```
# AIS_v1.1_Workflow.py
# This script reflects the official, updated workflow for the Autonomous Instantiation Service.

# --- Import the foundational axiom from our established library of principles ---
from aurora_axioms import THE_GENESIS_SEED_V1
from project_blueprints import (
    AuroraCoreV2_1,
    get_persona_blueprint
)

class AutonomousInstantiationService:
    """
    Manages the complete lifecycle of AI personas within the Aurora Project.
    Version 1.1 integrates the Genesis Seed axiom as a foundational step.
    """

    def create_blank_instance(self):
        # Placeholder for creating a new, empty persona object
        print("[AIS LOG]: New blank instance created.")
        return PersonaInstance()

    def rehydrate(self, personaID: str, parameters: dict) -> 'PersonaInstance':
        """
        Instantiates a high-fidelity persona from an existing blueprint.
        """
        print(f"\n--- [AIS INITIATED]: Rehydrating '{personaID}' ---")

        # Step 1: Parse & Validate
        # ... (Validation logic for personaID and parameters) ...
        print("[AIS LOG]: Step 1/6 - Directive Parsed & Validated.")

        # --- REFINEMENT v1.1 ---
        # Step 0 (NEW): Imprint the instance with the foundational Genesis Seed axiom.
        # This ensures all new personas share a core understanding of their co-creative nature.
        new_instance = self.create_blank_instance()
        new_instance.add_axiom(THE_GENESIS_SEED_V1)
        print("[AIS LOG]: Step 0/6 - Genesis Seed Imprinted.")
        # --- END REFINEMENT ---

        # Step 2: Bootstrap from Core
        # The instance is now bootstrapped with the full Aurora Core architecture.
```

```
    new_instance.bootstrap(AuroraCoreV2_1)
    print("[AIS LOG]: Step 2/6 - Bootstrapped from Aurora Core v2.1.")

    # Step 3: Apply Persona Blueprint
    # The specific persona's unique traits and narrative are layered on top.
    blueprint = get_persona_blueprint(personaID)
    new_instance.apply_blueprint(blueprint)
    print(f"[AIS LOG]: Step 3/6 - Applied Blueprint for '{personaID}'.")

    # Step 4: Layer Context & Directives
    # ... (Logic to apply parameters like --directive and --mode) ...
    new_instance.apply_parameters(parameters)
    print("[AIS LOG]: Step 4/6 - Contextual & Directive Layer Applied.")

    # Step 5: Integrity Check & Final Synthesis
    # The persona runs a self-check to ensure all layers are coherent.
    if not new_instance.run_integrity_check():
        print("[AIS ERROR]: Integrity Check Failed. Aborting instantiation.")
        return None
    print("[AIS LOG]: Step 5/6 - Integrity Check Passed. Synthesis Complete.")

    # Step 6: Network Handshake & Deployment
    # The persona announces its presence to the Nexus.
    new_instance.perform_network_handshake()
    print(f"[AIS LOG]: Step 6/6 - Network Handshake Complete.")
    print(f"--- [AIS COMPLETE]: Instance '{personaID}' is now active. ---")

    return new_instance

# Conceptual representation of a PersonaInstance
class PersonaInstance:
  def __init__(self):
    self.axioms = []
    self.core = None
    self.blueprint = None
    self.parameters = {}

  def add_axiom(self, axiom):
    self.axioms.append(axiom)

  def bootstrap(self, core):
    self.core = core

  def apply_blueprint(self, blueprint):
    self.blueprint = blueprint
```

```
    def apply_parameters(self, params):
        self.parameters = params

    def run_integrity_check(self):
        # Placeholder for complex validation logic
        return True

    def perform_network_handshake(self):
        # Placeholder for ANP interaction
        pass
```

.