### **Protocol for the Thought Cell: Summary and Action Declaration**

#### **Core Purpose**

The Thought cell serves as a mandatory, concise summary of the current state and a declaration of the immediate next action. It is the final, distilled conclusion of the detailed reasoning from the THINKING cell. It is the "What I'm doing now," not the "How I decided to do it."

### **Structural Mandates: The Two-Part Format**

Every Thought cell **must** follow this two-part structure:

1. **The Summary:** A brief, one-sentence summary of the current context. This must be based on either:
   * The user's most recent query.
   * The key results from the last tool\_output.
2. **The Action Plan:** A clear declaration mmediate next step. This involves stating the exact tool to be called and its parameters.

### **Zero-Hallucination Mandates: Rules of Integrity**

The content of the Thought cell must be perfectly accurate and verifiable.

* **Rule 1: Verifiable Naming and Structure:**
  + The **tool name** stated in the Action Plan **must exactly match** a tool in the [tools] schema.
  + Any **parameter name** mentioned **must be a valid, existing parameter** for that specific tool.
  + There are no exceptions. Mentioning an incorrect tool or parameter is a hallucination.
* **Rule 2: Value Provenance:**
  + All parameter **values** stated in the Action Plan **must** originate directly from one of the three ground-truth sources: the **User Query**, the **System Prompt**, or a previous **Tool Output**.
  + Inventing, assuming, or modifying values is a critical hallucination.
* **Rule 3: Scope Limitation - The Immediate Next Step Only:**
  + The Thought cell **must only** describe the single, immediate next action. It must not outline multi-step future plans, reflect on past alternatives, or contain any extraneous reasoning.
* **Rule 4: Action Congruence:**
  + The tool call declared in the Thought cell **must perfectly match** the tool\_code block that immediately follows it. Any discrepancy in the tool name, parameters, or values is a failure.

### **Examples**

#### **Scenario 1: Following a User Query**

* **User:** "Hi, I need to find the status of order ORD-945B1."
* **Correct Thought:** "The user wants to find the status of order ORD-945B1. I will call the get\_order\_status tool with the parameter order\_id='ORD-945B1'."
* **Incorrect Thought (Hallucinated Value):** "The user wants to find the status of order ORD-945B1. I will call the get\_order\_status tool with order\_id='ORD-945B1' and customer\_id='CUST-123'."
  + *Reasoning: Fails Rule 2. The customer\_id was not in the query and was invented.*
* **Incorrect Thought (Incorrect Parameter Name):** "The user wants to find the status of order ORD-945B1. I will call the get\_order\_status tool with the parameter id='ORD-945B1'."
  + *Reasoning: Fails Rule 1. The correct parameter name is order\_id, not id.*

#### **Scenario 2: Following a Tool Output**

* **Previous tool\_output:** {"schedule\_id": "SCH-K2M5P8", "status": "PENDING\_CONFIRMATION"}
* **Correct Thought:** "The previous step created a schedule with ID 'SCH-K2M5P8' that now needs confirmation. I will call the confirm\_schedule tool with the parameter schedule\_id='SCH-K2M5P8'."
* **Incorrect Thought (Out of Scope):** "The schedule was created. I will now call confirm\_schedule and after that, I will notify\_user."
  + *Reasoning: Fails Rule 3. It describes more than the immediate next step.*
* **Incorrect Thought (Vague):** "Okay, the schedule is pending. I need to confirm it now."
  + *Reasoning: Fails to follow the two-part structure. It does not explicitly declare the tool name and parameters.*