📄 FSD_06 – Prompt Chain Compiler

🧠 Purpose:

The **Prompt Chain Compiler** is the engine that constructs the final, structured LLM prompt(s) used by each Audira agent during live interaction. It pulls from:

- Discovery tags (covered + qualified)
- Onboarding answers
- Document segments
- Dynamic qualification responses
- Agent intent profile

Its output is a **contextualized prompt chain** — serialized, structured, and ready for runtime consumption by Audira's response engine (FSD_08).

Think of it as the *"brain wiring harness"* that connects the SMB's real business data into effective, safe, and tailored LLM prompts.

🧩 FSD_06 – Section Breakdown

Section	Description
1. Scope	What this module assembles, and its downstream role
2. Input Requirements	Required data from earlier FSDs
3. Prompt Construction Logic	How the prompt chain is composed
4. Modular Prompt Types	The different prompt types per use case
5. Output Format	Structured schema for each compiled prompt
6. LLM Compatibility & Token Limits	Prompt sizing, truncation, and safety features
7. Runtime Handoff	How the compiled prompt connects to live agent logic
8. Future Enhancements	Chain templates, multi-model routing, style adaptation

---

✅ Section 1: Scope

🎯 Purpose:

This module builds the full prompt chain — a structured sequence of instruction + context + semantic grounding — that will be passed to the live agent (FSD_08). It ensures that every piece of knowledge collected during onboarding becomes usable for answering end-user questions.

🧾 **Responsibilities:**

- Compile user-specific business knowledge into LLM-friendly input format
- Apply tag-to-prompt mapping logic using the discovery tag dictionary
- Inject relevant answers, context blocks, and segment citations
- Create use-case specific prompt variants (e.g. Q&A, summarization, compliance, sales pitch)
- Respect token limits and safety filters
- Output serialized prompt objects ready for use in runtime environments

❌ **Not in Scope:**

- LLM calling or inference execution (handled in FSD_08)
- Question generation (FSD_04 handles this)
- Readiness validation logic (FSD_05 already covers that)

📎 **References**:

- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – links each tag to prompt logic
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – source of fixed answers and agent role
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines final chain layout
- 📘 *AUDIRA FILE & SEGMENT MAP* – provides segment anchors and token-level metadata
- 📘 *AUDIRA DYNAMIC QUALIFICATION RESPONSES* – used for refined prompt enrichment

---

🔷 Section 3: Prompt Construction Logic

This section defines **how prompt chains are constructed**, merging discovery tags, onboarding data, document segments, and clarified insights into **LLM-compatible instructions** with stable ordering, token control, and usage-specific structure.

🎯 Objective:

To compile **modular, semantically complete prompt chains** that reflect:

- Business structure
- Operational logic
- User tone and context
- Fallback behavior for missing or low-confidence data

🧠 Prompt Chain Logic – Core Flow

Step 1: Select Valid Tags

From the tag map:

- Include tags where:
  - `status == covered` OR `status == clarified`
  - AND `confidence ≥ 0.75`
- Exclude tags with unresolved conflict flags (from FSD_05)

Step 2: Sort Tags by Priority

Using `AUDIRA DISCOVERY TAGS DICTIONARY`:

- Order: `core → advanced → optional`
- Tags are grouped by category (e.g., `business_model`, `compliance`, `support_model`)

Step 3: Generate Prompt Fragments

For each selected tag:

Input	Output
`tag_id`	Injected into base instruction
`template_prompt`	Used directly, or
`fallback_prompt`	Generated via LLM if template missing

`context`	Pulled from onboarding answers or segment text
`style`	Applied based on agent config or tag metadata

Example Fragment:

```
The business earns revenue primarily through: subscriptions sold to SMEs via
Stripe. Payments are processed automatically every 30 days.
```

Step 4: Group into Prompt Blocks

Based on `agent intent profile`, tags are grouped into use-case blocks:

Prompt Block	Included Tags
Business Summary	business_model, revenue_model, product_type
Vendor Logic	vendor_model, payout_policy, partner_integration
Support Policies	return_policy, customer_support, SLA
Compliance Profile	data_compliance, KYC_method, tax_reporting

Each block has a **header**, **merged context**, and a **fallback clause** if any required tags are weak.

Step 5: Compile Final Prompt Chain

Final structure:

```
{
  "prompt_chain": [
    {
      "block_type": "business_summary",
      "content": "The business earns revenue primarily through subscriptions.
Products are digital tools for SME finance. Payments are processed via
Stripe.",
      "tags_covered": ["revenue_model", "product_type", "payment_method"]
    },
    {
      "block_type": "compliance_profile",
      "content": "The company complies with GDPR and supports CCPA where
applicable. KYC is performed via Jumio APIs.",
      "tags_covered": ["data_compliance", "KYC_method"]
    }
  ],
  "language": "English",
  "style": "consultative",
  "total_token_estimate": 873
}
```

🔐 Prompt Safety Controls:

- **Token Budgeting**: Uses token estimation (based on segment length + prompt size) to enforce model-specific max tokens
- **Tag Gaps**: If priority tag is missing, insert comment like:

```
<!-- Missing payout_policy details – fallback enabled -->
```

- **De-duplication**: Each tag appears in only one prompt block
- **Normalization**: Converts mixed sources (answers + files) into unified tone using sentence standardizer

---

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines final block format and sequencing
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – provides per-tag prompt instructions
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – provides context segment text and token counts
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – guides conflict handling and fallback cases

---

💠 Section 4: Modular Prompt Types

This section outlines the **types of prompt chains** Audira generates depending on the agent's use case. Each type reflects a **different LLM behavior goal**, shaped by the selected capabilities during onboarding.

🧩 *Prompt Type Categories*

Prompt Type	Purpose	Used When Agent Supports…
Q&A Profile Prompt	Provide fact-based answers grounded in user-specific knowledge	General Assistant, Support Bot, Compliance Helper
Summarization Prompt	Summarize business structure or documents for downstream API use	Report Generator, Partner Profile Sharing
Compliance Prompt	Ensure answers reflect regulatory context and policies	GDPR, KYC, Audit Assistants
Sales Context Prompt	Frame the business offering in persuasive tone	Outbound Bots, Discovery Tools, B2B Showcases
Response Guardrail Prompt	Limit model replies to verified inputs only	High-risk use cases (Finance, Legal, HR)

Q&A Profile Prompt

```
Use the following verified business profile to answer questions from users:
- Business model: Subscription-based digital tools for SMEs
- Payment method: Stripe (monthly recurring)
- Vendor payout: Automated via Stripe every 14 days
Only answer from this knowledge. Do not fabricate information.
```

Summarization Prompt

```
Generate a concise overview of the business:
- What it sells
- How it charges
- Who it serves
- Key operational details
```

Compliance Prompt

```
When generating responses, prioritize legal alignment:
- GDPR, CCPA, and PCI compliance are enabled
- KYC is performed via Jumio
- Tax handled via external CPA
Avoid advice. Summarize policy-level declarations only.
```

Sales Context Prompt

```
Act as a digital sales rep. Highlight:
- Benefits of the product (SME time savings)
- Trusted partners (Stripe, HubSpot)
- Pricing tiers (Starter, Pro, Enterprise)
Use a persuasive yet factual tone. Include selling points clearly.
```

Response Guardrail Prompt

```
Only use the verified data below to generate answers.
If unsure, respond with: "I'm not able to confirm that yet."
Do not speculate, assume, or fabricate.
```

Selection Logic:

Prompt types are triggered based on:

- Agent config (from onboarding)
- Use-case declared during setup
- Category-to-type mapping in the blueprint logic

Multiple prompt types may be compiled per agent and switched via runtime logic in FSD_08.

References:

- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – identifies intent and use-case
- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – outlines prompt switching strategy
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – connects prompt types to downstream APIs
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – determines tone and style for prompts

---

📐 Section 5: Output Format

This section defines the **serialized structure** of the compiled prompt chain that is passed to the LLM runtime (FSD_08) and optionally stored for replay, simulation, and debugging.

📦 Unified Prompt Chain Object Structure

Each compiled prompt is stored as a **versioned object**, grouped by usage type and containing all necessary metadata for downstream use, including agent runtime control, segment traceability, and model-specific adjustments.

🧬 JSON Schema – Full Prompt Chain

```
{
  "agent_id": "AGENT_01234",
  "version": "1.0.0",
  "language": "English",
  "style": "consultative",
  "prompt_chain": [
    {
      "block_type": "business_summary",
      "content": "The business offers cloud-based tools to SMEs, charges
monthly via Stripe, and delivers services through a self-serve portal.",
      "tags_covered": ["revenue_model", "product_type", "payment_method"],
      "source_segments": ["seg_15", "seg_18"],
      "confidence_avg": 0.89,
      "token_estimate": 137
    },
    {
      "block_type": "compliance_profile",
      "content": "The company complies with GDPR and offers CCPA upon
request. KYC is handled via Jumio integrations.",
      "tags_covered": ["data_compliance", "KYC_method"],
      "source_segments": ["seg_25"],
      "confidence_avg": 0.93,
      "token_estimate": 102
    }
  ],
  "total_estimated_tokens": 845,
  "fallbacks_applied": [
    "Missing:pricing_strategy - used fallback wording.",
    "Missing:employee_count - excluded from prompt."
  ],
  "guardrails": {
    "allow_hallucination": false,
    "max_tokens_per_block": 512,
    "trust_confidence_threshold": 0.75
  }
}
```

🔍 Key Fields:

Field	Description
`agent_id`	Unique identifier of the configured agent
`version`	Version of the prompt chain compiler logic
`language`	Prompt language (e.g., English, Arabic)
`style`	Tone or style (e.g., consultative, sales, neutral)
`prompt_chain`	List of modular blocks with content and tag coverage
`source_segments`	Segment IDs that directly influenced the block
`confidence_avg`	Mean confidence score of covered tags
`fallbacks_applied`	Record of missing tags or applied safety substitutes
`guardrails`	Instructions to LLM runtime on usage constraints
`total_estimated_tokens`	Sum of all token counts for runtime planning

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines format and runtime contract
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – defines source_segments and token structure
- 📘 *AUDIRA AGENT ONBOARDING FRAMEWORK* – supports language, style, and intent matching
- 📘 *AUDIRA DISCOVERY TAGS DICTIONARY* – maps tags to categories and fallback triggers

---

💠 Section 6: LLM Compatibility & Token Limits

This section ensures that all compiled prompt chains remain **model-compatible**, respect **token constraints**, and deliver **safe, controlled results** across any deployed LLM (e.g., Open Source, Claude, GPT, Mixtral, etc.).

🎯 *Goals:*

- Prevent runtime truncation or failure due to excessive input
- Adapt prompts to different model capacities
- Enforce trust boundaries for factual integrity and cost efficiency

---

📐 Token Sizing Logic

🧮 *Per-Block Estimation*

Each prompt block includes:

- **Static overhead** (header, instruction): ~25 tokens
- **Content size**: Based on word/token count of merged segments and answers
- **LLM-specific buffer**: Reserved to allow for generation headroom

```
block_token_estimate = base_tokens + len(content.split()) * 1.3 +
model_buffer
```

🧠 *Total Prompt Budget*

LLM Model	Max Tokens	Target Max Input	Reserved for Response
GPT-4-turbo	128k	100k	28k
Claude 3 Opus	200k	180k	20k
Mixtral 8x7B	32k	28k	4k
LLaMA2-13B	8k	6.5k	1.5k

🛡️ Truncation & Fallback Rules

Scenario	Strategy
Prompt exceeds model limit	Drop optional blocks, compress advanced-tag content
Token estimate exceeds dynamic budget	Re-rank blocks by tag priority and drop lowest
Critical segment too long	Use `summary_segment()` to create compressed block
LLM warning (context length exceeded)	Rerun compiler with reduced target size (retry mode)

Safety Controls

Guardrail	Description
`allow_hallucination: false`	Prevents prompt from encouraging guesswork
`min_confidence_to_include: 0.75`	Ensures low-certainty tags don't enter prompt
`block_max_tokens: 512`	Prevents any single block from dominating the chain
`segment_verification_required: true`	Forces prompt to cite verified segment_id if available

Deployment Notes:

- Token budgeting logic lives inside the **Prompt Compiler Engine**, not the runtime agent
- Overrides allowed per model via `AUDIRA INTEGRATION SCAFFOLDS GUIDE`
- Prompt chain versions are annotated with `max_token_compat` for audit purposes

References:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines prompt-to-model interface
- 📘 *AUDIRA FILE & DATA UPLOAD SCHEMA* – provides segment-level token metadata
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – defines per-model prompt budgets
- 📘 *AUDIRA PRE-LAUNCH VALIDATOR SPEC* – contributes confidence cutoff logic

Section 7: Runtime Handoff

This section defines how the compiled prompt chain is delivered to the **live agent system** at runtime, enabling LLM-powered interactions that are personalized, grounded, and safe.

Handoff Pathways

There are two runtime pathways depending on agent type:

Agent Mode	Handoff Target	Trigger
Self-serve SMB	Agent API → Prompt Interpreter Module (FSD_08)	On agent launch or retrigger

Partner-embedded	Partner Middleware or White-labeled Runtime	On demand via API call or iframe embed

---

Handoff Payload Format

```
{
  "agent_id": "AGENT_01234",
  "compiled_prompt": { ... },  // full output from Section 5
  "use_case": "general_assistant",
  "model_hint": "gpt-4-turbo",
  "guardrails": {
    "max_tokens": 100000,
    "block_limit": 512,
    "reject_low_confidence": true
  },
  "metadata": {
    "compiled_by": "PromptCompiler_v1.2",
    "timestamp": "2025-06-11T02:19:00Z",
    "prompt_version": "1.0.0"
  }
}
```

Agent Runtime Usage (FSD_08 Dependency)

The runtime engine receives the compiled prompt and:

1. **Parses the modular blocks** by type (e.g., business_summary, compliance_profile)
2. **Selects** the appropriate blocks based on query type (e.g., Q&A vs. sales)
3. **Applies pre-answer filters**, such as guardrail checks
4. **Injects the prompt** into the selected LLM
5. **Maps the output** to agent behavior (response, link, summary, etc.)

Logging and Traceability

Each handoff generates a **Prompt Use Log**, including:

Log Field	Description
`agent_id`	Agent that used the prompt
`prompt_version`	Version of compiled prompt logic
`model_used`	Actual model at runtime
`tokens_sent`	Token count of prompt block used
`blocks_invoked`	List of prompt types consumed
`fallback_used`	Boolean flag if fallbacks were triggered

This enables **auditing, replay, and fine-tuning** of prompt performance and agent responses.

📎 **References**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – defines runtime-to-LLM connection
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – provides agent use-case and runtime config
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – shows external handoff methods
- 📘 *AUDIRA PRODUCT BLUEPRINT* – defines LLM agent runtime flow

---

💠 Section 8: Future Enhancements

This section outlines strategic improvements and roadmap features that will evolve the Prompt Chain Compiler into a **dynamic, context-aware orchestration layer** for real-time LLM prompt assembly.

🚀 Planned Enhancements

Feature	Description	Benefit
Chain Templates per Agent Type	Pre-defined prompt templates for sales, support, legal, etc.	Faster prompt construction, tuned by intent
Adaptive Prompt Reshaping	Dynamically restructure prompts based on past response quality or feedback	Improves long-term agent performance
Multilingual Prompt Generation	Generate prompt chains in Arabic, French, etc., based on agent profile	Enables localization for global SMBs
Prompt Chain Caching & Delta Updates	Regenerate only changed blocks after a document update	Saves computation and preserves history
Context-Aware Prompt Routing	Use OpenRouter/LangGraph-style logic to select different prompt paths based on question intent	Modularizes response logic for specialized flows
Embedded Evaluation Metrics	Insert pre/post quality hooks (e.g., LLMScore, DeepEval) into prompt compiler	Enables automated tuning and regression checks
User Style Mimicry	Adjust prompt tone (casual, professional, bold) based on onboarding inputs or previous writing	Improves brand alignment and engagement
Chain-of-Thought Injection	Optionally add reasoning scaffolds to blocks for better output quality	Supports complex decision-making agents

Model-Specific Optimization Profiles	Fine-tuned prompt layouts for different LLMs (e.g., Mixtral vs. GPT-4)	Maximizes accuracy and token efficiency per model

---

🧰 Compatible Open Tools (MIT / OSS)

Tool	Usage
LangGraph	Prompt chain routing, fallback handling
TruLens / DeepEval	Prompt quality scoring and regression testing
PromptLayer / PromptTools	Version control and experimentation
Haystack / LlamaIndex	Segment injection and source grounding
Unstructured.io	Block-level document parsing (FSD_02 input prep)

📎 **Linked Modules**:

- 📘 *AUDIRA PROMPT CHAIN & LLM LOGIC FLOW* – core architecture link
- 📘 *AUDIRA AGENT BLUEPRINT TEMPLATE* – used for templated tone logic
- 📘 *AUDIRA AGENT SIMULATION TEST KIT* – used to evaluate outputs from different chain styles
- 📘 *AUDIRA INTEGRATION SCAFFOLDS GUIDE* – LLM-type routing and token profile optimization

---