Atlas Parametric MGA: Architecture Overview with VeriCore

> **How VeriCore Powers Your Parametric Insurance MGA**

Executive Summary

This document shows how **VeriCore** serves as the trust and integrity foundation for your parametric insurance MGA platform. VeriCore solves the "Oracle Problem" by providing standardized, verifiable EO data credentials that enable instant payouts with regulatory compliance.

The Problem VeriCore Solves

Current Industry Challenge

The Oracle Problem:

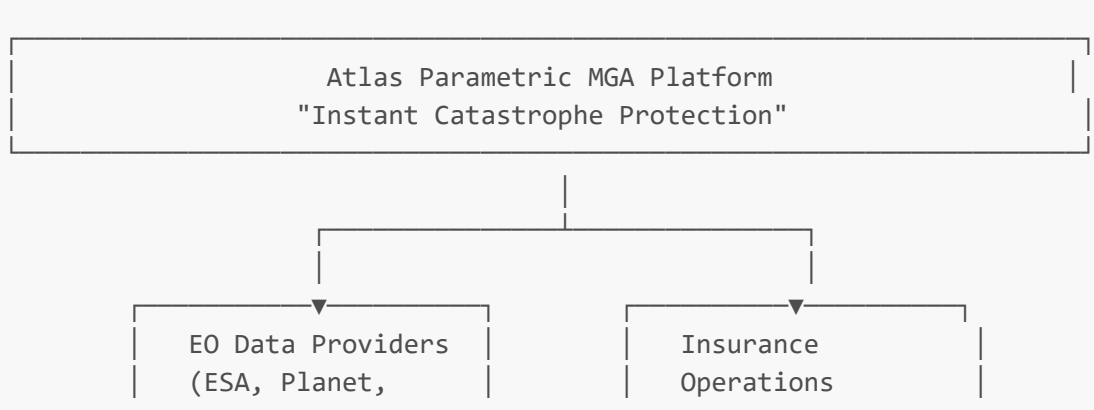
- Each insurer builds custom API integrations for each EO data provider
- No standardized way to accept data from multiple providers (ESA, Planet, NASA, NOAA)
- Trust issues: Need cryptographic proof that data used for $50M payout is authentic
- Regulatory compliance: Need tamper-proof audit trails

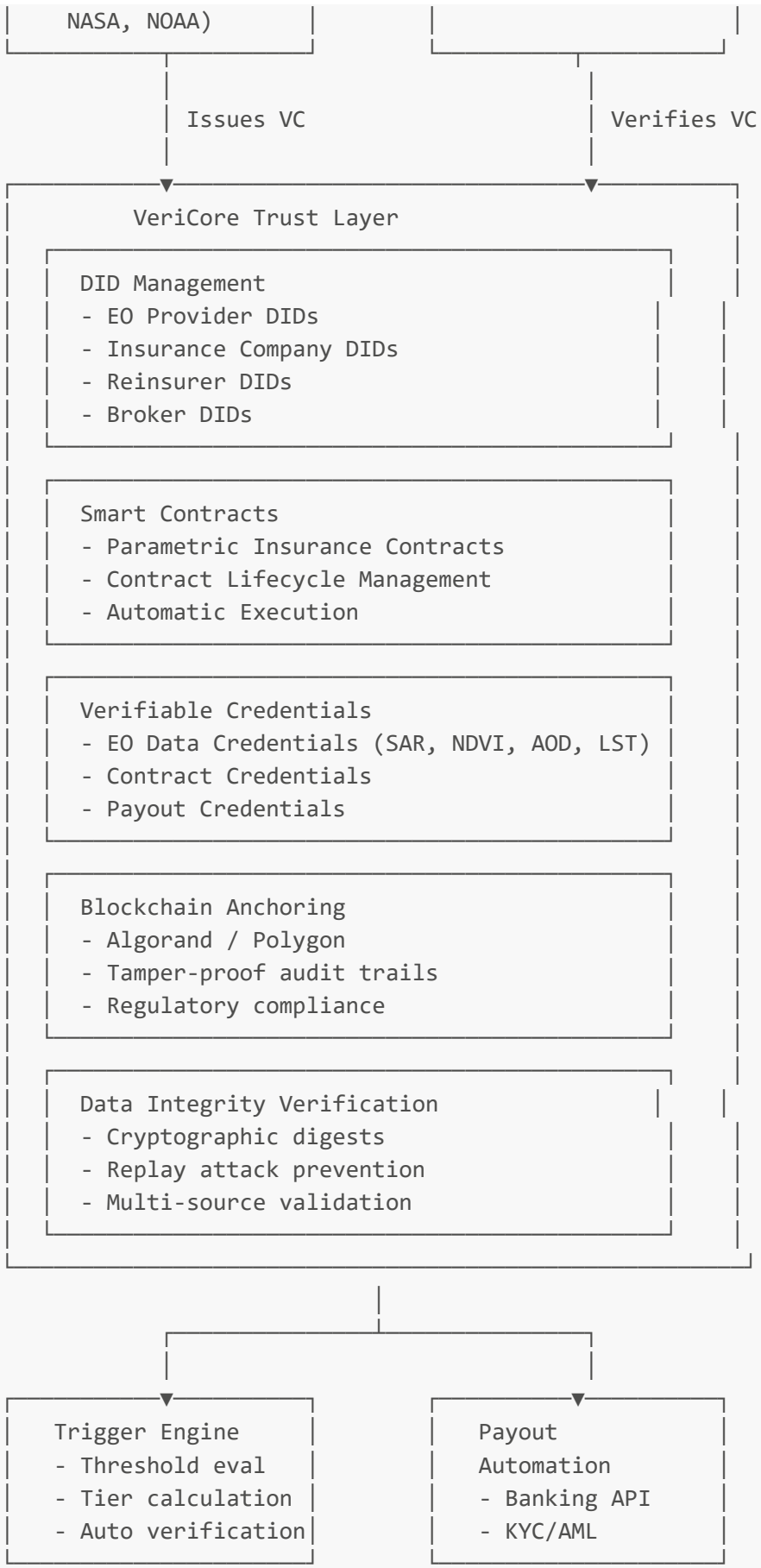
VeriCore Solution

Standardized EO Data Oracle:

- ☑ Accept EO data from any certified provider using W3C Verifiable Credentials
- ☑ Cryptographic proof of data integrity prevents tampering
- ☑ Blockchain anchoring for tamper-proof audit trails
- ☑ Multi-provider support without custom integrations
- ☑ Automated trigger verification for instant payouts

High-Level Architecture




NASA, NOAA)
Issues VC
Verifies VC
VeriCore Trust Layer
DID Management
- EO Provider DIDs
- Insurance Company DIDs
- Reinsurer DIDs
- Broker DIDs
Smart Contracts
- Parametric Insurance Contracts
- Contract Lifecycle Management
- Automatic Execution
Verifiable Credentials
- EO Data Credentials (SAR, NDVI, AOD, LST)
- Contract Credentials
- Payout Credentials
Blockchain Anchoring
- Algorand / Polygon
- Tamper-proof audit trails
- Regulatory compliance
Data Integrity Verification
- Cryptographic digests
- Replay attack prevention
- Multi-source validation
Trigger Engine
- Threshold eval
- Tier calculation
- Auto verification
Payout
Automation
- Banking API
- KYC/AML

Product Implementation Map

1. SAR Flood Parametric

VeriCore Components:

- **EO Data Credential**: Sentinel-1 SAR flood depth wrapped in VC
- **Blockchain Anchor**: Tamper-proof trigger record
- **Payout Credential**: Verifiable payout record

Flow:

```
Create Contract → Bind (VC + Anchor) → Activate →
Sentinel-1 SAR → EO Provider → Issues VC → Execute Contract →
Automatic Payout
```

2. Heatwave Parametric

VeriCore Components:

- **EO Data Credential**: MODIS LST + ERA5 temperature data
- **Multi-Day Verification**: Consecutive days above threshold
- **Payout Credential**: Verifiable heatwave payout

Flow:

```
Create Contract → Bind (VC + Anchor) → Activate →
MODIS LST → EO Provider → Issues VC → Execute Contract →
Automatic Payout
```

3. Solar Attenuation Parametric

VeriCore Components:

- **EO Data Credential**: AOD + Irradiance data
- **Attenuation Calculation**: Percentage drop verification
- **Payout Credential**: Verifiable solar payout

Flow:

```
Create Contract → Bind (VC + Anchor) → Activate →
AOD + Irradiance → EO Provider → Issues VC → Execute Contract →
Automatic Payout
```

Key VeriCore Features Used

1. DID Management

Purpose: Identity for all participants

```
// Create DIDs for EO providers, insurers, reinsurers
val eoProviderDid = vericore.dids.create(method = "key")
val insuranceDid = vericore.dids.create(method = "key")
```

2. Smart Contracts

Purpose: Executable agreements with automatic execution

```
// Create parametric insurance contract
val contract = vericore.contracts.draft(
    request = ContractDraftRequest(
        contractType = ContractType.Insurance,
        executionModel = ExecutionModel.Parametric(
            triggerType = TriggerType.EarthObservation,
            evaluationEngine = "parametric-insurance"
        ),
        parties = ContractParties(...),
        terms = ContractTerms(...)
    )
).getOrThrow()

// Bind contract (issues VC and anchors)
val bound = vericore.contracts.bindContract(
    contractId = contract.id,
    issuerDid = insurerDid,
    issuerKeyId = insurerKeyId
).getOrThrow()

// Activate contract
val active = vericore.contracts.activateContract(bound.contract.id).getOrThrow()
```

3. Verifiable Credentials

Purpose: Wrap EO data with cryptographic proof

```
// Issue EO data credential
val floodCredential = vericore.credentials.issue(
    issuer = eoProviderDid,
```

```
    subject = floodData,
    types = listOf("EarthObservationCredential", "InsuranceOracleCredential")
).getOrThrow()
```

4. Contract Execution

Purpose: Automatically execute contracts based on EO data

```
// Execute contract with EO data
val result = vericore.contracts.executeContract(
    contract = active,
    executionContext = ExecutionContext(
        triggerData = buildJsonObject {
            put("floodDepthCm", 75.0)
            put("credentialId", floodCredential.id)
        }
    )
).getOrThrow()

if (result.executed) {
    // Process automatic payout
    processPayout(result)
}
```

5. Credential Verification

Purpose: Verify EO data before using for triggers

```
// Verify credential before trigger evaluation
val verification = vericore.credentials.verify(floodCredential)
if (!verification.valid) {
    // Reject trigger
}
```

6. Blockchain Anchoring

Purpose: Tamper-proof audit trails

```
// Anchor trigger to blockchain
val anchorResult = vericore.blockchains.anchor(
    data = payoutCredential,
    chainId = "algorand:mainnet"
).getOrThrow()
```

7. Multi-Provider Support

Purpose: Accept EO data from any certified provider

```
// Accept data from ESA, Planet, NASA, NOAA
val eoData = acceptEoDataCredential(dataCredential)
// All providers use same VC format - no custom integrations!
```

Business Value Delivered

For Your MGA

1. **Cost Reduction**: Eliminate custom API integrations (80% cost savings)
2. **Speed to Market**: Launch products faster with standardized format
3. **Regulatory Compliance**: Blockchain-anchored audit trails
4. **Trust**: Cryptographic proof of data integrity
5. **Scalability**: Add new EO providers without code changes

For Reinsurers

1. **Objective Triggers**: Verifiable EO data prevents disputes
2. **Low Moral Hazard**: Cryptographic proof prevents fraud
3. **Automated Underwriting**: Standardized data format
4. **Portfolio Analysis**: Rich EO data for risk modeling

For Policyholders

1. **Transparency**: Verify data used for payouts
2. **Speed**: 24-72 hour payouts vs months
3. **Fairness**: Standardized data prevents manipulation
4. **Trust**: Cryptographic proof of integrity

Implementation Roadmap

Phase 1: MVP (Weeks 1-6)

VeriCore Setup:

- ☑ Initialize VeriCore with blockchain anchoring
- ☑ Create DIDs for EO providers
- ☑ Build SAR flood credential issuance
- ☑ Implement trigger verification

Deliverables:

- SAR flood product working
- EO data credentials issued
- Blockchain anchors created
- Basic trigger evaluation

Phase 2: Production (Months 2-12)

VeriCore Enhancements:

- ☑ Multi-provider EO data acceptance
- ☑ Heatwave product with LST credentials
- ☑ Solar attenuation product with AOD credentials
- ☑ Regulatory compliance features
- ☑ Reinsurer dashboard with VC verification

Deliverables:

- 3 products live
- Multi-provider support
- Regulatory compliance
- Reinsurer integration

Phase 3: Scale (Months 12-24)

VeriCore Scale:

- ☑ Hurricane product
- ☑ Drought/NDVI product
- ☑ Enterprise licensing
- ☑ Global expansion

Competitive Advantage

Why VeriCore Gives You an Edge

1. **Only EO-First MGA**: Full-spectrum EO integration (SAR, NDVI, AOD, LST, InSAR)
2. **Standardized Format**: W3C-compliant VCs work with all providers
3. **Instant Verification**: Cryptographic proof enables 24-72 hour payouts
4. **Regulatory Ready**: Blockchain-anchored audit trails
5. **Multi-Provider**: Accept data from any certified provider

vs. Competitors

Feature	Atlas Parametric (VeriCore)	FloodFlash	Arbol	Descartes
EO-First Design	☑	✗	✗	✗

Feature	Atlas Parametric (VeriCore)	FloodFlash	Arbol	Descartes
Multi-Provider Support	☑	✗	✗	✗
Blockchain Audit Trail	☑	✗	✗	✗
Standardized Format	☑	✗	✗	✗
Instant Verification	☑	✗	✗	✗

Technical Stack

Core Platform

- **VeriCore**: Trust and integrity foundation
- **Kotlin**: Primary language
- **Spring Boot**: API framework
- **PostgreSQL**: Policy and payout data

EO Data Processing

- **Sentinel-1 SAR**: Flood detection
- **MODIS LST**: Heatwave detection
- **AOD + Irradiance**: Solar attenuation
- **NDVI**: Drought/agriculture

Blockchain

- **Algorand** or **Polygon**: Anchoring (via VeriCore)
- **Low Cost**: Anchor digests only, not full data

Integration

- **Banking APIs**: Stripe, Plaid for payouts
- **KYC/AML**: Onfido, Jumio
- **Broker Portals**: React frontend

Getting Started

1. Review Implementation Guide

See Parametric Insurance MGA Implementation Guide for complete code examples.

2. Explore VeriCore

- Quick Start
- Parametric Insurance with EO Data
- Earth Observation Scenario

3. Start Building

```
// Initialize VeriCore
val vericore = VeriCore.create {
    blockchains {
        "algorand:mainnet" to AlgorandBlockchainAnchorClient(...)
    }
}

// Create EO provider DID
val eoProviderDid = vericore.dids.create()

// Issue EO data credential
val floodCredential = vericore.credentials.issue(
    issuerDid = eoProviderDid.id,
    credentialSubject = floodData,
    types = listOf("EarthObservationCredential")
).getOrThrow()

// Anchor to blockchain
vericore.blockchains.anchor(
    data = floodCredential,
    serializer = VerifiableCredential.serializer(),
    chainId = "algorand:mainnet"
)
```

Next Steps

1. **Read Implementation Guide**: parametric-insurance-mga-implementation-guide.md
2. **Review VeriCore Docs**: Getting Started
3. **Build MVP**: Start with SAR flood product
4. **Add Products**: Heatwave, solar attenuation
5. **Scale**: Multi-provider, global expansion

Questions?

- **VeriCore Documentation**: docs/README.md
- **API Reference**: API Reference
- **Scenarios**: Scenarios

Built with VeriCore - The Foundation for Decentralized Trust and Identity