

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

These snippets show connection patterns. Supply your configured wallet and recipient details, and follow the current documentation before sending real funds.

</> SDK

YOUR TYPESCRIPT APP

Request and send payments from TypeScript.

```
import { Conto } from '@conto_finance/sdk';

const conto = new Conto({ apiKey });
const p = await conto.payments.request({
  amount: 10,
  recipientAddress, // your approved recipient
  purpose: 'API usage',
});
if (p.status === 'APPROVED') {
  await conto.payments.execute(p.requestId);
}
```

🌐 REST

ANY LANGUAGE

Send payment requests from any language using an API key.

```
base=https://conto.finance/api/sdk
curl -X POST "$base/payments/request" \
  -H "Authorization: Bearer $CONT0_API_KEY" \
  -H "Content-Type: application/json" \
  --data @payment-request.json
# JSON: amount, recipientAddress, purpose
```

🔧 MCP

ASSISTANTS & TEAMS

Add the server, then your assistant can pay within policy.

```
// claude_desktop_config.json
{ "mcpServers": { "conto": {
  "command": "npx",
  "args": ["@conto_finance/mcp-server"],
  "env": { "CONT0_API_KEY": "conto_agent_..." }
}}}
// then ask: "Pay api.exa.ai $10" -> calls pay
```

> CLI

TERMINAL QUICKSTART

Set up an agent and its spending rules from your terminal.

```
# agent + wallet + policy + key, one guided run
npx @conto_finance/create-conto-agent
# verify scopes, custody, and balance
npx conto doctor
# first policy-checked payment
node example.mjs --execute
```



Conto Pay

NO CODE, HOSTED

Use Conto Pay to manage payment requests, approvals, and receipts from one account.

Pay @exa-api \$10 for API usage

Request the payment, check the rules, and send it if approved.

Each connection uses the same payment checks. Test with sandbox funds before going live.