下面把“**五个模块 + 1 个 LLMs Router**”之间通信所依赖的**抽象接口与契约**按文件 → 接口/数据模型列清楚（只列抽象层；实现类另外附带指向）：

---

1. 抽象接口（Protocols）

- **app/core/interfaces.py**
  - `GraphConstruction`: `build(GraphBuildIn) -> GraphBuildOut`（图构建模块统一入口）。
  - `RetrievalAgent`: `retrieve(RetrievalIn) -> RetrievalOut`（检索模块统一入口）。
  - `ReasoningAgent`: `reason(ReasoningIn) -> ReasoningOut`（推理模块统一入口）。
  - `VerifierAgent`: `verify(VerifyIn) -> VerifyOut`（验证模块统一入口）。
- **app/core/providers/base.py**
  - `LLMProvider`:
    - `complete(model, prompt, require) -> str`
    - `embed(model, texts, require) -> List[List[float]]`
      （Router 与底层提供方的抽象契约；OpenAI/Ollama 等具体实现都遵守它）。
- **app/telemetry/sinks.py**
  - `TelemetrySink`:
    - `record(TelemetryEvent) -> None`
    - `flush_run(trace_id, result) -> None`
      （可选注入；用于模块/Router 的日志采集与运行快照落盘）。

说明：**Orchestrator（LangGraph）本身不定义新接口**，而是通过 `NodeContext` 注入这些接口实例来编排调用；`NodeContext` 是组合对象，不是抽象协议。

---

2. 数据契约（DTO / Schemas）

- **app/core/dto.py**
  - 图构建：`GraphBuildIn`, `GraphBuildOut`
  - 检索：`RetrievalIn`, `RetrievalOut`, `Hit`
  - 推理：`ReasoningIn`, `ReasoningOut`
  - 验证：`VerifyIn`, `VerifyOut`
    模块之间**只通过这些 DTO 传参/回传**，实现了解耦。
- **app/schemas/graph_request_v2.py**（用于新版图装配场景的结构化输入）
  `AssembleGraphRequestV2` 及其子模型（`Sentence/Inputs/Provenance`）。

---

3. Router 的对外接口（模块→Router）与对内接口（Router→Provider）

- **app/core/llm_router.py**（Router 自身是稳定 API，而非 Protocol）
  - 对模块暴露：
    - `complete(module, purpose, prompt, require) -> Dict`
    - `embed(model_hint, texts, require) -> List[List[float]]`
      模块在需要 LLM 时**只调用 Router**，不依赖具体 Provider。
  - Router 内部再调用 **`LLMProvider`**（见上）完成实际推理/向量化。
  - 可选接入 **`TelemetrySink`** 记录 `llm_call`。
- **Provider 的具体实现（遵循 LLMProvider）**
  - OpenAI：**app/core/providers/openai_provider.py**（`LLMProvider` 实现）。
  - Ollama：**app/core/providers/ollama_provider.py**（`LLMProvider` 实现）。

---

4. 实现类（参考对应哪些接口）

下列实现类通过 **Protocol + DTO** 与编排/其它模块交互：

- **GraphConstruction 实现**
  - Flow：`app/modules/graph_construction/flow.py` → 实现 `GraphConstruction.build(...)`。
  - Impl：`app/modules/graph_construction/impl_networkx.py` → 亦实现同接口用于落盘。
- **RetrievalAgent 实现**
  - Flow：`app/modules/retrieval/flow.py`。
  - Impl：`app/modules/retrieval/impl_hybrid.py`。
- **ReasoningAgent 实现**
  - Flow：`app/modules/reasoning/flow.py`。
  - Impl：`app/modules/reasoning/impl_planner_synth.py`。
- **VerifierAgent 实现**
  - Flow：`app/modules/verification/flow.py`。
  - Impl：`app/modules/verification/impl_rules_llm.py`。

---

1. 一句话总括

- **抽象接口文件**：`app/core/interfaces.py`（四大业务接口）、`app/core/providers/base.py`（LLMProvider）、`app/telemetry/sinks.py`（TelemetrySink）。
- **数据契约文件**：`app/core/dto.py`（跨模块 I/O）、`app/schemas/graph_request_v2.py`（图装配输入）。
- **Router 对外 API**：`app/core/llm_router.py`（模块→Router），对内通过 `LLMProvider` 调用各 Provider。

这些正是五个模块与 LLMs Router **彼此独立且仅依赖抽象接口/DTO** 的边界。