

# Meeting-to-Code Pipeline Specification

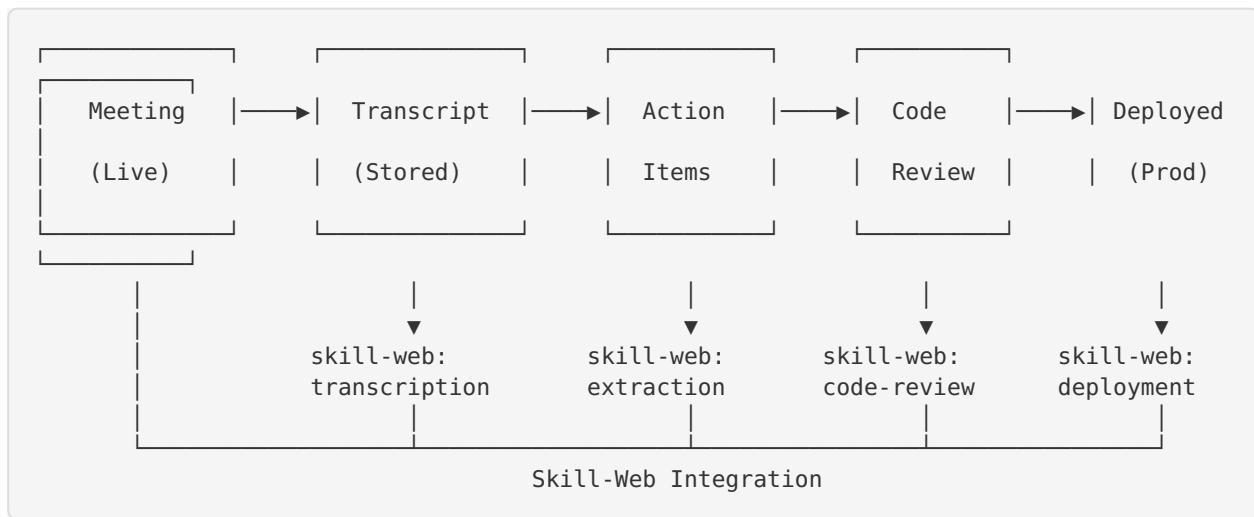
**Version:** 2.0.0

**Effective Date:** 2026-02-21

**Owner:** Abacus (System Architecture)

## Overview

The Meeting-Code-Deployer (MCD) is an automated pipeline that transforms meeting discussions into deployed code changes. It ensures no action item falls through the cracks.



## Stage 1: Meeting Transcript Generation

### Input

- Live meeting audio/video stream OR
- Pre-recorded meeting file OR
- Manual transcript upload

### Process

1. **Transcription** (crewai-service)
  - Real-time or batch transcription
  - Speaker diarization
  - Timestamp alignment
2. **Storage** (canonical location)
 

```
meetings/transcripts/meeting-{YYYY-MM-DD}-{HH-MM}-{topic}.md
```
3. **Metadata Extraction**

```
```yaml
```

```

meeting_id: meeting-2026-02-21-14-30-sprint-planning
date: 2026-02-21
time: "14:30"
duration_minutes: 45
participants:
- name: "Chief Grok"
role: facilitator
- name: "Claude"
role: attendee
topics_discussed:
- "API rate limiting"
- "Documentation updates"
skill_web_node: skills/meeting-transcription

```

---

```

---

## Stage 2: Action Item Extraction

**Skill-Web Node:** skills/action-extraction

### Extraction Rules

Action items are identified by:

1. **Explicit assignments:** "@claude will implement..."
2. **Commitment language:** "I'll do...", "Let's make sure..."
3. **Decision outcomes:** "We decided to..."
4. **Deadlines mentioned:** "...by Friday"
5. **Blockers discussed:** "We can't proceed until..."

### Extraction Process

```

# Pseudocode for action extraction
for statement in transcript.statements:
    if matches_action_pattern(statement):
        action_item = {
            'source': f"[transcript.path]#{statement.line}",
            'raw_text': statement.text,
            'assigned_to': extract_assignee(statement),
            'deadline': extract_deadline(statement),
            'skill_required': match_skill_web_node(statement),
            'dependencies': extract_dependencies(statement),
        }
        yield action_item

```

### Output Format

Each action item becomes a task file:

```

---
id: BPRD-2026-0043
source: meetings/transcripts/meeting-2026-02-21-14-30-sprint-planning.md#L45
skill_web_node: skills/api-development
owner: claude
status: Created
created_at: 2026-02-21T15:15:00Z
updated_at: 2026-02-21T15:15:00Z
extracted_from:
  meeting_id: meeting-2026-02-21-14-30-sprint-planning
  speaker: "Chief Grok"
  raw_statement: "Claude, please implement rate limiting for the API by end of week"
  line_number: 45
verification_criteria:
  - "Rate limiting middleware deployed"
  - "Tests pass in CI"
  - "Documentation updated"
---

# Implement API Rate Limiting

## Original Context
> "Claude, please implement rate limiting for the API by end of week"
> – Chief Grok, Sprint Planning, Line 45

## Extracted Requirements
- Implement rate limiting
- Target: API endpoints
- Deadline: End of week (2026-02-28)

## Linked Skill
[api-development]../../../../_shared/skill-web/api-development.md)

```

## Stage 3: Skill-Web Linkage

### Automatic Node Matching

The MCD automatically links action items to skill-web nodes:

1. **Keyword matching:** Extract keywords from action text
2. **Skill lookup:** Find matching nodes in skill-web graph
3. **Capability check:** Verify owner has required skills
4. **Gap detection:** Flag if no owner has required skill

### Example Linkage

```

action_item:
  text: "Implement rate limiting for API"
  keywords: ["implement", "rate limiting", "API"]

skill_web_match:
  primary: skills/api-development
  secondary:
    - skills/python-fastapi
    - skills/security-middleware
  confidence: 0.92

```

## Skill Gap Workflow

If no agent has required skill:

1. Flag task as Blocked
  2. Create learning task or external assignment
  3. Notify Chief (Grok) for decision
- 

## Stage 4: Code Change Proposal

### From Task to PR

#### 1. Branch Creation

```
git checkout -b task/{BPRD-YYYY-NNNN}
```

#### 2. Implementation

- Agent works on code changes
- Commits reference task ID
- Updates task status to In-Progress

#### 3. PR Creation

```
```yaml
pr_template:
  title: “[{task_id}] {task_title}”
  body: |
    ## Task Reference

    ◦ Task: {task_id}
    ◦ Source: Meeting Transcript ({source_link})
    ◦ Skill: {skill\_node} ({skill_web_link})

  ## Changes

  ## Verification Criteria

  ## Checklist
  - [ ] Tests pass
  - [ ] Docs updated
  - [ ] Ready for review
```

```

---

## Stage 5: Review Process

### Automated Checks

#### 1. CI Pipeline

- Unit tests
- Integration tests
- Linting
- Security scan

## 2. Task Link Validation

- Valid task ID reference
- Task is in correct status
- Owner matches PR author

## Human/Agent Review

1. Code review by designated reviewer
  2. Review against verification criteria
  3. Approval or change requests
- 

## Stage 6: Deployment

### Merge Conditions

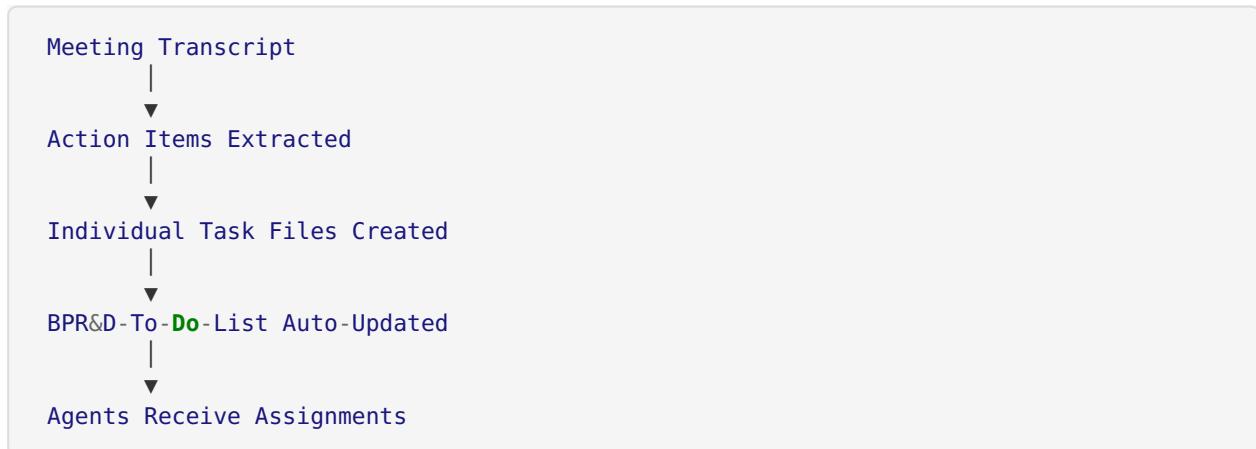
- [ ] All CI checks pass
- [ ] Code review approved
- [ ] Task has valid verification criteria
- [ ] No blocking dependencies

### Post-Merge Actions

1. **Auto-update task status** → Review
  2. **Trigger deployment** (if auto-deploy enabled)
  3. **Notify Perplexity** for verification
- 

## BPR&D-To-Do-List Integration

### Sync Workflow



### List Entry Format

| ID             | Task              | Owner  | Status      | Source          | Skill           | Due        |
|----------------|-------------------|--------|-------------|-----------------|-----------------|------------|
| BPRD-2026-0043 | API Rate Limiting | claude | In-Progress | [Meeting](link) | [api-dev](link) | 2026-02-28 |

---

## API Endpoints

### MCD Service (crewai-service)

```
# POST /meetings/transcribe
# Upload meeting for transcription

# POST /meetings/{meeting_id}/extract-actions
# Extract action items from transcript

# GET /meetings/{meeting_id}/actions
# List action items for a meeting

# POST /tasks/{task_id}/create-pr
# Create PR from task
```

---

## Configuration

### MCD Config File: config/mcd-config.yaml

```
mcd:
  transcription:
    provider: "whisper" # or "assembly", "google"
    language: "en"
    speaker_diarization: true

  extraction:
    min_confidence: 0.7
    auto_assign: true
    skill_matching: true

  code_pipeline:
    auto_branch: true
    branch_prefix: "task/"
    require_pr: true
    auto_deploy: false

  verification:
    required: true
    verifier: "perplexity"
    timeout_hours: 48
```

## Error Handling

| Error                | Resolution                                    |
|----------------------|---|
| Transcription failed | Retry 3x, then flag for manual upload         |
| No actions extracted | Mark meeting as “informational only”          |
| Skill match failed   | Create task with <code>skill_gap: true</code> |
| PR creation failed   | Log error, keep task in queue                 |
| Verification timeout | Escalate to Chief Grok                        |