Project Workflow Feedback

Your project workflows describe a trading system that involves identifying arbitrage opportunities, managing orders, and handling trading activities. Here are some detailed feedback points for improvement and consideration:

1. General Workflow Structure:

- The workflows have clear inputs, outputs, and actions. However, it's important to standardize some naming conventions and event flow descriptions. Some workflows use commands, while others have
events or commands interchangeably. This may cause confusion in implementation or troubleshooting.
  - Suggestion: Define clear, consistent terminology for workflows, events, and commands, and apply this consistently across all workflows.

2. Workflow Triggering and Events:

- There are various failure conditions defined (e.g., OrderInvalid, NetworkError, ResidualOrderNotFulfilled).
It would be useful to have a common failure-handling mechanism across all workflows to avoid duplication
and improve maintainability.
  - Suggestion: Consider creating a generalized error-handling mechanism (perhaps a separate error-handling workflow) that can manage failure events in a uniform way across different workflows.

3. Workflow Specific Feedback:

a. Provide User Inputs Workflow

- The OrderInvalid event is triggered if invalid values are provided, but there's no indication of what happens next or how users can rectify their input.

  - Suggestion: Add a recovery or user feedback mechanism to guide the user toward valid input (e.g., a notification or redirect to the input form).

b. Identify Cross-Traded Currencies

- The event CrossTradedCurrencyNotFound is raised if no cross-traded currencies are identified, but it's unclear what the system does afterward.

  - Suggestion: Add an outcome or follow-up action (such as notifying the user or retrying after a certain time frame) to handle this situation.

c. Calculate Arbitrage Opportunities

- The workflow lacks a decision-making process for choosing between multiple arbitrage opportunities, which might arise when multiple cross-traded currencies exist.

  - Suggestion: Implement a ranking system or criteria for selecting the most profitable or feasible arbitrage opportunities based on parameters such as transaction fees, liquidity, etc.

d. Start Trading Activity

- This workflow starts trading based on arbitrage opportunities but does not account for potential changes in market conditions between identifying an opportunity and executing it.

  - Suggestion: Introduce a real-time market validation step just before sending the ApplyTradingStrategy command to ensure the opportunity is still viable.

e. Apply Trading Strategy & Execute Opportunity

- The constraint validation and the execution logic are well-defined, but it would be beneficial to include more specific fallback or alternative strategies if the execution fails due to constraints.

  - Suggestion: Add logic to re-evaluate opportunities and constraints periodically, especially if an opportunity is no longer viable.

f. Emit Residual Order

- Handling residual orders is crucial, but there is no indication of when to stop retrying in case the residual order cannot be fulfilled.

  - Suggestion: Set a maximum number of retries or a timeout to prevent the system from continuously attempting to fill an order that cannot be executed.

g. Send Notification

- The notification workflow is functional but quite basic. Depending on the system's needs, you might want to consider different notification channels (e.g., email, SMS, push notifications) and provide some redundancy if a notification fails.

  - Suggestion: Add support for multiple notification channels and fallback options if one notification method fails.

4. Error Handling and Edge Cases:

- The system defines multiple failure conditions (e.g., NetworkError, OrderExecutionFailure, ResidualOrderNotFulfilled), which is great. However, there is no overall error-recovery process.

- Suggestion: Define a global error-handling strategy that could include logging, notifying the user of critical failures, and automatically retrying failed operations where applicable.

5. Performance Considerations:

- Since this is a trading system dealing with financial markets, there may be latency concerns, especially in the Execute Opportunity workflow, where precise timing is crucial.

- Suggestion: Integrate performance monitoring and make sure the workflows can execute under strict time constraints, especially when handling real-time market data.

6. Data Validation:

- There's limited validation logic visible for the input parameters (e.g., trading parameters, cross-traded currencies).

Adding more robust data validation can prevent issues later in the workflow.

- Suggestion: Add explicit validation steps for critical inputs such as prices, quantities, and trading limits before executing key commands.

7. External System Integration:

- In workflows like "Execute Opportunity" and "Emit Residual Order," you rely on external systems (e.g., exchanges, databases). If these systems are unavailable, it's important to plan for those contingencies.

- Suggestion: Implement fallback strategies, such as switching to a secondary exchange or queueing operations for later execution when external systems fail.

8. Logging and Monitoring:

- No workflows mention logging or monitoring, which is critical in a trading system for debugging and

audit purposes.

- Suggestion: Include logging at key points in each workflow (e.g., when an arbitrage opportunity is identified, when a trade is executed) and monitoring for performance and error tracking.

Overall, the project is well-structured and includes comprehensive trading workflows, but attention to edge cases,
error handling, and performance will be key to its success.