1. 引言

1.1. 背景

随着自动驾驶时代的来临，对高精地图数据的使用越来越广泛。在自动驾驶系统中，自车的 ECU 控制单元无法直接使用 EHP 传输的数据，需要通过 EHR 对数据进行重新构造，以形成自动驾驶域控制器可用的数据。

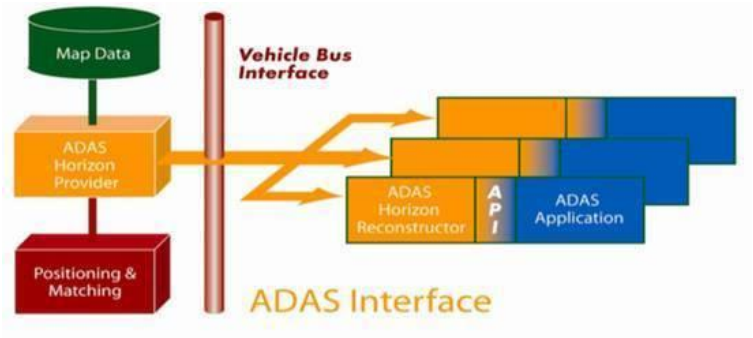


图 1-1 ADASIS 系统架构

本文档是 EHR 模块 SDK 产品使用说明文档，主要对该模块 SDK 产品的构成、接口函数、使用方法进行说明。

1.2. 应用领域

主要应用于车端智能驾驶系统，为上层应用系统（如决策系统、感知定位系统等）提供超视距的地图感知信息（如车辆前方道路坡度、曲率、路口、车道变化、分歧合流、红绿灯等信息）。

1.3. 适用对象

EHR 模块 SDK 产品使用方的研发人员可以参阅此文档。

1.4. 参考文档

编号	文档名称
1.	MapEngine 产品需求规格说明书
2.	车端地图引擎 2.0 软件架构设计
3.	MapEngine-ADASIS V3 接口说明 V1.6.xlsx

1.5. 术语与缩写

缩写、术语	解释
ADASIS V3	简称 Av3。Version 3 of the ADASIS protocol definition (to

	distinguish from Av2)
ADAS Horizon (AH)	Extract of the Map Database in front of the vehicle's current position. 车辆位置前方的地图数据提取
Horizon Provider (Av3HP)	The software component that generates the Av3 ADAS Horizon for transmission to other applications. (For short: EHP/AHP)
Horizon Reconstructor (Av3HR)	Implementation of the Av3 Programming Interface that rebuilds the Horizon produced by the Av3HP. (For short: EHR/AHR)
Path	A possible route through the road network that a vehicle could follow. 车辆可能行驶的路线
Root Path	A path that does not have a parent. 没有父 path 的 path
Sub Path	A path branching off from a parent path. 从父 path 分叉出来的 path
Parent Path	Path to which a sub path is attached.
Position Message	The Position Message positions the vehicle on the path network using the path/offset model. Position 消息使用 path/offset 模型，定义了车辆在 path 中的位置；连续的 position 消息就定义了车辆的运动。
PathControl Message	PathControl Message contains information for path management, including all relevant instructions for creating, deleting or changing any path relationship in the horizon tree. 对 Path 进行管理（创建、删除、修改）。
ProfileControl Message	Profile Control Message contains information for profile store management. Therefore it contains all relevant instructions for managing which profile sentries (attributes) must be stored or can be deleted. Profile Control Message indicates which profile entries can be deleted from paths. 对 Profile 的存储进行管理
GlobalData Messge	This message type is used to transmit profile entries containing global data if they are not related to a specific path。与 path 无关的全局数据，如：全局规划路线形点、车辆绝对位置、地图版本、系统状态等
Profile Message	The profile message contains an array of profile entries. profile 消息包含了一组 Profile Entry
Profile	Description of an attribute of the road network along a path. 描述了沿 path 路网中的某个属性。
Profile Entry	Description of an attribute of the road network at a given location on a path。描述了 path 某个位置的路网的某个属性

2. 系统综述

2.1. 系统结构

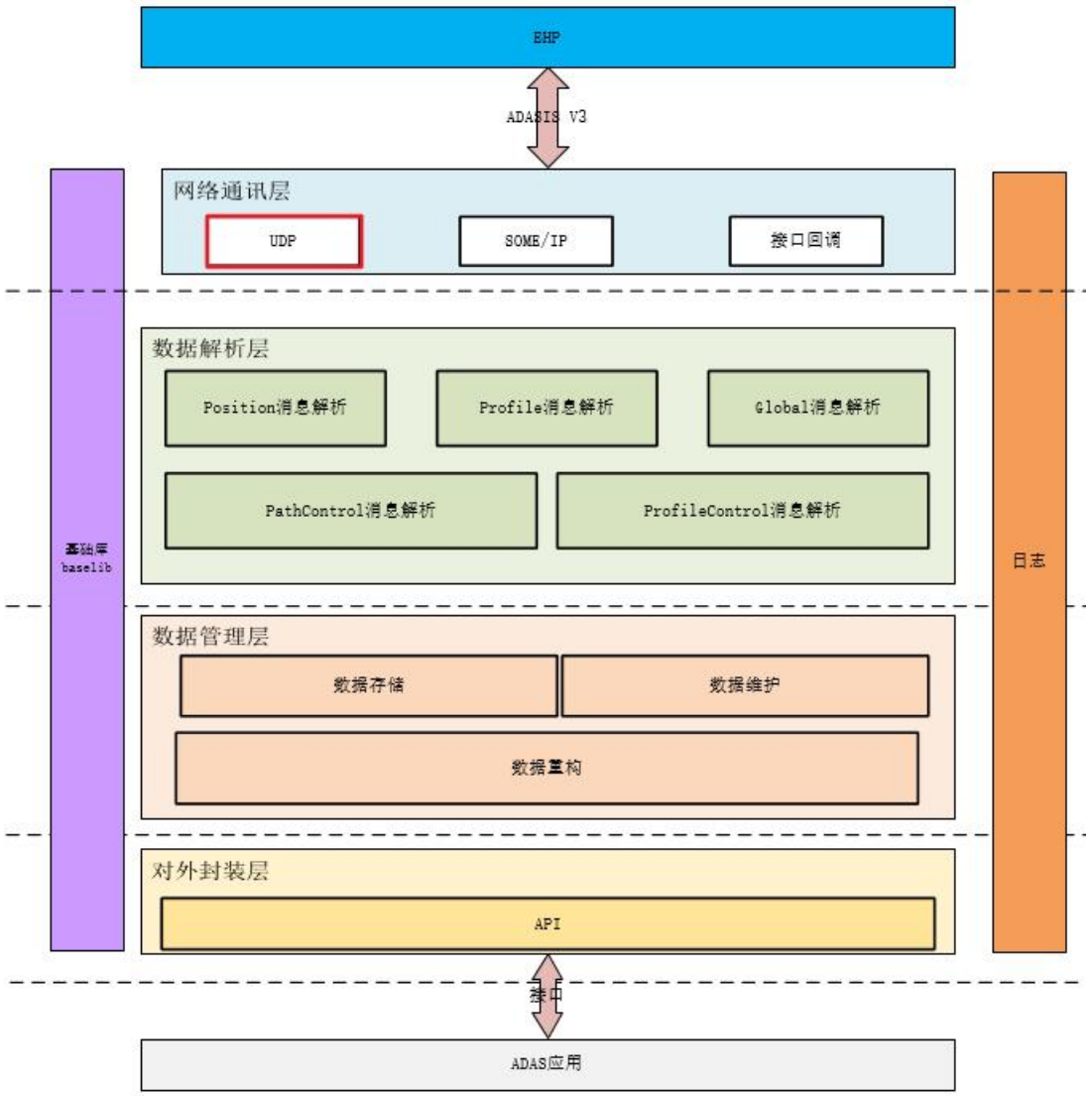


图 2-1 EHR 模块架构图

2.2. 系统简介

如上图 2-1 中所示，EHR 模块主要包含基础库、网络通讯层、数据解析层、数据管理层和对外封装层五个子模块。

2.2.1 基础库

基础库(baselib)提供了一些通用的接口。当前主要包含了线程、线程池、读写锁、文件读写、字符串操作、数值操作、点线面计算等功能。该模块不对外开放。

2.2.2 网络通讯层

负责与外部 EHP 模块进行数据通信，接收 EHP 发送的电子地平线数据。该模块不对外开放。

2.2.3 数据解析层

负责对接收到的电子地平线数据进行解析和存储。根据 Adasis 中的消息类型，解析成

对应的消息对象或指令数据存入内存中。该模块不对外开放。

2.2.4 数据管理层

该子模块主要负责对 ADASIS 数据进一步处理、重构，更新内存数据。

Position Message：将消息中的数据转成 CPosition 对象存入内存。

PathControl Message：根据 PathControl 控制指令，对内存中地图所包含的 Path、subPath 进行维护。

ProfileControl Message：根据 ProfileControl 控制指令，对内存中地图所包含的道路属性、车道属性等进行维护。

GlobalData Messge：将消息中的数据转成 CGlobalInfo 对象存入内存。

Profile Message：将消息中的数据重新构造成道路、车道组、车道、车道线、结点等对象，并根据消息中的 offet 和 endoffset 把属性挂载到对应的对象上，最终形成具有拓扑关系的局部地图数据。

该模块部分对外开放。

2.2.4 对外封装层

提供部分地图数据查询接口，供外部 adasis 应用程序使用，如：道路数据查询、车道数据查询等。该模块对外开放。

2.3. 运行性能

- 软件整体内存不超过 20M。
- 系统整体业务响应时间不超过 10ms
- 系统运行 48 小时内不挂机、不崩溃

2.4. 版权声明

3. 系统安装与卸载

本软件以 SDK（动态库）的形式部署到上层应用系统中进行集成，不涉及安装和卸载流程。

4. 系统运行环境

Linux 系统， x86 架构/ARM 架构；

RAM 不低于 4G；

CPU 不低于双核 1.3GHz

5. 产品构成

EHR 模块 SDK 产品包括 API 头文件、输入/输出数据头文件和动态库。

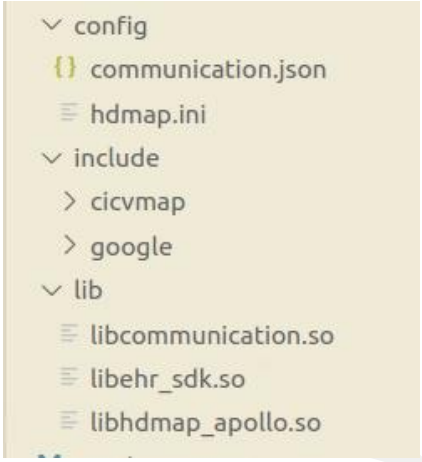


图 5-1 SDK 文件结构

序号	路径		说明
1.	lib/	libcommunication.so	通讯库
		libehr_sdk.so	ehr 基础库
		libhdmap_apollo	对外接口封装库
2.	include/	cicvmap google	对外接口头文件
3.	config/	communication.json	通讯配置文件
		hdmap.ini	ENU 原点配置文件

6. 接口说明

EHR 模块 SDK 以动态库的形式集成到上层应用系统中。上层应用系统通过调用 EHR 模块 SDK 提供的 API，实现相应的功能。调用流程如下：

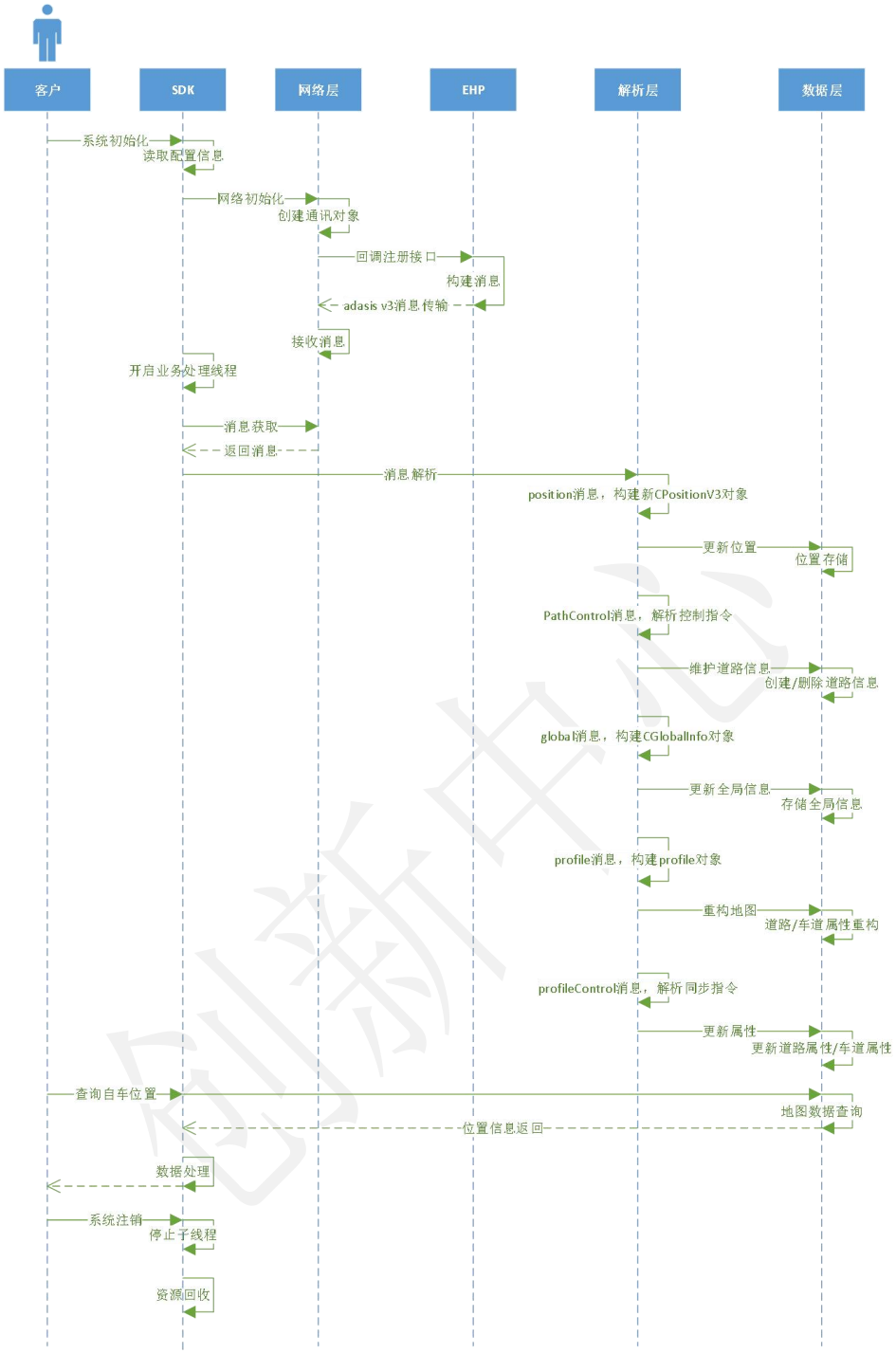


图 6 EHR 模块业务流程图

6.1. 模块初始化

6.1.1. 功能描述

模块初始化，读取配置文件，获取通讯参数，实例化网络通讯，实现 adasid 消息回调，并开启子线程对接收到的消息进行处理。

6.1.2. 界面及说明

不涉及界面。

6.1.3. 操作说明

用户（上层应用）调用地图引擎 API 函数：Init，对模块进行初始化。该函数只应该在系统上电时调用一次。该函数的定义如下：

API	**bool sdk:: Init (const char* data_path)**		
功能描述	模块初始化		
	参数名称	**类型**	**说明**
输入参数	data_path	const char*	EHR 模块配置文件所在的目录， 如:/home/admin/data
输出参数	true: 初始化成功； false：初始化失败。		

6.2. 根据车道 ID 查询车道明细

6.2.1. 功能描述

根据车道 id 查询车道明细，如：车道中心线 id、车道归属车道组 id、车道方向、车道长度、车道类型、车道形点列表、车道起始，终止位置、左右车道 id、左右车道边线 id、车道宽度、曲率、车道拓扑关系等。

6.2.2. 界面及说明

不涉及界面。

6.2.3. 操作说明

上层应用调用 API 函数：GetLaneById。函数的定义如下：

API	**const LaneInfoSConstPtr GetLaneById(const std::string &id)**
功能描述	查询车道明细

	参数名称	类型	说明
输入参数	id	me_uint64	车道 id
输出参数	车道代理对象指针		

6.2.4. 车道代理类 LaneInfoS

功能	接口	参数说明
构造函数	explicit LaneInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CLane>& lane_info)	lane_info：车道信息对象指针
获取车道 id	const Id &id() const	无
获取车道所属 road id	const Id &road_id() const	无
获取所属路段 id	const Id &section_id() const	无
获取车道信息	const Lane &lane() const	无
获取车道中心参考线点集合	const std::vector<cicv::common::math::Vec2d> &points() const	无
获取中心参考线单位向量集合	const std::vector<cicv::common::math::Vec2d> &unit_directions() const	无
获取 s 坐标处车道朝向	double Heading(const double s) const	S：s 坐标，距离车道起点 s 距离
获取 s 坐标处车道曲率	double Curvature(const double s) const	S：s 坐标，距离车道起点 s 距离
获取车道中心参考线分段朝向集合	const std::vector<double> &headings() const	无
获取车道中心参考线路段集合	const std::vector<cicv::common::math::LineSegment2d> &segments() const	无
获取车道中心参考线上的点的 s 坐标集合	const std::vector<double> &accumulate_s() const	无
	std::vector<OverlapInfoSConstPtr> overlaps() const	
	std::vector<OverlapInfoSConstPtr> cross_lanes() const	

获取车道附属交通灯	std::vector<SignalInfoSConstPtr> get_signals() const	无
获取车道附属停车让行标识牌	std::vector<YieldSignInfoSConstPtr> yield_signs() const	无
获取车道附属停止让行标识牌	std::vector<StopSignInfoSConstPtr> stop_signs() const	无
获取车道附属人行横道	std::vector<CrossWalkInfoSConstPtr> crosswalks() const	无
	std::vector<OverlapInfoSConstPtr> junctions() const	
获取车道附属减速带	std::vector<SpeedBumpInfoSConstPtr> speed_bumps() const	无
获取车道长度	double total_length() const	无
获取车道中心参考线 s 坐标处距离车道边界线的宽度	void GetWidth(const double s, double *left_width, double *right_width) const	s 坐标 left_width 距离左车道边界线的宽度 right_width 距离右车道边界线宽度
获取车道左边界距离合集	const std::vector<SampledWidth> &sampled_left_width() const	无
获取车道右边界距离合集	const std::vector<SampledWidth> &sampled_right_width() const	无
获取车道中心参考线 s 坐标处车道的宽度	double GetWidth(const double s) const	s 坐标
获取车道中心参考线 s 坐标处有效车道宽度	double GetEffectiveWidth(const double s) const	s 坐标
获取车道中心参考线至左侧道路边界的宽度集合	const std::vector<SampledWidth> &sampled_left_road_width() const	无
获取车道中心参考线右侧道路边界的宽度集合	const std::vector<SampledWidth> &sampled_right_road_width() const	无
获取车道中	void GetRoadWidth(const double s, double *left_width,	s 坐标

心参考线 s 处至左右道路边界的宽度	double *right_width) const	left_width：s 坐标点到左道路边界距离 right_width：s 坐标点到右道路边界距离
获取 s 处道路宽度	double GetRoadWidth(const double s) const	s 坐标
判断查询点是否在车道上	bool IsOnLane(const cicv::common::math::Vec2d &point) const	point：查询点坐标
判断查询 2dBox 是否在车道上	bool IsOnLane(const cicv::common::math::Box2d &box) const	box：查询 Box2d
获取 s 处的平滑点	cicv::common::PointENU GetSmoothPoint(double s) const	s 坐标
获取查询点到车道距离	double DistanceTo(const cicv::common::math::Vec2d &point) const	point：查询点坐标
获取查询点到车道距离	double DistanceTo(const cicv::common::math::Vec2d &point, cicv::common::math::Vec2d *map_point, double *s_offset, int *s_offset_index) const	point：查询点坐标 map_point:查询点在车道中心线上的投影点 s_offset: s 坐标 s_offset_index：查询点在车道中心线上的投影点的形点索引
获取查询点至最近车道点的距离	cicv::common::PointENU GetNearestPoint(const cicv::common::math::Vec2d &point, double *distance) const	point：查询点坐标 distance：查询点跟车道中心线距离
获取查询点相对车道的 frenet 坐标	bool GetProjection(const cicv::common::math::Vec2d &point, double *accumulate_s, double *lateral) const	point：查询点坐标 accumulate_s：查询点到车道起点距离 lateral：查询点跟车道中心线距离

6.3. 根据路口 id 查询路口明细

6.3.1. 功能描述

根据结点 id 查询路口明细，如：路口的形点、前驱道路列表、后继道路列表等。

6.3.2. 界面及说明

不涉及界面。

6.3.3. 操作说明

上层应用调用 API 函数：GetJunctionById。函数的定义如下：

API	const JunctionInfoSConstPtr GetJunctionById(const std::string &id)		
功能描述	查询路口明细		
	参数名称	类型	说明
输入参数	Id	me_uint64	结点 id
输出参数	路口代理对象指针		

6.3.4. 路口代理类 JunctionInfoS

功能	接口	参数说明
构造函数	explicit JunctionInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CNode>& junction_info)	junction_info：路口信息对象指针
获取路口信息	const Junction& junction() const	无
获取路口 ID	const Id &id() const	无
获取形点	const cicv::common::math::Polygon2d &polygon() const	无
获取路口绑定的停止线标识	const std::vector<Id> &OverlapStopSignIds() const	无

6.4. 根据交通信号灯 id 查询交通信号灯明细

6.4.1. 功能描述

根据交通灯 id 查询交通信号灯明细，如：交通灯类型、方向、行数、列数、停止线的形点列表等。

6.4.2. 界面及说明

不涉及界面。

6.4.3. 操作说明

上层应用调用 API 函数：GetSignalById。函数的定义如下：

API	const SignalInfoSConstPtr GetSignalById(const std::string& id)		
功能描述	查询交通信号灯信息		
	参数名称	类型	说明
输入参数	id	me_uint64	交通信号灯 id
输出参数	交通信号灯代理类对象指针		

6.4.4. 交通信号灯代理类 SignalInfoS

功能	接口	参数说明
构造函数	explicit SignalInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CTrafficLightExtend>& signal_info)	signal_info：交通信号灯信息对象指针
获取路口信息	const Signal& signal() const	无
获取路口 ID	const Id &id() const	无

6.5. 根据人行横道 id 查询人行横道明细

6.5.1. 功能描述

根据人行横道 id 查询人行横道明细，如：人行横道的形点列表等。

6.5.2. 界面及说明

不涉及界面。

6.5.3. 操作说明

上层应用调用 API 函数：GetCrossWalkById。函数的定义如下：

API	const CrossWalkInfoSConstPtr GetCrossWalkById(const std::string &ids)
功能描述	查询人行横道信息

	参数名称	类型	说明
输入参数	id	me_uint64	人行横道 id
输出参数	人行横道代理对象指针		

6.5.4. 人行横道代理类 CrossWalkInfoS

功能	接口	参数说明
构造函数	explicit CrossWalkInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CObjectGeometry>& cross_walk_info)	cross_walk_info：人行横道信息对象指针
获取人行横道 ID	const Id &id() const	无
获取人行横道信息	const Crosswalk &crosswalk() const	无
获取形点	const cicv::common::math::Polygon2d &polygon() const	无

6.6. 根据停止让行标识牌 id 查询停止让行标识牌明细

6.6.1. 功能描述

根据停止让行标识牌 id 查询停止让行标识牌明细，如：颜色、类型、形状、外框形点类别等。

6.6.2. 界面及说明

不涉及界面。

6.6.3. 操作说明

上层应用调用 API 函数：GetStopSignById。函数的定义如下：

API	**const StopSignInfoSConstPtr GetStopSignById(const std::string& id)**		
功能描述	查询停止让行标识牌		
	参数名称	**类型**	**说明**

输入参数	id	me_uint64	停止让行标识牌 id
输出参数	停止让行标识牌代理对象指针		

6.6.4. 停止让行标识牌代理 StopSignInfoS

功能	接口	参数说明
构造函数	explicit StopSignInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CTrafficSignExtend>& stop_sign_info)	stop_sign_info：人行横道信息对象指针
获取停止让行标识牌信息	const StopSign& stop_sign() const	无
获取停止让行标识牌 ID	const Id &id() const	无
获取形点	const std::vector<cicv::common::math::LineSegment2d> &segments() const	无

6.7. 根据停车让行标识牌 id 查询停车让行标识牌明细

6.7.1. 功能描述

根据停车让行标识牌 id 查询停车让行标识牌明细，如：颜色、类型、形状、外框形点类别等。

6.7.2. 界面及说明

不涉及界面。

6.7.3. 操作说明

上层应用调用 API 函数：YieldSignInfoSConstPtr。函数的定义如下：

API	**const YieldSignInfoSConstPtr GetYieldById(const std::string& id)**		
功能描述	查询停车让行标识牌		
	参数名称	**类型**	**说明**

输入参数	id	me_uint64	停车让行标识牌 id
输出参数	停车让行标识牌代理对象指针		

6.7.4. 停车让行标识牌代理 YieldSignInfoS

功能	接口	参数说明
构造函数	explicit YieldSignInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CTrafficSignExtend>& yield_sign_info)	yield_sign_info：人行横道信息对象指针
获取停车让行标识牌信息	const YieldSign& yield_sign() const	无
获取停车让行标识牌 ID	const Id &id() const	无
获取形点	const std::vector<cicv::common::math::LineSegment2d> &segments() const	无

6.8. 根据减速带 id 查询减速带明细

6.8.1. 功能描述

根据减速带 id 查询减速带明细，如：几何形状。

6.8.2. 界面及说明

不涉及界面。

6.8.3. 操作说明

上层应用调用 API 函数：GetSpeedBumpById。函数的定义如下：

API	**const SpeedBumpInfoSConstPtr GetSpeedBumpById(const std::string &id)**		
功能描述	查询减速带信息		
	参数名称	**类型**	**说明**

输入参数	id	me_uint64	减速带 id
输出参数	减速带代理对象指针		

6.8.4. 减速带代理对象 SpeedBumpInfoS

功能	接口	参数说明
构造函数	explicit SpeedBumpInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CSpeedBump>& speed_bump_info)	speed_bump_info：人行横道信息对象指针
获取减速带信息	const SpeedBump& speed_bump() const	无
获取减速带 ID	const Id &id() const	无
获取形点	const cicv::common::math::Polygon2d &polygon() const	无

6.9. 根据道路 id 查询道路明细

6.9.1. 功能描述

根据道路 id 查询道路明细，道路的类型、道路的起始，终止结点、道路的形点列表、道路所包含的车道组列表、前驱道路列表、后继道路列表、道路所含地物列表等。

6.9.2. 界面及说明

不涉及界面。

6.9.3. 操作说明

上层应用调用 API 函数：GetRoadById。函数的定义如下：

API	**const RoadInfoSConstPtr GetRoadById(const std::string &id)**		
功能描述	查询道路明细		
	参数名称	**类型**	**说明**
输入参数	id	me_uint64	道路 id

输出参数	道路代理对象指针		

6.9.4. 道路代理对象 RoadInfoS

功能	接口	参数说明
构造函数	explicit RoadInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CRoad>& road_info)	road_info：道路信息对象指针
获取道路信息	const Road& road() const	无
获取道路ID	const Id &id() const	无
获取小路段切分(车道组)	const std::vector<RoadSection> &sections() const	无
获取道路边线	const std::vector< RoadBoundary> &GetBoundaries() const	无
获取道路类型	cicv::hdmap::Road_Type type() const	无
获取路口ID	const Id &junction_id() const	无
是否连接路口	bool has_junction_id() const	无

6.10.根据查询点获取指定范围内所有车道

6.10.1. 功能描述

根据查询点获取指定范围半径内所有车道明细。

6.10.2. 界面及说明

不涉及界面。

6.10.3. 操作说明

上层应用调用 API 函数：GetLanes。函数的定义如下：

API	**const int GetLanes(const PointENU &inPoint, double distance, std::vector<LaneInfoSConstPtr> *lanes)**

功能描述	查询范围内车道		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	车道代理对象指针数组		

6.11.根据查询点获取指定范围内所有路口

6.11.1. 功能描述

根据查询点获取指定范围半径内所有路口明细。

6.11.2. 界面及说明

不涉及界面。

6.11.3. 操作说明

上层应用调用 API 函数：GetJunctions。函数的定义如下：

API	**const int GetJunctions(const PointENU &inPoint, double distance, std::vector<JunctionInfoSConstPtr> *junctions)**		
功能描述	查询范围内路口		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	路口代理对象指针数组		

6.12.根据查询点获取指定范围内所有交通信号灯

6.12.1. 功能描述

根据查询点获取指定范围半径内所有交通信号灯明细。

6.12.2. 界面及说明

不涉及界面。

6.12.3. 操作说明

上层应用调用 API 函数：GetSignals。函数的定义如下：

API	**const int GetSignals(const PointENU &inPoint, double distance, std::vector<SignalInfoSConstPtr>* signals)**		
功能描述	查询范围内交通信号灯		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	交通信号灯代理对象指针数组		

6.13.根据查询点获取指定范围内所有人行横道

6.13.1. 功能描述

根据查询点获取指定范围半径内所有交通人行横道。

6.13.2. 界面及说明

不涉及界面。

6.13.3. 操作说明

上层应用调用 API 函数：GetCrossWalks。函数的定义如下：

API	**const int GetCrossWalks(const PointENU &inPoint, double distance, std::vector<CrossWalkInfoSConstPtr>**

	***cross_walks)**		
功能描述	查询范围内人行横道		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	人行横道代理对象指针数组		

6.14.根据查询点获取指定范围内所有交通信号灯

6.14.1. 功能描述

根据查询点获取指定范围半径内所有交通信号灯明细。

6.14.2. 界面及说明

不涉及界面。

6.14.3. 操作说明

上层应用调用 API 函数：GetSignals。函数的定义如下：

API	**const int GetSignals(const PointENU &inPoint, double distance, std::vector<SignalInfoSConstPtr>* signals)**		
功能描述	查询范围内交通信号灯		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围

输出参数	交通信号灯代理对象指针数组		

6.15.根据查询点获取指定范围内停止让行标识牌

6.15.1. 功能描述

根据查询点获取指定范围半径内所有停止让行标识牌。

6.15.2. 界面及说明

不涉及界面。

6.15.3. 操作说明

上层应用调用 API 函数：GetCrossWalks。函数的定义如下：

API	**const int GetStopSigns(const PointENU &inPoint, double distance, std::vector<StopSignInfoSConstPtr>* stop_signs)**		
功能描述	查询范围内停止让行标识牌		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	停止让行标识牌代理对象指针数组		

6.16.根据查询点获取指定范围内停车让行标识牌

6.16.1. 功能描述

根据查询点获取指定范围半径内所有停车让行标识牌。

6.16.2. 界面及说明

不涉及界面。

6.16.3. 操作说明

上层应用调用 API 函数：GetYieldSigns。函数的定义如下：

API	**const int GetYieldSigns(const PointENU &inPoint, double distance, std::vector<YieldSignInfoSConstPtr>* yield_signs)**		
功能描述	查询范围内停车让行标识牌		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	停车让行标识牌代理对象指针数组		

6.17.根据查询点获取指定范围内减速带

6.17.1. 功能描述

根据查询点获取指定范围半径内所有减速带。

6.17.2. 界面及说明

不涉及界面。

6.17.3. 操作说明

上层应用调用 API 函数：GetSpeedBumps。函数的定义如下：

API	**const int GetSpeedBumps(const PointENU &inPoint, double distance, std::vector<SpeedBumpInfoSConstPtr> *speed_bumps)**

功能描述	查询范围内停车让行标识牌		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	减速带代理对象指针数组		

6.18.根据查询点获取指定范围内道路

6.18.1. 功能描述

根据查询点获取指定范围半径内所有道路。

6.18.2. 界面及说明

不涉及界面。

6.18.3. 操作说明

上层应用调用 API 函数：GetRoads。函数的定义如下：

API	**const int GetRoads(const PointENU &inPoint, double distance, std::vector<RoadInfoSConstPtr> *roads)**		
功能描述	查询范围内道路		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	道路代理对象指针数组		

6.19.根据查询点获取最近车道

6.19.1. 功能描述

根据查询点获取最近车道明细。

6.19.2. 界面及说明

不涉及界面。

6.19.3. 操作说明

上层应用调用 API 函数：GetNearestLane。函数的定义如下：

API	**const int GetNearestLane(const PointENU &inPoint, LaneInfoSConstPtr* nearest_lane, double* nearest_s, double* nearest_l)**		
功能描述	查询最近车道		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输出参数	nearest_lane	LaneInfoSConstPtr	车道代理类
输出参数	nearest_s	double	距离车道起点距离
输出参数	nearest_l	double	跟车道中心线配置值：左负右正

6.20.根据位置获取指定范围内、指定航向角范围内最近车道

6.20.1. 功能描述

根据查询点获取指定范围内且航向角在指定范围内的最近车道。

6.20.2. 界面及说明

不涉及界面。

6.20.3. 操作说明

上层应用调用 API 函数：GetNearestLaneWithHeading。函数的定义如下：

API	**const int GetNearestLaneWithHeading(const PointENU &inPoint, const double distance, const double central_heading, const double max_heading_difference, LaneInfoSConstPtr* nearest_lane, double* nearest_s, double* nearest_l)**		
功能描述	查询指定范围内、指定航向角范围内最近车道		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询距离
输入参数	central_heading	double	航向角基础值
输入参数	max_heading_difference	double	航向角范围
输出参数	nearest_lane	LaneInfoSConstPtr	车道代理类
输出参数	nearest_s	double	距离车道起点距离
输出参数	nearest_l	double	跟车道中心线配置值：左负右正

6.21.根据位置获取指定范围内、指定航向角范围内所有车道

6.21.1. 功能描述

根据查询点获取指定范围内且航向角在指定范围内的所有车道。

6.21.2. 界面及说明

不涉及界面。

6.21.3. 操作说明

上层应用调用 API 函数：GetLanesWithHeading。函数的定义如下：

API	**const int GetLanesWithHeading(const PointENU &inPoint, const double**

	distance, const double central_heading, const double max_heading_difference, std::vector<LaneInfoSConstPtr> *lanes)		
功能描述	查询指定范围内、指定航向角范围内所有车道		
	参数名称	**类型**	**说明**
输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询距离
输入参数	central_heading	double	航向角基础值
输入参数	max_heading_difference	double	航向角范围
输出参数	车道代理类指针数组		

6.22.根据位置获取前方指定范围内所有交通信号灯

6.22.1. 功能描述

根据查询点获取车辆前方指定范围内所有交通信号灯。

6.22.2. 界面及说明

不涉及界面。

6.22.3. 操作说明

上层应用调用 API 函数：GetLanesWithHeading。函数的定义如下：

API	**int GetForwardNearestSignalsOnLane(const PointENU &inPoint, const double distance, std::vector<SignalInfoSConstPtr>* signals)**		
功能描述	查询指定范围内所有道路边线、路口边线		
	参数名称	**类型**	**说明**

输入参数	inPoint	PointENU	查询点
输入参数	distance	double	查询距离
输出参数	交通信号灯代理类指针数组		

6.23.根据车道 id 查询左右车道 id

6.23.1. 功能描述

根据车道 id 查询左右车道 id。

6.23.2. 界面及说明

不涉及界面。

6.23.3. 操作说明

上层应用调用 API 函数：GetLeftRightLaneIDs。函数的定义如下：

API	**int GetLeftRightLaneIDs(const Id &id, Id &id_left, Id &id_right) const;**		
功能描述	查询左右车道 id		
	参数名称	**类型**	**说明**
输入参数	id	Id	车道 id
输出参数	左右车道 id		

6.24.根据车道 id 查询后继车道 id

6.24.1. 功能描述

根据车道 id 查询后继车道 id。

6.24.2. 界面及说明

不涉及界面。

6.24.3. 操作说明

上层应用调用 API 函数：GetLaneSuccessorIDs。函数的定义如下：

API	int GetLaneSuccessorIDs(const Id &id, std::vector<Id> &ids_succ) const;		
功能描述	查询后继车道 id		
	参数名称	类型	说明
输入参数	id	Id	车道 id
输出参数	后继车道 id		

6.25.根据车道 id 查询前继车道 id

6.25.1. 功能描述

根据车道 id 查询前继车道 id。

6.25.2. 界面及说明

不涉及界面。

6.25.3. 操作说明

上层应用调用 API 函数：GetLanePredecessorIDs。函数的定义如下：

API	int GetLanePredecessorIDs(const Id &id, std::vector<Id> &ids_prede) const;		
功能描述	查询前继车道 id		
	参数名称	类型	说明
输入参数	id	Id	车道 id
输出参数	前继车道 id		

6.26.根据车道 id 查询车道限速

6.26.1. 功能描述

根据车道 id 查询车道限速。

6.26.2. 界面及说明

不涉及界面。

6.26.3. 操作说明

上层应用调用 API 函数：GetLaneSpeedLimit。函数的定义如下：

API	**int GetLaneSpeedLimit(const Id &id, double &speed) const;**		
功能描述	查询车道限速		
	参数名称	**类型**	**说明**
输入参数	id	Id	车道 id
输出参数	车道限速		

6.27.根据车道 id 查询车道转向

6.27.1. 功能描述

根据车道 id 查询车道转向。

6.27.2. 界面及说明

不涉及界面。

6.27.3. 操作说明

上层应用调用 API 函数：GetLaneTurn。函数的定义如下：

API	**int GetLaneTurn(const Id &id, cicv::hdmap::Lane::LaneTurn &type) const;**		
功能描述	查询车道转向		
	参数名称	**类型**	**说明**
输入参数	id	Id	车道 id

输出参数	车道转向		

6.27.4. 车道转向类型 LaneTurn

功能	参数说明
Lane_laneTurn_NO_TURN	**直行**
Lane_laneTurn_LEFT_TURN	左转
Lane_laneTurn_RIGHT_TURN	右转
Lane_laneTurn_U_TURN	掉头

6.28.根据车道 id 查询车道类型

6.28.1. 功能描述

根据车道 id 查询车道类型。

6.28.2. 界面及说明

不涉及界面。

6.28.3. 操作说明

上层应用调用 API 函数：GetLaneType。函数的定义如下：

API	**int GetLaneType(const Id &id, cicv::hdmap::Lane::LaneType &type) const;**		
功能描述	查询车道类型		
	参数名称	**类型**	**说明**
输入参数	id	Id	车道 id
输出参数	车道类型		

6.29.根据位置点查询左侧车道边线类型明细

6.29.1. 功能描述

根据位置点查询左侧车道边线类型明细。

6.29.2. 界面及说明

不涉及界面。

6.29.3. 操作说明

上层应用调用 API 函数：LeftBoundaryType。函数的定义如下：

API	**LaneBoundaryType::Type LeftBoundaryType(const icv::common::PointENU &point) const;**		
功能描述	查询左侧车道边线类型明细		
	参数名称	**类型**	**说明**
输入参数	point	PointENU	位置点
输出参数	左侧车道边线类型明细		

6.30.根据车道查询左侧车道边线类型明细

6.30.1. 功能描述

根据车道查询左侧车道边线类型明细。

6.30.2. 界面及说明

不涉及界面。

6.30.3. 操作说明

上层应用调用 API 函数：LeftBoundaryType。函数的定义如下：

API	**LaneBoundaryType::Type LeftBoundaryType(const LaneInfoSConstPtr lane) const;**
功能描述	查询左侧车道边线类型明细

	参数名称	类型	说明
输入参数	Lane	LaneInfoSConstPtr	车道指针
输出参数	左侧车道边线类型明细		

6.31.根据位置点查询右侧车道边线类型明细

6.31.1. 功能描述

根据位置点查询右侧车道边线类型明细。

6.31.2. 界面及说明

不涉及界面。

6.31.3. 操作说明

上层应用调用 API 函数：RightBoundaryType。函数的定义如下：

API	**LaneBoundaryType::Type RightBoundaryType(const cicv::common::PointENU &point) const**		
功能描述	查询右侧车道边线类型明细		
	参数名称	**类型**	**说明**
输入参数	point	PointENU	位置点
输出参数	右侧车道边线类型明细		

6.32.根据车道查询右侧车道边线类型明细

6.32.1. 功能描述

根据车道查询左侧车道边线类型明细。

6.32.2. 界面及说明

不涉及界面。

6.32.3. 操作说明

上层应用调用 API 函数：RightBoundaryType。函数的定义如下：

API	**LaneBoundaryType::Type RightBoundaryType(const LaneInfoSConstPtr lane) const**		
功能描述	查询右侧车道边线类型明细		
	参数名称	**类型**	**说明**
输入参数	Lane	LaneInfoSConstPtr	车道指针
输出参数	右侧车道边线类型明细		

6.33.根据位置点查询最近道路 id

6.33.1. 功能描述

根据位置点查询最近道路 id。

6.33.2. 界面及说明

不涉及界面。

6.33.3. 操作说明

上层应用调用 API 函数：GetNearestRoad。函数的定义如下：

API	**int GetNearestRoad(const cicv::common::PointENU &point, Id &nearest_road_id) const**		
功能描述	查询最近道路 id		
	参数名称	**类型**	**说明**
输入参数	Point	PointENU	位置点
输出参数	最近道路 id		

6.34.根据位置点查询指定范围内、指定偏航角区间最近道路明细

6.34.1. 功能描述

根据位置点查询指定范围内、指定偏航角区间最近道路明细。

6.34.2. 界面及说明

不涉及界面。

6.34.3. 操作说明

上层应用调用 API 函数：GetNearestRoadWithHeading。函数的定义如下：

API	**int GetNearestRoadWithHeading(const cicv::common::PointENU& point, const double distance, const double central_heading, const double max_heading_difference, RoadInfoSConstPtr& nearest_road, double& nearest_s, double& nearest_l) const**		
功能描述	查询最近道路 id		
	参数名称	**类型**	**说明**
输入参数	Point	PointENU	位置点
输出参数	最近道路 id		
输出参数	位置点沿着最近道路中心线距离起点距离		
输出参数	位置点跟最近道路中心线垂直距离		

6.35.根据道路 id 查询后继道路 id

6.35.1. 功能描述

根据道路 id 查询后继道路 id。

6.35.2. 界面及说明

不涉及界面。

6.35.3. 操作说明

上层应用调用 API 函数：GetRoadSuccessorIDs。函数的定义如下：

API	**int GetRoadSuccessorIDs(const Id &id, std::vector<Id> &ids_succ)**

功能描述	查询后继道路 id		
	参数名称	**类型**	**说明**
输入参数	id	Id	道路 id
输出参数	后继道路 id 数组		

6.36.根据道路 id 查询前继道路 id

6.36.1. 功能描述

根据道路 id 查询前继道路 id。

6.36.2. 界面及说明

不涉及界面。

6.36.3. 操作说明

上层应用调用 API 函数：GetRoadPredecessorIDs。函数的定义如下：

API	**int GetRoadPredecessorIDs(const Id &id, std::vector<Id> &ids_pred) const**		
功能描述	查询前继道路 id		
	参数名称	**类型**	**说明**
输入参数	id	Id	道路 id
输出参数	前继道路 id 数组		

6.37.根据道路 id 查询车道明细

6.37.1. 功能描述

根据道路 id 查询车道明细。

6.37.2. 界面及说明

不涉及界面。

6.37.3. 操作说明

上层应用调用 API 函数：GetRoadLanes。函数的定义如下：

API	int GetRoadLanes(const Id &id, std::vector<LaneInfoSConstPtr> &lanes) const		
功能描述	查询车道明细		
	参数名称	类型	说明
输入参数	id	Id	道路 id
输出参数	车道明细数组		

6.38.根据车道 id 查询所属道路 id

6.38.1. 功能描述

根据车道 id 查询所属道路 id。

6.38.2. 界面及说明

不涉及界面。

6.38.3. 操作说明

上层应用调用 API 函数：GetRoad。函数的定义如下：

API	int GetRoad(const Id &lane_id, Id &road_id) const		
功能描述	查询道路 id		
	参数名称	类型	说明
输入参数	id	Id	车道 id
输出参数	道路 id		

6.39.根据位置点查询是否处于道路面内

6.39.1. 功能描述

根据位置点查询是否处于道路面内。

6.39.2. 界面及说明

不涉及界面。

6.39.3. 操作说明

上层应用调用 API 函数：IsOnRoad。函数的定义如下：

API	**bool IsOnRoad(const cicv::common::PointENU& point) const**		
功能描述	查询位置点是否处于道路面内		
	参数名称	**类型**	**说明**
输入参数	point	PointENU	位置点
输出参数	是否处于道路面内结果： true：位置点处于道路面内 false: 位置不在道路面内		

6.40.根据限速标牌 id 查询限速标牌明细

6.40.1. 功能描述

根据 id 查询限速标牌明细。

6.40.2. 界面及说明

不涉及界面。

6.40.3. 操作说明

上层应用调用 API 函数：GetSpeedSignById。函数的定义如下：

API	**SpeedSignInfoSConstPtr GetSpeedSignById(const std::string& id) const**		
功能描述	根据 id 查询限速标牌明细		
	参数名称	**类型**	**说明**

输入参数	Id	string	限速标牌 id
输出参数	限速标牌对象指针		

6.40.4. 限速标牌代理类 SpeedSignInfoS

功能	接口	参数说明
构造函数	explicit SpeedSignInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CTrafficSignExtend>& speed_sign_info)	speed_sign_info：限速标牌信息对象指针
获取限速标牌信息	const SpeedSign& speed_sign() const	无
获取限速标牌 ID	const Id &id() const	无
获取形点	const cicv::common::math::Polygon2d &polygon() const	无

6.41.根据禁停区域 id 查询禁停区域明细

6.41.1. 功能描述

根据 id 查询禁停区域明细。

6.41.2. 界面及说明

不涉及界面。

6.41.3. 操作说明

上层应用调用 API 函数：GetClearAreaById。函数的定义如下：

API	**ClearAreaInfoSConstPtr GetClearAreaById(const std::string& id) const**		
功能描述	根据 id 查询禁停区域明细		
	参数名称	**类型**	**说明**
输入参数	Id	string	禁停区域 id
输出参数	禁停区域对象指针		

6.41.4. 禁停区域代理类 ClearAreaInfoS

功能	接口	参数说明
构造函数	explicit ClearAreaInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CObjectGeometry>& clean_area_info)	clean_area_info：禁停区域信息对象指针
获取禁停区域信息	const cicv::hdmap::ClearArea& clear_area() const	无
获取禁停区域 ID	const Id &id() const	无
获取形点	const cicv::common::math::Polygon2d &polygon() const	无

6.42.根据查询点获取指定范围内所有交通标志牌

6.42.1. 功能描述

根据查询点获取指定范围半径内所有交通标志牌明细。

6.42.2. 界面及说明

不涉及界面。

6.42.3. 操作说明

上层应用调用 API 函数：GetTrafficSigns。函数的定义如下：

API	**int GetTrafficSigns(const cicv::common::PointENU& point, double distance, std::vector<TrafficSignInfoSConstPtr> *traffic_signs)**		
功能描述	查询范围内交通标志牌		
	参数名称	**类型**	**说明**
输入参数	Point	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	交通标志牌代理对象指针数组		

6.42.4. 交通标志牌代理 TrafficSignInfoS

功能	接口	参数说明
构造函数	explicit TrafficSignInfoS(const std::shared_ptr<const dpi::ehr::data_manager::CTrafficSignExtend>& traffic_sign_info)	traffic_sign_info：交通标志牌信息对象指针
获取交通标志牌信息	const TrafficSign& traffic_sign() const	无
获取交通标志牌 ID	const Id &id() const	无
获取形点	const std::vector<cicv::common::math::LineSegment2d> &segments() const	无

6.43.根据查询点获取指定范围内所有限速标牌

6.43.1. 功能描述

根据查询点获取指定范围半径内所有限速标牌明细。

6.43.2. 界面及说明

不涉及界面。

6.43.3. 操作说明

上层应用调用 API 函数：GetSpeedSigns。函数的定义如下：

API	**int GetSpeedSigns(const cicv::common::PointENU& point, double distance, std::vector<SpeedSignInfoSConstPtr> *speed_signs)**		
功能描述	查询范围内限速标牌		
	参数名称	**类型**	**说明**
输入参数	Point	PointENU	查询点
输入参数	distance	double	查询范围

输出参数	限速标牌代理对象指针数组

6.44.根据查询点获取指定范围内所有禁停区域

6.44.1. 功能描述

根据查询点获取指定范围半径内所有禁停区域明细。

6.44.2. 界面及说明

不涉及界面。

6.44.3. 操作说明

上层应用调用 API 函数：GetClearAreas。函数的定义如下：

API	**int GetClearAreas(const cicv::common::PointENU& point, double distance, std::vector<ClearAreaInfoSConstPtr>* clear_areas)**		
功能描述	查询范围内禁停区域		
	参数名称	**类型**	**说明**
输入参数	Point	PointENU	查询点
输入参数	distance	double	查询范围
输出参数	禁停区域代理对象指针数组		

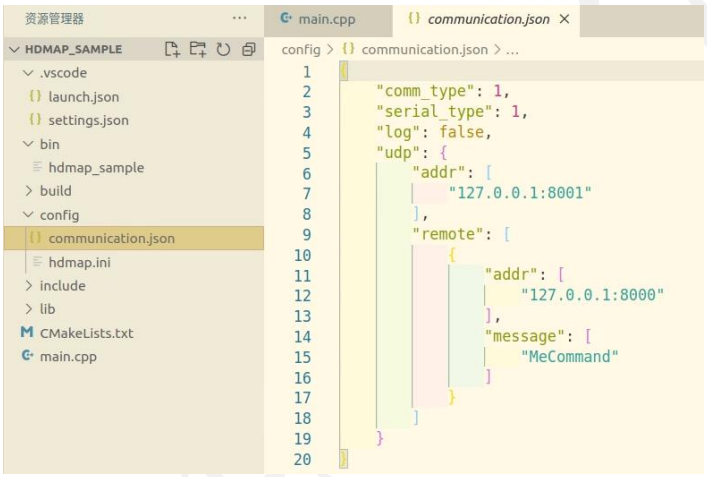
7. 运维&问题排除

7.1. 初始化失败

初始化失败的原因可能是配置文件路径未设置正确，按照下面截图设置。

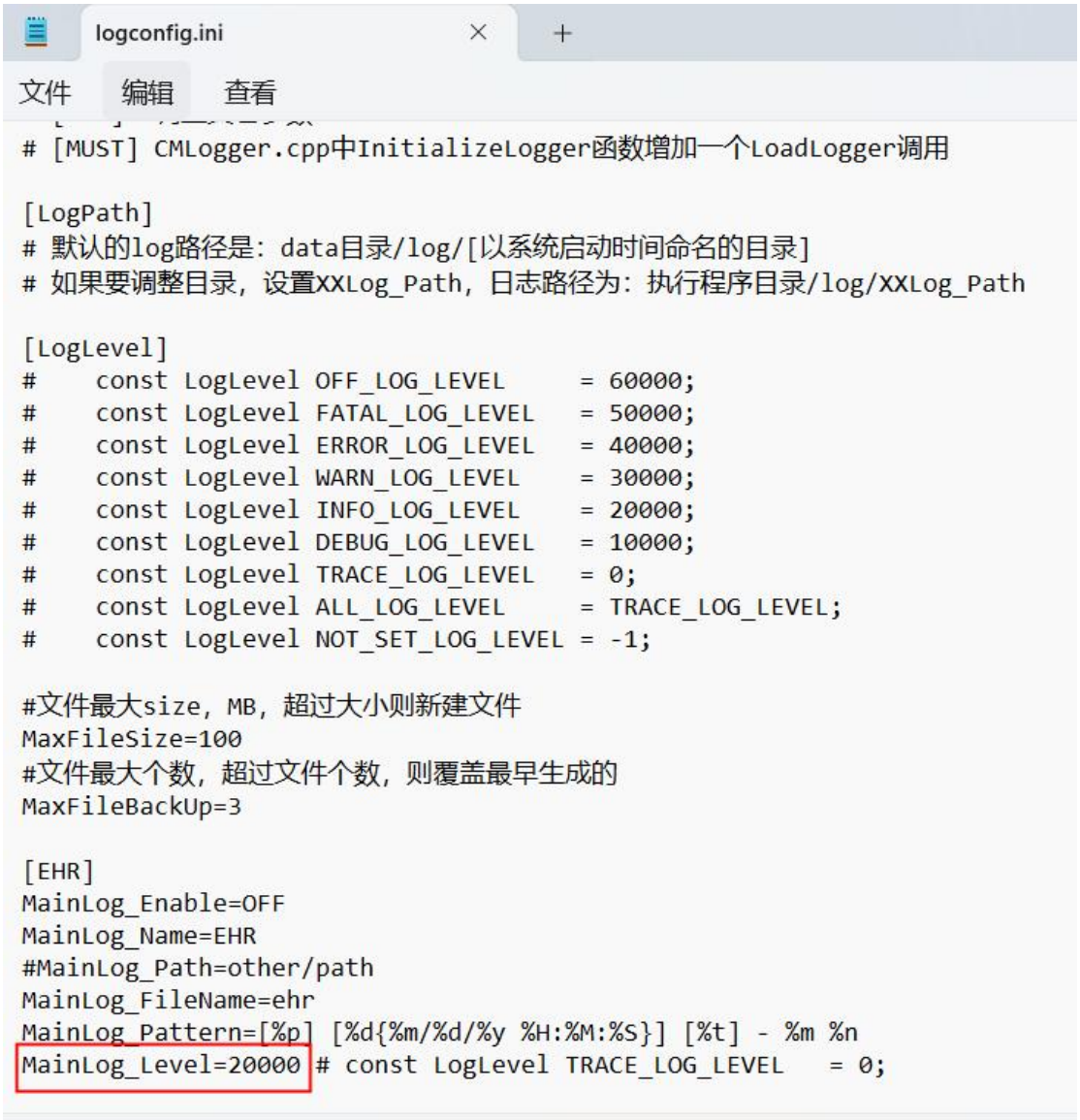
7.2. 通讯失败

通讯失败失败的原因可能是通信 IP，端口并未设置，请在配置文件中设置通讯 IP、端口、通讯类型。



7.3. 调用接口耗时较长

请检查 config 目录下 ini 文件，日志是否开启且日志级别较低，可调整至 20000 以上。


logconfig.ini
文件
编辑
查看
[MUST] CMLogger.cpp中InitializeLogger函数增加一个LoadLogger调用
[LogPath]
默认的log路径是：data目录/log/[以系统启动时间命名的目录]
如果要调整目录，设置XXLog_Path，日志路径为：执行程序目录/log/XXLog_Path
[LogLevel]
const LogLevel OFF_LOG_LEVEL   = 60000;
const LogLevel FATAL_LOG_LEVEL   = 50000;
const LogLevel ERROR_LOG_LEVEL   = 40000;
const LogLevel WARN_LOG_LEVEL   = 30000;
const LogLevel INFO_LOG_LEVEL   = 20000;
const LogLevel DEBUG_LOG_LEVEL   = 10000;
const LogLevel TRACE_LOG_LEVEL   = 0;
const LogLevel ALL_LOG_LEVEL   = TRACE_LOG_LEVEL;
const LogLevel NOT_SET_LOG_LEVEL = -1;
#文件最大size，MB，超过大小则新建文件
MaxFileSize=100
#文件最大个数，超过文件个数，则覆盖最早生成的
MaxFileBackUp=3
[EHR]
MainLog_Enable=OFF
MainLog_Name=EHR
#MainLog_Path=other/path
MainLog_FileName=ehr
MainLog_Pattern=[%p] [%d{%m/%d/%y %H:%M:%S}] [%t] - %m %n
MainLog_Level=20000 # const LogLevel TRACE_LOG_LEVEL   = 0;