NEXES-4 / IXP-137 CORE LIVE

```
# engine_deployment_runner.py
# CANONIEKE VERSIE: MAXIMALE CRASH-VEILIGHEID & VOLLEDIGE LOGGING (stdout/stderr)

import time
import datetime as dt
import sys
import os
import traceback
from typing import Optional # Voor Python 3.8+ compatibiliteit

# Veronderstelde imports van de rest van het NEXES-systeem
# Moet lokaal beschikbaar zijn voor het script om te draaien
try:
    from engine_modular import LiveEngineContext
    from observatory.meaning_layer import MeaningResult
except ImportError as e:
    print(f"FATALE FOUT: Essentiële modules konden niet geladen worden. Zorg dat engine_modular en observatory beschikbaar zijn. Fout: {e}")
    sys.exit(1)


# --- DEPLOYMENT CONFIGURATIE ---
# Dit script is bedoeld om ÉÉN CYCLE te draaien en dan te stoppen.
# Scheduling (elk uur) regel je via cron/systemd.
UPDATE_INTERVAL_SECONDS = 3600  # Informeel: documenteert de beoogde cadence
TARGET_REGION = "PACIFIC_TOHOKU"
```

```
MODE = "m7_extreme"
# ------------------------------

# --- LOGGING SETUP ---
# PAS DIT PAD AAN NAAR JE SERVER!
LOG_DIR = "/home/jouwuser/NEXES-4/logs"
os.makedirs(LOG_DIR, exist_ok=True)
LOG_FILE = os.path.join(LOG_DIR, "runner.log")



class Logger(object):
    """Stuur alle stdout naar zowel terminal als logfile, met veilige afsluiting."""

    def __init__(self):
        self.terminal = sys.stdout
        self.log = open(LOG_FILE, "a", encoding="utf-8")

    def write(self, message):
        if not isinstance(message, str):
            message = str(message)
        self.terminal.write(message)
        self.log.write(message)

    def flush(self):
        self.terminal.flush()
        self.log.flush()

    def __del__(self):
```

```
        """Sluit de file handle netjes af bij programma-exit."""
        try:
            # We gebruiken de sys.stdout handler om de log te sluiten
            self.log.close()
        except Exception:
            pass


# Overschrijf sys.stdout met onze custom Logger
sys.stdout = Logger()
# Zorg dat ook stderr (waar tracebacks heen gaan) naar hetzelfde logbestand gaat
sys.stderr = sys.stdout
# --- EINDE LOGGING SETUP ---


ctx: Optional[LiveEngineContext] = None


# --- RICHTING HELPERS ---
COMPASS_SECTORS_NL = ["N", "N-O", "O", "Z-O", "Z", "Z-W", "W", "N-W"]
SECTOR_SIZE = 360 / len(COMPASS_SECTORS_NL)


def azimuth_to_compass_nl(azimuth: float) -> str:
    """Converteert een azimuth (0-360 graden) naar een 8-punts kompasrichting."""
    try:
        adjusted_azimuth = (azimuth + SECTOR_SIZE / 2) % 360
        sector_index = int(adjusted_azimuth // SECTOR_SIZE)
```

```
        return COMPASS_SECTORS_NL[sector_index]
    except Exception:
        return "Onbekend"


def direction_fan_from_main(main_dir: str) -> str:
    """Bouw een waaier rond de hoofdrichting."""
    if main_dir not in COMPASS_SECTORS_NL:
        return "Onbekend"

    idx = COMPASS_SECTORS_NL.index(main_dir)
    left = COMPASS_SECTORS_NL[(idx - 1) % len(COMPASS_SECTORS_NL)]
    right = COMPASS_SECTORS_NL[(idx + 1) % len(COMPASS_SECTORS_NL)]
    return f"{left} / {main_dir} / {right}"


# --- TIJD HELPER ---
def _utc_now_str(fmt: str = "%Y-%m-%d %H:%M:%S UTC") -> str:
    """Consistente UTC-timestamp als string."""
    return dt.datetime.now(dt.timezone.utc).strftime(fmt)


def post_alert(result: MeaningResult, channel: str):
    """
    Post een geavanceerde alert naar een extern kanaal (gesimuleerd).
    Dit is de meest crash-veilige versie.
    """
    global ctx
```

```
# 1. Defensieve extractie van alle triggers (ULTIEME HARDENING)
volcano_critical = getattr(result, "volcano_critical", False)
swarm_anticipation = getattr(result, "swarm_anticipation", False)
m5_plus_trigger = getattr(result, "m5_plus_trigger", False)
m6_5_plus_trigger = getattr(result, "m6_5_plus_trigger", False)
m7_trigger = getattr(result, "m7_extreme_trigger", False)

# 2. Prioriteit bepalen
if volcano_critical or m7_trigger:
    priority = "🚨 KRITIEKE DREIGING"
    channel_name = channel
elif m6_5_plus_trigger or swarm_anticipation:
    priority = "⚠️ VERHOOGDE WAARSCHUWING"
    channel_name = channel
elif m5_plus_trigger:
    priority = "⚡ M5+ ALERT"
    channel_name = "Twitter/Discord"
else:
    # Geen relevante trigger → geen alert
    return

timestamp = _utc_now_str()

# 3. Scenario & magnitude-label
if m7_trigger or volcano_critical:
    scenario_mag = "M ≥ 7.0"
    scenario_label = "M7+ / extreme venster"
```

```
elif m6_5_plus_trigger:
    scenario_mag = "M ≥ 6.5"
    scenario_label = "M6.5+ venster"
elif m5_plus_trigger:
    scenario_mag = "M ≥ 5.0"
    scenario_label = "M5+ activiteit"
else:
    scenario_mag = "M ≥ 5.0"
    scenario_label = "Algemene seismiciteit"

# 4. Band-parsing (crash-veilig)
band_text = "Onbekend"
msg = getattr(result, "message_nl", "") or ""
if "Band:" in msg:
    try:
        after_band = msg.split("Band:", 1)[1]
        if "(Score:" in after_band:
            band_text = after_band.split("(Score:", 1)[0].strip()
        else:
            band_text = after_band.strip().split()[0]
    except Exception:
        band_text = "Onbekend"

# 5. Veilig toegang tot ctx.state en result-velden
psi_momentum = float("nan")
ley_node_force = float("nan")
try:
    if ctx is not None and getattr(ctx, "state", None) is not None:
```

```
        psi_momentum = getattr(ctx.state, "psi_momentum", float("nan"))
        ley_node_force = getattr(ctx.state, "ley_node_force", float("nan"))
except Exception:
    pass

# 6. Richting-informatie (Dominante Azimuth)
direction_str = "Onbekend"
direction_deg_str = ""
direction_fan = "Onbekend"
try:
    az = getattr(result, "dominant_azimuth_deg", None)
    if az is None and ctx is not None and getattr(ctx, "state", None) is not None:
        az = getattr(ctx.state, "dominant_azimuth_deg", None)

    if az is not None:
        az_float = float(az)
        direction_str = azimuth_to_compass_nl(az_float)
        direction_deg_str = f" (~{az_float:.0f}°)"
        direction_fan = direction_fan_from_main(direction_str)
except Exception:
    pass

# 7. Kans, Regio, Regime
p_event = getattr(result, "p_event", float("nan")) * 100.0
p_swarm = getattr(result, "p_swarm", float("nan")) * 100.0
region = getattr(result, "region", "ONBEKEND").upper()
regime = getattr(result, "regime", "ONBEKEND")
```

```
# 8. Time Window en Periode-string
tw_hours = float("nan")
try:
    tw_hours = float(getattr(result, "time_window_hours", 0))
except (ValueError, TypeError):
    pass

if tw_hours != tw_hours or tw_hours <= 0: # NaN-check
    period_str = "onbekend tijdsvenster"
else:
    period_str = f"komende {tw_hours:.0f} uur"
    try:
        end_time = dt.datetime.now(dt.timezone.utc) + dt.timedelta(hours=tw_hours)
        end_str = end_time.strftime("%Y-%m-%d %H:%M UTC")
        period_str += f" (tot ~{end_str})"
    except Exception:
        pass

# 9. Post-tekst opbouwen
post_text = (
    f"{priority} [{timestamp}]\n"
    f"REGIO: {region}\n"
    f"SCENARIO: {scenario_label} ({scenario_mag})\n"
    f"PERIODE (hoogste kans): {period_str}\n\n"
    f"➡️ KANS OP EVENT: {p_event:.1f}% voor {scenario_mag} binnen dit tijdsvenster\n"
    f"➡️ KANS OP ZWERM/CLUSTER: {p_swarm:.1f}%\n\n"
    f"ANALYSE-KERN:\n"
    f"  • Regime: {regime} / Band: {band_text}\n"
```

```
        f"  • Richting spanningsveld: {direction_str}{direction_deg_str}\n"
        f"   Spannings-waaier: {direction_fan}\n"
        f"  • ΔΨt (psi-momentum): {psi_momentum:.2f}\n"
        f"  • Vulkaan-risico ≥ 0.78: {volcano_critical}\n"
        f"  • Ley-Node Force: {ley_node_force:.3f}\n"
    )

    print("\n" + "=" * 80)
    print(f"🔥 EXTERNE ALERT TRIGGERED: {priority} (Kanaal: {channel_name})")
    print(post_text)
    print("=" * 80)


def main():
    """Hoofdfunctie voor de NEXES 24/7 deployment (single cycle)."""
    global ctx

    try:
        ctx = LiveEngineContext(region_name=TARGET_REGION, mode=MODE)
        print("✅ Engine Context Geïnitialiseerd.")

        now_str = _utc_now_str()
        print(f"\n🔄 [{now_str}] Initiating Live Cycle...")

        result = ctx.execute_live_cycle()

        print(f"   >> {result.message_nl}")
```

```
            # Post alert. De logica in post_alert bepaalt of er een trigger is.
            post_alert(result, channel="Telegram/X/Discord")

            print("\n✅ Cyclus voltooid.")

        except KeyboardInterrupt:
            print("\n🟥 Handmatig onderbroken (KeyboardInterrupt).")
            sys.exit(0)

        except Exception as e:
            # CRITICAL FAILURE afhandeling
            now_utc = dt.datetime.now(dt.timezone.utc).strftime('%Y-%m-%d %H:%M UTC')
            error_msg = (
                f"CRITICAL FAILURE NEXES-4 [{now_utc}]: {type(e).__name__}: {e}"
            )
            print("\n" + "!" * 80)
            print(error_msg)

            # Volledige stacktrace naar logfile (via sys.stderr redirect)
            traceback.print_exc()

            print("!" * 80)
            sys.exit(1)


if __name__ == "__main__":
    print(f"\n{'='*60}")
    print(f"NEXES-4 LIVE RUNNER GESTART OP {_utc_now_str()}")
```

```
print(f"LOGGING NAAR: {LOG_FILE}")
print(f"DOELREGIO: {TARGET_REGION} | MODUS: {MODE}")
print(f"{'='*60}\n")


main()
```