DLMS

DLMS是**通信协议**，应用层规范标准

支持诸如远程读取测量仪数据、远程控制和增值服务之类的应用，用于计量不同种类的能源，例如电、水、天然气或热能

COSEM

电能计量配套规范

用于与电能计量设备进行通信的**接口模型**，提供了可通过通信接口使用的功能的视图。该模型使用面向对象的方法

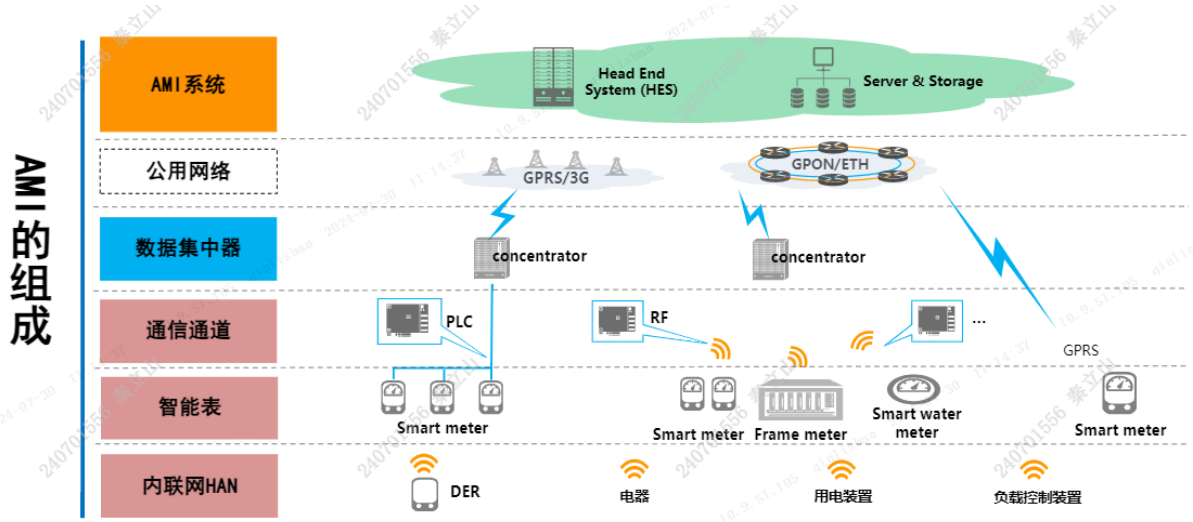
AMI

Advanced Metering Infrastructure（高级计量体系）的简称

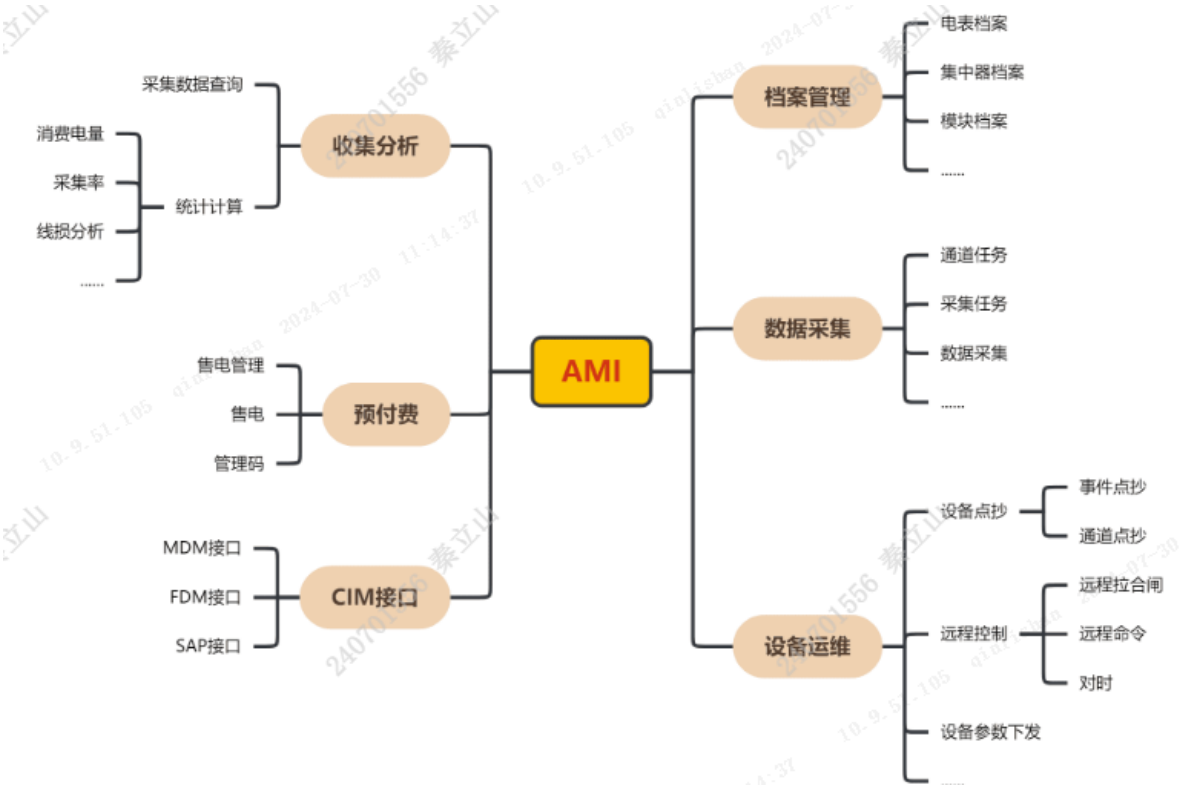
是一个用来测量、收集、存储、分析和运用用户用电信息的完整网络和系统

是智能电网的重要组成部分

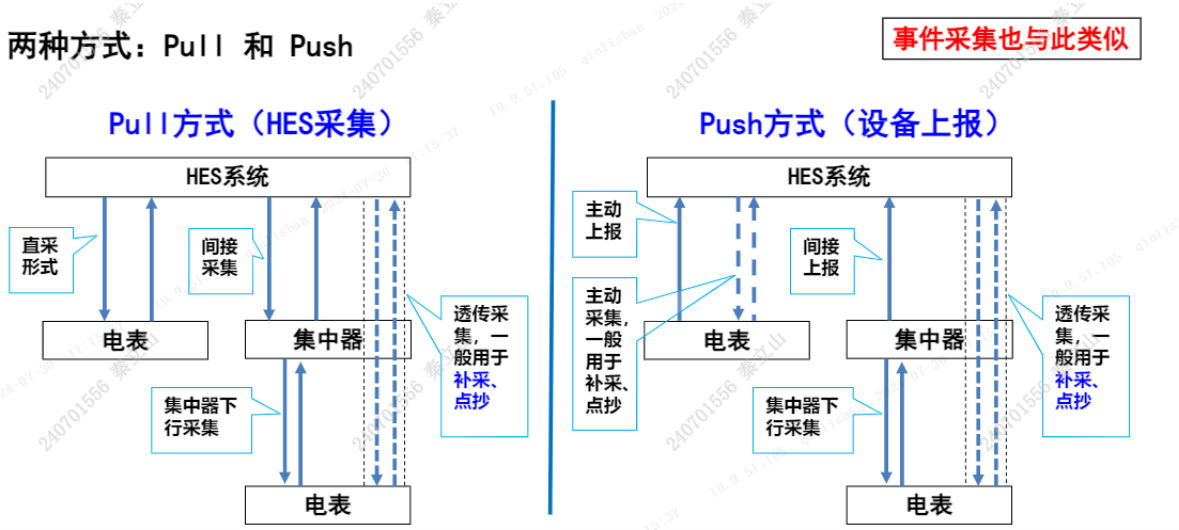
组成



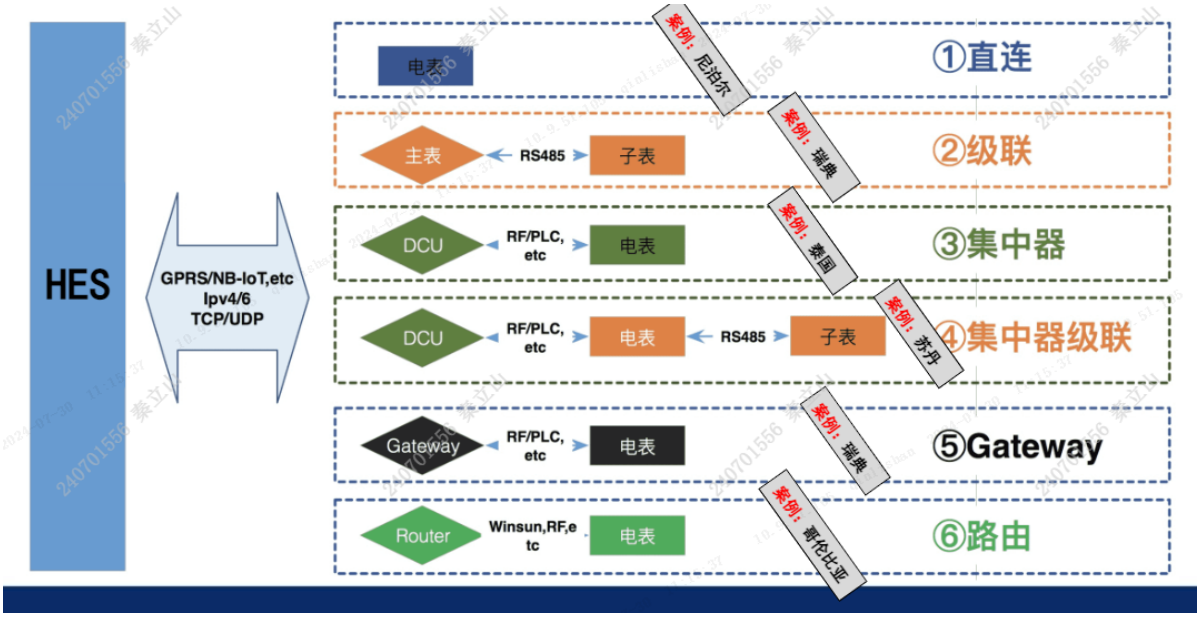
功能


AMI
收集分析
采集数据查询
消费电量
采集率
线损分析
统计计算
.....
预付费
售电管理
售电
管理码
CIM接口
MDM接口
FDM接口
SAP接口
档案管理
电表档案
集中器档案
模块档案
.....
数据采集
通道任务
采集任务
数据采集
.....
设备运维
设备点抄
事件点抄
通道点抄
远程控制
远程拉合闸
远程命令
对时
设备参数下发
.....

数据采集


两种方式：Pull 和 Push
事件采集也与此类似
Pull方式（HES采集）
HES系统
直采形式
间接采集
电表
集中器
透传采集，一般用于补采、点抄
集中器下行采集
电表
Push方式（设备上报）
HES系统
主动上报
主动采集，一般用于补采、点抄
间接上报
电表
集中器
透传采集，一般用于补采、点抄
集中器下行采集
电表

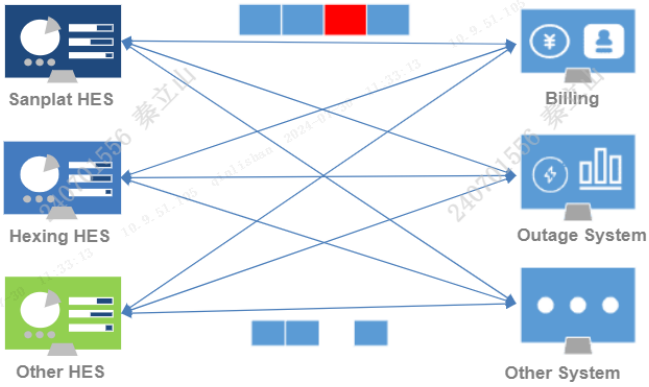
场景案例


HES
GPRS/NB-IoT,etc
Ipv4/6
TCP/UDP
电表
案例：尼泊尔
①直连
主表
RS485
子表
案例：瑞典
②级联
DCU
RF/PLC, etc
电表
案例：泰国
③集中器
DCU
RF/PLC, etc
电表
RS485
子表
案例：芬丹
④集中器级联
Gateway
RF/PLC, etc
电表
案例：瑞典
⑤Gateway
Router
Winsun,RF,etc
电表
案例：哥伦比亚
⑥路由

MDM

why

电力公司**面临的问题**



问题：

1. 不同的HES厂家，**规约可能不一致，接口不一致；**
2. 电力公司需要学习不同HES系统操作
2. HES与其他系统之间接口**错综复杂；**
3. HES接口上传的数据**参差不齐；**
4. 数据存储在HES系统中，**无法看到数据汇总，数据价值无法体现。**

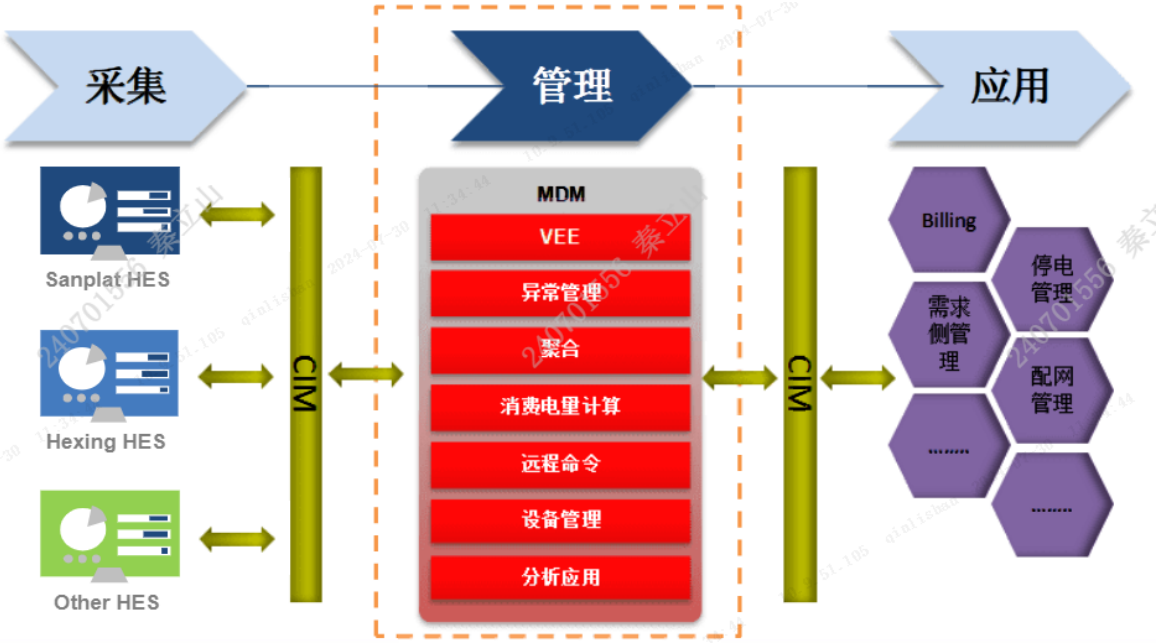
what

MeterDataManagement（量测数据管理系统）的简称

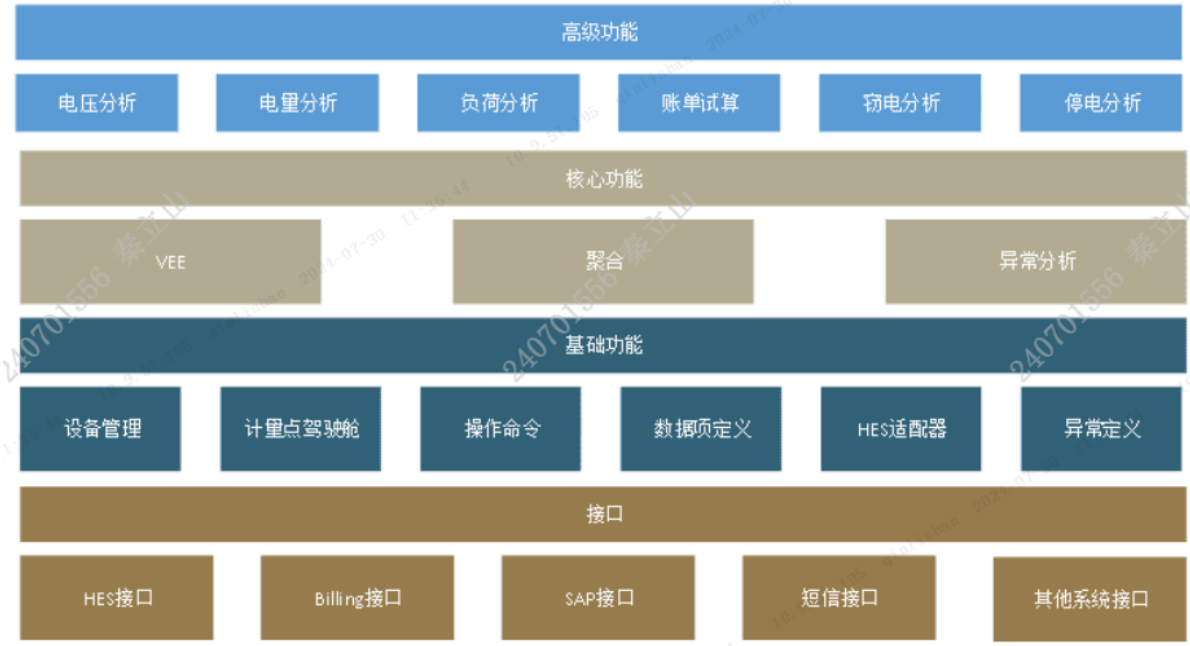
是AMI解决方案中的一个重要组成部分，是一个集收集、数据清洗、存储、分析于一身的高级应用系统

MDM是其他高级应用系统的数据源泉，提供完整、准确的数据

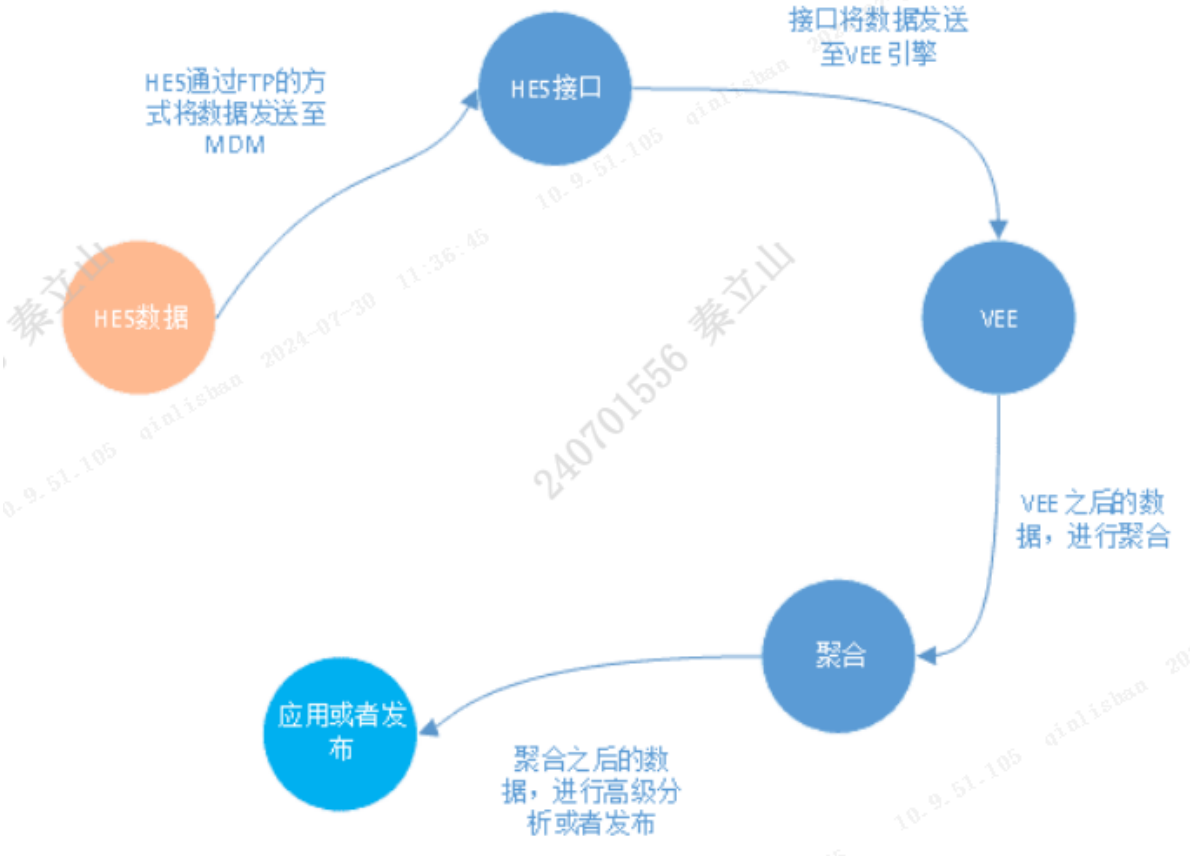
位置



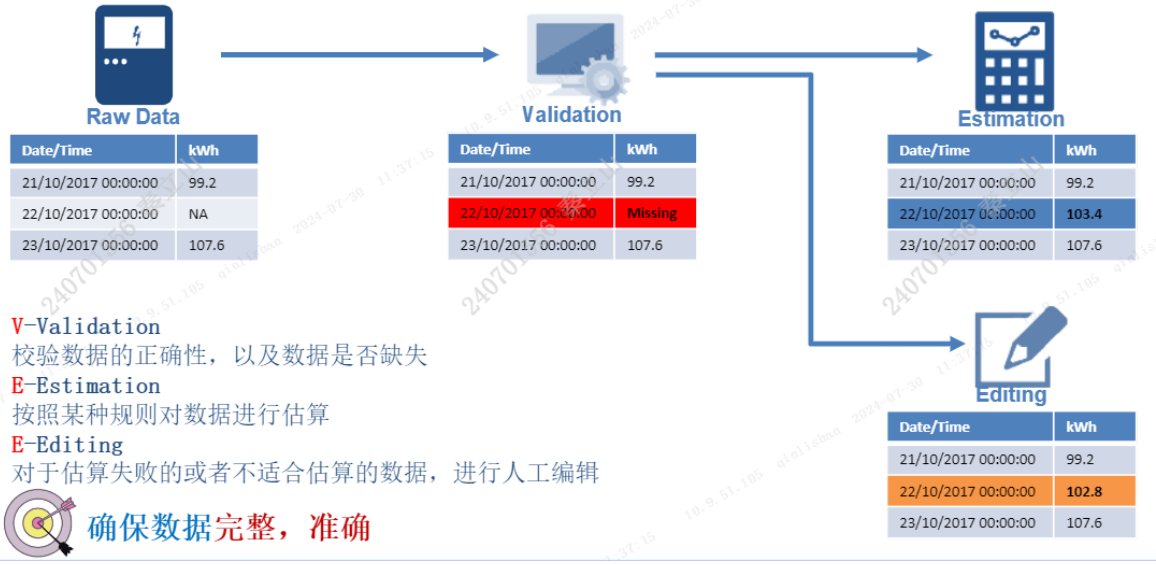
架构


高级功能
电压分析
电量分析
负荷分析
账单试算
窃电分析
停电分析
核心功能
VEE
聚合
异常分析
基础功能
设备管理
计量点驾驶舱
操作命令
数据项定义
HES适配器
异常定义
接口
HES接口
Billing接口
SAP接口
短信接口
其他系统接口

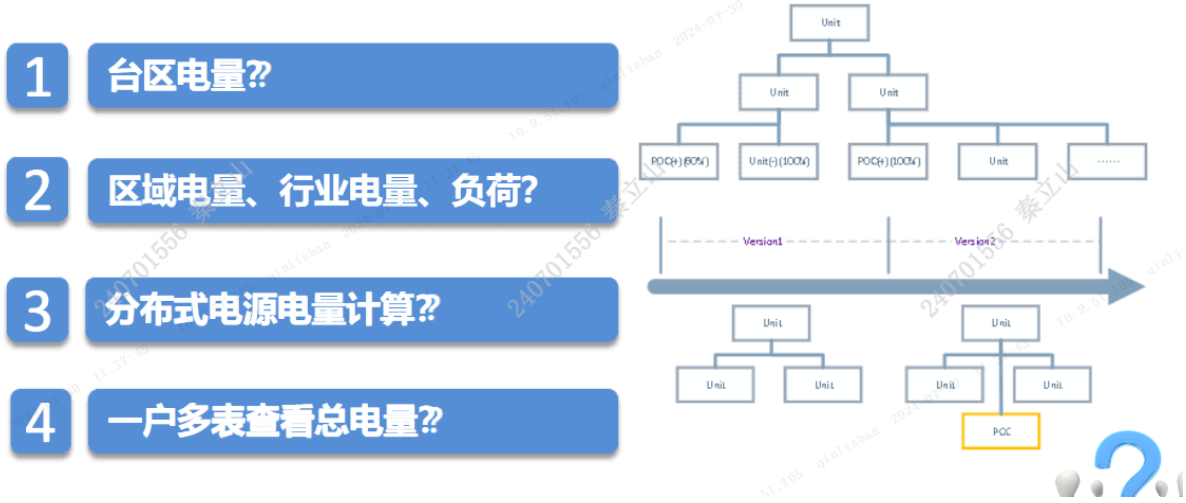
数据流


HES通过FTP的方式将数据发送至MDM
HES接口
接口将数据发送至VEE引擎
HES数据
VEE
VEE之后的数据，进行聚合
聚合
应用或者发布
聚合之后的数据，进行高级分析或者发布

VEE


Raw Data
Validation
Estimation
Editing
Date/Time
kWh
21/10/2017 00:00:00
99.2
22/10/2017 00:00:00
NA
23/10/2017 00:00:00
107.6
Missing
103.4
102.8
V-Validation
校验数据的正确性，以及数据是否缺失
E-Estimation
按照某种规则对数据进行估算
E-Editing
对于估算失败的或者不适合估算的数据，进行人工编辑
确保数据完整，准确

聚合


1 台区电量?
2 区域电量、行业电量、负荷?
3 分布式电源电量计算?
4 一户多表查看总电量?
Unit
POC
Version1
Version2

NMS

what


整体架构图
MDC（数据采集系统）
数据采集
电表管理
前置通信
远程操作
NMS（网络管理平台）
VPN服务器
DHCP服务器
Radius认证服务器
设备管理
Gateway（网关）
BR（网关模块）
VPN客户端
数据转发
黑白名单
DHCP客户端
LN入网管理
数据转发
网络拓扑
DHCP服务器
LN（节点/电表模块）
Meter（电表）
LN入网
DHCP客户端
数据转发
IP推送
HES通信
电表注册
事件上报
远程执行
局域网
（IPv6）
公网/专网
（IPv4/6）
Wi-SUN
（IPv6）

NMS：Network management platform（网络管理平台）

- **网络路由器（MDC与电表通信网络）**
- **Wi-SUN物联网方案**
- **管理Gateway、LN设备**
- **管理设备IPv6地址（DHCP）**
- **网络设备通讯安全保障（Radius、网关认证）**
- **网络设备在线检测（上线、掉线、掉电）**

VPN服务器：选用OpenVPN自研方案

- **网关安全隧道（证书加密），IP分配**
- **IPv4转IPv6，Wi-SUN网络基于IPv6协议**

DHCP服务器：动态主机配置协议服务器（自主开发）

- **基于DHCPv6通信协议，基于GUID分配BR IP**
- **网络节点添加动态路由**
- **基于DHCPv6协议**

Radius认证服务器：安全认证服务器（自主开发）

- **电表模块（LN）入网安全认证**
- **基于Radius协议**

MDC：电表数据采集系统（类似于HES）

Gateway：网关，负责透传数据，类似路中器

LN：Left node（节点），这里指的是电表模块

COAP： Constrained Application Protocol（WEB）

- **NMS、Gateway、BR、LN通信协议**

Wi-SUN：Wireless Utility Networks（智能无线网络）

远程操作：电表远程抄读数据、设置参数、拉合闸控制等

FDM

what

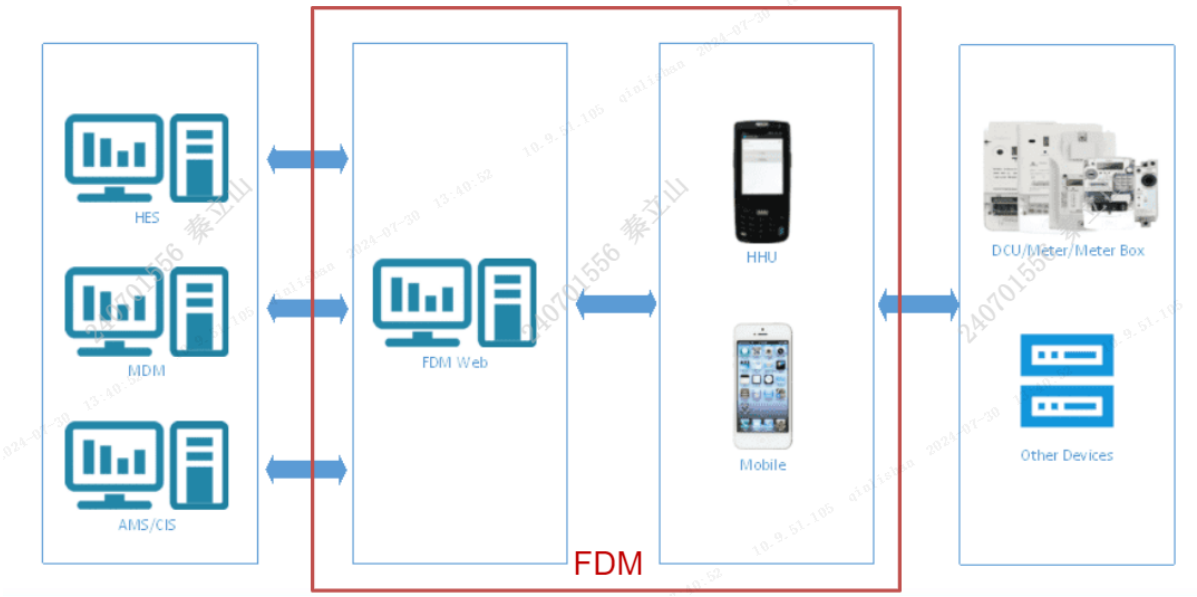
FieldDeviceManager（现场设备管理系统）的简称

是AMI解决方案中的比较重要的一个环节

FDM管理设备的整个生命周期

FDM负责对现场的设备进行配置、调试、维护和诊断。

位置



GPRS电表与4G电表区别

技术原理区别

GPRS电表：GPRS(General Packet Radio Service)是一种基于无线通信技术的数据传输方式，适用于低速、低功耗的远程通信。GPRS电表通过内置的GPRS模块与运营商基站建立连接，实现数据传输。由于其采用分组交换技术，通信成本相对较低。

4G电表：4G(FourthGeneration)是一种高速无线通信技术，相较于GPRS，4G电表具有更高的数据传输速率和更稳定的网络连接。4G电表同样内置通信模块，但与运营商4G基站进行连接，支持多种通信协议，可实现多媒体、高速数据传输等应用。

性能与功能区别

数据传输速率：4G电表具有更高的数据传输速率，能够满足实时监测、远程控制等高带宽需求场景。而GPRS电表传输速率相对较低，适用于轻度实时应用。

通信稳定性：4G网络覆盖范围更广，信号强度较高，因此4G电表的通信稳定性较好。GPRS网络覆盖范围有限，信号较弱，可能影响数据传输稳定性。

功耗：4G电表功耗相对较高，但可采用节能设计降低能耗。GPRS电表功耗较低，更适合能源监测与控制场景。

应用场景：GPRS电表适用于低速率、低功耗的远程抄表、用电监测等场景。4G电表适用于高速率、高稳定性要求的智能家居、工业自动化等领域。

应用优势与劣势

GPRS电表优势

通信成本较低，适用于远程抄表等大规模应用；

功耗低，适用于能源监测与控制场景；

抗干扰能力强，适应各种恶劣环境。

劣势

-数据传输速率有限，不适合高带宽需求场景；

-网络覆盖范围有限，可能影响通信稳定性；

-抗电磁干扰能力较弱。

2.4G电表优势

-数据传输速率高，支持实时监测与控制；

-通信稳定性较好，适用于复杂环境；

-支持多种通信协议，可实现多媒体、高速数据传输等应用。

不足

-功耗较高，对能源需求较大；

-通信成本相对较高；

-抗干扰能力一般。

熟练使用项目

官方文档

pom文件结构

什么是POM文件?

POM（Project Object Model）文件是Maven项目的核心文件之一。它是一个XML文件，描述了项目的基本信息、依赖项、构建和发布等信息。POM文件是Maven的重要组成部分，可以帮助开发者管理和构建项目。在使用Maven进行项目构建时，需要根据项目的需要配置POM文件

POM文件的基本结构

POM文件是一个XML文件，包含多个元素，每个元素代表一个特定的配置项。下面是一个POM文件的基本结构

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
         http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>example-project</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <!-- 依赖项配置 -->
    </dependencies>

    <build>
        <!-- 构建配置 -->
    </build>

    <repositories>
        <!-- 仓库配置 -->
    </repositories>
</project>
```

POM文件的常用配置项

坐标信息

坐标信息指的是、和元素。定义了项目的组织ID，定义了项目的唯一标识符，定义了项目的版本号。这些信息对于Maven的依赖管理和构建过程非常重要

```
<groupId>com.example</groupId>
<artifactId>example-project</artifactId>
<version>1.0-SNAPSHOT</version>
```

依赖项配置

依赖项配置用于定义项目所依赖的外部库。Maven会自动下载并管理这些依赖项。依赖项配置包含在元素中，每个依赖项使用一个元素进行描述

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.3.13</version>
    </dependency>
</dependencies>
```

> dependencies与dependencyManagement的区别
>
> **dependencyment**
>
> 在我们项目中，我们会发现在父模块的pom文件中常常会出现dependencyMent元素，这是因为我们可以通过其来管理子模块的版本号，也就是说我们在父模块中声明号依赖的版本，但是并不实现引入；

> **dependencies**
>
> 上面说到dependencyment只是声明一个依赖，而不实现引入，故我们在子模块中也需要对依赖进行声明，倘若不声明子模块自己的依赖，是不会从父模块中继承的；只有子模块中也声明了依赖。并且没有写对应的版本号它才会从父类中继承；并且version和scope都是取自父类；此外要是子模块中自己定义了自己的版本号，是不会继承自父类的
>
> **总结**
>
> dependencyment只是用来管理依赖，规定未添加版本号的子模块依赖继承自它，dependencies是用来声明子模块自己的依赖，可以在其中来写自己需要的版本号；

构建配置

构建配置用于定义项目的构建过程。它包含在元素中，包括了多个子元素。其中比较常用的子元素有

```
<build>
    <plugins>
        <!-- 插件配置 -->
    </plugins>
    <resources>
        <!-- 资源文件配置 -->
    </resources>
    <testResources>
        <!-- 测试资源文件配置 -->
    </testResources>
</build>
```

插件配置

插件是Maven的一个重要特性，它可以用于扩展Maven的功能。Maven自带了一些插件，比如maven-compiler-plugin、maven-jar-plugin等。插件配置包含在元素中，每个插件使用一个元素进行描述

```
<plugins>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
            <source>1.8</source>
            <target>1.8</target>
        </configuration>
    </plugin>
</plugins>
```

仓库配置

仓库配置用于定义Maven的依赖项下载地址。Maven默认使用中央仓库，但是也可以配置私有仓库或者本地仓库。仓库配置包含在元素中，每个仓库使用一个元素进行描述

```
<repositories>
    <repository>
        <id>central</id>
        <url>https://repo.maven.apache.org/maven2</url>
    </repository>
</repositories>
```

父子 POM 示例

Maven 父 POM （或超级 POM）用于构造项目，以**避免重复或重复使用 pom 文件之间的*继承配置***。 它有助于长期轻松维护。

如果在父 POM 和子 POM 中都使用不同的值配置了任何依赖项或属性，则子 POM 值将具有优先级

父 POM 内容

可以使用包 `pom` 声明父 POM。 它不打算分发，因为仅从其他项目中引用了它。

Maven 父 pom 可以包含几乎所有内容，并且可以继承到子 pom 文件中，例如：

- 通用数据 – 开发人员的姓名，SCM 地址，分发管理等
- 常数 – 例如版本号
- 共同的依赖项 – 所有子项共同的。 与在单个 pom 文件中多次写入它们具有相同的效果。
- 属性 – 例如插件，声明，执行和 ID。
- 配置
- 资源

父 POM 和子 POM 示例

为了匹配父 POM，Maven 使用两个规则：

1. 在项目的根目录或给定的相对路径中有一个 pom 文件。
2. 子 POM 文件中的引用包含与父 POM 文件中所述相同的坐标。

父 POM

此处，父 POM 为 JUnit 和 spring 框架配置了基本项目信息和两个依赖项。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd;
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.howtodoinjava.demo</groupId>
    <artifactId>MavenExamples</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>pom</packaging>

    <name>MavenExamples Parent</name>
    <url>http://maven.apache.org</url>
```

```
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <junit.version>3.8.1</junit.version>
        <spring.version>4.3.5.RELEASE</spring.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>

    </dependencies>
</project>
```

子 POM

现在，子 POM 需要使用 `parent` 标签并指定 `groupId/artifactId/version` 属性来引用父 POM。 这个 pom 文件将从父 POM 继承所有属性和依赖项，并且还可以包括子项目特定的依赖项。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

    <!--The identifier of the parent POM-->
    <parent>
        <groupId>com.howtodoinjava.demo</groupId>
        <artifactId>MavenExamples</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>MavenExamples</artifactId>
    <name>MavenExamples Child POM</name>
    <packaging>jar</packaging>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-security</artifactId>
            <version>${spring.version}</version>
        </dependency>
    </dependencies>
```

```
</project>
```

父 POM 相对路径

默认情况下，Maven 首先在项目的根目录下查找父 POM，然后在本地仓库中查找，最后在远程仓库中查找。 如果父 POM 文件不在任何其他位置，则可以使用代码标签。 该**相对路径应相对于项目根。**

如果未明确给出相对路径，则默认为`..`，即当前项目的父目录中的 pom。

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

    <!--The identifier of the parent POM-->
    <parent>
        <groupId>com.howtodoinjava.demo</groupId>
        <artifactId>MavenExamples</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <relativePath>../baseapp/pom.xml</relativePath>
    </parent>

    <modelVersion>4.0.0</modelVersion>
    <artifactId>MavenExamples</artifactId>
    <name>MavenExamples Child POM</name>
    <packaging>jar</packaging>

</project>
```

maven修改镜像地址

在maven中的conf文件中，找到settings.xml文件，将文件中的 mirrors 标签中 添加如下代码，即可将镜像地址改成其他，加快下载速度

```
<mirror>
     <id>alimaven</id>
     <name>aliyun maven</name>
     <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
     <mirrorOf>central</mirrorOf>
</mirror>
```

MongoClients是怎么实现MongoClient的