Jarvis Planner – Task Graph & DSL Blueprint

This document defines the planning core of the Jarvis system. It specifies how intents are transformed into executable task graphs using a deterministic, auditable Domain-Specific Language (DSL). This is a design-level blueprint, not an implementation guide.

1. Planner Purpose

The planner is responsible for converting user goals into structured, multi-step execution plans. Its output is a task graph, not a script or command list. The graph enables recovery, partial execution, replanning, and learning.

2. Task Graph Model

A task graph is a directed graph where nodes represent tasks and edges represent execution dependencies. Graphs are typically acyclic but may contain controlled loops for polling or retries.

3. Task Node Schema

```
{
   "task_id": "open_project_folder",
   "type": "action | decision | loop | composite",
   "description": "Open the main project directory",
   "inputs": { "path": "~/projects/jarvis" },
   "preconditions": ["path_exists"],
   "postconditions": ["folder_opened"],
   "on_fail": "retry | skip | replan | abort",
   "retries": 1, "risk": "low | medium | high",
   "controller_action": "open_folder"
}
```

4. Task Types

- **Action Task:** Atomic, single controller call, no branching or logic.
- **Decision Task:** Evaluates state and selects execution path.
- **Composite Task:** Reusable subgraph representing higher-level goals.
- **Loop Task:** Controlled repetition for waiting or polling (use sparingly).

5. DSL Design Principles

The DSL is declarative, serializable, deterministic, and controller-agnostic. It prevents ad-hoc logic and enables replay, auditing, and learning.

6. DSL Structure Example

```
task_graph:

  name: prepare_work_environment
  version: "1.0"
  entry: check_workspace

  tasks:

    check_workspace:
        type: decision
        condition: workspace_ready
        on_true: done
        on_false: open_project_folder

    open_project_folder:
        type: action
        controller: open_folder
        args:
            path: "~/projects/jarvis"
        on_success: launch_vscode
        on_failure: abort

    launch_vscode:
        type: action
        controller: launch_app
        args:
            app: vscode
        retries: 1
        risk: low
        on_success: done

    done:
        type: action
        controller: notify
        args:
            message: "Workspace ready"
```

7. Preconditions & Postconditions

Preconditions are validated before task execution to avoid redundant or unsafe actions. Postconditions verify task success and enable recovery or replanning.

8. Failure Semantics

Every task declares explicit failure behavior: retry, skip, replan, or abort. Implicit failure handling is prohibited.

9. Learning Integration

```
{
   "task_graph": "prepare_work_environment",
   "success": true,
   "failures": [],
   "execution_time": 12.4,
   "user_interrupted": false
}
```

Planner outcomes are stored and used to bias future graph selection, reduce risk, and improve efficiency.