

get_current_context

Fetches general context information relevant for use with other MCP tools (so it does not need any parameters); you should run this tool before addressing requests, if you need information about the user currently logged in the CRM

Input schema

-

Output schema

context	array	OPTIONAL	Array of markdown-styled lines for the MCP client to read, containing the information of the user currently logged in.

listtypes

List the modules that have fields with `uitype` of the specified formats

Input schema

fieldTypeList	array	REQUIRED	Array of requested primitive types examples: [["integer","file"]]

Output schema

types	array	OPTIONAL	The list of modules with the requested uitypes
information	object	OPTIONAL	The list of fields' information, grouped by modules

process_text

Rewrite or improve the supplied text using an optional tone of voice and complexity level.

Input schema

text	string	REQUIRED	The source text to process.
tone_of_voice	string	OPTIONAL	Optional tone of voice, for example professional, friendly, formal, informative, reassuring, convincing, or a custom tone.
complexity	string	OPTIONAL	Optional response complexity. enum: ["simple","intermediate","advanced"]

Output schema

result	string	OPTIONAL	The processed text.
error	string	OPTIONAL	Error message if the text could not be processed.

query

Safely execute a SQL-like query and return the result's rows. Additional notes: - The `FROM` clause accepts module names from Modules ONLY, DO NOT invent new ones. - DO NOT attempt to query Modules that cannot be queried: see `tools_manual`, if available, for information. - There are NO tables with the `vtiger` prefix. - Aggregate functions (e.g. `SUM`, `MAX`, `MIN`, `COUNT`...), the operators `BETWEEN`, `NOT`, and `OFFSET` are NOT supported. - When selecting in "inventory" modules, NEVER work with `product_block`, since it is not a real column. - All ids are accepted in the format `module_id` + `x` + `record_id`. Always use the tool `describe` each time on the module of interest to understand the columns before executing the query.

Input schema

query	string	REQUIRED	The SQL query to execute, note that: - The `FROM` clause accepts module names ONLY. - If you need to work with the id column of a module, just use `id` as the column name: this webservice will figure out which id column it actually is. - Aggregate functions (e.g. `SUM`, `MAX`, `MIN`, `COUNT`...), the operators `BETWEEN`, `NOT`, `<>`, and `OFFSET` are NOT supported. - Nested queries are NOT supported. - NEVER user parenthesis, EXCEPT with the `IN` operator. - query MUST ended by semicolon `;` examples: ["SELECT * FROM Accounts WHERE accountname LIKE '%vte%';"]

relate

Relate the record with all the records in a relatelist array

Input schema

id	string	REQUIRED	The id of the module's record, with the format `module_id` + `x` + `record_id` examples: ["3×310"]
relatelist	array	REQUIRED	The list of IDs (see `id`) to relate the record with
relationid	string	OPTIONAL	If provided, the specific relation id to relate the record with

retrieve

Illustrate the fields of the specified record and their relative values

Input schema

id	string	REQUIRED	The id of the module's record, with the format `module_id` + `x` + `record_id` examples: ["4×44"]

retrieveInventory

Illustrate the fields of the specified inventory record, their relative values and the product block's information

Input schema

id	string	REQUIRED	The id of the module's record, with the format `module_id` + `x` + `record_id` examples: ["16×104"]

summarize

Generate a summary of a VTENEXT record. Use this tool whenever the user asks for a summary, recap, abstract, synthesis or overview of a record. Currently, only the HelpDesk, ModComments (chat) and

Input schema

id	string	REQUIRED	The id of the record to summarize, preferably with the format `module_id` + `x` + `record_id`; a bare numeric CRM id is also accepted. examples: ["3×310"]

For each tool, you can view:

- tool name
- description
- input parameters
- required and optional fields
- structure of the returned output

This information is stored locally, so the tools can then be invoked within agents and processes without having to reinterpret the remote server structure each time.

Periodic Synchronization

If you enable **Synchronize**, a scheduled process can periodically check whether new or modified tools are available on the MCP server.

- If changes are detected, the local tool configuration is updated.
- If no differences are found, the date of the last synchronization is updated anyway.
- If you have also enabled **Notification**, you will receive an alert when the set of tools changes.

> **Warning:** when tools change on the MCP server, it may be necessary to check any processes or agents that use them, especially if parameters or outputs have changed.

Practical examples

Example 1: Connecting to an external MCP server

You can configure an MCP client that points to an external server published by another system. In this case, enter the URL, the required authentication method, and then verify the available tools.

Example 2: Connecting to a vtenext MCP Server

If you have already created an MCP Server in the relevant section, you can connect the client to the same endpoint and use **VTE (Access Key)** authentication to simplify the configuration.

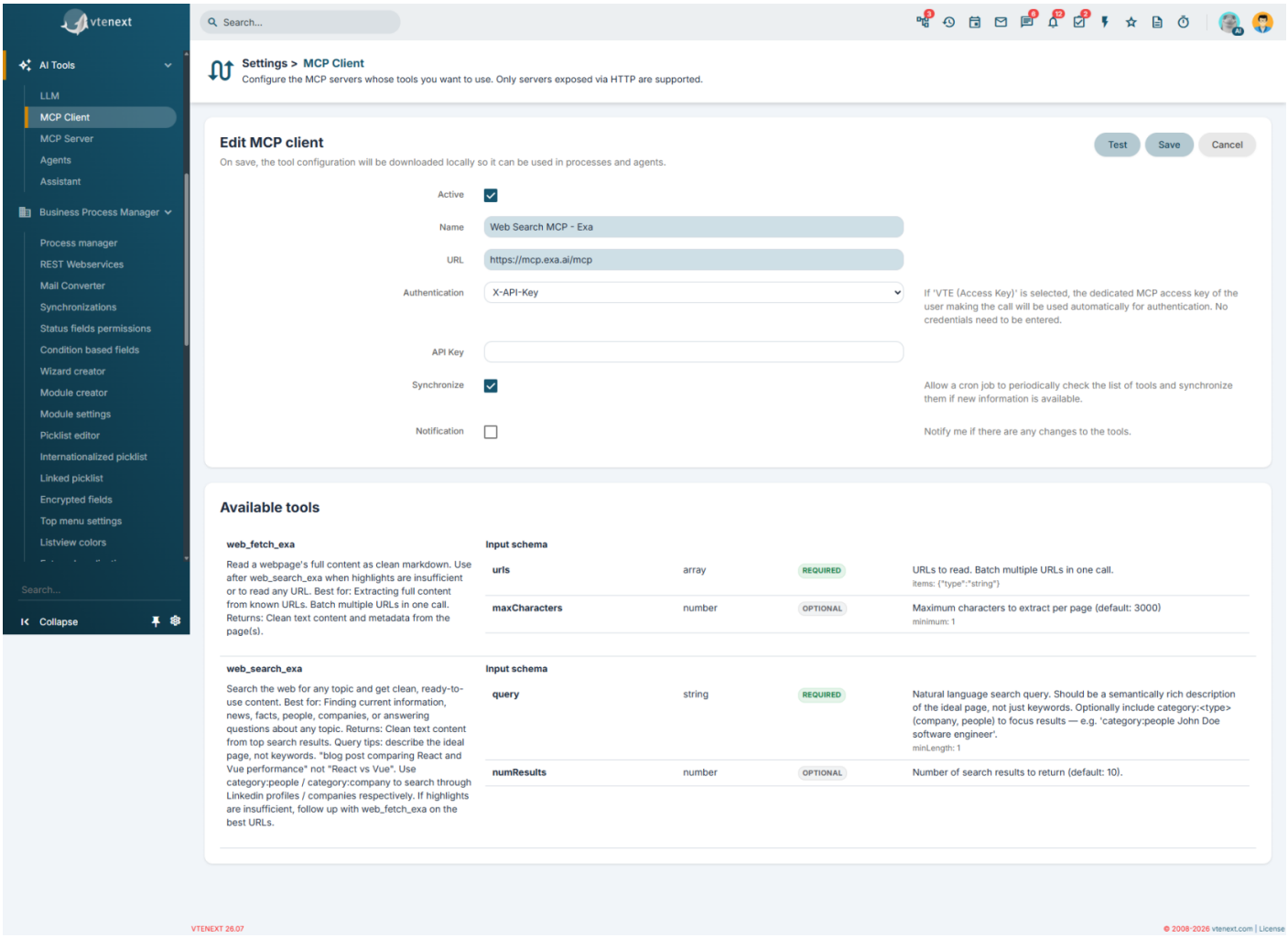
Example 3: Controlled tool updates

If the tools on the remote server may change over time, it is recommended to enable periodic synchronization and, if necessary, notifications as well, so that processes and agents remain aligned.

Web Search MCP - Exa

vtenext also provides a preconfigured MCP client called **Web Search MCP - Exa**. This entry is created automatically but remains disabled until you decide to use it.

Exa is a web search service designed for AI agents. It allows real-time web searches, retrieval of content from specific web pages, and provides models with more up-to-date information compared to the knowledge already embedded within the model.



How to activate it

1. Open the **Web Search MCP - Exa** entry in the **MCP Clients** list.
2. Leave the preconfigured URL unchanged, as it points to Exa’s public MCP server.
3. Set or complete the **X-API-Key** authentication with the API key from your Exa account.
4. Enable the client.
5. Save and verify the available tools.

In this case, the authentication method expected by vtenext is **X-API-Key**, which is already selected in the initial client configuration.

Available Tools with Exa

The MCP server exposes the following tools:

- **web_search_exa:** performs web searches and returns content already prepared for use by the agent.
- **web_fetch_exa:** retrieves the full content of a web page starting from a known URL.

> **When it can be useful:** Exa is particularly suitable when you want to provide agents and processes with access to up-to-date web searches, online page analysis, or the rapid collection of external information.

For further details about the service and the features of the Exa MCP server, refer to the official website: https://exa.ai/mcp.

Revision #11
Created 2026-07-13 08:58:10 UTC by Admin
Updated 2026-07-13 13:50:00 UTC by Admin