EventBus 服务与 TRN 系统整合设计文档

🎯 项目目标

设计一个支持多实例、持久化、规则触发、并与 TRN 系统深度整合的事件总线系统。该服务将作为独立模块运行，并通过 JSON-RPC 提供标准通信接口，支持事件的发布、订阅、拉取与规则驱动执行。

———

📦 模块划分（Rust 多包架构）

eventbus/
├── event-core # 基础结构体和 trait 定义（EventEnvelope, EventBus, Rule 等）
├── event-storage # SQLite/Postgres 持久化实现（可插拔）
├── event-server # JSON-RPC over TCP 服务端，支持多总线实例
├── event-client # JSON-RPC 客户端，用于模块访问事件总线
├── event-router # 规则引擎（可选）：事件模式匹配 → 调用工具
├── event-config # 多总线配置解析器

———

📐 核心数据结构设计

EventEnvelope

pub struct EventEnvelope {
pub topic: String,
pub payload: Value,
pub timestamp: i64,
pub metadata: Option,

```
// ✅ TRN 整合
pub source_trn: Option<String>,     // 谁发出的（事件来源）
pub target_trn: Option<String>,     // 与谁有关（受影响资源）
pub correlation_id: Option<String>, // 链路追踪 ID，可用于 tracing
```

}

ToolInvocation（规则触发的目标）

pub struct ToolInvocation {
pub tool_id: String, // TRN 字符串：trn:user:alice:openapi:github-api:get-repo:v1
pub input: Value, // 工具输入
}

EventTriggerRule

pub struct EventTriggerRule {
pub id: String,
pub topic: String,
pub match_fields: HashMap<String, Value>, // 可用 JSONPath 匹配扩展
pub action: RuleAction,
}

pub enum RuleAction {
InvokeTool { tool_id: String, input: Value },
EmitEvent { topic: String, payload: Value },
}

———

🔌 JSON-RPC 接口定义

方法名 参数结构 描述
emit_event EventEnvelope 发布一个事件
poll_events { topic, since } 拉取历史事件
register_rule EventTriggerRule 注册触发规则
list_topics {} 返回当前可用主题列表
subscribe_topic { topic } 保留接口（未来推送）

———

🧱 总线实例结构设计

pub struct EventBusInstance {
pub id: String,
pub config: EventBusConfig,
pub storage: Option<Arc>,
pub dispatcher: EventDispatcher,
pub rule_engine: Option,
}

支持每个实例独立运行，可监听不同端口、配置不同持久化策略。

———

⚙️ 配置结构示例（event_buses.json）

[
{
"id": "global",
"listen": "127.0.0.1:4010",
"persist": true,
"storage_path": "data/global.db",
"enable_rules": true,
"allowed_sources": ["trn:user:*", "trn:org:*"],

"max_concurrency": 8
},
{
"id": "workflow",
"listen": "127.0.0.1:4011",
"persist": false,
"enable_rules": false
}
]

🎮 并发控制策略

- 每个 EventBusInstance 内部使用 tokio::sync::mpsc 管道管理 emit 队列
- 使用 Semaphore 控制规则触发并发数
- 每个 topic 使用独立广播通道，支持 buffer + backpressure

🗃️ 持久化表结构（SQLite）

```
CREATE TABLE event_log (
event_id INTEGER PRIMARY KEY AUTOINCREMENT,
topic TEXT NOT NULL,
payload TEXT NOT NULL,
timestamp INTEGER NOT NULL,
metadata TEXT,
source_trn TEXT,
target_trn TEXT,
correlation_id TEXT
);
```

支持按 topic、trn、时间戳查询，可用于恢复、调试、重放。

🧩 TRN 权限/过滤建议（进阶）

- EventBus 实例可配置 allowed_sources 限制 source_trn
- router 可支持规则限定 target_trn 范围（如 trn:org🔤workflow:*）
- 后续可结合认证模块，对 TRN 进行权限控制

✅ 总结

功能点	说明
多实例总线	配置多个 bus（global / workflow 等）
TRN 整合	每个事件携带来源/目标/追踪 ID
可持久化	SQLite/Postgres 插拔存储

工具触发 使用 TRN 字符串调用工具系统
JSON-RPC 接口 emit / poll / rule / subscribe
并发控制 使用 Semaphore / mpsc 队列
可扩展规则路由 基于 topic + payload 字段匹配

该系统将成为事件驱动架构的基础通信与调度层，可用于 AI、工具、调度、工作流等模块之间解耦通信与驱动。