Granda.ATTS.CIM使用说明

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Arthur | Func. | Date | Version |
| IvanJLZhang | 编辑类库使用说明 | 2018/4/17 | 1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 类库结构图

Granda.ATTS.CIM.Data.dll

Granda.ATTS.CIM.dll

CIMBASE

CIM4HST

CIM4EQT

Initialize Scene

Data Collection

Equipment Terminal Service

Clock

Process Program Management

Alarm Management

Remote Control

……

Granda.CIM模块软件设计架构图

## 类库说明

### Granda.ATTS.CIM.dll

主要处理和CIM Server端的消息通信，根据消息的功能以场景形式向Equipment/Host端提供相应API，响应事件以及接口等。

### Granda.ATTS.CIM.Data.dll

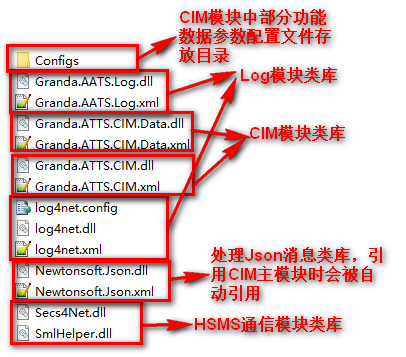
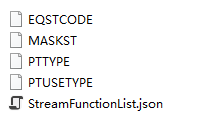
提供各场景功能所涉及数据的结构体定义。

### Granda.ATTS.Log.dll

记录模块运行日志信息，此部分与主体模块相同。

## 类库使用说明

### 类库文件目录：



### 类库使用详细说明：

使用时添加Granda.ATTS.CIM.dll和Granda.ATTS.CIM.Data.dll类库引用，CIM模块主入口类CIM4EQT和CIM4HST分别对应Equipment和Host端的入口类，调用方法采用公开的API/接口/事件三种形式，其中接口目前按照Scenario分为七个主要功能接口。以下介绍CIM4EQT类的初始化以及相关Scenario功能方法介绍

#### 类库初始化：

##### 1）CIM4EQT提供两种类构造方法：

/// <summary>

/// 构造方法， 提供SecsGem参数

/// </summary>

/// <param name="secsGem">HSMS通信模块</param>

/// <param name="deviceId">设备Id号， 默认为1</param>

public CIM4EQT(SecsGem secsGem, short deviceId = 1);

/// <summary>

/// 构造方法，提供创建SecsGem所需参数

/// </summary>

/// <param name="ipAddress">Server端IP地址</param>

/// <param name="port">Server端端口号</param>

/// <param name="isActive">是否为主动模式</param>

/// <param name="deviceId">设备Id号， 默认为1</param>

public CIM4EQT(string ipAddress, int port, bool isActive, short deviceId = 1);

##### 2）初始化类以后需要调用一次Scenario初始化方法，以注册并初始化Scenario功能：

/// <summary>

/// 场景初始化

/// <para>接口初始化和事件初始化不可同时使用，即如果初始化了相应场景的接口之后，对应场景下所有的事件将不会再被响应</para>

/// </summary>

/// <param name="itializeScenario">initialize场景回调方法接口</param>

/// <param name="rCSCallBack">Remote Control场景回调方法接口</param>

/// <param name="aMSCallBack">Alart Management场景回调方法接口</param>

/// <param name="clock">Clock场景回调方法接口</param>

/// <param name="eqtTerminalService">Equipment Terminal Service场景回调方法接口</param>

/// <param name="recipeManagement">Prcess Program(Recipe) Management场景回调方法接口</param>

/// <param name="dataCollection">Data Collection场景回调方法接口</param>

public void ScenarioInitialize(

IInitializeScenario itializeScenario = null,

IRCSCallBack rCSCallBack = null,

IAMSCallBack aMSCallBack = null,

IClock clock = null,

IEqtTerminalService eqtTerminalService = null,

IRecipeManagement recipeManagement = null,

IDataCollection dataCollection = null);

##### 3) 在调用Scenario初始化方法后，如果有相应接口没有初始化实例则需要注册相应Scenario下的事件（相应Scenario下事件下面会详细介绍）。

**（\*Tip: 接口和事件响应不可同时使用， 即如果在初始化时实现了某个Scenario的接口的话， 那么在这个Scenario下的所有事件将不会被触发。）**

#### Initialize Scenario

接口：IInitializeScenario

事件：

1. ControlStateChanged;// 设备控制状态变化事件
2. DateTimeUpdate;// Date and Time 更新事件

API：

/// <summary>

/// local端设置online/offline状态

/// </summary>

/// <param name="onLine">true表示请求在线，反之离线</param>

/// <param name="equipmentInfo">设备当前信息</param>

public bool LaunchOnOffLineProcess(bool onLine, EquipmentInfo equipmentInfo);

/// <summary>

/// 向Host端发送更新时间请求

/// </summary>

public bool LaunchRequestDateTimeProcess();

#### Data Collection

接口：

IDataCollection

事件：

// Selected Equipment Status Request event

OnSelectedEquipmentStatusRequest;

// Equipment Constants Request event

OnEquipmentConstantsRequest;

// Formatted Status Request event

OnFormattedStatusRequest;

// Enable Disable event report request event

OnEnableDisableEventReportRequest;

// Trace Data Initialization Request event

OnTraceDataInitializationRequest;

API：

/// <summary>

/// report Glass Process data

/// </summary>

public bool LaunchProcessResultReportProcess(ProcessResultReport report);

/// <summary>

/// Trace Data Send

/// </summary>

public bool LaunchTraceDataInitializationReportProcess(TraceDataInitializationReport report);

/// <summary>

/// Selected Equipment Status Data report

/// </summary>

public bool LaunchSelectedEquipmentStatusReportProcess(string[] report);

/// <summary>

/// Formatted Status Report

/// </summary>

public bool LaunchFormattedStatusReportProcess(FormattedProcessProgramReport report);

/// <summary>

/// Equipment Constants data report

/// </summary>

public bool LaunchEquipmentConstantsReportProcess(EquipmentConstantChangeReport report);

#### Remote Control

接口：IRCSCallBack

事件：

// Remote Control Scen下Host所发送的命令

OnRemoteControlCommandRequest;

API：

/// <summary>

/// Remote Control Scenario:

/// Process (Start/Cancel/Abort/Pause/Resume/Operator Call) Report

/// </summary>

public bool LaunchProcessReport(RCMD rcmd, ProcessLaunchReport processLaunchReport, EquipmentInfo equipmentInfo);

#### Alarm Management

接口：IAMSCallBack

事件：

// Alarm Enable Disable Request Event

OnAlarmEnableDisableRequest;

// Current Alarm List Request

OnCurrentAlarmListRequest;

API：

/// <summary>

/// 对RequestAlarmList的回复

/// </summary>

public bool LaunchCurrentAlarmListReport(CurrentAlarmListReport currentAlarmListReport);

#### Process Program (Recipe) Management

接口：IRecipeManagement

事件：

// Current EPPD Request event

OnCurrentEPPDRequest;

// Formatted Process Program Request event

OnFormattedProcessProgramRequest;

API：

/// <summary>

/// local端recipe发生变化时向host发送通知

/// </summary>

public bool LaunchRecipeChangeProcess(RecipeChangeReport report);

/// <summary>

/// 对CurrentEPPDRequest的回复

/// </summary>

public bool LaunchCurrentEPPDReportProcess(CurrentEPPDReport report);

#### Clock

接口：IClock

事件：

// Date and Time 更新事件

DateTimeUpdate;

API：

无

#### Equipment Terminal Service

接口：IEqtTerminalService

事件：

// 接收Equipment Terminal Service Scen下Display Message事件，

// （消息有每次最大十条限制）

ReceiveDisplayMessage;

// 发送Equipment Terminal Service Scen下Display Message消息成功事件，

// （消息有每次最大十条限制）

SendDisplayMessageDone;

API：

/// <summary>

/// 向Host发送display message

/// </summary>

/// <param name="messages">消息内容，最多十条</param>

public bool LaunchSendDisplayMessageProcess(string[] messages);