Skill Template Reference

Canonical template for creating a FrameworX AI Skill page. Duplicate this page and replace placeholder content with your skill implementation.

Skill Frontmatter (copy this block to the top of your .md file)

```
---
title: "Skill Template — Replace This Title"
tags: [replace, with, relevant, keywords]
description: "One-line summary explaining what this skill teaches Claude to do"
version: "1.0"
author: "Tatsoft"
---
```

Canonical template for creating FrameworX AI Skills — markdown playbooks that teach the AI assistant how to build specific solutions. Duplicate this page to create your own skill.

⚠ **Publishing checklist:** When publishing a skill page to Confluence, you must add the `skill` label to the page. Without this label, `search_docs(query, labels='skill')` will not discover your skill. Add additional labels for relevant modules (e.g. `alarm`, `script`, `ml`).

To create a new skill: Click **...** **Copy** on this page, then replace all placeholder content with your implementation.

What This Skill Does

Brief one-paragraph summary of what this skill accomplishes. Focus on the outcome, not the process. Write this for Claude — it reads this after deciding to load the skill, so get straight to the point.

Example: "This skill creates a complete alarm pipeline: groups for severity classification, alarm items with limit-based triggers on existing tags, and verification via AlarmsMonitor."

On this page:

What This Skill Does
When to Use This Skill
Prerequisites
MCP Tools and Tables Involved
Implementation Steps
Code Examples
Verification
Common Pitfalls
Variations
Related Skills
See also

When to Use This Skill

Describe the trigger conditions. Be specific — Claude uses this to decide whether to load the skill.

Use this skill when:

- The user asks to *‹describe specific scenario›*
- The user mentions *‹specific keywords or concepts›*
- The solution needs *‹specific capability›*

Do NOT use this skill when:

- *‹Scenario where a different approach is better›*
- *‹Common misapplication to avoid›*

Prerequisites

What must exist in the solution before this skill can be applied.

- Solution must be open (`open_solution` or `create_solution` completed)
- *‹Specific tags, channels, or objects that must exist›*

- *‹Required ScriptsReferences or NuGet packages›*

MCP Tools and Tables Involved

Quick reference for Claude on which tools and table types this skill uses.

Category	Items
Tools	`get_table_schema`, `write_objects`, `list_protocols`, `get_objects`, `list_elements`
Tables	`UnsTags`, `DevicesChannels`, `DevicesNodes`, `DevicesPoints`

Always pass `solution_id`. The `solution_id` returned by `open_solution` / `create_solution` is the required routing key on the session-bound tools — `get_objects`, `write_objects`, `get_state`, `browse_object_model`, `designer_action` and their peers. Omitting it fails the call with `MISSING_SOLUTION_ID`. Solution-independent tools (`get_table_schema`, `list_protocols`, `list_dynamics`, `search_docs`) do not take it.

Implementation Steps

Step 1: ‹Action Name›

Explain what to do and why. Include the exact MCP tool call.

First, fetch the schema to confirm field names:

```
get_table_schema('TableType')
```

Then write the objects:

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "table_type": "TableType",
  "data": [
    {
      "Name": "ObjectName",
      "Property1": "Value1",
      "Property2": "Value2"
    }
  ]
}
```

Key decisions:

- Why this value for Property1
- What to change for different scenarios

Step 2: ‹Action Name›

Continue the pattern. Each step should be self-contained enough that Claude can execute it and verify before moving on.

Error rendering macro 'code': Invalid value specified for parameter 'com.atlassian.confluence.ext.code.render.InvalidValueException'

```
{
  "table_type": "AnotherTable",
  "data": [
    {
      "Name": "ObjectName",
      "DependsOn": "ObjectFromStep1"
    }
  ]
}
```

Step 3: ‹Action Name›

Final configuration step.

Code Examples

Include C# script examples when the skill involves ScriptsClasses or ScriptsTasks.

ScriptsClasses — ClassContent for ‹purpose›

```
public static void MethodName(string tagPath)
{
    double value = @Tag.TagPath.Value;
    // Process logic here
    @Tag.OutputTag.Value = result;
}
```

Expression to trigger this class:

ObjectName	Expression	Execution
TriggerName	`Script.Class.ClassName.MethodName(Tag.InputTag)`	OnChange

Verification

How to confirm the implementation is correct. List concrete checks.

1. `get_objects('TableType', solution_id=<id>)` — verify the new objects appear
2. `get_state(target='designer', solution_id=<id>)` — confirm no compilation errors (for scripts)
3. Start runtime check that *‹expected behavior occurs›*
4. `browse_object_model('Tag.Path', solution_id=<id>)` — verify tags have expected values

Common Pitfalls

Mistake	Why It Happens	How to Avoid
‹Mistake 1›	*‹Root cause›*	*‹Correct approach›*
‹Mistake 2›	*‹Root cause›*	*‹Correct approach›*
‹Mistake 3›	*‹Root cause›*	*‹Correct approach›*

Variations

Optional section for common variations of this skill.

Variation A: ‹Name›

- Change Step 2 to use *‹alternative approach›*
- Useful when *‹scenario›*

Variation B: ‹Name›

- Skip Step 3 and instead *‹alternative approach›*
- Useful when *‹scenario›*

Related Skills

- `skill-related-name` — *‹How it connects›*
- `skill-another-name` — *‹How it connects›*

See also

- Skill Authoring Guide. How to write, test, and publish a custom skill end-to-end.
- Skill Acme Naming Conventions. Worked example of a published user-created skill.