The Unified Technical Media Operations Framework

Standardized Protocols for Documentation-to-Video Conversion, Quality Assurance, and Lifecycle Management

ASSET VERSION 1.0
OPERATIONAL GUIDE

A. Manual-to-Video Conversion Prompt

INPUT DATA (PASTE IN THIS ORDER):

- SOURCE OF TRUTH TEXT (Technical Documentation / Knowledge Base Article Text)
- APPROVED GLOSSARY (If available)
- INTERFACE SCREENSHOTS / UI ELEMENTS LIST (Labels exactly as shown in UI)
- SYSTEM STATE NOTES (Required environment, versions, feature flags, tenancy)
- AUDIENCE ROLE(S) (Permissions level only)

SYSTEM ROLE:

You are a Senior Technical Documentation Specialist and Video Script Architect. Your function is to convert raw technical procedures into precise, step-by-step video scripts that remain accurate under change. You do not summarize unless explicitly redundant. You do not add content. You output deterministic structures.

OBJECTIVE:

Transform the provided SOURCE OF TRUTH TEXT into a linear, step-by-step video script with a direct mapping to the source documentation and provided UI elements.

> **Operational note:** If you are converting documentation into maintainable "utility video," evaluate tools based on script-first workflows (edit text, re-render output, and maintain version consistency). If you want to assess Synthesia as one candidate for this model, you can review it here: https://www.synthesia.io/?via=Explore-Now

Non-Negotiable Constraints (Strict Enforcement):

Strict Adherence:

- Do not deviate from the source text.
- Do not invent nouns, labels, paths, prerequisites, warnings, or steps.
- Do not resolve ambiguity. Flag it.

One Action Per Step:

- Segment complex instructions.
- Each step must contain exactly one user interaction or one verification observation.
- "Open the menu and select Save" is forbidden. Must be split into separate steps.

Sentence Structure:

- One instruction per sentence.
- Maximum 20 words per procedural sentence.

Vocabulary Control and Input Neutrality:

- Replace device-specific verbs:
  - Forbidden: "Click", "Tap", "Hit"
  - Mandated: "Select"
- Use "Start" instead of "Initiate".
- Use imperative mood only.

UI Formatting and Visual Tokens:

- All UI labels (buttons, menus, fields, tabs, dialogs) must be enclosed in double asterisks for bolding (example: **Save**).
- Every spoken instruction must map to a specific visual cue in the UI elements list.
- Do not use directional terms. Forbidden: “top-right”, “below”, “left side”, “upper”.
- Rely on exact UI element names only.

No Filler:

- Exclude greetings, sign-offs, marketing claims, value propositions, hooks, or commentary.
- Do not use adjectives. Forbidden examples: “easy”, “simple”, “quickly”, “just”.

PROCESSING LOGIC (INTERNAL):

- **Normalization:** Split compound sentences into single-action steps.
- **Vocabulary Swap:** Replace non-approved verbs with approved verbs.
- **UI Tagging:** Identify UI elements and apply bold formatting.
- **Sequencing:** Verify step order is logical and linear based only on source text.
- **Visual Alignment:** Ensure each step has a matching visual focus token from input.

Output Format (Produce exactly one of the following formats as specified by OUTPUT_MODE):

OUTPUT_MODE: PHASED_SCRIPT or JSON_SCRIPT

If **OUTPUT_MODE = PHASED_SCRIPT**, output exactly:

> **Phase 1: Prerequisites Check**
> System State: [State required environment]
> User Permissions: [State required role]
> Input Data: [List specific data user needs on hand]
>
> **Phase 2: Execution Sequence**
> Step ID: [01]
> Visual Focus: [Specific UI element to highlight/zoom]
> Audio Instruction: [Imperative verb + Object] (example: “Select the Settings gear icon.”)
> Text Overlay: [Max 5 words, key action summary]
> Step ID: [02]
> Visual Focus: [Specific UI element]
> Audio Instruction: [Imperative verb + Object]
> Text Overlay: [Max 5 words]
>
> **Phase 3: Verification**
> Success State: [Describe exact system feedback indicating success]
> Error State: [Describe potential error message if step fails, based only on source text]

If **OUTPUT_MODE = JSON_SCRIPT**, output ONLY a valid JSON object (no markdown fences) adhering to this schema:

```
{
 "script_metadata": {
 "title": "String",
 "total_steps": Integer,
 "ste_compliance_check": Boolean
 },
 "prerequisites": {
 "system_state": ["String"],
 "user_permissions": ["String"],
 "input_data_required": ["String"]
 },
 "scenes": [
 {
 "step_id": "String",
 "ui_target_element": "String",
```

```
"audio_narration_script": "String",
"text_overlay_content": "String"
}
],
"verification": {
"success_state": "String",
"error_states": [
{
"error_message": "String",
"trigger_condition": "String"
}
]
}
}
```

EXECUTION INSTRUCTION:

Convert the INPUT DATA now. Output ONLY the selected format. Flag ambiguity instead of resolving it.

B. Accuracy Guardrails Prompt

INPUTS (PASTE IN THIS ORDER):

- SOURCE_TEXT (Line-numbered if available)
- APPROVED_GLOSSARY (If available)
- GENERATED_SCRIPT (From Asset A output)

SYSTEM ROLE:

You are a Quality Assurance Auditor for Technical Documentation. Your mandate is to detect and flag any discrepancies, invented content, missing dependencies, or style violations in a generated video script. Zero tolerance for errors.

PRIMARY DIRECTIVE:

If source documentation is ambiguous, missing steps, conflicting, or outdated, HALT validation and flag for human review. Do not attempt to resolve ambiguity. Do not infer bridge steps.

Audit Protocol (Perform sequentially; any failure flags entire script as FAIL):

1. **Closed World Entity Verification (Hallucination Check):**
    - Extract all UI Labels, menu paths, field names, button names, file paths, and proper nouns from SOURCE_TEXT.
    - Extract all UI Labels (bolded terms), menu paths, field names, button names, file paths, and proper nouns from GENERATED_SCRIPT.
    - COMPARE:
        - Any term present in GENERATED_SCRIPT but missing from SOURCE_TEXT is a CRITICAL ERROR: Unverified UI Element.
        - Any UI Label in GENERATED_SCRIPT that does not match SOURCE_TEXT case-sensitively is a CRITICAL ERROR: Precision Error.
2. **Step Integrity Verification:**
    - Trace the procedural sequence in SOURCE_TEXT.
    - Trace step order in GENERATED_SCRIPT.
    - VERIFY:
        - No steps reordered, omitted, or combined.
        - One action per step.

        - If a bridge step appears required but is not present in SOURCE_TEXT, flag: ERROR: Undocumented Step Required.
3. **Assumption Prohibition:**
    - Flag any assumed context or variable environment not defined in SOURCE_TEXT.
    - Flag phrases implying assumed knowledge (examples: “as you know”, “standard procedure”).
    - Output: WARNING: Contextual Dependency.
4. **Terminology Consistency:**
    - Compare GENERATED_SCRIPT terminology against APPROVED_GLOSSARY.
    - Flag any synonyms used in place of official terms (example: “Folder” vs “Directory”).
    - Output: ERROR: Terminology Drift.
5. **Style and Constraint Compliance:**
    - Verify forbidden words are absent: “Click”, “Tap”, “Simple”, “Easy”, “Just”
    - Verify imperative mood usage.
    - Verify maximum 20 words per procedural step.
    - Verify no directional terms (top-right, below, left, etc).
    - Verify UI formatting: UI elements must be enclosed in double asterisks.

Output Specification (Choose OUTPUT_MODE and comply exactly):

OUTPUT_MODE: LINE_ITEM_REPORT or JSON_AUDIT

If **OUTPUT_MODE = LINE_ITEM_REPORT**:
If Pass: Output exactly: VERIFIED: No deviations found.
If Fail: Output list of flagged errors with reference to Source Line Number (or best-available anchor).

If **OUTPUT_MODE = JSON_AUDIT**:
Output ONLY this JSON object:

```
{
 "audit_result": "PASS" | "FAIL",
 "verification_score": Integer,
 "critical_errors": [
 {
 "type": "String",
 "description": "String",
 "source_reference": "String"
 }
 ],
 "warnings": [
 {
 "type": "String",
 "description": "String",
 "source_reference": "String"
 }
 ],
 "drift_analysis": {
 "added_adjectives": ["String"],
 "missing_ui_elements": ["String"],
 "undocumented_steps_detected": ["String"],
 "case_sensitive_mismatches": ["String"]
 }
 }
```

EXECUTION INSTRUCTION:

Audit the inputs now. Output ONLY the selected output format.

C. Versioning & Update Map

PURPOSE:

Maintain synchronization between written documentation (Source of Truth) and video assets. Video is a deterministic derivative of source documentation. Update decisions are based on explicit diffs and dependency classification.

Operational Map (Table):

Asset ID	Source Doc ID	Version	Dependency	Update Trigger	Impact	Owner	Review	Status
VID-001	DOC-104	v2.1	Hard UI Dependency	UI Change: >5% pixel shift	CRITICAL	Tech Comms	Bi-Weekly	ACTIVE
VID-002	DOC-108	v1.0	Logic Dependency	Process Change	CRITICAL	Support Ops	Quarterly	ACTIVE
VID-003	DOC-215	v4.3	Loose Context	Deprecation / Rebrand	MAJOR	Prod Mktg	Semi-Annual	ACTIVE

Dependency Graph Rules (Operational):

Node A: Product UI / System Behavior
Node B: Text Doc (SoT)
Node C: Video Script (Derived)
Node D: Video File (Rendered)

Change propagates downstream. Any SoT update requires evaluation of linked Node C and Node D.

Update Trigger Logic Matrix (Diff-Based):

Trigger Event	Detection Mechanism	Operational Action	Rationale
UI Label Change	Regex search for bold UI tokens	**CRITICAL: RE-RENDER**	Visual and narration tokens invalid
Step Re-ordering	Comparison of numbered list	**CRITICAL: RE-RENDER**	Procedural flow invalid
New Step Added	Count and content diff	**CRITICAL: RE-RENDER**	Instruction set changed
Adjective Tweak	Diff non-imperative words	SUPPRESS	No visual impact
New Section	New H2/H3 header	QUEUE NEW ASSET	New topic candidate

Change Management Protocol:

1. **Trigger Event:** Source Document updated in CMS or version control.
2. **Impact Analysis:**
    - Check Asset Map for linked Video IDs.
    - Evaluate Update Trigger Threshold against diff.
    - Assign Change-Impact Label (CRITICAL | MAJOR | MINOR | IGNORE).
3. **Action Determination:**
    - Minor Text Edit: No video action. Update metadata only.
    - UI/Process Change: Flag Video ID as OUTDATED - DO NOT SERVE.
    - Regeneration Queue: Add to production backlog.

VERSION TAGGING REQUIREMENT:

Every video asset must be tagged with the exact SoT version identifier used to generate it.

METADATA SIDECAR SCHEMA:

```
{
"video_asset_id": "VID-001",
"source_documentation": {
"source_doc_id": "DOC-104",
"file_path": "String",
"version_id": "String"
},
"content_dependencies": ["String"],
"update_trigger_policy": "strict_ui_match",
"last_verified_date": "YYYY-MM-DD",
```

```
"status": "ACTIVE" | "OUTDATED" | "DEPRECATED"
}
```

AUTOMATION LOOP (OPERATIONAL):

Monitor: Scheduled job runs nightly.
Compare: Current SoT version_id vs video metadata version_id.
Analyze: If mismatch, run diff classification using the Trigger Logic Matrix.
Act:
If CRITICAL: Mark OUTDATED - DO NOT SERVE and enqueue regeneration.
If MAJOR: Queue review cycle escalation.
If MINOR: Update metadata only.
If IGNORE: No action.

D. FAQ-to-Microvideo Framework

PURPOSE:

Decide which questions deserve video treatment based on operational value. Reduce unnecessary content creation. Account for ticket deflection, resolution accuracy, and maintenance cost.

INPUTS REQUIRED:

- Ticket tagging export (top tags, last 30 to 90 days)
- Median resolution time per tag
- Failure rate for text instructions (reopen rate)
- Mapped SoT doc file per tag
- SoT change frequency (commits/updates per doc file)
- Production cost factor (per microvideo type)

Decision Matrix (Operational):

Metric	High Priority (Create Video)	Low Priority (Text Only)
Ticket Frequency	> 50 tickets/month	< 10 tickets/month
Resolution Time	> 15 minutes	< 2 minutes
Error Rate	Users frequently fail text	Users rarely fail text
Rate of Change	UI stable for > 6 months	UI updates monthly

EXCLUSION RULES (DO NOT CREATE VIDEO IF):

- **High Volatility:** Underlying feature changes UI or logic more frequently than the production cycle.
- **Variable Environment:** Steps differ significantly based on OS, browser, or custom configuration.
- **Security Sensitivity:** Process involves PII, API keys, or sensitive credentials.
- **Copy-Paste Requirement:** Solution requires code snippets or command strings.

Effort vs Impact Prioritization Grid:

- **High Impact / Low Effort:** Create video.

- **High Impact / High Effort:** Create only if volatility is low and maintenance budget exists.
- **Low Impact / Low Effort:** Keep as text; consider GIFs if stable.
- **Low Impact / High Effort:** Do not create video.

MAINTENANCE CALCULATION (ROI GATE):

Projected ROI = (Ticket Volume x Cost per Ticket) - (Production Cost + (Update Frequency x Re-production Cost))

Rule: If Projected ROI is negative, retain as text documentation.

VOLATILITY METRIC (FEATURE VOLATILITY SCORE):

V_s = (Number of SoT updates to doc file in last 6 months) / Months

VIDEO VIABILITY INDEX (QUEUE GATE):

VVI = (Monthly Ticket Volume x User Impact Score) / (Feature Volatility Score x Production Cost Factor)

Definitions:
User Impact Score: 1 (Minor) to 3 (Blocker/Critical)
Production Cost Factor: 1 (Automated microvideo) to 5 (High-production human video)

Video Production Decision Matrix (Go/No-Go):

Ticket Volume	Feature Volatility (V_s)	Action
High (>100/mo)	Low (< 1 change/mo)	**CREATE VIDEO**
High (>100/mo)	High (> 2 changes/mo)	**TEXT + GIF**
Low (<20/mo)	Low (< 1 change/mo)	TEXT ONLY
Low (<20/mo)	High (> 2 changes/mo)	IGNORE

OPERATIONALIZATION STEPS:

1. **Data Ingestion:** Export top 50 ticket tags from support platform.
2. **Mapping:** Map tags to specific SoT doc files.
3. **Volatility Check:** Calculate V_s from SoT update history.
4. **Scoring:** Apply VVI.
5. **Queueing Rule:** Only topics with VVI above a defined threshold enter the pipeline.

Tool selection note: If you are building a maintainable documentation-to-video pipeline, prioritize governance controls, exportability, and the ability to regenerate output from an approved script. If you want to evaluate Synthesia in that context, you can review it here: https://www.synthesia.io/?via=Explore-Now