物联云平台架构设计

修订记录：

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **修订版本** | **描述** | **作 者** |
| 2017-4-14 | REV.0 | 初次发布 | 马 翔 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[产品服务 4](#_Toc482748534)

[让设备轻松接入云端 5](#_Toc482748535)

[保护设备和数据 5](#_Toc482748536)

[存储设备数据 6](#_Toc482748537)

[计算设备数据 6](#_Toc482748538)

[产品优势 6](#_Toc482748539)

[产品架构 7](#_Toc482748540)

[设备SDK 8](#_Toc482748541)

[设备网关 8](#_Toc482748542)

[身份验证和授权 8](#_Toc482748543)

[物接入 9](#_Toc482748544)

[物管理 10](#_Toc482748545)

[物解析 10](#_Toc482748546)

[设备影子 11](#_Toc482748547)

[规则引擎 12](#_Toc482748548)

[产品组件 12](#_Toc482748549)

[物接入 12](#_Toc482748550)

[物解析 16](#_Toc482748551)

[模块 17](#_Toc482748552)

[物管理 17](#_Toc482748553)

[模块 20](#_Toc482748554)

[API 28](#_Toc482748555)

[模块 29](#_Toc482748556)

[规范 29](#_Toc482748557)

[附录 29](#_Toc482748558)

[HTTP Status Code 29](#_Toc482748559)

[通用错误码 30](#_Toc482748560)

[C Data Type 30](#_Toc482748561)

[Microsoft C++ Data Type Ranges 31](#_Toc482748562)

[struct — Interpret bytes as packed binary data 34](#_Toc482748563)

物联网 (IoT) 是 Kevin Ashton 创造的一个术语，他是英国射频识别 (RFID) 领域的技术先驱，构想出了一种通过无处不在的传感器将现实世界与互联网连在一起的系统。虽然物品、互联网和连接是 IoT 的三个核心组件，但其价值却是在自我增强和自我完善的系统中消除现实与数字世界之间的鸿沟。物联网通过将具有唯一的标识的物品接入互联网， 构成一个具有一定上下文信息，物品能够感知网络，感知自身，感知所处环境的系统。通过丰富的数据集和高级分析，物联网能让我们对世界有更清晰深入的认识：检测风力涡轮机叶片的震动，并进行实时分析，能在叶片失效前就进行维护；通过控制建筑内无人楼层的灯光，减少能源损耗；制造自动驾驶汽车，处理环境信息并能够进行瞬时决策，避免事故的发生。通过物联网，得到基于真实世界的集体智慧，促进效率提高，产生新的商业模型，更低的污染以及更健康的环境。

当我们能够了解世上万物的情况且能够以数据说话，就能够解决以上问题。为了解决这些问题并获享互联设备的价值，构建智能物联网平台。物联网平台提供的功能及服务有助于收集物品数据并将其发送到云中，还能轻松加载和分析信息以及管理设备，这样便可开发适合特定需求的应用程序。

物联云平台（Internet of Things）为设备提供与云应用程序以及其他设备交互的平台，通过搭建安全性能强大的数据通道，方便终端（如传感器、执行器、嵌入式设备或智能家电等等）和云端的双向通信。多节点部署让海量设备低延时接入，并提供安全可靠的消息处理能力。物联云平台提供一站式服务，满足数据计算、存储、加速、分析等多样化需求，快速搭建稳定可靠高效的物联网络。

## 产品服务

传统企业基于物联网进行业务创新，通过运营设备数据来实现效益的提升。这样的做法基本上已经是行业的共识，大势所趋。但是企业的物联网转型或者建设物联网平台还是会遇到各种各样的阻碍，这已经严重制约了物联网的发展。对此，针对行业的痛点，提供了一系列服务帮助企业建设物联网平台。

### 让设备轻松接入云端

首先，企业的物联网转型会遇到设备接入云端的痛点，尤其是大规模设备的接入，云端必然要考虑分布式架构，网络延时性，设备安全性等等问题，这些问题成了第一道拦路虎。所以，首当其冲，物联云平台就要解决这个问题，提供设备端SDK让设备轻松接入。实现大规模的设备接入云端，保障设备与云端通信的质量，性能，安全以及网络等等问题。提供设备管理服务

设备接入云端之后，当然要知道哪个设备接上来了，这时云端就要提供设备管理的服务。套件提供树形结构的设备管理模型，以及提供设备的状态管理。

### 保护设备和数据

安全是物联网最重要的话题。物联网套件在安全上做了很多事情，包括在所有节点提供身份验证和端到端加密服务，这些节点包括设备端和物联云平台各个云服务。换句话说，如果没有通过身份验证，是不能够进行节点间的数据交换的，即意味着设备无法与物联网套件通信。此外，物联网套件还提供了设备级的权限粒度服务，这个服务保证设备或者应用程序只有具有相应的访问权限，才能操作某些资源，简单来说，你想控制某个设备或者你想拿到某个设备的数据，都必须具有相应的权限，不是你想拿就能拿的。

### 存储设备数据

企业要进行物联网业务创新，必然需要对设备数据进行存储分析。物联云平台提供采集+存储的完整解决方案。

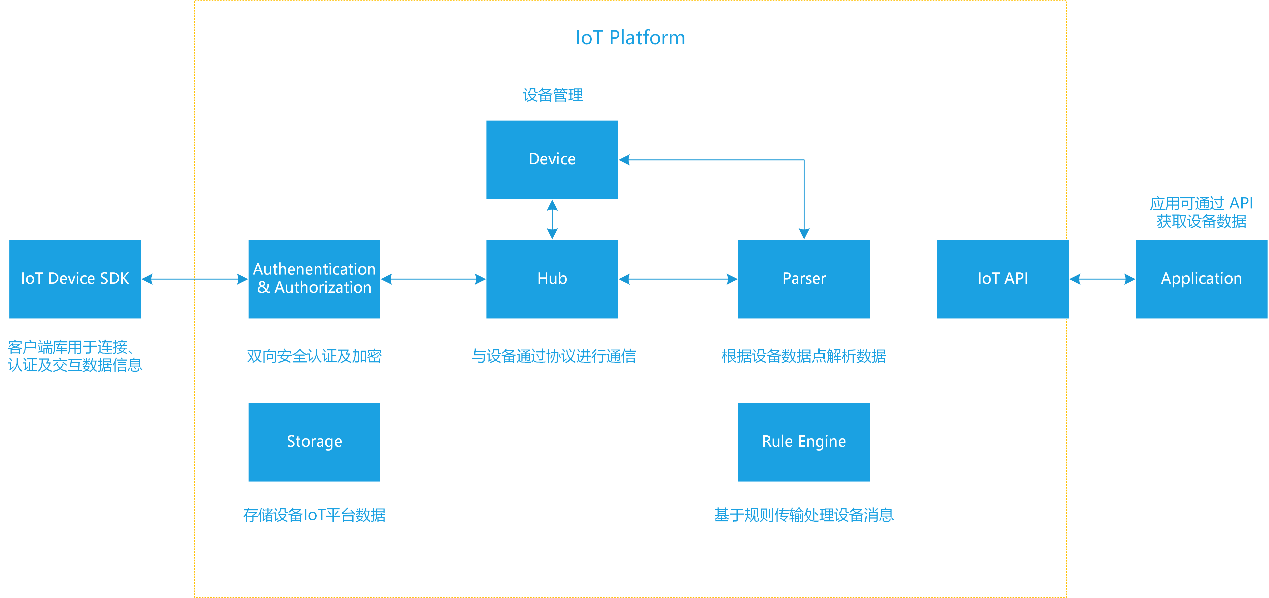
### 计算设备数据

有了数据，必然少不了计算。物联云平台会通过规则引擎与计算服务无缝打通，例如流式计算，大规模计算。具体的来说，就是将接入的设备数据按照意愿转发到计算产品中，就可以对设备数据进行各种各样的计算，可以是实时计算，也可以是离线计算。

## 产品优势

| **优势** | **物联云平台 IoT** | **传统物联网** |
| --- | --- | --- |
| 安全 | 设备接入可信，数据传输安全 | 需要专门部署与管理证书服务器，开发中处理传输加解密。 |
| 稳定 | 稳定支持海量服务，链路传输可靠 | 投入专门运维人员，不仅关注业务还需要关于网络连接等基础服务。 |
| 快速 | 快速接入物联网，开发省时省力 | 构建物联网网络架构，开发设备网关等功能。 |
| 高效 | 设备管理高效，提升企业效率 | 从终端接入、设备网关实现到数据处理等基础服务都需要投入研发。 |
| 开放 | 采用行业标准协议，避免厂商锁定 | 私有协议与接口，高成本对接不同厂商。 |
| 便捷 | 云服务无缝对接，迅速提升能力 | 需要自主开发数据处理分析套件。 |

## 产品架构



### **设备SDK**

IoT 提供 SDK，以帮助轻松快速地连接硬件设备或移动应用程序。利用 IoT 设备 SDK，设备可以使用 MQTT、HTTP 或 WebSockets 协议连接和验证并交换消息。

### 设备网关

IoT 设备网关支持设备安全高效地与 IoT 进行通信。设备网关可以使用发布/订阅模式交换消息，从而支持一对一和一对多的通信。凭借此一对多的通信模式，IoT 将支持互连设备向多名给定主题的订阅者广播数据。设备网关支持 MQTT、WebSocket 和 HTTP 1.1 协议，可以轻松实现对专有或陈旧协议的支持。设备网关可自动扩展，以支持 10 亿多台设备，而无需预配置基础设施。

### 身份验证和授权

物联云平台在所有连接点处提供相互身份验证和加密，因此，绝不会在无可靠身份的设备和 IoT 之间交换数据。IoT 支持私有身份验证方法以及基于身份验证的 X.509 证书。使用 HTTP 的连接可以使用任一方法，使用 MQTT 的连接可以使用基于证书的身份验证，使用 WebSockets 的连接可以使用私有身份验证方法。借助 IoT，您可以使用 IoT 生成的证书以及由首选证书颁发机构 (CA) 签署的证书。可以将所选的角色和/或策略映射到每个证书，以便授予设备或应用程序访问权限，或在改变主意时撤消访问权限（甚至都不需触碰设备）。80px-space

### 物接入

物接入帮助建立设备与云端之间安全可靠的双向连接，以支撑海量设备的数据收集、监控、故障预测等各种物联网场景。

安全可靠的双向连接:物联网服务是全托管的服务，用户可以快速创建物联网服务的实例并安全可靠地连接设备与云端并而不用为运维操心。

认证与授权:提供设备级别的认证，以及基于策略的授权，能够通过 SSL 保证数据安全传输，允许控制设备对特定主题的读写等权限，保障物联网应用的安全

支持主流物联网协议:支持 MQTT 标准物联网协议，TCP 透传自定义解析协议，Modbus 协议及 HTTP RESTful API，持从设备到云端安全可靠地传输大规模消息，也可以从云端向设备安全地发送命令

数据分析:提供开放标准的API，可通过调用API实现控制台操作，方便第三方应用快速集成云端服务，无缝连接物联网服务与大数据服务，通过大数据来处理分析收集的遥感数据，驱动业务的升级与转型。

### 物管理

提供覆盖设备全生命周期的、一站式的设备管理服务，包括设备的层级管理、监测、遥控、固件升级和维护保养等各种场景。

管理中心:通过可视化界面 Web UI 来管理企业设备，依据企业组织架构和业务流程，提供基于层级的设备管理模型。

注册管理:轻松创建和删除设备，有效识别设备的合法身份，并将之纳入设备层级体系中进行管理

设备影子:设备在云端的映射，对元数据的管理和操作，时监测设备状态，并提供反控能力

操作管理:支持对设备在线操作，大大提高效率并降低成本，重启设备、设备属性读取和写入；提供开放标准的API，可通过调用API实现控制台操作，方便第三方应用快速集成云端服务

### 物解析

简单快速完成各种设备数据协议解析，如TCP、HTTP、Modbus、OPC等。

通过物解析服务，可在云端解析和计算各种设备中的数据，大大节约数据流量，并降低设备成本。在云端解析数据，可以随时统一调整解析规则以适应业务的变化。利用云端强大的弹性计算能力、无限的数据存储能力和丰富多样的数据分析方案，帮助企业核心业务更加平稳高效运行，并激发更多融合创新。

IoT SDK:在设备端集成 SDK 后，设备即可通过物管理生成的密钥 SECRET 与云端通讯。

IoT Edge SDK:在网关端集成IoT Edge SDK后，只需要配置一个云端生成的密钥即可与云端连接，实现与云端通讯配置。

设备轮询管理:支持通过云端控制台的可视化Web UI操作或者Restful API调用方式选择要向哪些设备发送轮询配置。

协议数据解析:负责把网关或设备发到云端的设备原始数据（一般为未解析的二进制数据包）进行数据解析，然后进行数据存储，以进行后续的分析工作。

### 设备影子

凭借 IoT，您可以创建每台设备的持久虚拟版（或“影子”），它包含设备的最新状态，因此应用程序或其他设备可以读取消息并与此设备进行交互。设备影子保留每台设备的最后报告状态和期望的未来状态，即便设备处于离线状态。您可以通过 API 或使用规则引擎，获取设备的最后报告状态或设置期望的未来状态。

设备影子提供始终可用的 REST API，使得构建与您的设备进行交互的应用程序更加轻松。此外，应用程序可以设置设备的期望未来状态，而无需说明设备的当前状态。IoT 将比较期望未来状态和最后报告状态之间的差异，并命令设备“弥补差异”。

### 规则引擎

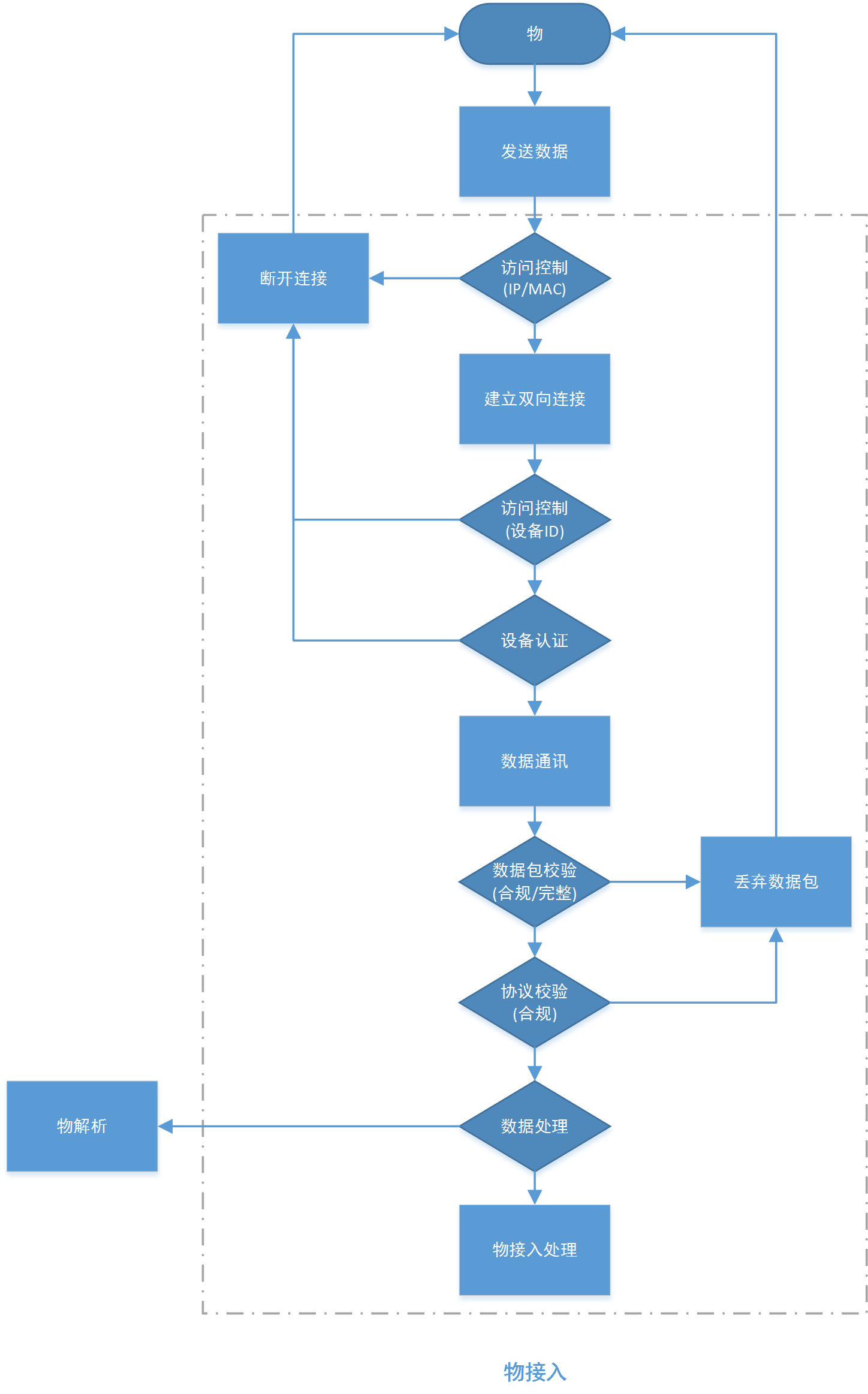
规则引擎可以构建 IoT 应用程序，这些应用程序将收集、处理和分析互连设备在全局范围内生成的数据并根据数据执行操作，且无需管理任何基础设施。规则引擎评估发布到 IoT 的入站消息，并根据定义的业务规则转换这些消息并将它们传输到另一台设备或云服务。规则可以应用至一台或多台设备中的数据，并且它可以并行执行一个或多个操作。

可以在管理控制台中或者使用类似 SQL 的语句编写规则。规则可以编写为采用不同的方式表示，具体取决于消息内容。例如，如果温度读数超出特定阈值。规则还可以编写为考虑云中的其他数据，例如其他设备中的数据。例如，您可以在规则中编写，如果此温度超出其他 5 台设备的平均值 15%，则应采取措施。

## 产品组件

### 物接入

物接入帮助建立设备与云端之间安全可靠的双向连接。



#### 模块

##### 负载均衡

以腾讯云为例，有两种解决方案：基于A记录和用云解析实现负载均衡。A记录负载均衡原理是给用户返回不同的IP地址，解析返回得到的IP地址是轮询随机得到的IP地址，不会根据服务器负载和运行状况进行分配；如果在同一线路下有多个服务器，需要将访问流量均摊到各个服务器上，可以使用云解析来实现负载均衡。

##### 访问控制（基于IP和MAC）

通过Nginx+Lua、OpenResty及iptables实现基于IP地址和MAC地址的访问控制及防火墙，通过黑白名单控制连接或恶意攻击，避免接入服务压力过载。

黑名单规则：

* 多次发送不合规数据
* 高频率（超出规定采集周期）发送数据
* 高频率（超出规定采集周期）发送不合规数据

##### 访问控制（基于设备通讯ID）

通过解析数据包，获取设备通讯ID，基于设备历史通讯行为，控制设备与云平台通讯，并与上一层访问控制联动。

黑名单规则：

* 多次与云平台双向验证失败的设备
* 验证成功但重复发送不合规数据的设备
* 频繁建立TCP半连接，但多次超时不发送数据的设备

##### 设备认证

根据通信协议规定的验证命令数据包规范，获取设备通讯ID及设备密钥，根据所选择的安全认证方法，进行安全认证，认证成功，则设备与云平台可保持长连接，进行数据交互；认证失败，则平台主动断开连接。

为了满足未来的安全需求，需要设计基于ID及密钥的多种安全加密方法。

##### 数据包校验

根据通信协议，判断数据包长度不大于1024字节，并且根据CRC校验值判断包头及数据的完整性。

根据通信协议，判断数据包头中协议命令是否合规。

当以上访问控制、认证校验均成功后，将数据包转发至物解析进行下一步处理。

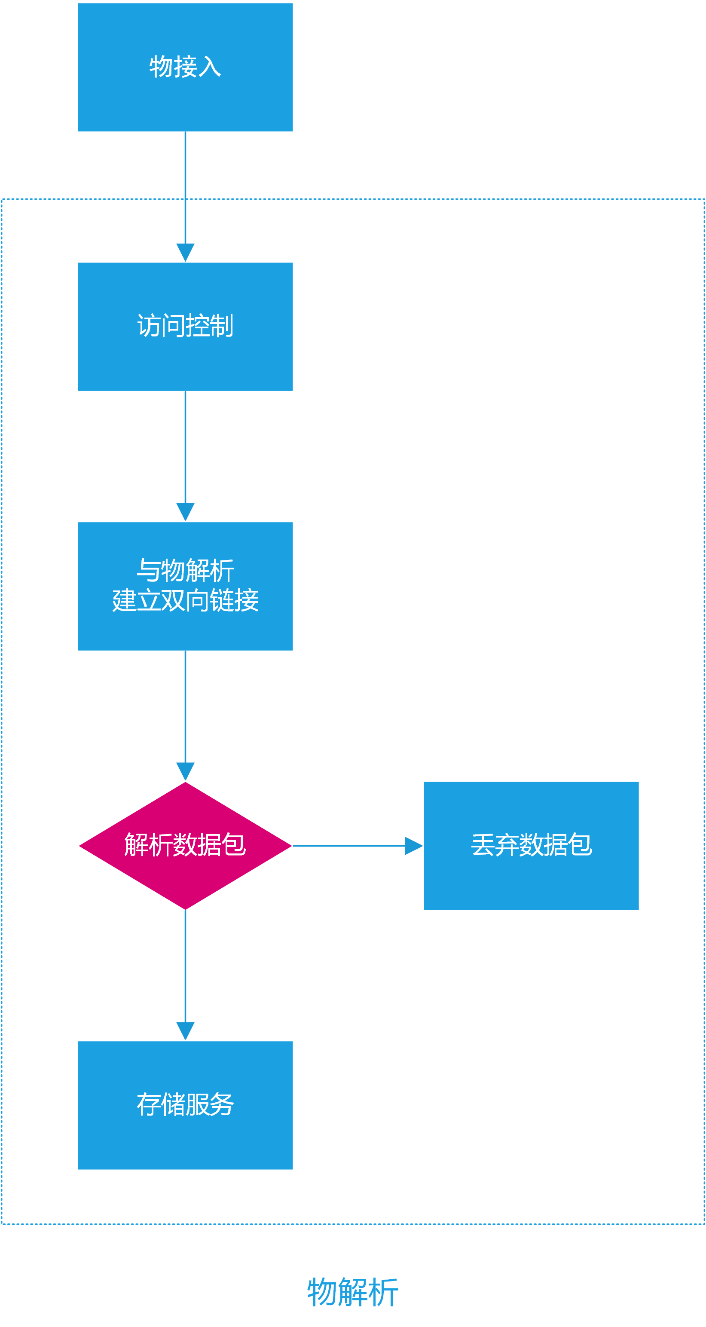
#### 开发技术选型

鉴于团队初创，为简化开发难度，统一技术栈，选择Twisted。依据项目实际进展情况，测试Eventlet及gevent等支持并发的网络库性能，与Linux调优同时进行，最终将整个接入服务向现代并发语言（如Golang 、Scala、Rust等）转换。

为了保证开发测试与线上环境一致，团队需部署测试环境，或团队成员使用统一的虚拟环境，如Vagrant + VirtualBox实现。

### 物解析

简单快速完成各种设备数据协议解析，如TCP、HTTP、Modbus、OPC等。



#### 模块

##### 访问控制

只有已注册的物接入服务才能与物解析进行数据交互，保证数据合法安全可靠传输。

##### 解析数据

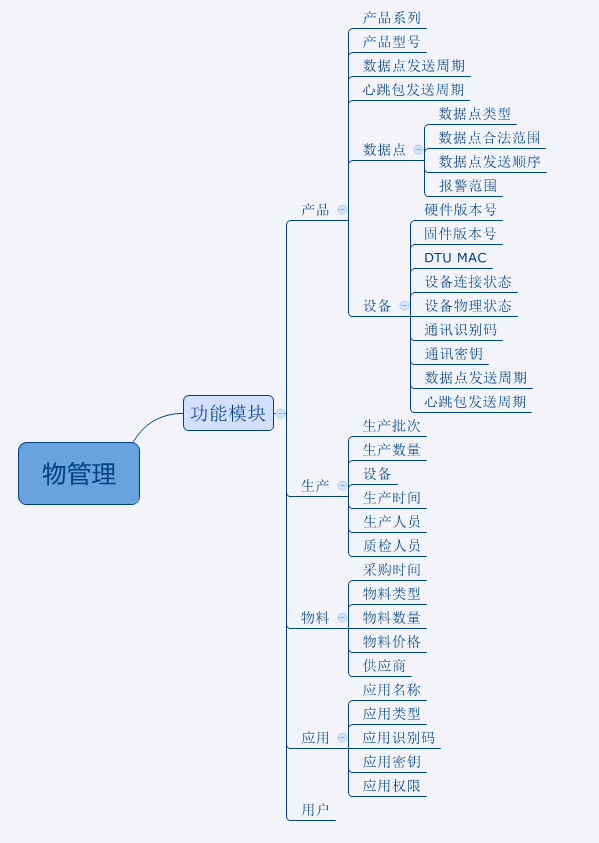
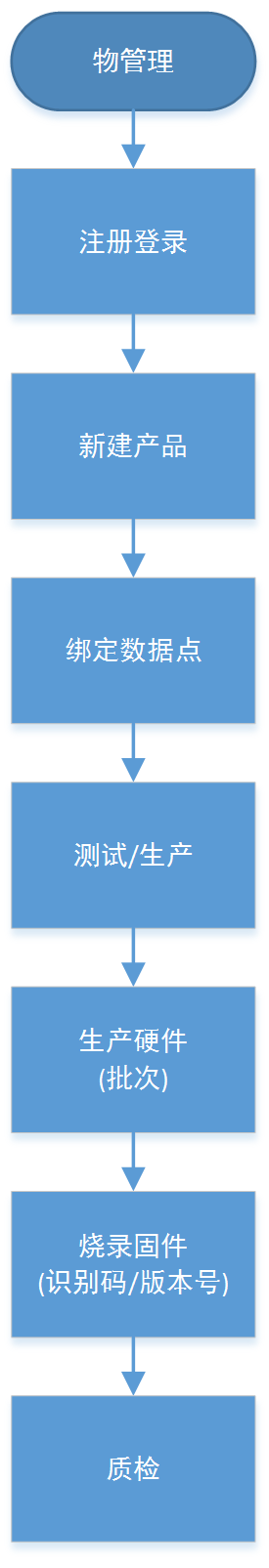
根据通讯协议，根据数据传输命令及设备数据点定义,如数据点顺序、数据点类型、数据点范围等参数，解析数据包，解析成功，转发至云平台存储服务，否则丢弃此数据包并进行记录。

#### 开发技术选型

与物接入选型相似，初期选择Twisted。依据项目实际进展情况，向现代并发语言过渡，同时进行Linux系统调优。

### 物管理

提供覆盖设备全生命周期的、一站式的设备管理服务，包括设备的层级管理、监测、遥控、固件升级和维护保养等各种场景。



#### 模块

##### 产品管理

研发人员立项，新建产品类型、产品系列、产品备注及产品识别码。通过权限管理，多个研发人员可管理同一产品。

数据表

**Product**

Table name: products

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| model | VARCHAR(60) | NN | 产品型号 |
| serie\_id | INT | FK,NN,UN |  |
| stage | SMALLINT | NN | 0-研发，1-生产 |
| measurement\_period | SMALLINT |  | 采集周期 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator | INT | NN |  |

**Product Serie**

Table name: product\_series

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| name | VARCHAR(60) | NN | 产品系列 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator\_id | INT | NN |  |

##### 数据点管理

产品数据点设置，包含数据点类型，数据类型，合法数据范围，数据处理规则。

数据点发送周期，初始化时，所有设备数据点发送周期与产品数据点发送周期一致；实际使用时，根据需要每个设备的数据点发送周期不尽相同。

生成供设备使用的标准数据解析收发库。

数据表

**Datapoint**

Table name: datapoints

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| name | VARCHAR(60) | NN | 标识名称（英文） |
| label | VARCHAR(60) |  | 显示名称 |
| resource\_type | SMALLINT |  | 0-上报/资源数据点，1-控制数据点 |
| method | SMALLINT |  | 0-只读，1-可写 |
| data\_type | SMALLINT |  | 数据类型 |
| data\_max | DECIMAL |  |  |
| data\_min | DECIMAL |  |  |
| precision | DECIMAL |  | 精度，实际值=采集值\*精度 |
| offset |  |  | 实际值=（采集值+偏移量）\*精度 |
| order | SMALLINT |  | 数据点顺序 |
| product\_id | INT | FK,NN,UN | 产品 |
| description | VARCHAR(256) |  | 数据点描述 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator\_id | INT | NN |  |

##### 设备管理

数据点

**Device**

Table name: devices

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| uuid | VARCHAR(256) | NN | 唯一识别码 |
| token | VARCHAR(256) | NN | 令牌 |
| mac | VARCHAR(256) |  | DTU、网卡物理地址 |
| latest\_online\_date | TIMESTAMP/ DATETIME |  | 最新上线时间 |
| latest\_offline\_date | TIMESTAMP/ DATETIME |  | 最新离线时间 |
| firmware\_id | INT | FK,NN,UN | 固件 |
| hardware\_id | INT | FK,NN,UN | 硬件 |
| batch\_id | INT | FK,NN,UN | 批次 |
| product\_id | INT | FK,NN,UN | 产品 |
| installation | INT |  | 0-未安装，1-安装 |
| installation\_date | TIMESTAMP/ DATETIME |  | 安装日期 |
| longtitude | INT |  | 经度，实际值=存储\* 10^-精度 |
| latitude | INT |  | 纬度，实际值=存储\* 10^-精度 |
| quality\_inspector | INT | FK,NN,UN |  |
| measurement\_period | SMALLINT |  | 采集周期 |
| *public\_key* |  |  |  |
| *private\_key* |  |  |  |
| created | TIMESTAMP/ DATETIME | NN | 生产日期 |
| modified | TIMESTAMP/ DATETIME | NN |  |

**Firmware**

Table name: firmwares

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| local\_file\_path | VARCHAR(256) |  | 本地存储 |
| remote\_file\_pat | VARCHAR(256) |  | 远程存储路径 |
| major | SMALLINT | NN,UN | 主版本号 |
| minor | SMALLINT | NN,UN | 子版本号 |
| revision | SMALLINT | NN,UN | 修正版本号 |
| build | SMALLINT | NN,UN | 编译版本号 |
| version | VARCHAR(60) | NN,UN | 版本号 |
| product\_id | INT | FK,NN,UN | 产品 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator\_id | INT | NN |  |

##### 生产管理

生产人员根据实际物料及下游需求，设置生产批次及数量，组装硬件产品并写入固件程序；质检人员保证产品质量良好，运行正常。

数据表

**Product Batch**

Table name: product\_batches

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| number | VARCHAR(60) | NN | 批次号 |
| product\_id | INT | FK,NN,UN | 产品 |
| description | VARCHAR(256) |  | 批次描述 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator\_id | INT | NN |  |

##### 物料管理

为了保证当产品出现问题时，能够对设备硬件进行溯源，保留硬件采购记录是非常有必要的。

##### 应用管理

为了让物联云平台的数据能够以更直观、合理及用户友好的方式展示给最终用户，需要为每一个行业应用进行开发，如Web App及Mobile App，为了对应用的数据获取进行访问控制，需要事先在云平台中配置应用的访问权限及通讯密钥。

数据表

**Applciation**

Table name: applications

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype |  | Comments |
| id | INT | PK, AI,UN |  |
| name | VARCHAR(60) | NN | 应用名称 |
| uuid | VARCHAR(256) | NN | 唯一识别码 |
| key | VARCHAR(256) | NN | API KEY |
| secret | VARCHAR(256) | NN | API SECRET |
| platform | SMALLINT | NN | 平台 |
| level | SMALLINT | NN | 权限等级 |
| description | VARCHAR(256) |  | 应用描述 |
| product\_id | INT | NN | 产品 |
| created | TIMESTAMP/ DATETIME | NN |  |
| modified | TIMESTAMP/ DATETIME | NN |  |
| creator\_id | INT | NN |  |

##### 用户管理

研发、生产及质检人员可对与产品生产流程相关的功能进行相应的操作。

云平台管理人员可实时监控物联云平台运行状态。

### API

1. API认证统一使用OAuth2
2. 数据返回格式统一使用JSON，返回代码参考附录
3. 可接受callback参数，使返回的数据为jsonp。callback只能包含数字、字母、下划线，长度不大于50
4. 需要授权的API，需要加access\_token的Header，并且使用https协议。不需要授权公开api可以使用http，参数里面如果不包含apikey的话，限制单ip每小时150次，带的话每小时500次。
5. API里面的通配符，:id代表纯数字， :name代表由数字+字母+[-\_.]这些特殊字符
6. 使用HTTP Status Code表示状态
7. 列表参数使用start和count
8. POST/PUT 时中文使用UTF-8编码
9. 时间格式：yyyy-MM-dd HH:mm:ss, 如"2007-06-28 11:16:11"

#### 模块

##### 产品

查询产品信息，返回产品所属设备。

##### 设备

查询设备状态，控制设备，修改设备数据发送周期，查询设备历史数据。

### 规则引擎

规则引擎帮助用户灵活地处理设备消息，用户可通过规则引擎设定消息处理规则，对指定的消息采取相应的措施来对设备进行监控和处理，如发送预警或报警信息、推送给Mobile APP等，也可以将设备消息无缝转发到时序数据库、关系型数据库和对象存储中进行存储。

### 存储

提供物联云平台存储服务，提供关系存储及NoSQL存储功能,集中数据管理，保证数据一致性，提高运维方便性。

#### 开发技术选型

使用主流数据存储技术，关系型数据存储采用MySQL数据库，可使用MongoDB进行数据缓存，类Key-Value型数据及缓存采用Redis或MongoDB进行存储。合理选择架构（主从、热备）方案，提供安全稳定、支持高并发、高可用且具有一定容灾性能的存储服务。

# 规范

[The OAuth 2.0 Authorization Framework](https://tools.ietf.org/html/rfc6749)

[MQTT V3.1 Protocol Specification](http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html)

# 附录

## HTTP Status Code

|  |  |  |
| --- | --- | --- |
| 状态码 | 含义 | 说明 |
| 200 | OK | 请求成功 |
| 201 | CREATED | 创建成功 |
| 202 | ACCEPTED | 更新成功 |
| 400 | BAD REQUEST | 请求的地址不存在或者包含不支持的参数 |
| 401 | UNAUTHORIZED | 未授权 |
| 403 | FORBIDDEN | 被禁止访问 |
| 404 | NOT FOUND | 请求的资源不存在 |
| 500 | INTERNAL SERVER ERROR | 内部错误 |

## 通用错误码

|  |  |  |  |
| --- | --- | --- | --- |
| 错误码 | 错误信息 | 含义 | status code |
| 999 | unknow\_v2\_error | 未知错误 | 400 |
| 1000 | need\_permission | 需要权限 | 403 |
| 1001 | uri\_not\_found | 资源不存在 | 404 |
| 1002 | missing\_args | 参数不全 | 400 |
| 1003 | input\_too\_short | 输入为空，或者输入字数不够 | 400 |
| 1004 | target\_not\_fount | 相关的对象不存在 | 400 |

## C Data Type

|  |  |  |
| --- | --- | --- |
| **Type** | **Storage size** | **Value range** |
| char | 1 byte | -128 to 127 or 0 to 255 |
| unsigned char | 1 byte | 0 to 255 |
| signed char | 1 byte | -128 to 127 |
| int | 2 or 4 bytes | -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 |
| unsigned int | 2 or 4 bytes | 0 to 65,535 or 0 to 4,294,967,295 |
| short | 2 bytes | -32,768 to 32,767 |
| unsigned short | 2 bytes | 0 to 65,535 |
| long | 4 bytes | -2,147,483,648 to 2,147,483,647 |
| unsigned long | 4 bytes | 0 to 4,294,967,295 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Storage size** | **Value range** | **Precision** |
| float | 4 byte | 1.2E-38 to 3.4E+38 | 6 decimal places |
| double | 8 byte | 2.3E-308 to 1.7E+308 | 15 decimal places |
| long double | 10 byte | 3.4E-4932 to 1.1E+4932 | 19 decimal places |

## Microsoft C++ Data Type Ranges

<https://msdn.microsoft.com/en-us/library/s3f49ktz.aspx>

| **Type Name** | **Bytes** | **Other Names** | **Range of Values** |
| --- | --- | --- | --- |
| int | 4 | signed | –2,147,483,648 to 2,147,483,647 |
| unsigned int | 4 | unsigned | 0 to 4,294,967,295 |
| \_\_int8 | 1 | char | –128 to 127 |
| unsigned \_\_int8 | 1 | unsigned char | 0 to 255 |
| \_\_int16 | 2 | short, short int, signed short int | –32,768 to 32,767 |
| unsigned \_\_int16 | 2 | unsigned short, unsigned short int | 0 to 65,535 |
| \_\_int32 | 4 | signed, signed int, int | –2,147,483,648 to 2,147,483,647 |
| unsigned \_\_int32 | 4 | unsigned, unsigned int | 0 to 4,294,967,295 |
| \_\_int64 | 8 | long long, signed long long | –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| unsigned \_\_int64 | 8 | unsigned long long | 0 to 18,446,744,073,709,551,615 |
| bool | 1 | none | false or true |
| char | 1 | none | –128 to 127 by default  0 to 255 when compiled by using [/J](https://msdn.microsoft.com/en-us/library/0d294k5z.aspx) |
| signed char | 1 | none | –128 to 127 |
| unsigned char | 1 | none | 0 to 255 |
| short | 2 | short int, signed short int | –32,768 to 32,767 |
| unsigned short | 2 | unsigned short int | 0 to 65,535 |
| long | 4 | long int, signed long int | –2,147,483,648 to 2,147,483,647 |
| unsigned long | 4 | unsigned long int | 0 to 4,294,967,295 |
| long long | 8 | none (but equivalent to \_\_int64) | –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| unsigned long long | 8 | none (but equivalent to unsigned \_\_int64) | 0 to 18,446,744,073,709,551,615 |
| enum | varies | none | See Remarks later in this article |
| float | 4 | none | 3.4E +/- 38 (7 digits) |
| double | 8 | none | 1.7E +/- 308 (15 digits) |
| long double | same as double | none | Same as double |
| wchar\_t | 2 | \_\_wchar\_t | 0 to 65,535 |

## struct — Interpret bytes as packed binary data

Format strings are the mechanism used to specify the expected layout when packing and unpacking data. They are built up from [Format Characters](https://docs.python.org/3/library/struct.html#format-characters), which specify the type of data being packed/unpacked. In addition, there are special characters for controlling the [Byte Order, Size, and Alignment](https://docs.python.org/3/library/struct.html#struct-alignment).

**7.1.2.1. Byte Order, Size, and Alignment**

By default, C types are represented in the machine’s native format and byte order, and properly aligned by skipping pad bytes if necessary (according to the rules used by the C compiler).

Alternatively, the first character of the format string can be used to indicate the byte order, size and alignment of the packed data, according to the following table:

| **Character** | **Byte order** | **Size** | **Alignment** |
| --- | --- | --- | --- |
| @ | native | native | native |
| = | native | standard | none |
| < | little-endian | standard | none |
| > | big-endian | standard | none |
| ! | network (= big-endian) | standard | none |

If the first character is not one of these, '@' is assumed.

Native byte order is big-endian or little-endian, depending on the host system. For example, Intel x86 and AMD64 (x86-64) are little-endian; Motorola 68000 and PowerPC G5 are big-endian; ARM and Intel Itanium feature switchable endianness (bi-endian). Use sys.byteorder to check the endianness of your system.

Native size and alignment are determined using the C compiler’s sizeof expression. This is always combined with native byte order.

Standard size depends only on the format character; see the table in the [Format Characters](https://docs.python.org/3/library/struct.html#format-characters) section.

Note the difference between '@' and '=': both use native byte order, but the size and alignment of the latter is standardized.

The form '!' is available for those poor souls who claim they can’t remember whether network byte order is big-endian or little-endian.

There is no way to indicate non-native byte order (force byte-swapping); use the appropriate choice of '<' or '>'.

Notes:

1. Padding is only automatically added between successive structure members. No padding is added at the beginning or the end of the encoded struct.
2. No padding is added when using non-native size and alignment, e.g. with ‘<’, ‘>’, ‘=’, and ‘!’.
3. To align the end of a structure to the alignment requirement of a particular type, end the format with the code for that type with a repeat count of zero. See [Examples](https://docs.python.org/3/library/struct.html#struct-examples).

**7.1.2.2. Format Characters**

Format characters have the following meaning; the conversion between C and Python values should be obvious given their types. The ‘Standard size’ column refers to the size of the packed value in bytes when using standard size; that is, when the format string starts with one of '<', '>', '!' or '='. When using native size, the size of the packed value is platform-dependent.

| **Format** | **C Type** | **Python type** | **Standard size** | **Notes** |
| --- | --- | --- | --- | --- |
| x | pad byte | no value |  |  |
| c | char | bytes of length 1 | 1 |  |
| b | signed char | integer | 1 | (1),(3) |
| B | unsigned char | integer | 1 | (3) |
| ? | \_Bool | bool | 1 | (1) |
| h | short | integer | 2 | (3) |
| H | unsigned short | integer | 2 | (3) |
| i | int | integer | 4 | (3) |
| I | unsigned int | integer | 4 | (3) |
| l | long | integer | 4 | (3) |
| L | unsigned long | integer | 4 | (3) |
| q | long long | integer | 8 | (2), (3) |
| Q | unsigned long long | integer | 8 | (2), (3) |
| n | ssize\_t | integer |  | (4) |
| N | size\_t | integer |  | (4) |
| e | (7) | float | 2 | (5) |
| f | float | float | 4 | (5) |
| d | double | float | 8 | (5) |
| s | char[] | bytes |  |  |
| p | char[] | bytes |  |  |
| P | void \* | integer |  | (6) |

Changed in version 3.3: Added support for the 'n' and 'N' formats.

Changed in version 3.6: Added support for the 'e' format.

Notes:

1. The '?' conversion code corresponds to the \_Bool type defined by C99. If this type is not available, it is simulated using a char. In standard mode, it is always represented by one byte.
2. The 'q' and 'Q' conversion codes are available in native mode only if the platform C compiler supports C long long, or, on Windows, \_\_int64. They are always available in standard modes.
3. When attempting to pack a non-integer using any of the integer conversion codes, if the non-integer has a [\_\_index\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__index__) method then that method is called to convert the argument to an integer before packing.

Changed in version 3.2: Use of the [\_\_index\_\_()](https://docs.python.org/3/reference/datamodel.html#object.__index__) method for non-integers is new in 3.2.

1. The 'n' and 'N' conversion codes are only available for the native size (selected as the default or with the '@' byte order character). For the standard size, you can use whichever of the other integer formats fits your application.
2. For the 'f', 'd' and 'e' conversion codes, the packed representation uses the IEEE 754 binary32, binary64 or binary16 format (for 'f', 'd' or 'e' respectively), regardless of the floating-point format used by the platform.
3. The 'P' format character is only available for the native byte ordering (selected as the default or with the '@' byte order character). The byte order character '=' chooses to use little- or big-endian ordering based on the host system. The struct module does not interpret this as native ordering, so the 'P' format is not available.
4. The IEEE 754 binary16 “half precision” type was introduced in the 2008 revision of the [IEEE 754 standard](https://en.wikipedia.org/wiki/IEEE_floating_point#IEEE_754-2008). It has a sign bit, a 5-bit exponent and 11-bit precision (with 10 bits explicitly stored), and can represent numbers between approximately 6.1e-05 and 6.5e+04 at full precision. This type is not widely supported by C compilers: on a typical machine, an unsigned short can be used for storage, but not for math operations. See the Wikipedia page on the [half-precision floating-point format](https://en.wikipedia.org/wiki/Half-precision_floating-point_format) for more information.

A format character may be preceded by an integral repeat count. For example, the format string '4h' means exactly the same as 'hhhh'.

Whitespace characters between formats are ignored; a count and its format must not contain whitespace though.

For the 's' format character, the count is interpreted as the length of the bytes, not a repeat count like for the other format characters; for example, '10s' means a single 10-byte string, while '10c' means 10 characters. If a count is not given, it defaults to 1. For packing, the string is truncated or padded with null bytes as appropriate to make it fit. For unpacking, the resulting bytes object always has exactly the specified number of bytes. As a special case, '0s' means a single, empty string (while '0c' means 0 characters).

When packing a value x using one of the integer formats ('b', 'B', 'h', 'H', 'i', 'I', 'l', 'L', 'q', 'Q'), if x is outside the valid range for that format then [struct.error](https://docs.python.org/3/library/struct.html#struct.error) is raised.

Changed in version 3.1: In 3.0, some of the integer formats wrapped out-of-range values and raised [DeprecationWarning](https://docs.python.org/3/library/exceptions.html#DeprecationWarning) instead of [struct.error](https://docs.python.org/3/library/struct.html#struct.error).

The 'p' format character encodes a “Pascal string”, meaning a short variable-length string stored in a *fixed number of bytes*, given by the count. The first byte stored is the length of the string, or 255, whichever is smaller. The bytes of the string follow. If the string passed in to [pack()](https://docs.python.org/3/library/struct.html#struct.pack) is too long (longer than the count minus 1), only the leading count-1 bytes of the string are stored. If the string is shorter than count-1, it is padded with null bytes so that exactly count bytes in all are used. Note that for [unpack()](https://docs.python.org/3/library/struct.html#struct.unpack), the 'p' format character consumes count bytes, but that the string returned can never contain more than 255 bytes.

For the '?' format character, the return value is either [True](https://docs.python.org/3/library/constants.html#True) or [False](https://docs.python.org/3/library/constants.html#False). When packing, the truth value of the argument object is used. Either 0 or 1 in the native or standard bool representation will be packed, and any non-zero value will be True when unpacking.