Using Agent Skills with the API

Learn how to use Agent Skills to extend Claude's capabilities through the API.

Copy page

Agent Skills extend Claude's capabilities through organized folders of instructions, scripts, and resources. This guide shows you how to use both pre-built and custom Skills with the Claude API.

For complete API reference including request/response schemas and all parameters, see:

- Skill Management API Reference - CRUD operations for Skills
- Skill Versions API Reference - Version management

Quick Links

Get started with Agent Skills
Create your first Skill

Create Custom Skills
Best practices for authoring Skills

Overview

Ask Docs

Skills integrate with the Messages API through the code execution tool. Whether using pre-built Skills managed by Anthropic or custom Skills you've uploaded, the integration shape is identical—both require code execution and use the same `container` structure.

Using Skills

Skills integrate identically in the Messages API regardless of source. You specify Skills in the `container` parameter with a `skill_id`, `type`, and optional `version`, and they execute in the code execution environment.

You can use Skills from two sources:

Aspect	Anthropic Skills	Custom Skills
Type value	`anthropic`	`custom`
Skill IDs	Short names: `pptx`, `xlsx`, `docx`, `pdf`	Generated: `skill_01AbCdEfGhIjKlMnOpQrStUv`
Version format	Date-based: `20251013` or `latest`	Epoch timestamp: `1759178010641129` or `latest`
Management	Pre-built and maintained by Anthropic	Upload and manage via Skills API
Availability	Available to all users	Private to your workspace

Both skill sources are returned by the List Skills endpoint (use the `source` parameter to filter). The integration shape and execution environment are identical—the only difference is where the Skills come from and how they're managed.

Prerequisites

To use Skills, you need:

1. **Anthropic API key** from the Console
2. **Beta headers**:
   - `code-execution-2025-08-25` - Enables code execution (required for Skills)
   - `skills-2025-10-02` - Enables Skills API
   - `files-api-2025-04-14` - For uploading/downloading files to/from container

Ask Docs

Using Skills in Messages

Container Parameter

Skills are specified using the `container` parameter in the Messages API. You can include up to 8 Skills per request.

The structure is identical for both Anthropic and custom Skills—specify the required `type` and `skill_id`, and optionally include `version` to pin to a specific version:

Python

```python
import anthropic

client = anthropic.Anthropic()

response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {
                "type": "anthropic",
                "skill_id": "pptx",
                "version": "latest"
            }
        ]
    },
    messages=[{
        "role": "user",
        "content": "Create a presentation about renewable energy"
    }],
    tools=[{
        "type": "code_execution_20250825",
        "name": "code_execution"
    }]
)
```

Ask Docs

When Skills create documents (Excel, PowerPoint, PDF, Word), they return `file_id` attributes in the response. You must use the Files API to download these files.

How it works:

1. Skills create files during code execution
2. Response includes `file_id` for each created file
3. Use Files API to download the actual file content
4. Save locally or process as needed

Example: Creating and downloading an Excel file

Python

Ask Docs

Claude Docs

```
client = anthropic.Anthropic()

# Step 1: Use a Skill to create a file
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"}
        ]
    },
    messages=[{
        "role": "user",
        "content": "Create an Excel file with a simple budget spreadsheet"
    }],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)

# Step 2: Extract file IDs from the response
def extract_file_ids(response):
    file_ids = []
    for item in response.content:
        if item.type == 'bash_code_execution_tool_result':
            content_item = item.content
            if content_item.type == 'bash_code_execution_result':
                for file in content_item.content:
                    if hasattr(file, 'file_id'):
                        file_ids.append(file.file_id)
    return file_ids

# Step 3: Download the file using Files API
for file_id in extract_file_ids(response):
    file_metadata = client.beta.files.retrieve_metadata(
        file_id=file_id,
        betas=["files-api-2025-04-14"]
    )
    file_content = client.beta.files.download(
        file_id=file_id,
        betas=["files-api-2025-04-14"]
    )

    # Step 4: Save to disk
    file_content.write_to_file(file_metadata.filename)
    print(f"Downloaded: {file_metadata.filename}")
```

Ask Docs

Python

```python
# Get file metadata
file_info = client.beta.files.retrieve_metadata(
    file_id=file_id,
    betas=["files-api-2025-04-14"]
)
print(f"Filename: {file_info.filename}, Size: {file_info.size_bytes} bytes")

# List all files
files = client.beta.files.list(betas=["files-api-2025-04-14"])
for file in files.data:
    print(f"{file.filename} - {file.created_at}")

# Delete a file
client.beta.files.delete(
    file_id=file_id,
    betas=["files-api-2025-04-14"]
)
```

For complete details on the Files API, see the Files API documentation.

Multi-Turn Conversations

Reuse the same container across multiple messages by specifying the container ID:

Python

Ask Docs

```python
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"}
        ]
    },
    messages=[{"role": "user", "content": "Analyze this sales data"}],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)

# Continue conversation with same container
messages = [
    {"role": "user", "content": "Analyze this sales data"},
    {"role": "assistant", "content": response1.content},
    {"role": "user", "content": "What was the total revenue?"}
]

response2 = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "id": response1.container.id,  # Reuse container
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"}
        ]
    },
    messages=messages,
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)
```

Long-Running Operations

Skills may perform operations that require multiple turns. Handle `pause_turn` stop reasons:

Python

Ask Docs

```python
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {"type": "custom", "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv", "
        ]
    },
    messages=messages,
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)

# Handle pause_turn for long operations
for i in range(max_retries):
    if response.stop_reason != "pause_turn":
        break

    messages.append({"role": "assistant", "content": response.content})
    response = client.beta.messages.create(
        model="claude-sonnet-4-5-20250929",
        max_tokens=4096,
        betas=["code-execution-2025-08-25", "skills-2025-10-02"],
        container={
            "id": response.container.id,
            "skills": [
                {"type": "custom", "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv
            ]
        },
        messages=messages,
        tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
    )
```

The response may include a `pause_turn` stop reason, which indicates that the API paused a long-running Skill operation. You can provide the response back as-is in a subsequent request to let Claude continue its turn, or modify the content if you wish to interrupt the conversation and provide additional guidance.

Using Multiple Skills

Combine multiple Skills in a single request to handle complex workflows:

Ask Docs

```
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {
                "type": "anthropic",
                "skill_id": "xlsx",
                "version": "latest"
            },
            {
                "type": "anthropic",
                "skill_id": "pptx",
                "version": "latest"
            },
            {
                "type": "custom",
                "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv",
                "version": "latest"
            }
        ]
    },
    messages=[{
        "role": "user",
        "content": "Analyze sales data and create a presentation"
    }],
    tools=[{
        "type": "code_execution_20250825",
        "name": "code_execution"
    }]
)
```

Managing Custom Skills

Creating a Skill

Upload your custom Skill to make it available in your workspace. You can upload using either a directory path or individual file objects.

Ask Docs

```python
import anthropic

client = anthropic.Anthropic()

# Option 1: Using files_from_dir helper (Python only, recommended)
from anthropic.lib import files_from_dir

skill = client.beta.skills.create(
    display_title="Financial Analysis",
    files=files_from_dir("/path/to/financial_analysis_skill"),
    betas=["skills-2025-10-02"]
)

# Option 2: Using a zip file
skill = client.beta.skills.create(
    display_title="Financial Analysis",
    files=[("skill.zip", open("financial_analysis_skill.zip", "rb"))],
    betas=["skills-2025-10-02"]
)

# Option 3: Using file tuples (filename, file_content, mime_type)
skill = client.beta.skills.create(
    display_title="Financial Analysis",
    files=[
        ("financial_skill/SKILL.md", open("financial_skill/SKILL.md", "rb"), "
        ("financial_skill/analyze.py", open("financial_skill/analyze.py", "rb"
    ],
    betas=["skills-2025-10-02"]
)

print(f"Created skill: {skill.id}")
print(f"Latest version: {skill.latest_version}")
```

Requirements:

- Must include a SKILL.md file at the top level
- All files must specify a common root directory in their paths
- Total upload size must be under 8MB
- YAML frontmatter requirements:
  - `name` : Maximum 64 characters, lowercase letters/numbers/hyphens only, no XML tags, no reserved words ("anthropic", "claude")
  - `description` : Maximum 1024 characters, non-empty, no XML tags

Ask Docs

Claude Docs

Listing Skills

Retrieve all Skills available to your workspace, including both Anthropic pre-built Skills and your custom Skills. Use the `source` parameter to filter by skill type:

Python

```python
# List all Skills
skills = client.beta.skills.list(
    betas=["skills-2025-10-02"]
)

for skill in skills.data:
    print(f"{skill.id}: {skill.display_title} (source: {skill.source})")

# List only custom Skills
custom_skills = client.beta.skills.list(
    source="custom",
    betas=["skills-2025-10-02"]
)
```

See the List Skills API reference for pagination and filtering options.

Retrieving a Skill

Get details about a specific Skill:

Python

```python
skill = client.beta.skills.retrieve(
    skill_id="skill_01AbCdEfGhIjKlMnOpQrStUv",
    betas=["skills-2025-10-02"]
)

print(f"Skill: {skill.display_title}")
print(f"Latest version: {skill.latest_version}")
print(f"Created: {skill.created_at}")
```

Deleting a Skill

Ask Docs

To delete a Skill, you must first delete all its versions:

Claude Docs

```python
# Step 1: Delete all versions
versions = client.beta.skills.versions.list(
    skill_id="skill_01AbCdEfGhIjKlMnOpQrStUv",
    betas=["skills-2025-10-02"]
)

for version in versions.data:
    client.beta.skills.versions.delete(
        skill_id="skill_01AbCdEfGhIjKlMnOpQrStUv",
        version=version.version,
        betas=["skills-2025-10-02"]
    )

# Step 2: Delete the Skill
client.beta.skills.delete(
    skill_id="skill_01AbCdEfGhIjKlMnOpQrStUv",
    betas=["skills-2025-10-02"]
)
```

Attempting to delete a Skill with existing versions will return a 400 error.

Versioning

Skills support versioning to manage updates safely:

Anthropic-Managed Skills:

- Versions use date format: `20251013`
- New versions released as updates are made
- Specify exact versions for stability

Custom Skills:

- Auto-generated epoch timestamps: `1759178010641129`
- Use `"latest"` to always get the most recent version
- Create new versions when updating Skill files

Python

Ask Docs

```python
new_version = client.beta.skills.versions.create(
    skill_id="skill_01AbCdEfGhIjKlMnOpQrStUv",
    files=files_from_dir("/path/to/updated_skill"),
    betas=["skills-2025-10-02"]
)

# Use specific version
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [{
            "type": "custom",
            "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv",
            "version": new_version.version
        }]
    },
    messages=[{"role": "user", "content": "Use updated Skill"}],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)

# Use latest version
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [{
            "type": "custom",
            "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv",
            "version": "latest"
        }]
    },
    messages=[{"role": "user", "content": "Use latest Skill version"}],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)
```

See the Create Skill Version API reference for complete details.

Ask Docs

When you specify Skills in a container:

1. **Metadata Discovery**: Claude sees metadata for each Skill (name, description) in the system prompt
2. **File Loading**: Skill files are copied into the container at `/skills/{directory}/`
3. **Automatic Use**: Claude automatically loads and uses Skills when relevant to your request
4. **Composition**: Multiple Skills compose together for complex workflows

The progressive disclosure architecture ensures efficient context usage—Claude only loads full Skill instructions when needed.

Use Cases

Organizational Skills

Brand & Communications

- Apply company-specific formatting (colors, fonts, layouts) to documents
- Generate communications following organizational templates
- Ensure consistent brand guidelines across all outputs

Project Management

- Structure notes with company-specific formats (OKRs, decision logs)
- Generate tasks following team conventions
- Create standardized meeting recaps and status updates

Business Operations

- Create company-standard reports, proposals, and analyses
- Execute company-specific analytical procedures
- Generate financial models following organizational templates

Personal Skills

Ask Docs

- Specialized formatting and styling
- Domain-specific content generation

Data Analysis

- Custom data processing pipelines
- Specialized visualization templates
- Industry-specific analytical methods

Development & Automation

- Code generation templates
- Testing frameworks
- Deployment workflows

Example: Financial Modeling

Combine Excel and custom DCF analysis Skills:

Python

Ask Docs

```
dcf_skill = client.beta.skills.create(
    display_title="DCF Analysis",
    files=files_from_dir("/path/to/dcf_skill"),
    betas=["skills-2025-10-02"]
)

# Use with Excel to create financial model
response = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02"],
    container={
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"},
            {"type": "custom", "skill_id": dcf_skill.id, "version": "latest"}
        ]
    },
    messages=[{
        "role": "user",
        "content": "Build a DCF valuation model for a SaaS company with the at
    }],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)
```

Limits and Constraints

Request Limits

- **Maximum Skills per request**: 8
- **Maximum Skill upload size**: 8MB (all files combined)
- **YAML frontmatter requirements**:
  - `name` : Maximum 64 characters, lowercase letters/numbers/hyphens only, no XML tags, no reserved words
  - `description` : Maximum 1024 characters, non-empty, no XML tags

Ask Docs

Environment Constraints

- **No runtime package installation** - Only pre-installed packages available
- **Isolated environment** - Each request gets a fresh container

See the code execution tool documentation for available packages.

Best Practices

When to Use Multiple Skills

Combine Skills when tasks involve multiple document types or domains:

Good use cases:

- Data analysis (Excel) + presentation creation (PowerPoint)
- Report generation (Word) + export to PDF
- Custom domain logic + document generation

Avoid:

- Including unused Skills (impacts performance)

Version Management Strategy

For production:

```
# Pin to specific versions for stability
container={
    "skills": [{
        "type": "custom",
        "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv",
        "version": "1759178010641129"  # Specific version
    }]
}
```

For development:

Ask Docs

```
    "skills": [{
        "type": "custom",
        "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv",
        "version": "latest"  # Always get newest
    }]
}
```

Prompt Caching Considerations

When using prompt caching, note that changing the Skills list in your container will break the cache:

Python

```
# First request creates cache
response1 = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02", "prompt-caching-2
    container={
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"}
        ]
    },
    messages=[{"role": "user", "content": "Analyze sales data"}],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)

# Adding/removing Skills breaks cache
response2 = client.beta.messages.create(
    model="claude-sonnet-4-5-20250929",
    max_tokens=4096,
    betas=["code-execution-2025-08-25", "skills-2025-10-02", "prompt-caching-2
    container={
        "skills": [
            {"type": "anthropic", "skill_id": "xlsx", "version": "latest"},
            {"type": "anthropic", "skill_id": "pptx", "version": "latest"}  #
        ]
    },
    messages=[{"role": "user", "content": "Create a presentation"}],
    tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
)
```

Ask Docs

Claude Docs

Error Handling

Handle Skill-related errors gracefully:

Python

```python
try:
    response = client.beta.messages.create(
        model="claude-sonnet-4-5-20250929",
        max_tokens=4096,
        betas=["code-execution-2025-08-25", "skills-2025-10-02"],
        container={
            "skills": [
                {"type": "custom", "skill_id": "skill_01AbCdEfGhIjKlMnOpQrStUv
            ]
        },
        messages=[{"role": "user", "content": "Process data"}],
        tools=[{"type": "code_execution_20250825", "name": "code_execution"}]
    )
except anthropic.BadRequestError as e:
    if "skill" in str(e):
        print(f"Skill error: {e}")
        # Handle skill-specific errors
    else:
        raise
```

Next Steps

API Reference

Complete API reference with all endpoints

Ask Docs

Authoring Guide

Claude Docs

>_

Code Execution Tool

Learn about the code execution environment

Claude Docs

Solutions

AI agents

Code modernization

Coding

Customer support

Education

Financial services

Government

Life sciences

Partners

Amazon Bedrock

Google Cloud's Vertex AI

Learn

Blog

Catalog

Courses

Use cases

Connectors

Customer stories

Engineering at Anthropic

Events

Powered by Claude

Service partners

Startups program

Company

Anthropic

Careers

Economic Futures

Research

News

Responsible Scaling Policy

Security and compliance

Transparency

Ask Docs

Claude Docs

Status

Support

Discord

Terms and policies

Privacy policy

Responsible disclosure policy

Terms of service: Commercial

Terms of service: Consumer

Usage policy

Ask Docs