

PrintKK MCP - Usage Guide

Version 1.0.1

This guide covers obtaining credentials, installing and configuring the MCP server, verifying the connection, finding a product, uploading artwork, creating a design, creating an unpaid test order, updating or canceling it, and paying a production order.

Security: Never commit, publish, screenshot, or paste your PrintKK API key or secret key into source code. If a secret is exposed, rotate it in the PrintKK Dashboard immediately.

1. Requirements

- PrintKK account
- PrintKK API key and secret key
- Node.js 18.14.1 or newer
- MCP-compatible client such as Codex or Claude Desktop
- JPG, JPEG, or PNG artwork (maximum 100 MB)

The PrintKK API rate limit is 60 requests per minute. API timestamps use fixed UTC-8.

Simple prompt

What do I need before using PrintKK MCP?

What do I need before using PrintKK MCP?

I'll check PrintKK's MCP documentation for the current account, authentication, and workflow prerequisites so the checklist matches the integration you're using.

Used Printkk integration

Worked for 18s

Before using PrintKK MCP, you need:

1. A PrintKK account

Create or sign in at the [PrintKK Dashboard](#).

2. API credentials

In **Dashboard** → **Setting** → **API Management**, generate an API key and secret key. Configure both in the PrintKK MCP server, preferably through environment variables or your MCP configuration rather than placing secrets in prompts or source code.

3. Required API permissions

Enable the modules you plan to use:

Product

 for searching and viewing products

Image

 for uploading artwork

Design

 for generating designs

Order

 for creating and managing orders

General

 for shared API functions

4. Valid, unexpired keys

Keys expire. They can only be extended by 90 days when no more than seven days of validity remain; expired keys cannot be renewed.

5. Artwork that meets the upload limits

Images must be JPG, JPEG, or PNG and no larger than 100 MB. Public URL uploads require an accessible HTTPS URL.

6. A funded PrintKK wallet, only when placing orders

Product browsing, uploads, and design creation do not require payment. The wallet needs sufficient balance before calling `pay_order`.

↓

The normal MCP workflow is:

2. Get API credentials

1. Open <https://dashboard.printkk.com>.
2. Go to Setting -> API Management.
3. Create or reveal an API Key and Secret Key.
4. Store both values in a password manager or secret store.
5. Enable the API module permissions required by your workflow.

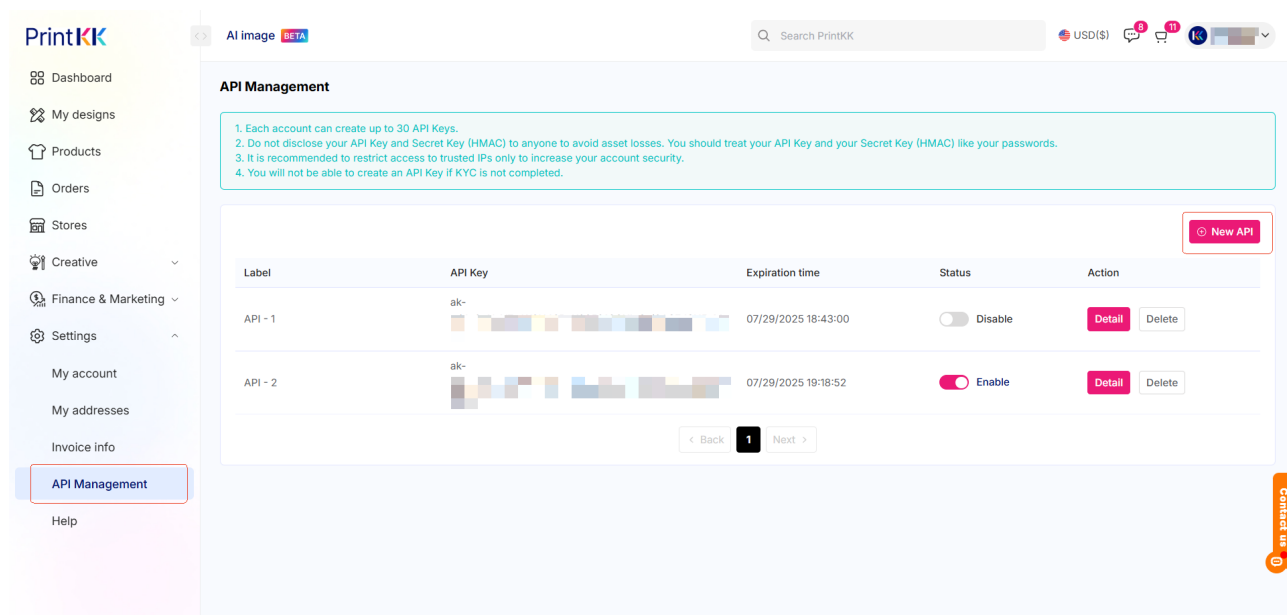
Use these environment variable names:

```
PRINTKK_API_KEY=your-api-key
PRINTKK_SECRET_KEY=your-secret-key
```

The key needs product-read permission to browse products, plus image, design, order, or payment permissions for those write operations.

Simple prompt

Show me where to get my PrintKK API key and secret key, and tell me which permissions I need for designing and ordering.



3. Configure PrintKK MCP in Codex

```
codex mcp add printkk \
--env PRINTKK_API_KEY=your-api-key \
--env PRINTKK_SECRET_KEY=your-secret-key \
-- npx -y @printkk/mcp@latest
```

For a pinned release, replace @latest with @1.0.0. Check the configuration:

```
codex mcp get printkk
codex mcp list
```

The CLI masks secret values. Restart Codex or open a new task after changing MCP configuration.

For JSON-based MCP clients:

```
{
  "mcpServers": {
    "printkk": {
      "command": "npx",
      "args": ["-y", "@printkk/mcp@latest"],
      "env": {
        "PRINTKK_API_KEY": "your-api-key",
        "PRINTKK_SECRET_KEY": "your-secret-key"
      }
    }
  }
}
```

On Windows, use npx.cmd if your client cannot find npx.

Optional variables: PRINTKK_RECV_WINDOW (milliseconds, default 5000) and PRINTKK_READ_ONLY. Set PRINTKK_READ_ONLY to the string 1 or true to expose only read/query tools. Use strings for environment values.

Simple prompt

Configure PrintKK MCP in Codex using my API key and secret key, then show me whether the server is enabled.

4. Verify the connection

Ask the MCP client:

Use PrintKK to search the first 10 template products and include categories.

A successful result proves that the server started, credentials were accepted, and the key can read products.

Use PrintKK to search the first 10 template products and include categories.

I'll query the available PrintKK catalog tools, then return the first 10 template products with their category names.

🔍 Search products

Worked for 30s

The first 10 PrintKK template products, from 744 total:

#	Product	Code	Category
1	Crossbody Cell Phone Purse	WJH72	Bags & Organizers > Satchel
2	Large Outdoor Decor Flags Double Sided (2:3)	WJMC5	Garden & Outdoor > Garden Decor
3	Oxford Cloth Mailbox Covers	WJMF8R	Garden & Outdoor > Garden Decor
4	Large Twill Cloth Backpack	WJMT8H	Bags & Organizers > Schoolbag
5	Thickened Spare Tire Cover	WJMAE	Car Accessories > Car Exterior Accessories > License Plates & Wheels
6	Women's V-Neck Short-Sleeve T-Shirt	N9Y2R	Fashion Clothing > Women's Clothing > Women's T-shirts
7	Yoga Sports Bra Top	WJHQ4	Fashion Clothing > Women's Clothing > Women's Sportswear
8	Canvas Prints Wall Art (2:1)	N9XZP	Home & Living > Wall Decor > Wall Art

Simple prompt

Test my PrintKK MCP connection by finding three random products.

5. Available tools

Tool	Purpose
search_products	Browse products and categories
get_product	Read product variants and printable areas
upload_image	Upload local JPG/PNG or public HTTPS image
list_images	Browse uploaded images
create_design	Apply images to product print areas
get_design	Read design details and variant codes
list_designs	Browse existing designs
create_order	Create an unpaid order
get_order	Read order and optional payment status
list_orders	Browse orders with filters
pay_order	Pay a pending order from the wallet
cancel_order	Cancel a pending-payment order
update_order_address	Update an unpaid order address
get_countries	Read supported shipping countries
read_api_docs	Read embedded API documentation

With PRINTKK_READ_ONLY=1, write tools are hidden.

6. Design workflow

search_products -> get_product -> upload_image -> create_design -> get_design

6.1 Find and inspect a product

Search PrintKK for women's T-shirts. Return product codes, names, sizes, prices, and printable areas.

Save the returned productCode. Then ask:

Get the full PrintKK product detail for product code PRODUCT_CODE.

Record isWhole (1 = full-surface, 0 = placement), valid printAreaCode values, and available sizes/colors. Always copy print area codes from get_product.

Simple prompt

Find any suitable women's clothing product for this image, then show its sizes, price, isWhole value, and printable areas.

6.2 Upload artwork

Local file:

Upload this local image to PrintKK: /absolute/path/to/artwork.png

Public image:

Upload this public HTTPS image to PrintKK: https://example.com/artwork.png

The server returns an `imageId` required by `create_design`. Provide exactly one of `filePath` or `fileUrl`. Local paths must be absolute; public URLs must use HTTPS and cannot point to private/internal hosts.

Simple prompt

Upload this image to PrintKK: /absolute/path/to/my-image.png

 <https://example.com/20260731092001.jpg>
Upload this public HTTPS image to PrintKK

I'll upload that image to the root of your PrintKK media library, using the timestamped filename from the URL.

🔄 Upload image

Worked for 17s

Uploaded successfully to the PrintKK media library.

- Image ID: 2082999905907093506
- Filename: 20260731092001.jpg
-  [View uploaded image](#)



6.3 Create the design

Create a PrintKK design using:

- `productCode`: PRODUCT_CODE
- `isWhole`: IS_WHOLE
- `printAreaCode`: PRINT_AREA_CODE
- `imageId`: IMAGE_ID
- `imageFillingMode`: contain

Apply the artwork to the front print area and tag it as my-project.

Create a PrintKK design using:

- productCode: WJM72
- imageId: 2082999905907093506
- imageFillingMode: contain

Apply the artwork to the front print area and tag it as my-project.

I'll verify WJM72's current print-area code and print type first, then create the design with the artwork contained inside the front-facing printable surface and add the `my-project` tag.

% Get product

WJM72 exposes one full-surface print area (`Surface` , code `181267365233338626`), so that is the product's applicable front-facing artwork area. I'm creating the design with that mapping now.

% Create design

PrintKK accepted the design request and is rendering it asynchronously. I'm waiting for the completed design and its orderable specification code.

Worked for 43s

Design created successfully.

- Design: `Crossbody Cell Phone Purse 20260731092001`
- Design code: `nuoUYLLRuyC`
- Specification code: `nuoUYLLRuyCnr`
- Print area: `Surface` (the product's single full-surface/front-facing area)
- Filling mode: `contain`
- Tag: `my-project`
- Status: `mutable`



Filling modes: `contain` preserves the entire image but may leave blank space; `cover` fills the area but may crop edges; `fill` stretches to the exact area and may distort the image.

`create_design` is asynchronous. It waits for the task and returns `designCode`, preview URLs, and `designSpecifications`. Each `designSpecificationCode` identifies a size/color variant and is required for ordering.

Simple prompt

Use this image to create a design on the selected product. Put it on the front, preserve the full image, and show me the preview and all size specification codes.

7. Create an unpaid test order

Creating an order creates a real PrintKK record even when unpaid. Use a clearly fake address only for testing and mark it non-fulfillable.

```
Create one unpaid PrintKK test order using:
- designSpecificationCode: DESIGN_SPECIFICATION_CODE
- quantity: 1
- address: Test Recipient, 123 Test Street, San Francisco, CA 94105, US
- shipping preference: economy, standard, express
- note: UNPAID TEST ORDER - DO NOT FULFILL
Do not call pay_order.
```

The `shippingType` array must contain `standard`, `economy`, and `express` exactly once. The first value is preferred. Verify with `get_order` that `orderStatus` is `pendingPayment`, `payTime` is empty, and `product`, `variant`, `quantity`, `address`, `shipping cost`, and `total` are correct.

Simple prompt

Create one unpaid test order for this design using a clearly fake address. Use size M, quantity 1, and do not pay.

8. Update or cancel an unpaid order

While the order is pendingPayment, use `update_order_address` to replace a test address with a real address, or `cancel_order` to cancel it. Once paid, the order is locked for API modification and cancellation.

Simple prompt

Cancel unpaid test order ORDER_ID.

9. Pay a production order

`pay_order` spends PrintKK wallet funds and cannot be undone through the MCP. Before calling it, verify order ID, pendingPayment status, real address, size/color, quantity, final total, and wallet balance.

Use explicit confirmation:

I confirm payment for PrintKK order ORDER_ID. The verified total is AMOUNT and the shipping address is ADDRESS.

After payment, call `get_order` with `includePayment=true`. Successful payment normally changes the order to `waitingForFulfillment` and sets `payTime`.

Simple prompt

Review order ORDER_ID one more time and pay it only if the address and total are correct.

10. Order lifecycle

```
pendingPayment -> waitingForFulfillment -> beingFulfilled  
-> partiallyShipped -> shipped -> completed
```

Orders can also transition to canceled and then closed.

11. Troubleshooting

- Missing credentials: add `PRINTKK_API_KEY` and `PRINTKK_SECRET_KEY` to the MCP env section and restart.
- Permission denied: enable the relevant API module permissions in Setting -> API Management.
- Wrong design specification: call `get_design` and use the exact `designSpecificationCode`; never use `productCode` or `productSpecificationId` in `create_order`.
- Invalid print area: call `get_product` and copy a valid `printAreaCode` with the matching `isWhole`.
- Image rejected: check JPG/JPEG/PNG format, file size under 100 MB, absolute path, and public HTTPS URL requirements.
- Design still processing: do not submit repeatedly; query later with `get_design`.
- Payment lookup HTTP 400: an unpaid order may have no payment record; check `orderStatus` and `payTime` and do not treat the failed lookup as payment success.

12. Operational rules

1. Use read-only mode while exploring or integrating.
2. Use real addresses only for production shipments.
3. Mark test orders as unpaid and non-fulfillable.
4. Review the design preview before creating an order.
5. Review the address and total immediately before payment.
6. Never call `pay_order` as part of an automated design test.
7. Rotate keys if they appear in logs, screenshots, commits, or chat.

13. Quick reference

Production: Get API key + secret -> install `@printkk/mcp` -> configure MCP -> `search_products` -> `get_product` -> `upload_image` -> `create_design` -> `get_design` -> `create_order` -> `get_order` (verify) -> `pay_order` (explicit confirmation) -> `get_order` (verify fulfillment).

Safe test: Configure read-only MCP -> create tagged design -> create marked unpaid test order -> verify `pendingPayment` -> do not call `pay_order` -> cancel when finished.

Official links

- PrintKK Dashboard: <https://dashboard.printkk.com>
- PrintKK API documentation: <https://api-docs.printkk.com>
- PrintKK website: <https://www.printkk.com>
- npm package: <https://www.npmjs.com/package/@printkk/mcp>