# Motorsport Network - Pre-Proposal

**Date:** January 14, 2026 **From:** Tim Mitchell

## Assessmnent

From my research, I understand that Motorsport Network is a trusted, authoratative voice with a strong history, but also an industry leader and looking ahead (InsideEVs.com). My overarching lens on any recommendations is that I would seek to *enhance* and enable the editorial vision and talent with technological solutions, while also balancing the business goals.

- 40M+ monthly unique users
- 15M+ social media followers
- 65M+ monthly ad impressions
- Since 1950

## Problem Statement

Currently, we've only had a brief discussion, and the challenge has been framed as some general maintenance and operational bottlenecks, as well as a desire to look ahead at how AI or other platform changes could improve the product and grow the business.

## A Note on Assumptions

This proposal is based on external analysis - HTTP headers, public JavaScript, org chart review, Ahrefs data, etc. I haven't seen your internal systems, so some of these observations may be off-base or flat-out wrong, but the investigation gave me an opportunity to at least get the lay of the land.

---

## The Good News

Your publishing platform appears to be **modern and well-architected**:

- React/Next.js with Tailwind - current industry standard, data and AI/agent friendly
- Unified codebase across 6 properties - smart architecture
- AWS CloudFront CDN - enterprise-grade delivery
- RelevantYield + Google Ad Manager - proper ad stack

Your video platform (Motorsport.tv) is also **professional-grade**:

- React SPA with Bitmovin Player - enterprise video (used by Hulu, Red Bull Media, BBC)
- Google IMA DAI - server-side ad insertion (the right way to do video ads)
- NPAW/Youbora - video analytics and quality monitoring
- Content protection via service worker encryption

---

## Possible Bottlenecks and Challenges

You mentioned the platform is "hard to maintain." Based on external analysis, I can surmise a few likely factors:

### Organization and Scale

**Scale complexity:** - 6 properties × 20+ language editions = 100+ configurations - Each configuration has locale-specific rules, ad configs, SEO settings, legal compliance

**Engineering concentration:** - ~45+ engineers appear to be in Ukraine (Dnipro/Lviv hub) - Timezone gaps and knowledge concentration - If key people leave, institutional knowledge leaves with them

### Custom Implementation for Critical Components

**Custom component library (`msnt-*`):** - You have a custom component library (visible in the HTML/JS) - Custom libraries carry a maintenance tax and can suggest deeper tech debt - The “Article Card” needs to work across all properties and localizations with different theming, ad rules, and content structures - That’s legitimately complex and business critical, but the business logic is embedded in the code - This coupling of data and display layers can be arbitrary tech debt, or it can be a symptom of upstream data issues

---

## Hypotheses

These recommendations these issues based on my assumptions, but they’re also sound practice and forward thinking regardless of the exact diagnosis. They reduce risk, improve velocity, and position you for Ai integration and new growth ideas.

### Frontend Configuration

**What:** Document the component library, extract configuration from code, and establish a shared knowledge base. My assumption is that the business logic captured in the code reflects upstream business decisions by non-tech-teams as well, and this is a pattern I’ve seen many times.

**Why it helps:** - **Knowledge transfer** - Captures tribal knowledge and shares to all stakeholders - **100+ configurations** - Changes become data, not code; new properties launch faster - **Organizational distribution** - Knowledge accessible across timezones, teams, and roles - **AI-assisted development** - Coding assistants and agents can work with documented context, especially data objects

**The progression:**

| Phase | What | Risk |
|---|---|---|
| Document | Create COMPONENT.md for each component with business rules, property/series behavior | Lowest - no code changes |
| Extract | Move hardcoded property/edition logic to external YAML config files | Medium - refactoring |
| Standardize | Gradually adopt community UI primitives (Radix, Chakra) for base components | Higher - incremental migration |

See **Appendix A** for detailed implementation patterns for the fix.

**Risk level:** Starts low, increases with each phase. Can stop at any phase with value delivered. Inclusion of other stakeholders could improve communication.

---

## Evaluate and Address Upstream Data

**Problem:** Overloaded display components can often be necessary solution revealing a symptom of a lack of upstream data architecture/ infrastructure. Even if this isn't the historical cause, an evaluation and deep dive into this area might be the most valuable investment you can make for a number of reasons.

**What:** Create a unified data layer that is all about your domain. It resolves "what is this content about?" at a very human/semantic level, regardless of what the front-end is or even if its for a customer, advertiser, or internal employees.

**Why it helps:**

- **Knowledge silos** - Editorial knows quotes, Stats has performance data, Photo has LAT metadata, Ad sales knows advertiser relationships. A domain layer makes all of this accessible to everyone.
- **Geographic distribution** - Shared source of truth works across timezones and teams
- **Frontend simplicity** - Components receive pre-resolved data, not raw inputs to interpret
- **Cross-property-language-location consistency** - Same definitions everywhere, regardless of language, location
- **Cross-platform integration** - Articles and videos share entity metadata, and adaptations are decoupled.
- **AI/search/recommendations** - Unified structured data opens up many Ai integration opportunities without heavy lifting (RAG pipelines, auto-tagging, cluster/sentiment analysis, content operations, automated SEO, intelligent site features). More on this below.
- **Audience and Advertiser Analytics with domain context** - Web and advertising analytics gain domain dimensionality. You know not just pageviews, but how many articles mentioned Hamilton, which teams drove traffic, which circuits correlate with ad performance.
- **Editorial intelligence** - Content operations can understand audience response by semantic entities (drivers, teams, circuits) rather than vague web analytics categories. "Hamilton content outperforms Verstappen 2:1 in UK" becomes a queryable fact.
- **Premium ad inventory packaging** - Entity-tagged content enables data-driven ad products. Instead of selling "motorsport audience," you can offer "Ferrari content bundle" or "Monaco GP package" as premium first-party segments. This is how the big players operate: ESPN sells custom audience segments built on first-party behavioral data; Google DV360 enables custom intent au-

diences tied to semantic categories. Contextual advertising is projected to reach $335B by 2026 precisely because it delivers relevance without privacy concerns—and your domain entities are the foundation for that relevance.

**The hypothesis:**

Based on the org chart, you have several data sources (CMS, Forix/Wildsoft stats, LAT photos, Motorsport.tv video, DFP/Analytics). If there's no unified layer resolving "what is this content about", components may be doing data integration work:

- "Is this an F1 article?" → parse tags, check categories, infer
- "What drivers are mentioned?" → text extraction or manual tagging
- "Of those drivers, what teams do they drive for?" → graph traversal or repeated lookups
- "What ads apply?" → property-specific hardcoding

If so, the fix isn't just better components - it's resolving these questions upstream.

**What it would look like:**

```
Article arrives from CMS (English)
         │
         ▼
┌──────────────────────────────────────────────────────────┐
│                   Entity Resolution                       │
│                                                           │
│ "mentions hamilton"                                       │
│ → driver:lewis_hamilton                                   │
│ → team:ferrari                                            │
│ → series:f1                                               │
│ → nationality:british                                     │
└──────────────────────────────────────────────────────────┘
         │
         ├───────────────────────────────────────┐
         ▼                                       ▼
┌─────────────────────────┐           ┌─────────────────────────┐
│ Autosport UK (en-GB)    │           │ Motorsport.com FR       │
│                         │           │                         │
│ language: english       │           │ language: french        │
│ → no translation        │           │ → translate or skip     │
│                         │           │                         │
│ local driver check:     │           │ local driver check:     │
│ hamilton = british ✓    │           │ hamilton ≠ french       │
│ → feature prominently   │           │ → check for Gasly,      │
│                         │           │   Ocon, Hadjar content  │
│ displayConfig:          │           │                         │
│ → hero placement        │           │ displayConfig:          │
│ → premium ad tier       │           │ → standard placement    │
└─────────────────────────┘           │ → local driver sidebar  │
                                      └─────────────────────────┘
         │                                       │
         ▼                                       ▼
┌─────────────────────────┐           ┌─────────────────────────┐
```

```
│  Frontend receives:       │          │  Frontend receives:       │
│  { article,               │          │  { article (translated),│
│    entities,              │          │    entities,              │
│    displayConfig,         │          │    displayConfig,         │
│    localRelevance: high   │          │    localRelevance: low,   │
│  }                        │          │    localAlternatives:     │
│                           │          │      [vettel_article]     │
│  (simple render)          │          │  }                        │
└───────────────────────────┘          └───────────────────────────┘
```

The key insight: entity metadata (driver nationality, team origin, circuit location) enables *rules* about local relevance. In the example, we have a hypothetical feed-back loop from analytics which is basically telling us "Too much Hamilton in Ger-many", so you pull a different article on a French driver instead.

**What it would enable:** - "Everything about Hamilton" on demand (articles, videos, stats, photos) - Auto-tag content with drivers, teams, circuits - Cross-platform recommendations - Stats integration in editorial - API access for partners

## What I'd Want to Validate

| Question | Why It Matters |
|---|---|
| **Where does maintenance time actually go?** | Is it `msnt-*`, or somewhere else? |
| **What is the editorial strategy and thinking about data and Ai and automation?** | Are they open to ideas? |
| **What are the processes/ channels that bridge tech/non-tech?** | What is the culture between engineering and everyone else |
| **What are the top 10 repetitive tasks?** | Concrete examples beat theory |
| **How are entities (drivers, teams, series) managed?** | Is there a unified catalog? |
| **What does Forix/Stats integration look like?** | Rich domain data - how's it connected? |
| **What's the appetite for change and Ai?** | "Make do with what we have" vs "I want the future" |

## Other Possibilities

I have many other ideas we could discuss, and I've included them in **Appendix B: Additional Possibilities**

## What I Bring

**Background:** 20 years product leadership at Microsoft, Amazon, Roku, and more - building data driven products, specializing in digital media, data, ML, and Ai.

**Current focus:** Helping organizations develop Data and Ai strategies to achieve their products and business goals. Developing practical AI integration strategies through a framework called "Semantic Operations".

**Relevant experience:** - **Media & Advertising** - 15+ years in digital media content + advertising - **Data Systems** - content catalogs, ML pipelines, analytics, e-commerce, experimentation, RAG pipelines and databases, data engineering and data science skills) - **Cross-functional Utility** - Product Management owner of products in large, highly matrixed organizations. - **Product strategy for AI readiness** - structuring data so AI tools can actually use it - **International** - Experience working for non-US multi-national companies, launched many products accross 20+ countries, versed in international data compliance.

**What I don't bring:** Motorsport domain expertise. Your team has 75 years of heritage. I'd help with the data architecture and integration patterns, not racing knowledge.

---

## Next Steps

If this resonates, a 30-minute call would help validate assumptions:

1. Does the maintenance burden picture ring true?
2. Is there an entity catalog I didn't see, or is that actually missing?
3. How does video/publishing metadata work today?
4. What's the appetite for this kind of work right now?

## Appendix A: Frontend Documentation & Configuration Details

This appendix provides detailed implementation patterns for the three phases of frontend documentation and configuration work.

### Phase 1: Document the Component Library

**What:** Create structured documentation for each `msnt-*` component.

**Directory structure:**

```
components/
├── msnt-article-card/
│   ├── index.tsx           # The component (unchanged)
│   ├── COMPONENT.md        # Human-readable documentation
│   ├── variants.json       # All supported configurations
│   └── examples/           # Usage examples
```

**COMPONENT.md template:**

```
# Article Card Component

## Purpose
Displays article preview in list/grid contexts.

## Props
| Prop | Type | Required | Description |
|------|------|----------|-------------|
| article | Article | Yes | Article entity |
| variant | 'compact' | 'hero' | 'sidebar' | No | Display variant |

## Business Rules
- Hero variant only for featured articles
- Compact variant max 2 lines of title
- Author hidden on mobile for compact
```

```
## Property-Specific Behavior
- Autosport: Premium styling, Piano paywall eligible
- Motor1: Higher ad density, no paywall

## Series-Specific Overrides
- F1: Show driver tags prominently
- NASCAR: Include sponsor logos
```

**Why this helps:** - New engineers read docs, not reverse-engineer code - Business rules become explicit - AI coding assistants can use docs to make correct changes - Knowledge survives personnel changes

---

### Phase 2: Extract Configuration from Code

**What:** Move hardcoded property/edition logic to external configuration files.

**Before (in component):**

```
function ArticleCard({ article, property, edition }) {
  const showAuthor = property !== 'motor1' || edition !== 'us';
  const showSeriesBadge = ['f1', 'motogp'].includes(article.series);
  const adDensity = property === 'motor1' ? 'high' : 'standard';
  // ... 200 more lines of this
}
```

**After (config-driven):**

```
# properties/autosport-uk.yaml
property:
  id: autosport-uk
  domain: autosport.com

features:
  showAuthor: true
  showSeriesBadge: true
  adDensity: standard
  paywall: piano

# Business context (for humans and AI)
notes: |
  Autosport is the UK flagship property, established 1950.
  Premium positioning - higher ad rates than motorsport.com.
```

```
function ArticleCard({ article, config }) {
  return (
    <Card>
      {config.features.showAuthor && <AuthorByline />}
      {config.features.showSeriesBadge && <SeriesBadge />}
    </Card>
  );
}
```

**Why this helps:** - Config changes don't require code deploys - New properties/editions are template copies - Non-engineers can understand behavior - Business context lives alongside technical config

**Publishing and visibility:**

Config files live in a shared code repository (Git), providing: - Version history - who changed what, when, and why - Review process - changes can be reviewed before going live - Rollback - easy to revert if something breaks

But non-technical teams (marketing, ad ops, sales) also need visibility into these configs - they're often the source of change requests. Options:

| Approach | How It Works | Who Can Use |
|---|---|---|
| **Read-only dashboard** | Render YAML configs as browsable UI | Anyone |
| **Config editor UI** | Form-based editing with validation, commits to Git | Product, marketing with training |

| Approach | How It Works | Who Can Use |
|---|---|---|
| **Documentation export** | Auto-generate readable docs from configs | Anyone |

The goal: Marketing says “we want to change ad density on Motor1 US” and can either (a) see current config and request the change precisely, or (b) make the change themselves through a validated UI.

---

### Phase 3: Adopt Standard UI Primitives

**What:** Gradually replace custom UI primitives with community-maintained libraries (Radix, Chakra, etc.).

**The separation of concerns:**

```
┌─────────────────────────────────────────────────────────┐
│                  Presentation Layer                      │
│           (Radix, Chakra, or similar)                   │
│  Button, Card, Modal, Grid, Typography                   │
│  → Community maintains accessibility, browser compat     │
└─────────────────────────────────────────────────────────┘
                          │
                          ▼
┌─────────────────────────────────────────────────────────┐
│                   Composition Layer                      │
│              (Your "recipe" components)                  │
│  ArticleCard = Card + Image + Typography + Link          │
│  → You maintain publishing-specific composition          │
│  → Documented in COMPONENT.md                            │
└─────────────────────────────────────────────────────────┘
                          │
                          ▼
┌─────────────────────────────────────────────────────────┐
│                  Configuration Layer                     │
│               (Property/edition YAML)                    │
│  → Business logic externalized from code                 │
└─────────────────────────────────────────────────────────┘
```

**Why this helps:** - Community fixes UI bugs, accessibility issues - Your team focuses on publishing-specific logic - New hires already know standard library patterns - Reduces maintenance surface area

---

## Shared Documentation Layer Structure

Beyond component docs, establish a central knowledge base:

```
platform-docs/
├── architecture/
│   ├── ARCHITECTURE.md         # System overview
│   ├── DOMAIN_MODEL.md         # Entity definitions
│   └── decisions/             # Architecture Decision Records
│
├── properties/
│   └── [property.yaml + notes for each]
│
├── components/
│   └── [COMPONENT.md for each]
│
├── runbooks/
│   ├── deploy-ad-unit.md      # Step-by-step for common tasks
│   ├── add-new-language.md
│   └── update-series-branding.md
│
└── onboarding/
    └── [guides for new engineers]
```

**Why this helps:** - Knowledge accessible across timezones - Institutional memory survives personnel changes - Repetitive tasks documented once, executed consistently - AI/agents can read and follow documented processes

## Appendix B: Additional Possibilities

These opportunities emerged from research but aren't covered in the main proposal. They could become relevant depending on priorities and what we learn in discovery.

### Video Platform Modernization (Motorsport.tv)

Motorsport.tv appears to run on a separate, older tech stack - simpler JavaScript (not React/Next.js), different CDN patterns, likely separate DAM and metadata systems.

**Potential work:**

- **DAM integration** - Unified asset catalog linking video to domain entities (drivers, teams, circuits)
- **Metadata enrichment** - AI-assisted tagging from transcripts and visual content
- **Post-publish workflow** - Metrics feedback loop tying performance back to assets
- **Cross-platform integration** - Videos and articles share entity metadata, enabling "all Hamilton content" queries

**Why it matters:** Video is increasingly important for engagement and monetization. A disconnected video platform limits cross-property recommendations and unified analytics.

---

### AI-Powered Content Strategy (Ahrefs Pipeline)

Ahrefs provides content gap, backlink, and keyword data - but turning this into content decisions is manual work.

**Potential work:**

- **Automated ingestion** - Import Ahrefs exports, normalize to common schema
- **Domain enrichment** - Tag keywords by series (F1, MotoGP, NASCAR), link to existing content
- **AI analysis** - For each opportunity: analyze competitor content, identify angle, estimate effort, generate brief
- **Prioritized queue** - Score and rank opportunities, surface to editorial dashboard

**Example:** "lewis hamilton 2026 contract" shows as a content gap → AI analyzes competitor coverage (thin, rumor-focused), recommends updating existing article + adding timeline infographic, generates headline and outline with sources from entity catalog.

**Why it matters:** Turns SEO data into actionable editorial decisions without manual analyst work.

---

### Data Unification & Analytics Transformation

Likely data silos: Google Analytics (traffic), Google Ad Manager (revenue), Piano (subscriptions), social platforms (engagement), CRM (advertiser relationships). No unified view of Content → Audience → Revenue.

**Potential work:**

- **Unified data model** - dim_content, dim_audience, dim_advertiser linked to fact tables
- **ETL pipeline** - Standard patterns (dbt, Airbyte) connecting source systems
- **Key metrics** - Content ROI, author performance, series value, advertiser fit

**Example queries enabled:**

- "What's each article actually worth?" (ad revenue + subscription attribution)
- "Which series do high-value users prefer?"

- “Which advertisers align with which audience segments?”

**Why it matters:** Data-driven decisions about content investment, sales packaging, and editorial strategy require unified metrics. The entity catalog (from main proposal) becomes even more valuable when analytics can be sliced by domain dimensions.

# Appendix C: About Tim Mitchell & Semantic Operations

## About Tim Mitchell

**20+ years in data-driven product leadership at Microsoft, Amazon, and Roku.**

I've spent my career translating complex data and ML capabilities into business value. Now I'm focused on the question every organization is asking: *How do we actually benefit from AI?*

**Current focus:** Helping organizations develop practical Data and AI integration strategies and solutions.

**LinkedIn:** linkedin.com/in/timjmitchell **Website:** timjmitchell.com

---

## The Problem I Solve

Most organizations treat AI transformation as a technology problem. They hire engineers, buy tools, and build pipelines. Then they discover the actual failures stem from:

- **Business domain blindness** - Strategies don't transfer; your industry determines what data exists
- **Dissolved ownership** - Nobody owns semantic integrity across systems
- **Trust collapse** - Silent analytics failures erode credibility
- **AI amplification** - Every organizational gap becomes an agent failure mode

**The pattern:** Organizations invest heavily in generic capabilities (auth, billing, infrastructure) while under-investing in their actual differentiators. When AI enters the picture, it amplifies this misallocation.

---

## Semantic Operations Framework

After working with ML, analytics, and data systems throughout my career—and helping consulting clients with AI integration recently—I developed a framework called **Semantic Operations (SemOps)**. It's a methodology based on pattern recognition across dozens of data/AI initiatives.

**SemOps is a practical framework for aligning technology and organization to materially benefit from AI.**

### Three Pillars

| Pillar | What It Solves |
|---|---|
| **Strategic Data** | Data as first-class citizen, not afterthought. Structure enables AI; AI accelerates structure. |
| **Symbiotic Architecture** | Your software = your organization = your product. Intentional architecture that aligns with business and scaffolds data-driven operations. |
| **Semantic Optimization** | Measuring and maintaining *meaning* as operational infrastructure. Growth without coherence collapse. |

Learn more: github.com/semops-ai

---

## How This Applies to Motorsport Network

Your publishing platform is modern and sound. Your competitive advantage is **75 years of motorsport expertise**—the data, relationships, and domain knowledge that no one else has.

The opportunity isn't "modernize your stack." It's:

1. **Unify your domain knowledge** - Drivers, teams, circuits, series as structured entities, not scattered data
2. **Make meaning operational** - Frontend gets `{ article, entities, displayConfig }`—done. No 500-line conditionals.
3. **Enable AI on your terms** - With structured domain data, AI agents work *with* your expertise, not around it

**The pitch:** Build an Entity Catalog + Resolution Layer, and your maintenance burden drops, your frontend simplifies, and your AI future becomes possible.