智能物联网平台需求分析

修订记录：

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 描述 | 作者 |
| 2017-5-3 | REV.0 |  | 马翔 |
|  |  |  |  |

目录

[智能物联网平台需求分析 1](#_Toc481615080)

[物接入- IoT Hub 3](#_Toc481615081)

[功能 3](#_Toc481615082)

[实现 3](#_Toc481615083)

[TCP 接入 3](#_Toc481615084)

[API 接口（HTTP） 5](#_Toc481615085)

[数据表 7](#_Toc481615086)

[物管理 – IoT Device 8](#_Toc481615087)

[功能 8](#_Toc481615088)

[实现 9](#_Toc481615089)

[数据表 9](#_Toc481615090)

[MySQL 9](#_Toc481615091)

[物解析 – IoT Parser 11](#_Toc481615092)

[功能 11](#_Toc481615093)

[实现 12](#_Toc481615094)

[数据表 12](#_Toc481615095)

[MySQL 12](#_Toc481615096)

[规则引擎 12](#_Toc481615097)

[应用管理 13](#_Toc481615098)

[数据表 13](#_Toc481615099)

[MySQL 13](#_Toc481615100)

[API 13](#_Toc481615101)

[Appendix 14](#_Toc481615102)

[C Data Type 14](#_Toc481615103)

[Microsoft C++ Data Type Ranges 14](#_Toc481615104)

[struct — Interpret bytes as packed binary data 15](#_Toc481615105)

# 物接入- IoT Hub

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。

## 功能

1. 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。

1. 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

1. 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

1. 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。

* 接入防火墙，根据IP地址进行黑白名单过滤。
* 接入负载均衡。
* *接入安全，可采用不对称加密方法，由平台生成公钥及私钥，并且将私钥置入设备固件。*
* 设备唯一识别码和密钥（非加密密钥）是唯一证明设备合法性的数据。
* 完整性校验，应用于所有数据包，计算包头及包身部分哈希。
* 对于TCP长连接，设置超时时间，增强服务端可用性。
* 合规性校验，应用于所有数据包，判断设备命令合法。

## 实现

技术选型：[Python](https://www.python.org/) ([Twisted](https://twistedmatrix.com/trac/), [gevent](http://www.gevent.org/index.html), [Flask](http://flask.pocoo.org/), [Tornado](http://www.tornadoweb.org/)), [MySQL](https://www.mysql.com/), [Go](https://golang.org), [Nginx](http://nginx.org/)/[OpenResty](https://openresty.org), [Lua](http://www.lua.org/)

TCP 接入，HTTP 接入，在线/离线设备统计，在线时长统计（设备在线总时长，设备当前在线时长），认证统计

### TCP 接入

协议基本格式：包头+包头CRC+数据+数据CRC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 包头 | 包头CRC | 数据 | 数据CRC |
| 长度（Byte） |  | 1 |  | 1 |

包头

用于标识设备识别码、命令编号、读写操作类型、包头CRC16

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | CRC |  |
| 长度（Byte） | 2 | 1 | 1 | 1 | 1 |  |
| C Type | unsigned int | unsigned char | \_Bool/ char |  |  |  |
| Python type | Integer | integer | Bool/ bytes of length 1 |  |  |  |
| 范围 | 4294967295 | 255 |  |  |  |  |
|  |  |  |  | 未定义 |  |  |

心跳命令：物接入无返回数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | 预留 | CRC |  |
| 长度 |  | 1 |  |  |  |  |  |
| 内容 |  | 1 |  |  |  |  |  |

认证命令：

物接入发送：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | 预留 | CRC |
| 长度 |  | 1 |  |  |  |  |
| 内容 |  | 2 |  |  |  |  |

设备回复：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | 预留 | CRC | 设备密钥 | CRC |
| C Type |  |  |  |  |  |  | char[] |  |
| Python type |  |  |  |  |  |  | bytes |  |
| 长度 |  | 1 |  |  |  |  | 16 |  |
| 内容 |  | 2 |  |  |  |  |  |  |

当IoT设备与物接入建立TCP连接后，物接入向设备发送认证指令，设备收到后回复设备密钥，如果认证成功，记录设备上线时间，并在上线设备列表中添加此设备唯一识别码，如果认证失败，断开与设备的连接。如果设备超过10秒不回复认证消息，断开连接。不论认证成功与否，都需要记录设备唯一识别码，认证时间及认证结果。在设备未认证成功前

发送的任何信息均被丢弃。

校时命令：

物接入发送：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | 预留 | CRC | 当前时间 | CRC |
| C Type |  |  |  |  |  |  | char[] |  |
|  |  |  |  |  |  |  |  |  |
| Python type |  |  |  |  |  |  | bytes |  |
| 长度 |  | 1 |  |  |  |  |  |  |
| 内容 |  | 3 |  |  |  |  |  |  |

时间表示方案

1. 年(unsigned short)+月(unsigned char)日(unsigned char)+时(unsigned char)+分(unsigned char)+秒(unsigned char)
2. char[]
3. Unix Time Stamp

设备回复：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 设备唯一识别码 | 命令编号 | 读写操作类型 | 预留 | 预留 | CRC |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| 长度 |  | 1 |  |  |  |  |
| 内容 |  | 3 |  |  |  |  |

如果超过10秒超时时间，设备未确认时间，再次发送当前时间。

### API 接口（HTTP）

获取物接入状态

|  |  |
| --- | --- |
| Method | GET |
| URL | /v1/hub/status |

|  |
| --- |
| 在线 离线 设备数量 比率  最近1小时/分钟/秒 数据量 收发率  # 验证命令比例  # 时间验证比例  # 数据转发比例 |

添加设备唯一识别码和设备密钥（需通过API Gateway 认证鉴权）

|  |  |
| --- | --- |
| Method | POST |
| URL | /v1/device |

|  |  |  |
| --- | --- | --- |
| 参数 | 意义 | 备注 |
| id | 设备唯一识别码 | 必填 |
| secret | 设备密钥 | 必填 |
| expire | 过期时间 | 选填，默认为永久可连接 |
|  |  |  |

获取设备在线/离线统计

|  |  |
| --- | --- |
| Method | GET |
| URL | /v1/device/status |

|  |
| --- |
| {  ‘total’: 300  ‘online: 200,  ‘offline’: 100  } |

获取某个设备的在线/离线状态

|  |  |
| --- | --- |
| Method | GET |
| URL | /v1/device/:id/status |

|  |
| --- |
| {  ‘device\_id: 123  ‘status: ‘online’,  }  {  ‘device\_id: 124  ‘status: ‘offline’,  ‘last\_online\_date’: ‘2016-02-03 20:01:12’  } |

获取某个设备在某时间段的接入记录

|  |  |
| --- | --- |
| Method | GET |
| URL | /v1/device/:id/log |

|  |  |  |
| --- | --- | --- |
| 参数 | 意义 | 备注 |
| id | 设备唯一识别码 | 必填 |
| start | 起始时间 |  |
| end | 结束时间 | 时间段不超过1天，记录数不超过100条 |

|  |
| --- |
| {  ‘device\_id’: 123  ‘authentication’: [  {  ‘created’: ‘2016-02-03 20:01:12’,  ‘status’: ‘Success’  },  {  ‘created’: ‘2016-03-03 20:01:12’,  ‘status’: ‘Fail  },  ],  ‘network’: [  {  ‘start: ‘2016-02-03 20:01:12’,  ‘end: ‘2016-03-03 20:01:12’,  },  {  ‘start: ‘2016-03-03 20:01:12’,  ‘end: ‘2016-03-08 20:01:12’,  },  ],  } |

### 数据表

#### MySQL

#### Redis

db0 – Hub stat

|  |  |  |
| --- | --- | --- |
| KEY | Memo | Expire(seconds) |
| hub\_total\_devices | 总设备数量 | 0 |
| hub\_online\_devices | 在线设备数量 | 0 |
| total\_daily | 每日数据量 | 86400 |
| heartbeat\_daily | 每日心跳包量 | 86400 |
| auth\_daily | 每日认证消息量 | 86400 |
| ntp\_daily | 每日时间协议量 | 86400 |
| parser\_daily | 每日转发物解析量 | 86400 |
| total\_hourly |  | 3600 |
| heartbeat\_hourly |  | 3600 |
| auth\_hourly |  | 3600 |
| ntp\_hourly |  | 3600 |
| parser\_hourly |  | 3600 |
| total\_per\_minute |  | 60 |
| heartbeat\_per\_minute |  | 60 |
| auth\_per\_minute |  | 60 |
| ntp\_per\_minute |  | 60 |
| parser\_per\_minute |  | 60 |

db1 – Devices

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KEY | Field | Field | Field | Field |  |
|  | secret | added | expire | Online/Offline |  |
| 设备ID | 设备密钥 | 添加时间 | 过期时间 | 1/0 |  |

# 物管理 – IoT Device

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。

## 功能

1. 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。

1. 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

1. 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

1. 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

* 产品管理，包含产品名称、型号、备注及产品识别码。
* 产品权限管理，多个用户可管理同一个产品。
* 设备管理，设备影子，包含设备唯一识别码（长/短），设备密钥，硬件版本号，软件版本号，检验员，生产日期，状态（是否安装），位置。
* 设备消息安全，采用非对称加密。

## 实现

## 数据表

### MySQL

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 |  |

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 |  |

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 |  |

# 物解析 – IoT Parser

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。

## 功能

1. 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。

1. 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。

1. 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。

1. 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。

* 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。

## 实现

解析传感器数据：根据设备唯一识别码，获取所关联的产品及数据点，生成解析规则，解析数据并存储。

## 数据表

### MySQL

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 |  |

# 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。

# 应用管理

## 数据表

### MySQL

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 |  |

# API

## Appendix

### 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" \l "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" \l "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" \l "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" \l "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" \l "object.__index__" \o "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" \l "object.__index__" \o "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" \l "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" \l "struct.error" \o "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" \l "DeprecationWarning" \o "DeprecationWarning) instead of [struct.error](https://docs.python.org/3/library/struct.html" \l "struct.error" \o "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" \l "struct.pack" \o "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" \l "struct.unpack" \o "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" \l "True" \o "True) or [False](https://docs.python.org/3/library/constants.html" \l "False" \o "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.