Best practices for skill creators

This is how to write clearly defined skills that are appropriate for the task!

This is how to write clearly defined skills that are appropriate for the task!

Start with practical expertise.

A common mistake in skill creation is asking LLMs to generate skills without providing specific domain context—relying solely on the LLM's general training knowledge. The result is vague, generic processes ('handle errors appropriately', 'follow best practices for validation') instead of specific API patterns, exceptions, and project conventions that make skills valuable.

Effective skills are built upon practical expertise. The key is to provide specific field context into the skill development process.

Extracted from a real-world task.

Complete a practical task within a conversation with an agent, providing context, input, and preferences throughout the process. Then, extract a reusable template to create a skill. Pay attention to:

1. **The steps that worked** - the sequence of actions that led to success.
2. **The edits you made** – the places where you instructed the agent's approach (e.g., 'use library X instead of Y', 'check exception Z')
3. **Input/Output Format** - What does input and output data look like?
4. **The context you provided** —specific project events, conventions, or constraints that the agent is unaware of.

Compiled from existing project documents.

Once you have a body of existing knowledge, you can feed it into an LLM and ask it to synthesize a skill. A data path skill synthesized from your team's actual incident reports and operating manuals will perform better than one synthesized from a generic article on 'data engineering best practices,' because it captures your schema, failure modes, and recovery processes. The key is project-specific documentation, not generic references. Good sources include:

1. Internal documents, operating manuals, and style guides.
2. API specifications, schema, and configuration files.

3. Provide feedback on the code and troubleshooting tools (record common concerns and expectations of the reviewers).
4. Version control history, especially bug fixes and patches (reveals patterns through what actually changed).
5. Real-world error scenarios and how to resolve them.

Refine through practical implementation.

The first draft of a skill often needs refinement. Run the skill with real-world tasks, then feed the results—everything, not just the errors—back into the creation process. Ask yourself: what caused the false positives? What was missed? What can be eliminated?

Even a single trial run followed by revisions can significantly improve quality, and complex fields often benefit from multiple such runs.

? Read the agent's execution trail, not just the final output. If the agent wastes time on inefficient steps, common causes include overly vague instructions (the agent tries several approaches before finding one that works), instructions that don't apply to the current task (the agent still follows them), or too many options provided without a clear default option.

To adopt a more structured approach to the iterative process, including test cases, validation, and scoring.

Use context wisely.

Once a skill is activated, its entire SKILL.md file is loaded into the agent's context window along with conversation history, system context, and other active skills. Every word in your skill competes for the agent's attention with everything else in that window.

Add what the agent is missing, ignore what is already known.

Focus on what the agent wouldn't know without your skills: project-specific conventions, industry-specific processes, unclear exceptions, and specific tools or APIs to use. You don't need to explain what PDFs are , how HTTP works, or what database migration entails.

```
 ## Trích xu?t v?n b?n t? PDF File PDF (Portable Document Format) là m?t ??
nh d?ng file ph? bi?n ch?a v?n b?n, hình ?nh và các n?i dung khác. ??
 trích xu?t v?n b?n t? PDF, b?n c?n s? d?ng th? vi?n. pdfplumber ???c khuy?
n ngh? vì nó x? lý t?t h?u h?t các tr??ng h?p.  ## Trích xu?t v?n b?n t?
 PDF S? d?ng pdfplumber ?? trích xu?t v?n b?n. ??i v?i tài li?u ???
c scan, hãy s? d?ng pdf2image v?
i pytesseract. ```python import pdfplumber with pdfplumber.open("file.pdf") as p
```

Ask yourself about each item: 'Would the agent make a mistake doing this without instructions?' If the answer is no, eliminate it. If you're unsure, double-check. And if the agent handled the entire task well without that skill, then that skill might not be valuable.

Design coherent units

Deciding what a skill should include is similar to deciding what a function should do: You want it to encompass a coherent unit of work that integrates well with other skills. Skills that are too narrow force multiple skills to be loaded for a single task, leading to wasted resources and instruction conflicts. Skills that are too broad become difficult to trigger correctly. A skill for querying a database and formatting results might be a coherent unit, while a skill that also includes database administration might be trying to do too much.

Aiming for a moderate level of detail.

Overly comprehensive skills can do more harm than good – the agent will struggle to extract relevant information and may pursue inefficient paths due to instructions that don't apply to the current task. Concise, step-by-step instructions with practical examples are often more effective than overly detailed documentation. When you find yourself dealing with numerous exceptions, consider whether most of those cases could be handled better by the agent's judgment.

The structure of major skills is revealed gradually.

The recommended specifications suggest keeping SKILL.md under 500 lines and 5,000 tokens – only the core instructions the agent needs for each run. When a skill truly requires more content, move detailed references to separate files in the references/ directory or similar folders.

It's important to tell the agent when to load each file. 'Read references/api-errors.md if the API returns a status code other than 200' is more useful than the general statement 'see references/ for details'. This allows the agent to load the context on demand rather than right from the start, which is how incremental disclosure is designed to work.

Adjusting control

Not every part of a skill needs the same level of detail. Adjust the specificity of the instructions to match the sensitivity of the task.

The right balance between specificity and vulnerability.

Allowing agents freedom of action when multiple valid methods and tasks permit variation is important. For flexible instructions, explaining the reasoning can be more effective than rigid directives – an agent that understands the purpose behind the instruction will make better context-dependent decisions. Code review skills can describe what to look for without specifying precise steps:

```
## Quy trình ki?m tra code 1. Ki?m tra t?t c? các truy v?n c? s? d? li?
u xem có l?i t?n công SQL injection hay không (s? d?ng truy v?n tham s?
 hóa) 2. Xác minh ki?m tra xác th?c trên m?i endpoint 3. Tìm ki?m các ?i?
u ki?n tranh ch?p trong nh?ng ???ng d?n code ??ng th?i 4. Xác nh?
n thông báo l?i không làm l? thông tin n?i b?
```

Provide specific guidance when activities are susceptible to disruption, consistency is important, or a particular sequence needs to be followed:

```
## Di chuy?n c? s? d? li?u Ch?y chính xác chu?i l?
nh sau: ```bash python scripts/migrate.py --verify --backup ``` Không ???
c s?a ??i l?nh ho?c thêm các flag b? sung.
```

Most skills require a combination of many factors. Adjust each part independently.

Provides default settings, not menus.

When multiple tools or methods might be effective, choose a default setting and briefly mention the alternatives instead of presenting them as equally good options.

```
 B?n có th? s? d?ng pypdf, pdfplumber, PyMuPDF ho?c pdf2image.  S? d?
ng pdfplumber ?? trích xu?t v?n b?n: ```python import pdfplumber ``` ??i v?
i các file PDF ???c scan c?n nh?n d?ng ký t? quang h?c (OCR), hãy s? d?
ng pdf2image v?i pytesseract.
```

Prioritize process over claims.

A skill should teach learners how to approach a group of problems, not how to create a product for a specific case. Compare:

```
 K?t n?i b?ng `orders` v?i b?ng `customers` d?a trên `customer_id`, l?
c theo ?i?u ki?n `region = 'EMEA'`, và tính t?ng c?t `amount`.  1. ??c l??
c ?? t? `references/schema.yaml` ?? tìm các b?ng liên quan 2. K?t n?i các b
?ng b?ng cách s? d?ng quy ??c foreign key `_id` 3. Áp d?ng b?t k? b? l?
c nào t? yêu c?u c?a ng??i dùng d??i d?ng m?nh ?? WHERE 4. T?ng h?p các c?
t s? khi c?n và ??nh d?ng thành b?ng markdown
```

This doesn't mean skills can't include specific details—output formatting templates, constraints like 'never output personally identifiable information,' and tool-specific instructions are all valuable. The point is that the approach should be general even when individual details are very specific.

Effective guidance templates

These are reusable techniques for structuring skill content. You don't need every skill – use only those that are relevant to your tasks.

Important notes

The most valuable content in many skills is the list of unforeseen problems—specific environmental facts that contradict reasonable assumptions. This isn't general advice ('handle errors appropriately') but rather specific remedies for mistakes the agent would make if not instructed otherwise:

```
## Nh?ng ?i?m c?n l?u ý - B?ng `users` s? d?ng xóa m?m. Các truy v?n ph?
i bao g?m `WHERE deleted_at IS NULL` n?u không k?t qu? s? bao g?
m các tài kho?n ?ã b? vô hi?u hóa. - ID ng??i dùng là `user_id` trong c? s?
 d? li?u, `uid` trong d?ch v? xác th?
c, và `accountId` trong API thanh toán. C? ba ??u tham chi?u ??n cùng m?
t giá tr?. - Endpoint `/health` tr? v? mã tr?ng thái 200 mi?n là máy ch?
```

```
 web ?ang ch?y, ngay c? khi k?t n?i c? s? d? li?u b? ng?t. S? d?
ng `/ready` ?? ki?m tra ??y ?? tình tr?ng ho?t ??ng c?a d?ch v?.
```

Store common errors (gotchas) in the SKILL.md file so the agent can read them before encountering a situation. A separate reference file is also useful if you tell the agent when to load it, but for unclear issues, the agent might not recognize the trigger.

? When the agent makes a mistake that you need to fix, add a bug fix to the gotchas. This is one of the most direct ways to improve skills through iterative practice.

Output format template

When you need the agent to produce output in a specific format, provide a template. This is more reliable than describing the format in prose, as agents match patterns well to specific structures. Short templates can be stored directly in SKILL.md; for longer templates or templates only needed in certain cases, store them in the assets/ directory and reference them from SKILL.md so they are only loaded when needed.

```
## C?u trúc báo cáo S? d?ng template này, ?i?u ch?nh các ph?n khi c?n thi?
t cho phân tích c? th?: ```markdown # [Tiêu ?? phân tích] ## Tóm t?t [T?
ng quan m?t ?o?n v? các phát hi?n chính] ## Các phát hi?n chính - Phát hi?
n 1 kèm d? li?u h? tr? - Phát hi?n 2 kèm d? li?u h? tr? ## Ki?n ngh? 1. Ki?
n ngh? c? th? có th? th?c hi?n 2. Ki?n ngh? c? th? có th? th?c hi?n ```
```

Checklist for a multi-step workflow

A clear checklist helps the implementer track progress and avoid missing steps, especially those involving dependencies or validation checks.

```
## Quy trình x? lý bi?u m?u Ti?n ??: - [ ] B??c 1: Phân tích bi?u m?u (ch?
y `scripts/analyze_form.py`) - [ ] B??c 2: T?o ánh x? tr??ng (ch?nh s?
a `fields.json`) - [ ] B??c 3: Xác th?c ánh x? (ch?
y `scripts/validate_fields.py`) - [ ] B??c 4: ?i?n bi?u m?u (ch?
y `scripts/fill_form.py`) - [ ] B??c 5: Ki?m tra k?t qu? (ch?
y `scripts/verify_output.py`)
```

Validation loop

Instruct the agent to self-validate its job before proceeding. The model is: Perform the job, run the validator (a script, reference checklist, or self-check), fix any issues, and repeat until the validation process is successful.

```
## Quy trình ch?nh s?a 1. Th?c hi?n ch?nh s?a 2. Ch?y ki?m tra h?p l?
: `python scripts/validate.py output/` 3. N?u ki?m tra h?p l? th?t b?
i: - Xem l?i thông báo l?i - Kh?c ph?c s? c? - Ch?y ki?m tra h?p l? l?
i 4. Ch? ti?p t?c khi ki?m tra h?p l? thành công
```

References can also serve as a 'verification tool' – guiding the agent to check their work against the references before completion.

Plan - Verify - Execute

For batch or destructive operations, have the agent create an intermediate plan in a structured format, verify it against the correct information source, and only then execute it.

```
## ?i?n bi?u m?u PDF 1. Trích xu?t các tr??ng bi?u m?
u: `python scripts/analyze_form.py input.pdf` ? `form_fields.json` (li?
t kê tên, lo?i và tr?ng thái b?t bu?c c?a t?ng tr??ng) 2. T?
o `field_values.json` ánh x? t?ng tên tr??ng v?i giá tr? t??ng ?
ng 3. Xác th?
c: `python scripts/validate_fields.py form_fields.json field_values.json` (ki
?m tra xem m?i tên tr??ng có t?n t?i trong bi?u m?u, lo?i t??
ng thích và không thi?u các tr??ng b?t bu?c) 4. N?u xác th?c th?t b?
i, hãy s?a ??i `field_values.json` và xác th?c l?i 5. ?i?n bi?u m?
u: `python scripts/fill_form.py input.pdf field_values.json output.pdf`
```

The crucial component is step 3: A validation script checks the plan (field_values.json) against the correct data source (form_fields.json). Errors like 'Field 'signature_date' not found - available fields: customer_name, order_total, signature_date_signed' provide the agent with enough information to self-correct.

Package reusable scripts.

When iterating through the development of a skill, compare the agent's execution trace across test cases. If you notice the agent automatically regenerating the same logic each time—building graphs, parsing a specific format, validating output—that's a sign that you need to write a script that has been tested once and package it in the scripts/ directory.