📄 FSD_10 – Agent Update & Knowledge Refresh Module

🧠 Purpose:

This module governs how an **already-launched Audira agent is updated** over time, as the business evolves — including:

- New document uploads
- Answer changes
- Memory patching
- Segment invalidation
- Auto-refresh of the prompt chain and readiness status

It ensures that **long-lived agents remain accurate, current, and reliable** — without requiring full re-onboarding each time.

🧩 FSD_10 – Section Breakdown

Section	Description
1. Scope	What triggers an update, and what gets refreshed
2. Input Detection Logic	How new data is identified and categorized
3. Refresh Pathways	The decision tree: what updates quietly, what triggers retraining
4. Memory Update Protocol	How existing memory blocks are overwritten, merged, or frozen
5. Prompt Chain Refresh Rules	How prompt content gets rebuilt safely
6. Validator Re-run Conditions	When a post-launch readiness check is required
7. Role-Based Access & Permissions	Who can update what — user vs. admin vs. system
8. Future Enhancements	Continuous agent learning, update journaling, change approval flows

🔷 Section 1: Scope

🎯 Purpose:

The **Agent Update & Knowledge Refresh Module** ensures that a live agent continues to reflect the SMB's most recent and accurate business data — even after launch. It allows Audira to deliver **dynamic knowledge alignment** across:

- New files uploaded post-launch
- Changes to prior answers
- Updated business policies, prices, or vendor info
- Admin-triggered memory corrections
- QA-based improvements or corrections from feedback

✅ Responsibilities:

Function	Description
Detect Post-Launch Changes	Monitor for new files, changed responses, admin edits
Update Memory Blocks	Apply versioned overwrites or merges to FSD_07 memory
Refresh Prompt Chain	Recompile prompt (FSD_06) using updated tag map or memory
Optional Re-Validation	Rerun readiness check (FSD_05) if update affects critical tags
Audit Change History	Log what changed, why, when, and who triggered it
Trigger Retraining Queue	Flag meaningful changes for inclusion in future LLM updates

---

❌ Not in Scope:

Excluded	Reason

Manual override of validator results	Managed in FSD_09
Real-time agent response logic	Handled in FSD_08
Prompt chain design and structure	Defined in FSD_06
Memory conflict resolution	Managed in FSD_07 unless retriggered by update logic

---

📌 **When This Module Is Activated:**

Trigger	Scenario
✅ **New file uploaded**	User adds a newer “2025 Pricing Sheet”
✅ **Onboarding answer changed**	Customer updates “payment method” from Stripe to Paddle
✅ **Memory block edited by admin**	Conflict about refund policy is resolved and replaced
✅ **Segment invalidated**	Outdated file (pre-2023) auto-flagged by system logic
✅ **Feedback confirmed**	QA confirms hallucinated answer; memory patch is required

---

📎 **References**:

- 📘 *AUDIRA PRODUCT BLUEPRINT* – defines post-launch agent lifecycle and update triggers
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – source of new documents, timestamps, and file changes
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – sets ownership, permissions, and agent class
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – reused when re-checking post-update readiness

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – used for rebuilding prompt chain post-update

---

🔷 Section 2: Input Detection Logic

This section defines how the system **detects new or changed inputs** after an agent is already live, and classifies the update to determine the appropriate refresh action.

🕵️ Monitored Update Sources

Source Type	What Is Monitored
File Uploads	New documents uploaded post-launch
Answer Edits	User or admin modifies onboarding answers
Admin Interventions	Manual memory or segment updates
QA Feedback Confirmations	Validated hallucinations or corrections
System Rules	Time-based triggers (e.g., auto-check file age)

🧠 Input Change Classification

Every detected change is classified into one of the following update types:

Update Type	Trigger Example	Classification Tag
new_segment	New file uploaded with unseen content	SEGMENT_ADDITION
segment_replace	Updated version of previously uploaded doc	SEGMENT_VERSION_REPLACE
answer_change	Edited onboarding response	ANSWER_EDIT
memory_patch	Admin correction of conflict	MANUAL_MEMORY_UPDATE
auto_invalidation	File marked as stale or superseded	SEGMENT_EXPIRED

📜 Detection Record Example


```
{
 "agent_id": "AGENT_0221",
 "detected_change": {
  "type": "ANSWER_EDIT",
  "tag_id": "payment_method",
  "old_value": "Stripe",
  "new_value": "Paddle",
  "source": "user",
  "timestamp": "2025-06-11T04:28:00Z"
 },
 "requires_prompt_refresh": true,
 "requires_memory_update": true,
 "triggers_revalidation": true
}
```

🔄 Update Detection Workflow

1. **Trigger Check**
   → Event occurs (e.g., file upload)
2. **Fingerprinting & Diffing**
   → Compare file contents, segment count, tag coverage
3. **Tag Impact Analysis**
   → Recalculate what tags are affected by the change
4. **Refresh Decision Flagging**
   → System determines if prompt, memory, or validator need updates
5. **Log for Audit + Notification**
   → Change recorded in agent lifecycle journal

📎 **References**:

- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – includes document hash, upload timestamp, and version linking
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – provides source of answer edits and user permissions
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – triggers refresh if tag-linked prompt blocks are affected
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – defines roles allowed to initiate updates

---

🔷 Section 3: Refresh Pathways

This section defines the **decision tree** that determines what parts of the agent's logic need to be refreshed — and how — based on the type and severity of detected input changes.

🧭 Refresh Pathway Decision Matrix

Change Detected	Memory Update	Prompt Refresh	Re-run Validator	Trigger Retraining
✅ **New file uploaded**	Yes	Yes if new tags covered	Yes if core tag affected	Optional
✅ **Answer edit**	Yes	Yes	Yes if validator score impacted	Optional
✅ **Segment invalidated**	Yes (soft-delete)	Yes (if prompt referenced)	Yes	No
✅ **Admin memory patch**	Yes	Yes if affects prompt	No	Optional
✅ **Repeated feedback match**	Yes	Yes	Yes (feedback rules)	Yes

---

📦 Pathway Objects

Each update is logged with a refresh_action payload:

```
{
 "agent_id": "AGENT_0187",
 "refresh_action": {
  "memory_update_required": true,
  "prompt_refresh_required": true,
  "validator_rerun_required": false,
  "retraining_flag": true
 },
 "reason": "Admin updated vendor_model after conflict resolution"
}
```

🔄 Refresh Workflow Steps

1. **Classify Change**
   → Based on FSD_10 Section 2 logic
2. **Analyze Impacted Tags**
   → Cross-check with FSD_03 tag map + FSD_06 prompt blocks
3. **Determine Triggered Modules**
   - Memory (FSD_07)
   - Prompt Chain (FSD_06)
   - Validator (FSD_05)
   - Feedback/Training Router (FSD_09)
4. **Queue Updates**
   → Asynchronous job scheduler applies safe refresh order:
   - First: Update memory
   - Then: Rebuild prompt

- Then: (if needed) re-run readiness validator
- Last: Route delta into training queue

🧪 Special Cases

Scenario	Behavior
New segment triggers unknown tag	Auto-add tag, prompt refresh only, no validator rerun unless it's a core tag
Prompt already contains superseded segment	Flag for overwrite; triggers automatic block replacement
Tag replaced but prompt block unchanged	Version the block but suppress prompt rebuild to save cost (if tag is optional)

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – maps which prompt blocks depend on which tags
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – defines when validator must be re-run post-update
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to preview and test updated prompt/memory logic
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – defines per-agent refresh tolerance and automation scope

---

🔷 Section 4: Memory Update Protocol

This section defines **how existing memory blocks are updated**, replaced, merged, or invalidated when new inputs are detected after agent launch.

🧠 Update Modes for Memory Blocks

Mode	Trigger	Description
Replace	New value has higher confidence, newer timestamp, or confirmed override	Overwrites the full memory block
Merge	New value complements existing memory (e.g., adds delivery timelines to support_model)	Combines value fields into enriched memory block
Ignore	Change is redundant or low-confidence (< 0.75)	Memory remains unchanged
Flag for Review	New input contradicts existing memory and conflict resolution is pending	Marks tag as unstable and sends to admin dashboard (FSD_09)
Invalidate	Original segment or source is deprecated	Soft-deletes memory and logs delta

---

📦 Memory Block Replacement Example


```
{
 "tag_id": "support_model",
 "previous_value": "Email support, Mon–Fri",
 "new_value": "Email + live chat, 24/7",
 "change_type": "replace",
 "source": "file_upload:support_policy_2025.pdf",
 "timestamp": "2025-06-11T04:38:00Z",
 "version": 3
```

}

🔁 Merge Logic

- Applies only to additive attributes:
  - Feature lists
  - Partner networks
  - Operating hours
- Result:

"value_text": "Live chat, phone, and email — available 24/7"

🗂️ Versioning & History

Each memory update triggers a new version:

"memory_versions": {

"support_model": [

{"version": 1, "value": "Email only", "timestamp": "..."},

{"version": 2, "value": "Email + live chat", "timestamp": "..."},

{"version": 3, "value": "Live chat, phone, email – 24/7", "timestamp": "..."}

]

}

🔐 Memory Update Safeguards

Rule	Enforcement
Critical tags cannot be overwritten unless confirmed	Enforced by admin or validator logic
Admin overrides are locked from auto-edit	Only another admin can change

Contradictory inputs from same session require review	Auto-flagged for FSD_09
Frozen tags (e.g., legal_structure) require override key	Controlled via onboarding config

---

📎 **References**:

- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – defines which tags are "frozen" vs. dynamic
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – sets which reviewer roles can approve memory overwrites
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – defines tag criticality and update logic
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to preview agent behavior before and after memory patch

---

🔷 Section 5: Prompt Chain Refresh Rules

This section defines **how and when the prompt chain is rebuilt** after memory changes, new data uploads, or admin interventions — ensuring the LLM continues to receive accurate, aligned context.

🔄 When Prompt Chain Refresh is Triggered

Trigger	Prompt Refresh?	Notes
✅ **New tag added to memory**	Yes	New prompt block may be required
✅ **Memory block replaced**	Yes	Inject updated value in relevant block
⚠️ **Optional tag updated**	Maybe	Refresh only if block uses that tag
🚫 **Minor edits to unchanged segments**	No	Token buffer logic prevents refresh

✅ **Prompt block flagged as stale**	Yes	Rebuild block using latest memory snapshot

---

🧠 Prompt Chain Rebuild Logic

1. **Detect affected tags**
   → Map changed tags to prompt blocks using FSD_06 dependency tree.
2. **Mark outdated blocks**
   → Flag only those blocks tied to changed tags.
3. **Inject updated memory content**
   → Reconstruct block content using updated value_text.
4. **Recalculate token budget**
   → Ensure rebuilt prompt fits within the model's token limits.
5. **Update prompt chain version**
   → Increment version number; mark as REFRESHED.

📜 Example Block Refresh Event

```
{
 "block_type": "compliance_profile",
 "old_version": "v1.0",
 "new_version": "v1.1",
 "triggered_by": "memory_patch:gdpr_scope",
 "updated_tags": ["data_compliance", "data_storage_policy"]
}
```

🧩 Partial vs. Full Chain Refresh

Refresh Mode	Trigger	Behavior
Partial	Only some tags/blocks changed	Rebuild affected blocks only

Full	Agent has major input change, or validator re-run	Rebuild entire chain from scratch
Simulated Preview	Admin clicks “Simulate Prompt” in FSD_09	Rebuilds live prompt in preview sandbox

---

🔐 Prompt Refresh Safeguards

Rule	Description
Do not reuse prompt blocks with outdated tag references	Token conflict risk
Always re-calculate estimated token count	Prevents runtime truncation
Only include memory with confidence ≥ 0.75	Avoids hallucination from weak sources
Prompt refresh must preserve tone and agent style	Tone is pulled from onboarding config

---

📎 References:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – controls structure, block types, and injection order
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – determines which tags map to prompt blocks
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – provides new segment triggers and block anchors
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to preview and validate refreshed prompts

---

🔷 Section 6: Validator Re-run Conditions

This section defines **when the Readiness Validator (FSD_05)** must be re-triggered after a post-launch update, to ensure the agent still meets the minimum launch and operational quality thresholds.

🎯 Purpose of Post-Launch Validation

- Ensure the **integrity** of the agent after updates
- Detect regressions in tag coverage or segment quality
- Automatically catch critical gaps introduced by edits or removals
- Block agents from continuing if they become non-compliant

🔄 Validator Re-run Triggers

Trigger	Re-run Validator?	Notes
✅ **Critical tag edited or removed**	Yes	Required for launch-critical tag list (e.g., pricing, compliance)
✅ **New documents introduce conflicting info**	Yes	May affect tag confidence and memory clarity
✅ **Segment invalidation impacts current memory**	Yes	If memory depends on now-expired document
⚠️ **Optional tag updated**	No	Unless agent config requires strict tag coverage
⚙️ **Admin requests manual revalidation**	Yes	Triggered via FSD_09 dashboard
🚫 **Minor UI-only corrections**	No	Label edits, token trims, tone rephrasing only

---

🧠 Revalidation Logic Path

1. **Detect change impact on tag map**
2. **Recalculate tag coverage & confidence**
3. **Re-run scoring algorithm (FSD_05 Section 4)**
4. **Update readiness_score, status, and failure reasons (if any)**
5. **Push report to admin dashboard (FSD_09)**

📜 Sample Revalidation Result


```
{
 "agent_id": "AGENT_0042",
 "revalidation_score": 77,
 "status": "FAIL",
 "reason": ["payout_policy now undefined after document removal"],
 "required_action": "Answer onboarding Q7 or upload updated payout policy",
 "timestamp": "2025-06-11T04:55:00Z"
}
```

🔐 Safety Constraints

Constraint	Behavior
Agent enters “validation required” state	Pauses agent in production until override or update
Admin override used during revalidation	Triggers auto-flag for audit trail
Score drops >15 points from previous score	Escalation suggested via FSD_09 dashboard

Validator error (e.g., corrupted tag map)	Retries allowed once; fallback to admin-only launch gate

---

📎 **References**:

- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – scoring logic and threshold definitions
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – defines tag criticality per agent type
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – includes per-partner launch criteria (strict vs. flexible)
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – controls dependency between prompt blocks and tag validity

---

🔷 Section 7: Role-Based Access & Permissions

This section defines **who is allowed to update agent knowledge**, under what circumstances, and what actions are restricted to internal administrators or authorized partner reviewers.

👥 User Roles & Update Capabilities

Role	Can Upload Docs	Can Edit Answers	Can Trigger Refresh	Can Modify Memory	Can Override Prompt Blocks
SMB End User	✅	✅	🚫 (auto-triggered only)	🚫	🚫
Partner Consultant	✅	✅	✅ (non-critical updates)	⚠️ (propose only)	🚫

Audira Admin	✅	✅	✅	✅ (full memory access)	✅
QA Team	🚫	🚫	🚫	⚠️ (submit feedback only)	🚫
System (LLM Router / Validator)	✅ (read access)	🚫	✅ (auto-trigger)	⚠️ (with confidence threshold)	⚠️ (flag only)

🔐 Restricted Actions

Action	Who Can Perform	Justification Required
Force refresh of prompt chain	Admin only	No
Bypass validator after update	Admin only	✅ Yes
Freeze memory block	Admin only	✅ Yes
Approve memory overwrite with conflict	Admin or Partner (with role=editor)	✅ Yes
Upload tagged legal/compliance files	Admin or Trusted Partner	Optional

---

🧠 Soft vs. Hard Permissions

Type	Description
Soft Permissions	Action creates suggestion (e.g., partner edits memory → sends request to admin)
Hard Permissions	Action executes immediately (e.g., admin updates memory block)

🔐 Approval Chains for High-Risk Tags

Certain tags (e.g., financial_policy, legal_structure, tax_reporting) are designated as **high-risk**. Updates to these trigger a **dual-review process**:

- Action must be approved by:
    - Internal admin OR
    - Trusted partner AND Audira validator module

📊 Agent Owner Visibility Controls

Visibility Rule	Behavior
Memory change logs visible to agent creator	✅ Enabled
Prompt refresh logs visible to partner consultant	✅ Enabled
Feedback-triggered retraining history visible to user	🚫 Disabled (admin only)
Version history exports (PDF or JSON)	✅ Admin + Partner only

📎 **References**:

- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – defines user roles and partner tier permissions
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – used to validate permissions for prompt overwrite
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – specifies editable answers vs. locked system tags
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – includes tag sensitivity level for override rules

---

🔷 Section 8: Future Enhancements

This section outlines upcoming capabilities to further automate, audit, and optimize the agent update cycle — transforming updates from manual corrections into proactive, AI-assisted knowledge alignment.

🚀 Planned Enhancements

Feature	Description	Value
Auto-Refresh Suggestions	AI proactively recommends memory or prompt refresh when new data is detected	Increases update frequency and reduces admin effort
Time-Based Memory Decay Logic	Flags or expires memory blocks based on age or staleness	Ensures long-term accuracy
Continuous Prompt Health Monitor	Tracks prompt chain alignment with memory and flags inconsistencies	Prevents drift between facts and prompt narrative
User-Side Prompt Comparison Tool	Shows end users a “before vs. after” view of memory or prompt content	Boosts trust and explainability
Knowledge Update Sandbox Mode	Allows safe preview/testing of refresh effects before pushing live	Helps partners and admins validate major updates
Approval Workflow for Regulated Tags	Adds multi-stage signoff flow for updates to tags like tax_policy, KYC, medical_compliance	Enhances auditability and reduces liability
Delta-Only Prompt Compiler	Rebuilds only affected blocks of prompt rather than full regeneration	Saves compute cost and preserves continuity
Versioned Knowledge Export	Lets agents export full memory and prompt state (JSON or PDF) per version	Enables audit, retraining, and third-party reviews

---

🧰 Compatible OSS Tools

Tool	Usage
LangGraph / LangFlow	Multi-step update workflows and refresh triggers
TruLens / DeepEval	Evaluating impact of updated memory on output quality
PromptLayer	Version-tracking for before/after prompt comparisons
Label Studio / Weights & Biases	Annotating memory edits and retraining triggers
Unstructured.io	Supports dynamic file parsing for time-triggered memory validation

---

📎 Linked Modules:

- 📘 *AUDIRA PRODUCT BLUEPRINT* – defines continuous learning and lifecycle automation goals
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – dependency logic between memory and prompt refresh
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to validate new prompt/memory states before final push
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – used to re-check updated agent readiness post-refresh

---