OBINexus Access Credit Score (OACS) - Rust SemVerX Specification

Repository: github.com/obinexus/oacs
Framework: SemVerX with Constitutional Compliance
Author: Nnamdi Michael Okpala / OBINexus
Version: v1.0.0
License: OBINexus NT License v1.0

📖 Overview

The **OBINexus Access Credit Score (OACS)** is a constitutional compliance and governance tracking system that operationalizes the principles outlined in "Living Society-How We Govern Ourselves and Each Other." This repository provides Rust-based semantic versioning infrastructure for human-in-the-loop (HITL) and human-out-of-the-loop (HOTL) pipeline management.

Purpose Statement

"When systems fail, build your own." - OACS provides automated credit scoring for access rights within the OBINexus constitutional framework, ensuring systematic protection for neurodivergent individuals and dignified progression through governance tiers.

🧬 SemVerX Integration Architecture

Extended Semantic Versioning Schema

rust

```rust
#[derive(Debug, Clone, PartialEq)]
pub struct OACSVersion {
    pub major: u32,
    pub minor: u32,
    pub patch: u32,
    pub lifecycle: LifecycleState,
    pub pipeline: PipelineMode,
    pub constitutional_compliance: ComplianceLevel,
}

#[derive(Debug, Clone, PartialEq)]
pub enum LifecycleState {
    Stable,       // Production-ready, long-term support
    Legacy,       // Deprecated, backward-compatible
    Experimental, // Cutting-edge, test-only
}

#[derive(Debug, Clone, PartialEq)]
pub enum PipelineMode {
    HITL(f32),    // Human-in-the-loop with confidence percentage
    HOTL(f32),    // Human-out-of-the-loop with automation level
    Bridge(f32, f32), // Bidirectional with (HITL%, HOTL%)
}

#[derive(Debug, Clone, PartialEq)]
pub enum ComplianceLevel {
    Constitutional,  // Full OBINexus framework compliance
    Standard,        // Basic accessibility compliance
    Minimal,         // Legal minimum requirements
}
```

Version Format Examples

```
v1.stable.3.experimental.14.legacy+hitl70.hotl30.constitutional
v2.stable.0.stable.0.stable+bridge50.50.standard
v1.experimental.2.stable.5.legacy+hotl95.minimal
```

🔄 Pipeline State Management

V1 Standard: Foundation Pipeline

Focus: Basic constitutional compliance and accessibility

```rust
```

```rust
pub struct V1Pipeline {
    pub foundation_track: FoundationComponents,
    pub accessibility_requirements: AccessibilitySpec,
    pub basic_governance: GovernanceMinimum,
}

impl V1Pipeline {
    pub fn validate_constitutional_compliance(&self) -> ComplianceResult {
        // Human Rights Respect (Anti-Harassment Clause)
        // Constructive Engagement (Anti-Entitlement Clause)
        // OpenSense Extended (Sustainability Clause)
    }
}
```

V2 Standard: Advanced Life-Work Cycle

Focus: Full constitutional governance with life-work balance integration

rust

```rust
pub struct V2Pipeline {
    pub aspiration_track: AspirationComponents,
    pub life_work_balance: LifeWorkEngine,
    pub advanced_governance: ConstitutionalGovernance,
    pub neurodivergent_support: OpenSenseFramework,
}

impl V2Pipeline {
    pub fn compute_credit_score(&self, user_behavior: &BehaviorMatrix) -> CreditScore {
        // Systematic progression rewards
        // Automated consequence enforcement
        // Dignity-preserving access control
    }
}
```

🧠 Human-Machine Loop Integration

HITL Configuration (Human-in-the-Loop)

rust

```rust
```

```rust
#[derive(Debug, Clone)]
pub struct HITLConfig {
    pub human_oversight_level: f32,      // 0.0 to 1.0
    pub decision_checkpoints: Vec<CheckpointType>,
    pub manual_review_triggers: Vec<TriggerCondition>,
    pub accessibility_accommodations: AccessibilityMatrix,
}

impl HITLConfig {
    pub fn ensure_neurodivergent_support(&mut self) {
        // Sensory load management
        // Cognitive processing accommodations
        // Family network recognition patterns
    }
}
```

HOTL Configuration (Human-out-of-the-Loop)

rust

```rust
#[derive(Debug, Clone)]
pub struct HOTLConfig {
    pub automation_confidence: f32,      // 0.0 to 1.0
    pub fallback_strategies: Vec<FallbackMode>,
    pub constitutional_constraints: Vec<ConstraintType>,
    pub safety_thresholds: SafetyMatrix,
}

impl HOTLConfig {
    pub fn maintain_constitutional_protection(&self) -> ProtectionResult {
        // Automated bias detection
        // Systematic exclusion prevention
        // Machine-verifiable governance
    }
}
```

📊 Access Credit Score Matrix

Behavioral Governance Engine

rust

```rust
pub struct AccessCreditScore {
    pub constitutional_compliance: f32,  // 0.0 to 1.0
    pub community_contribution: f32,     // Constructive engagement level
    pub accessibility_support: f32,      // Neurodivergent inclusion score
    pub systematic_progression: f32,     // Growth trajectory
    pub dignity_preservation: f32,       // Anti-harassment protection
}

impl AccessCreditScore {
    pub fn compute_tier_access(&self) -> AccessTier {
        match (self.constitutional_compliance, self.dignity_preservation) {
            (0.9..=1.0, 0.9..=1.0) => AccessTier::EzeAccess,  // Safety-critical sovereignty
            (0.7..=0.89, 0.7..=1.0) => AccessTier::UcheAccess, // Collaborative projects
            (0.5..=0.69, 0.5..=1.0) => AccessTier::OpenSource,  // Community heart
            _ => AccessTier::Restricted
        }
    }
}
```

Tiered Behavioral Governance

rust

```rust
#[derive(Debug, Clone, PartialEq)]
pub enum AccessTier {
    OpenSource,   // Public pulse of the ecosystem
    UcheAccess,   // Knowledge-keys for partner counties
    EzeAccess,    // Safety-critical sovereignty keys
}

impl AccessTier {
    pub fn available_services(&self) -> Vec<ServiceType> {
        match self {
            AccessTier::OpenSource => vec![
                ServiceType::CommunityMesh,
                ServiceType::BasicPolyglot,
                ServiceType::GeographicDiscovery,
            ],
            AccessTier::UcheAccess => vec![
                ServiceType::EnterpriseOrchestration,
                ServiceType::AdvancedTopology,
                ServiceType::ProfessionalConsultation,
            ],
            AccessTier::EzeAccess => vec![
                ServiceType::PartnershipCollaboration,
                ServiceType::CulturalIntegration,
                ServiceType::CustomDeployment,
            ],
        }
    }
}
```

🛡️ Constitutional Protection Implementation

Anti-Sabotage Architecture

rust

```rust
pub struct ConstitutionalProtection {
    pub malicious_intent_detection: MaliciousIntentDetector,
    pub bias_prevention_engine: BiasPreventionEngine,
    pub accessibility_enforcement: AccessibilityEnforcer,
    pub systematic_protection: SystematicProtector,
}

impl ConstitutionalProtection {
    pub fn detect_constitutional_violation(&self, behavior: &UserBehavior) -> ViolationResult {
        // Real-time violation detection
        // Automated escalation protocols
        // Blockchain verification anchoring
        // Universal Pension Allocation penalties
    }

    pub fn enforce_neurodivergent_protection(&self) -> ProtectionStatus {
        // Phenotype-based UI/UX adaptation
        // Sensory processing support
        // Bidirectional communication learning
        // Family network recognition
    }
}
```

Machine-Verifiable Governance

```rust
pub trait MachineVerifiableGovernance {
    fn verify_constitutional_compliance(&self) -> ComplianceProof;
    fn generate_accountability_evidence(&self) -> BlockchainProof;
    fn enforce_automated_consequences(&self, violation: &Violation) -> EnforcementResult;
    fn preserve_audit_trail(&self) -> ImmutableRecord;
}
```

🌐 Integration with OBINexus Ecosystem

Division Coordination

```rust
```

```rust
pub struct EcosystemIntegration {
    pub legal_division: LegalInfrastructure,      // legal.obinexus.org
    pub computing_division: ComputingHub,         // computing.obinexus.org
    pub obiaxis_division: EpistemologicalAxis,    // obiaxis.obinexus.org
    pub publishing_division: CreativeExpression, // publishing.obinexus.org
}

impl EcosystemIntegration {
    pub fn coordinate_constitutional_compliance(&self) -> CoordinationResult {
        // Cross-division constitutional validation
        // Systematic accessibility accommodation
        // Protected contribution pathways
        // Emergency protocol activation
    }
}
```

Service Discovery and Routing

```rust
pub fn route_service_request(
    request: ServiceRequest,
    user_tier: AccessTier,
    geographic_location: GeographicPoint,
) -> RoutingResult {
    match user_tier {
        AccessTier::OpenSource => route_to_community_mesh(request, geographic_location),
        AccessTier::UcheAccess => route_to_enterprise_service(request, geographic_location),
        AccessTier::EzeAccess => route_to_sovereign_infrastructure(request, geographic_location),
    }
}
```

📋 Implementation Checklist

Phase 1: Foundation (V1 Standard)

- [ ] Basic constitutional compliance framework
- [ ] HITL/HOTL pipeline infrastructure
- [ ] Access credit score computation engine
- [ ] Anti-harassment detection system
- [ ] OpenSense accessibility integration

Phase 2: Advanced Governance (V2 Standard)

- [ ] Life-work balance engine implementation
- [ ] Advanced neurodivergent support systems
- [ ] Cross-division coordination protocols
- [ ] Machine-verifiable governance automation
- [ ] Emergency constitutional protection

Phase 3: Ecosystem Integration

- [ ] Full SemVerX compatibility
- [ ] Geographic intelligence routing
- [ ] Polyglot service coordination
- [ ] Blockchain verification anchoring
- [ ] Global constitutional expansion

🔧 Usage Examples

Basic Credit Score Computation

rust

```rust
use oacs::{AccessCreditScore, UserBehavior, ConstitutionalCompliance};

fn compute_user_access() -> AccessTier {
    let user_behavior = UserBehavior {
        harassment_incidents: 0,
        constructive_contributions: 15,
        accessibility_support_actions: 8,
        constitutional_violations: 0,
    };

    let credit_score = AccessCreditScore::from_behavior(&user_behavior);
    credit_score.compute_tier_access()
}
```

Pipeline Configuration

rust

```rust
use oacs::{PipelineMode, ConstitutionalProtection};

fn configure_pipeline() -> PipelineConfiguration {
    PipelineConfiguration {
        mode: PipelineMode::Bridge(0.7, 0.3), // 70% HITL, 30% HOTL
        constitutional_protection: ConstitutionalProtection::new(),
        accessibility_accommodations: AccessibilityMatrix::neurodivergent_optimized(),
    }
}
```

📚 Documentation and Support

Constitutional Resources

- **Framework Document**: OBINexus Constitutional Framework
- **Living Society**: "How We Govern Ourselves and Each Other" by Nnamdi Michael Okpala
- **OpenSense Framework**: Neurodivergent-Friendly Infrastructure

Technical Integration

- **SemVerX Core**: github.com/obinexus/rust-semverx
- **Vision Documents**: github.com/obinexus/vdocs
- **License Framework**: OBINexus NT License

Community and Support

- **Constitutional Violations**: constitutional-violations@obinexus.org
- **Accessibility Accommodation**: accessibility@obinexus.org
- **Technical Support**: computing@obinexus.org

🎯 Success Metrics

V1 Targets

- **Constitutional Compliance**: 95.4% minimum
- **Accessibility Coverage**: 100% WCAG 3.0+
- **Pipeline Reliability**: 99.9% uptime
- **Anti-harassment Effectiveness**: Zero tolerance violations

V2 Targets

- **Life-Work Balance Integration**: Full persona development support
- **Neurodivergent Optimization**: Bidirectional adaptation learning
- **Cross-division Coordination**: Seamless constitutional compliance
- **Global Expansion Readiness**: International human rights protection

---

"From filter to flash, from human to system, from heart to connection - building infrastructure that heals."

Built with Constitutional Integrity. Operated through Systematic Protection. Maintained by Community Governance.

Implementation Status: Specification Complete - Ready for Development
Last Updated: October 2025 | Specification Version: 1.0