USER MANUAL

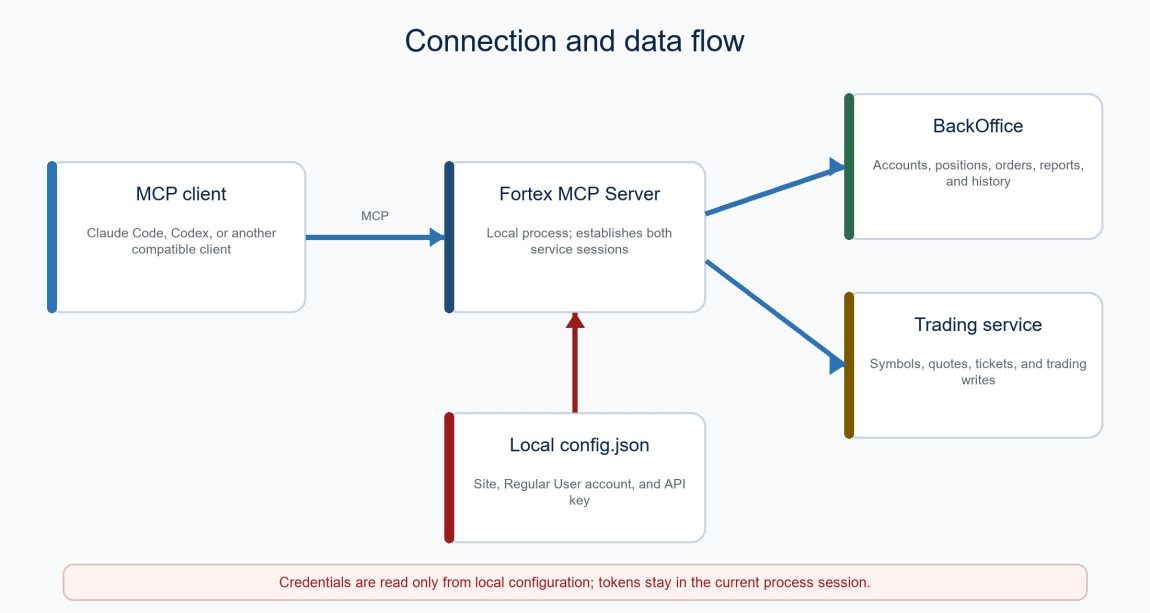
Fortex API MCP Server User Manual

This manual is for end users working through Claude Code, Codex, or another MCP-compatible client. It explains how to install, configure, verify, and safely use Fortex reporting, market-data, and trading capabilities.

INSTALL · CONFIGURE · READ · TRADE · TROUBLESHOOT

For Claude Code, Codex, and MCP-compatible clients

> **Important:** Trading write operations take effect immediately. There is no preview step, and changes may not be reversible. Use a dedicated test account for initial setup and testing.



Connection and data-flow overview

1. Before you start

Prepare the following:

- A Windows or macOS computer.
- An MCP-compatible client. Claude Code CLI and Codex support reads and trading. The Claude Code desktop app supports reads but does not allow trading write operations.
- An API key and raw API-key password for a Fortex **Regular User** account. Administrator accounts are refused.
- The Fortex service URL, such as `https://bo.yourcompany.com`.
- A test account for first-time verification. Do not test new trading workflows directly on a production account.

Public repository and setup entry point: Fortexinc/mcp-server. Manual packages are available from Releases.

2. Installation

2.1 Claude Code: plugin installation (recommended)

Run these commands in Claude Code:

```
/plugin marketplace add Fortexinc/mcp-server
/plugin install fortex-api-windows@fortex    # Windows
/plugin install fortex-api@fortex            # macOS
```

Start a new session. On first use, the plugin downloads the server and creates a blank `config.json` in its data directory. The terminal prints the exact path. Use that path instead of guessing the plugin directory name.

Common locations:

Platform	Plugin configuration file
Windows	`%USERPROFILE%\.claude\plugins\data\fortex-api-windows-fortex\config.json`
macOS	`~/.claude/plugins/data/fortex-api-fortex/config.json`

Use the update command to keep the configuration file:

```
/plugin update fortex-api-windows
```

Uninstalling the plugin deletes its `config.json`, including the API key. Do not use uninstall-and-reinstall as an upgrade method.

2.2 Codex

The Codex plugin expects `fortex-api-mcp` to be available on the system `PATH`. First install the manual package for your platform as described below. Then add `Fortexinc/mcp-server` as a plugin marketplace in Codex, install through `/plugins`, and start a new session.

2.3 Manual installation

Windows

1. Download the current Windows package from Releases.
2. Extract every file into a new directory.
3. Fill in the `config.json` in that directory as described in section 3.
4. Run `install.bat`.
5. Close and reopen the Claude Code or Codex session.

macOS

1. Download the current macOS package from Releases.
2. Extract every file into a new directory.
3. Fill in the `config.json` in that directory as described in section 3.
4. Run:

```
chmod +x install.sh fortex-api-mcp
./install.sh
```

Finally: Start a new Claude Code or Codex session.

The first launch of a new binary can take about 30 seconds while the operating system scans it. Later launches are normally faster. If macOS still blocks the program, right-click it in Finder and select **Open** once.

3. Configure the connection

`config.json` stores the service URL and API-key credentials. Credentials belong only in the local configuration file or the supported environment variables. Do not place them in an MCP registration command, a chat message, or tool arguments.

```
{
  "net": {
    "timeout": 30,
    "verify_tls": true
  },
  "sites": {
    "main": {
      "url": "https://bo.yourcompany.com",
      "accounts": {
        "YOUR_ACCOUNT": {
          "apikey": "your-api-key",
          "password": "your-api-key-raw-password"
        }
      }
    }
  }
}
```

Field guidance:

- `url`: one Fortex site. The same URL covers BackOffice reporting and trading.
- `YOUR_ACCOUNT`: the Regular User account name exactly as issued by the broker.
- `apikey`: the API key created for that account.
- `password`: the API key's raw password, not a hash.
- `timeout`: the normal network-request timeout in seconds.
- `verify_tls`: keep this `true` in production. Set it to `false` only in a development environment that uses a self-signed certificate.

Save the file as **UTF-8 without BOM**. It is read as JSONC, so `//` and `/* ... */` comments are accepted.

3.1 Multiple sites and accounts

At startup, the server selects only the first entry under `sites` and the first entry under that site's `accounts`. It then establishes both reporting and trading sessions automatically.

- Other sites and accounts may remain in the file, but the server does not switch between them at runtime.
- To change the active account, reorder the JSON entries and start a new session.
- When a tool omits `account`, the current login account is filled in automatically.
- Do not ask a tool to read another account, `*`, or `all`. Customer operations are restricted to the current login account.

3.2 Environment-variable fallback

Using `config.json` is recommended. If policy requires secrets in environment variables, use:

```
FORTEX_APIKEY_<ACCOUNT>
FORTEX_APIKEYPW_<ACCOUNT>
```

`<ACCOUNT>` corresponds to the configured account name. Never expose the variable values in screenshots, logs, or support tickets.

4. Verify the connection

After installation or a configuration change, start a new session so the client reloads the MCP tools.

4.1 Check client registration

```
claude mcp list
codex mcp list
```

Expect `fortex-api` to appear as `Connected` or `enabled`. In a session, `/mcp` should show `fortex-api connected`.

4.2 Check service readiness

Begin with a read-only request:

```
Use fortex-api and check the service status.
```

Continue to account reads only after `service_status` reports `ready: true`. Do not attempt trading while the service is not ready.

4.3 Standalone self-test

You can test without a client by running the following in the directory that contains `config.json`:

```
fortex-api-mcp.exe --selftest
```

```
./fortex-api-mcp --selftest
```

A failed self-test exits with a non-zero status and prints what must be corrected.

5. Capability overview

5.1 Account and reporting reads

Scenario	Available capabilities	Example request
Account information	User profile, balance, account summary, assets and collateral	“Show my balance, equity, and asset overview.”
Current risk	Positions, working orders, current tickets	“Show my open positions and orders that are still working.”
History	Closed tickets, end-of-day positions, P&L, funding history	“Show closed EUR/USD tickets last month, by close date.”
Carry costs	Overnight financing rates	“What is the overnight financing rate for EUR/USD?”

5.2 Market data and trading views

- Trading-account information.
- Tradable symbols and their pricing, minimum size, lot size, and quantity precision.
- Open and historical trading tickets.
- A finite number of quote updates for one or more symbols.

5.3 Trading write operations

- Place an order.
- Cancel a working order.
- Modify a working order. Modification uses cancel/replace and returns a new order ID.
- Close all or part of an open ticket.
- Set or update stop-loss and take-profit protection on an open ticket.

6. Recommended read patterns

6.1 General reporting workflow

1. Call `service_status` and confirm that the connection is ready.
2. State the intended account context, symbol, date range, and status.
3. Start with a narrow request, such as one symbol or a short date range.
4. Check dates, currencies, units, and pagination in the result.
5. Request another page or expand the date range only when needed. Avoid unbounded history requests.

Use explicit dates such as “2026-07-01 through 2026-07-31” instead of relying on how different clients interpret “last month” or “today.”

6.2 Distinguish the business objects

Object	Meaning	Common use
Position	Current market exposure aggregated by symbol	Review net risk and direction
Order	An instruction that has not reached a final filled or cancelled state	Confirm an order ID before cancel or modify
Ticket	The business record for an opened trade	Close a trade or set stop-loss/take-profit
Closed ticket	A finished trade and period totals	History, analysis, and reconciliation

Do not use an order ID as a ticket number. Do not infer which individual ticket to close from a net-position summary alone.

6.3 Quote requests

For `market_quotes`, `count` defaults to 1 and must be from 1 through 1000. The `window` or timeout is the total wait for the call, not a separate wait for each quote.

Recommended pattern:

1. Specify the symbol list and required market depth.
2. Start with a small `count`, such as 10 to 20 updates.
3. Set a reasonable total wait.
4. If the result reports `complete=false` and `timed_out=true`, use the quotes already collected. Retry only when the business need justifies it, preferably with a narrower request.

The quote tool is not a permanent subscription. For continued observation, the caller should control a finite count and time window. Do not create an unbounded polling loop.

6.4 Large requests and call rate

The complete tradable-symbol list is about 90 KB; a single-symbol result is normally about 500 bytes. When answering sizing, lot conversion, or minimum-size questions, query the individual symbol first.

When a reporting tool supports pagination, use a reasonable page size and read successive pages. The README does not publish one universal requests-per-second limit, so do not invent a fixed rate. Avoid request bursts, rapid repetition of failed calls, and long-running polling. Follow the service policy for the deployment.

7. Standard trading workflow

Safe trading workflow

7.1 Before an order

1. Call `service_status` and confirm `ready: true`.
2. Query one `symbol` to confirm the instrument code, minimum size, lot size, and quantity precision.
3. Read the balance, current positions, and working orders to confirm account context and risk.
4. Restate the symbol, side, order type, size and unit, price conditions, time in force, stop-loss, and take-profit.

5. Wait for explicit agreement before calling a trading tool. Silence, ambiguity, or a past preference is not authorization for the current write.

7.2 Quantity and lots

`place_order`, `modify_order`, and partial `close_ticket` sizing uses one of these forms:

- If the user states lots, pass the same number as `lots`.
- If the user states instrument units, pass the same number as `quantity`.
- Never pass both `lots` and `quantity`.
- Never convert lots with a fixed multiplier. Lot size differs by instrument; the server reads the instrument configuration and performs the conversion.
- For a full close, `close_ticket` can omit the size. A partial close must state one sizing form.

If a size is rejected for minimum, maximum, or precision, read the stated limit, propose the nearest tradable value, and obtain confirmation again. Do not silently round or rapidly retry the same value.

7.3 Multi-tool operating patterns

New order: symbol rules → balance/positions → restate and confirm → `place_order` → re-read orders or tickets to verify.

Modify a working order: read orders → confirm order ID and current fields → restate the change → `modify_order` → retain the new order ID → re-read orders to verify.

Cancel a working order: read orders → confirm order ID, symbol, and side → `cancel_order` → re-read orders to confirm the order is gone or final.

Full or partial close: read tickets → confirm ticket number and remaining size → restate the close size → `close_ticket` → re-read tickets and positions to verify.

Stop-loss/take-profit: read the open ticket and current quote → confirm ticket number, direction, and price levels → `set_ticket_protection` → re-read the related order or ticket to verify.

7.4 Model-independent request structure

Do not depend on prompt wording that is specific to one model. Any compatible client should receive the same explicit business context:

```
Goal: read / place / modify / cancel / close / set protection
Account: the currently configured account
Object: symbol, order ID, or ticket number
Scope: dates, status, page, or quote count
Value: number and unit (lots or quantity)
Constraints: price, order type, time in force, stop-loss, take-profit, maximum wait
Authorization: reads may run; restate writes and wait for explicit confirmation
```

8. Boundaries and prohibited actions

8.1 Permission boundaries

- Only Regular User API keys are accepted. Administrator accounts cannot establish a session.
- Customer tools are limited to the current login account and do not provide runtime account switching.
- User creation or deletion, lock or unlock, API-key creation or revocation, and cash deposit or withdrawal operations are not exposed.
- `funding_history` reads history only; it does not move funds.
- The Claude Code desktop app blocks trading tools under its own client safety policy. Use Claude Code CLI or Codex when trading is required.

8.2 A model or automation must not

- Put an API key, password, or complete configuration in chat, registration commands, tool arguments, or public screenshots.
- Guess size, price, direction, order ID, or ticket number from surrounding context.
- Place, modify, cancel, close, or change protection without explicit authorization for that specific write.
- Disable TLS verification in production as a troubleshooting shortcut.
- Repeatedly retry rejected credentials; repeated bad sign-ins can lock the account.
- Poll quotes, history, or status in a high-frequency infinite loop.
- Describe a write as a preview, simulation, or guaranteed-reversible action.

9. Troubleshooting

Call `service_status` first and act on `problem` and `message`:

Problem	Meaning	Action
`configuration`	Site URL or account configuration is missing	Correct `config.json`, save it, and start a new session
`credentials`	Account, API key, or raw password was refused	Stop retrying; verify the credentials before a new session
`permission`	The account is not a Regular User	Request a Regular User API key from the broker
`connection`	The service URL could not be reached	Check URL, network, proxy, firewall, and TLS; connection failures may retry automatically

Also check:

- A new session was started after the configuration change.
- `config.json` is at the exact plugin path that was printed, or beside the manually installed executable.
- The file is UTF-8 without BOM and has valid JSONC structure.
- `logs/fortex-api-mcp.log` contains an error that support can use. Before sharing a log, review and redact account, host, and other sensitive details.
- A first launch is not still within the operating system's security-scan period.