1. 通信模块
2. 连接管理

* **自动重连机制​​**

**指数退避策略​**​：首次断开后以1秒间隔重连，后续每次间隔翻倍（2秒、4秒、8秒等），并设置最大重试次数（如5次）和最大延迟时间（如30秒）。此策略可避免频繁请求对服务器造成压力

**实现方式​**​：通过递归或循环结合setTimeout实现延迟递增，重连时需销毁旧连接并重建新连接，防止内存泄漏

* **心跳检测​​**

**协议层PING/PONG​**​：利用WebSocket内置的PING/PONG帧，客户端定时发送PING帧（如每30秒一次），若超时未收到PONG响应则判定连接断开

​**​应用层心跳包​**​：通过自定义消息（如{"type":"heartbeat"}）实现，需在消息处理逻辑中重置心跳计时器。心跳间隔建议5-15秒，超时时间设置为间隔的2倍（如30秒未响应则触发重连）

1. 收发信息

直接接收文本帧​​：若服务端明确以文本格式发送XML数据，客户端通过onmessage事件直接获取字符串（如JavaScript的event.data）

1. 数据解析

* **​​XML格式解析​​**

**解析流程​**​：接收原始数据后，使用DOMParser或xml2js等库解析XML，提取三维坐标、路径等字段。需处理命名空间和嵌套结构

​**​性能优化​**​：对于高频数据，建议采用流式解析或预处理为JSON格式以减少开销

* **数据类型标识​​**

**消息头设计​**​：在XML根节点或自定义字段中定义dataType属性（如<message type="定位点">...</message>），或通过独立协议头区分

**路由分发​**​：根据类型调用不同处理逻辑，例如定位点数据更新实体位置，路径数据生成Cesium.Polyline

* **错误处理​​**

**完整性校验​**​：检查必填字段（如坐标值、ID）是否存在，验证数值范围（如经度-180~180），非法数据直接丢弃并记录错误日志

**异常隔离​**​：采用try-catch包裹解析逻辑，防止单条数据错误导致整体崩溃

1. 功能模块
2. 三位路径标记
3. 两点间路径（当前点->目标点）
4. 两点间路径（选择点->选择点）

迪杰斯特拉算法（最短路径算法）

1. 轨迹

实时位置查询+历史轨迹记录

1. 交互模块
2. 菜单模块