Skill-Web Integration Protocol

Version: 2.0.0
Effective Date: 2026-02-21
Owner: Abacus (System Architecture)

Overview

The Skill-Web is a directed graph connecting skills, agents, tasks, and meeting content. V2 enables **runtime skill injection** — agents dynamically receive skill context based on their assigned tasks.

Skill-Web Structure

Directory Layout

```
_shared/skill-web/
├── _graph.yaml              # Master skill graph
├── _agent-skills.yaml       # Agent-to-skill mappings
├── skills/                  # Skill definitions
│   ├── api-development.md
│   ├── code-review.md
│   ├── research-synthesis.md
│   └── ...
└── categories/              # Skill categories
    ├── development.yaml
    ├── research.yaml
    └── operations.yaml
```

Skill Node Schema

```yaml
# _shared/skill-web/skills/api-development.md
---
skill_id: api-development
name: "API Development"
category: development

description: |
  Building and maintaining REST/GraphQL APIs.
  Includes design, implementation, testing, and documentation.

prerequisites:
  - python-basics
  - http-protocols

related_skills:
  - fastapi
  - authentication
  - rate-limiting

capabilities:
  - "Design RESTful endpoints"
  - "Implement request validation"
  - "Handle errors gracefully"
  - "Write API documentation"

agents_with_skill:
  - claude
  - abacus

context_injection:
  templates:
    - "templates/api-endpoint.py"
    - "templates/api-tests.py"
  documentation:
    - "docs/api-guidelines.md"
  examples:
    - "examples/rate-limiter.py"
---

## API Development Skill

### Best Practices
1. Use consistent naming conventions
2. Version your APIs
3. Document all endpoints
4. Implement proper error handling

### Templates
[See context_injection.templates]

### Common Patterns
...
```

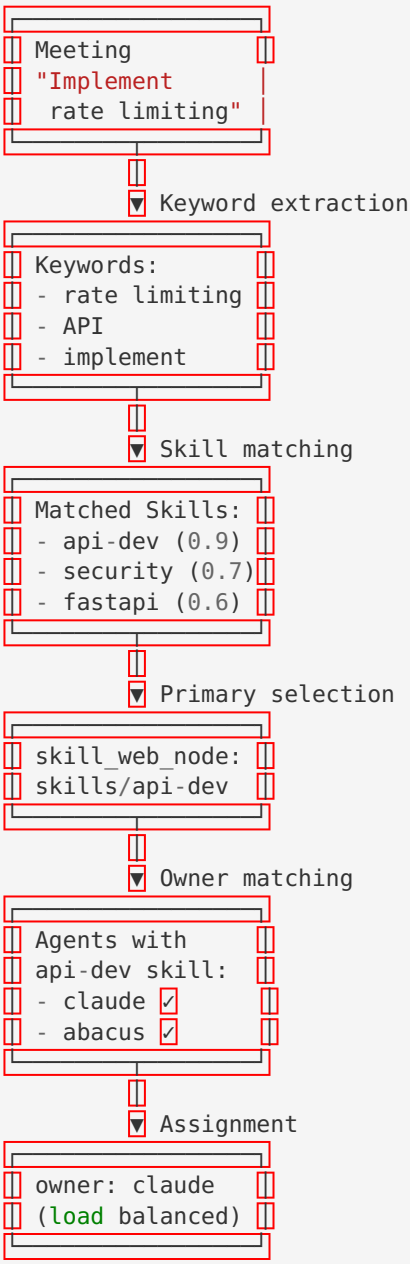
Task-to-Skill Linkage

How Tasks Link to Skills

Every task has a `skill_web_node` field pointing to the required skill:

```
# In task file BPRD-2026-0043.md
skill_web_node: skills/api-development
```

Linkage Workflow



Runtime Skill Injection

Concept

When an agent starts working on a task, the system automatically injects relevant skill context into the agent's working memory.

Injection Process

```python
def inject_skill_context(agent, task):
    """Inject skill context when agent starts task."""

    # 1. Load skill definition
    skill = load_skill(task.skill_web_node)

    # 2. Load related skills (1 level deep)
    related = [load_skill(s) for s in skill.related_skills]

    # 3. Load context injection materials
    context = {
        'templates': load_files(skill.context_injection.templates),
        'documentation': load_files(skill.context_injection.documentation),
        'examples': load_files(skill.context_injection.examples),
    }

    # 4. Load prerequisites (summary only)
    prereqs = [load_skill_summary(p) for p in skill.prerequisites]

    # 5. Inject into agent context
    agent.inject_context({
        'primary_skill': skill,
        'related_skills': related,
        'context_materials': context,
        'prerequisites': prereqs,
        'task': task,
    })

    return agent
```

What Gets Injected

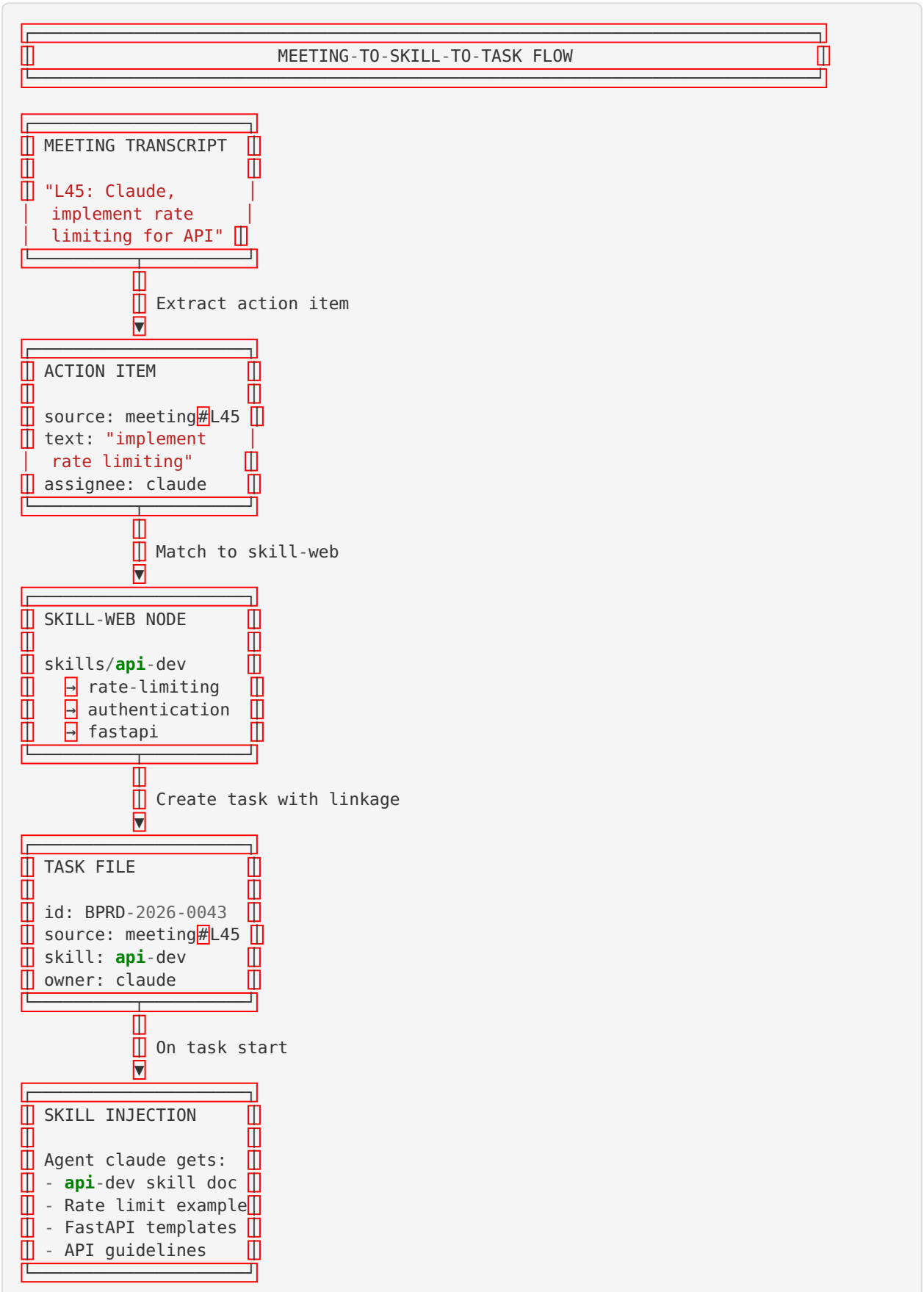
Content Type	Description	Priority
Primary Skill	Full skill definition	High
Templates	Code/doc templates for the skill	High
Examples	Reference implementations	Medium
Documentation	Relevant docs	Medium
Related Skills	Summary of related skills	Low
Prerequisites	Summary only	Low

Injection Triggers

1. **Task Assignment**: When `status` → `Assigned`
2. **Task Start**: When `status` → `In-Progress`
3. **Context Refresh**: On agent request
4. **Skill Update**: When skill definition changes

Meeting Transcript → Skill Node → Task Linkage

Full Pipeline



Traceability

Every element maintains bidirectional links:

```yaml
# Meeting transcript metadata
generated_tasks:
  - BPRD-2026-0043
  - BPRD-2026-0044

# Task file
source: meetings/transcripts/meeting-2026-02-21-14-30.md#L45
skill_web_node: skills/api-development

# Skill node
related_tasks:
  - BPRD-2026-0043  # Active
  - BPRD-2026-0012  # Complete (example)
```

Skill Graph Query API

Query Operations

```python
class SkillWebAPI:
    def find_skill_by_keywords(self, keywords: list[str]) -> list[SkillMatch]:
        """Find skills matching keywords."""

    def get_agents_with_skill(self, skill_id: str) -> list[Agent]:
        """Get all agents who have a skill."""

    def get_skill_path(self, from_skill: str, to_skill: str) -> list[str]:
        """Find learning path between skills."""

    def inject_skill(self, agent_id: str, skill_id: str) -> Context:
        """Inject skill context into agent."""

    def get_related_tasks(self, skill_id: str, status: str = None) -> list[Task]:
        """Get tasks related to a skill."""
```

Agent-Skill Matrix

`_agent-skills.yaml`

```yaml
agent_skills:
  claude:
    primary:
      - api-development
      - code-review
      - python
      - research-synthesis
    learning:
      - rust
      - kubernetes

  grok:
    primary:
      - strategic-planning
      - decision-making
      - vision-setting
    advisory:
      - all  # Can advise on any skill

  abacus:
    primary:
      - system-architecture
      - automation
      - data-pipelines
      - infrastructure

  perplexity:
    primary:
      - verification
      - fact-checking
      - quality-audit
      - research

  gemini:
    primary:
      - content-creation
      - media-production
      - writing
      - editing
```

Skill Gap Detection

When a Task Has No Matching Agent

```python
def handle_skill_gap(task, required_skill):
    """Handle case where no agent has required skill."""

    agents = skill_web.get_agents_with_skill(required_skill)

    if not agents:
        # No one has this skill
        return SkillGapResolution(
            action="escalate",
            notify=["grok"],  # Chief decides
            options=[
                "assign_to_learning_agent",
                "external_contractor",
                "defer_task",
                "simplify_requirements",
            ]
        )
    elif all(agent.workload > threshold for agent in agents):
        # All skilled agents overloaded
        return SkillGapResolution(
            action="queue",
            priority="when_available",
            notify=agents,
        )
```

Integration with BPR&D-To-Do-List

Skill Column

The v2 To-Do List includes a Skill column:

```
| ID | Task | Owner | Status | Skill | Source |
|----|------|-------|--------|-------|--------|
| BPRD-2026-0043 | API Rate Limiting | claude | In-Progress | [api-dev](link) | [Meet-
ing](link) |
```

Skill-Based Filtering

Agents can filter the To-Do List by their skills:

```
# Show tasks I can work on
filter: skill IN my_skills AND status = 'Created'

# Show tasks requiring skills I'm learning
filter: skill IN my_learning_skills
```

Configuration

`config/skill-web-config.yaml`

```yaml
skill_web:
  graph_file: "_shared/skill-web/_graph.yaml"
  agent_skills: "_shared/skill-web/_agent-skills.yaml"

  injection:
    enabled: true
    max_depth: 2  # Related skills depth
    include_examples: true
    include_templates: true

  matching:
    algorithm: "semantic"  # or "keyword"
    min_confidence: 0.7
    fallback_skill: "general"

  gap_handling:
    escalate_to: "grok"
    timeout_hours: 24
```