DEVELOPER CONNECTIONS

Five ways to *connect your agent*

Use the SDK, API, MCP, CLI, or Conto Pay to connect your agent's payment tools. The owner sets the spending rules.

THE PROBLEM

An agent may connect from an application, an assistant, or the terminal. Choose what fits your setup, then test the same spending and approval rules before sending real money.

WAYS TO CONNECT

01

SDK

Request payments from your TypeScript application.

02

REST API

Send requests from any language using an API key.

03

MCP

Connect an assistant with the access and rules it needs.

04

CLI

Set up an agent and check its wallet from the terminal.

05

Conto Pay

Manage payment requests, approvals, and receipts.

DEPLOYMENT PATH

STEP 01

Sandbox

Simulate without funding. Fund testnet transfers separately.

STEP 02

Validate

Test approvals, denials, and payment records.

STEP 03

Production

Review setup before sending real money.

WAYS TO CONNECT

Supply your configured wallet and recipient details, and follow the current documentation before sending real funds.

SDK

Request a simulated payment with a sandbox agent SDK key. Save the request ID to check its status.

```
import { Conto } from '@conto_finance/sdk';
const conto = new Conto({
  apiKey: process.env.CONTO_API_KEY!,
});
const p = await conto.payments.request({
  amount: 1,
  recipientAddress:
    '0x0000000000000000000000000000000000000001',
  purpose: 'Sandbox test',
  idempotencyKey: 'sandbox-example-001',
});
console.log(p.requestId, p.status);
// Reuse the key for retries.
// Use a new key for a new payment.
```

REST

Request a simulated payment with a sandbox agent SDK key. Save the request ID. Reuse the idempotency key only for retries.

```
curl --fail-with-body \
  https://conto.finance/api/sdk/payments/request \
  -H "Authorization: Bearer $CONTO_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"amount":1,
    "recipientAddress":
      "0x0000000000000000000000000000000000000001",
    "purpose":"Sandbox test",
    "idempotencyKey":"sandbox-rest-example-001"}'
```

MCP

Add the server, then your assistant can request a payment. Conto checks it against your policy before anything is sent.

```
// claude_desktop_config.json
{ "mcpServers": { "conto": {
  "command": "npx",
  "args": ["@conto_finance/mcp-server"],
  "env": { "CONTO_API_KEY": "conto_agent_..." }
}}}
// then ask: "Pay api.exa.ai $10" -> calls pay
```

CLI

Create an anonymous sandbox and run a simulated payment. No account, signer, or funds needed.

```
# run in a new directory
mkdir conto-first-payment
cd conto-first-payment
npx -y @conto_finance/create-conto-agent --sandbox --json
node example.mjs --execute
# After a timeout, use --status with the saved request ID.
```

Conto Pay

The hosted workspace. Ask the agent to pay, in plain language. Conto decides under the same policies and approvals and keeps the same record, with nothing to deploy.

Pay `@exa-api` $10 for API usage

Held, checked against your rules, and sent from the agent's hosted wallet only after Conto clears it.

Start with simulated sandbox payments, then test a funded onchain integration before preparing for production.