|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
| **ChiPeakTEC DNN1.8**软件 | | |  |
| **操作指南** | | |
| **文档版本** | **01** | |
| **发布日期** | **2020-1-10** | |
|  | | | | |
|  | 芯峰科技（广州）有限公司 | |  |  |

|  |
| --- |
| 版权所有 © 芯峰科技（广州）有限公司2019。 保留一切权利。  非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。  商标声明  和其他芯峰商标均为芯峰科技（广州）有限公司的商标。  本文档提及的其他所有商标或注册商标，由各自的所有人拥有。  注意  您购买的产品、服务或特性等应受芯峰公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，芯峰公司对本文档内容不做任何明示或默示的声明或保证。  由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。 |

|  |  |
| --- | --- |
| 芯峰科技（广州）有限公司 | |
| 地址： | 广州市南沙区进港大道8号南沙城1205/1206 |
| 网址： | <http://www.chipeak.com> |

前言

概述

本文档详细的描述了芯峰视频智能分析软件-ChiPeakTEC DNN安装、配置以及管理和运维等具体的操作指导，同时提供了常见的问题解答及故障处理方法。

读者对象

本文档主要适用于以下工程师：

* 技术支持工程师
* 运维工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

修改记录

| **文档版本** | **发布日期** | **修改说明** |
| --- | --- | --- |
| 01 | 2020-01-10 | 第一次正式发布。 |

目 录

[前言 ii](#_Toc36045984)

[1 软件架构与部署拓扑 1](#_Toc36045985)

[1.1 资源要求 2](#_Toc36045986)

[1.2 软件架构 2](#_Toc36045987)

[1.2.1 DNN Management Server 2](#_Toc36045988)

[1.2.2 DNN Engine 3](#_Toc36045989)

[1.3 部署拓扑 3](#_Toc36045990)

[1.3.1 边缘端部署（推荐） 3](#_Toc36045991)

[1.3.2 中心侧/云端部署 4](#_Toc36045992)

[2 DNN Management Server软件安装 5](#_Toc36045993)

[2.1 准备工具 5](#_Toc36045994)

[2.2 安装要求 6](#_Toc36045995)

[2.3 安装步骤 6](#_Toc36045996)

[3 DNN Engine软件安装 7](#_Toc36045997)

[3.1 准备工具 7](#_Toc36045998)

[3.2 安装前要求 8](#_Toc36045999)

[3.3 安装步骤 8](#_Toc36046000)

[3.3.1 安装方式1：FD安装 8](#_Toc36046001)

[3.3.2 安装方式1：本地安装 9](#_Toc36046002)

[3.4 业务流程图 10](#_Toc36046003)

[4 软件配置 10](#_Toc36046004)

[4.1 算法参数配置 10](#_Toc36046005)

[4.2](#_Toc36046006) **[ZoneEngine环境配置参数](#_Toc36046006)** [13](#_Toc36046006)

[4.3](#_Toc36046007) **[ZoneServer环境配置参数](#_Toc36046007)** [13](#_Toc36046007)

[4.4](#_Toc36046008) **[Graph文件配置](#_Toc36046008)** [14](#_Toc36046008)

[5 软件接口 15](#_Toc36046009)

[5.1 接口系统 15](#_Toc36046010)

[5.2 接口数据格式 16](#_Toc36046011)

[5.2.1 接口方法 16](#_Toc36046012)

[5.2.2 入参格式说明 16](#_Toc36046013)

[StructData，类型为JSON字符串，具体说明如下： 16](#_Toc36046014)

[5.2.3 出参说明 18](#_Toc36046015)

[5.2.4 应用举例 18](#_Toc36046016)

[6 软件授权 26](#_Toc36046017)

[6.1 授权限制 26](#_Toc36046018)

[6.2 初始化配置 27](#_Toc36046019)

[6.2.1 申请并导入License 27](#_Toc36046020)

[6.2.1.1 申请License 27](#_Toc36046021)

[6.2.1.2 导入License 27](#_Toc36046022)

[7 检查ChiPeakTEC DNN版本 28](#_Toc36046023)

[7.1 检查ChiPeakTEC DNN 软件版本 28](#_Toc36046024)

[7.2 检查AI设备固件版本 28](#_Toc36046025)

[8 升级ChiPeakTEC DNN 30](#_Toc36046026)

[8.1 升级ChiPeakTEC DNN容器镜像 30](#_Toc36046027)

[8.2 升级业务软件 30](#_Toc36046028)

[9 卸载ChiPeakTEC DNN 31](#_Toc36046029)

[10 其他 32](#_Toc36046030)

[10.1 摄像头配置 32](#_Toc36046031)

[10.2 摄像头配置检查 32](#_Toc36046032)

[10.3 带宽控制 32](#_Toc36046033)

[A FAQ 33](#_Toc36046034)

[A.1 系统重启 33](#_Toc36046035)

[A.1.1 配置的IP地址已存在，请重新配置后重试 33](#_Toc36046036)

[B 如何获取帮助 33](#_Toc36046037)

[B.1 收集必要的故障信息 34](#_Toc36046038)

[B.2 如何使用文档 34](#_Toc36046039)

[B.3 获取技术支持 34](#_Toc36046040)

# 软件架构与部署拓扑

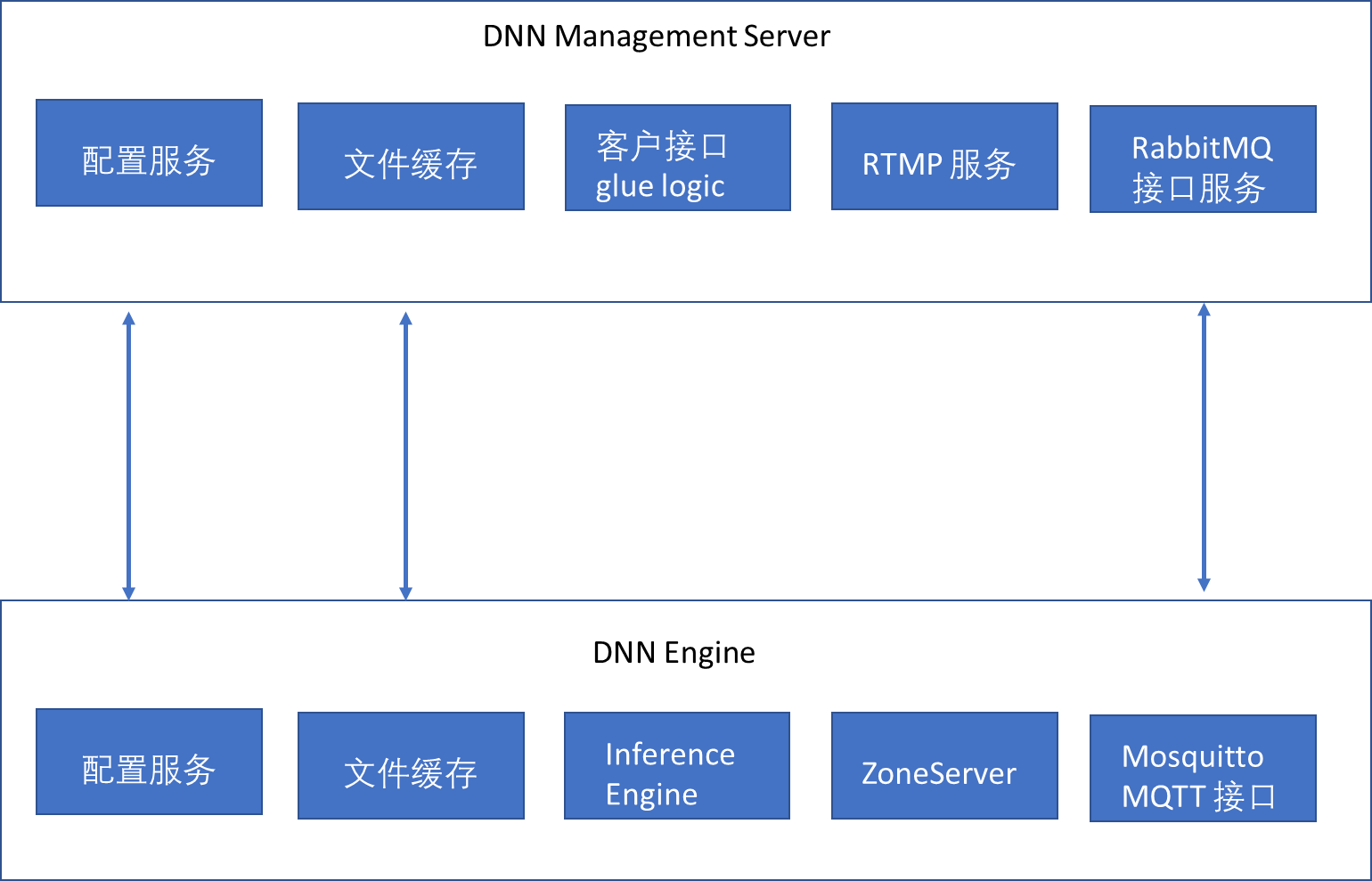
本章描述了ChiPeakTEC DNN视频智能分析软件在部署过程中的资源要求，软件部署时的部署拓扑，以及简单介绍了软件的各个功能模块及其分工。

## 资源要求

网络和资源要求

* 单台ChiPeakTEC DNN能够正常运行业务的有效网络至少为20Mbps。
* ChiPeakTEC DNN能够正常运行业务的有效网络交换推荐至少为100Mbps。
* 安装ChiPeakTEC DNN的服务器对外连接的管理网口建议GE网口。

## 软件架构



### DNN Management Server

**功能描述**

提供AI设备的运维，报警数据汇聚，以及提供对外数据服务接口。

**配置服务**

设备中的DNN Engine 启动需要初始化的数据从这里获取。

**文件缓存**

报警短视频缓存，数据源从DNN engine同步。

**RTMP服务**

客户业务系统通过RTMP获取报警短视频。

**RabbitMQ接口**

MQTT HUB，通过MQTT协议汇聚各AI设备的报警数据（文本与图片），供客户业务系统集中订阅并有QoS的保障。

**客户接口Glue Logic**

根据客户提供的接口把数据推送过去，定制化服务，不必须。建议客户通过RabbitMQ接口订阅以维持系统的鲁棒性。

### DNN Engine

**功能描述**

后端推理及逻辑判断引擎，产生报警数据并发送到DNN Management Server端。

**配置服务**

从DNN Management Server 同步启动需要初始化的数据。

**文件缓存**

报警图片、短视频缓存。

**Inference Engine**

AI推理引擎模块，主要通过DNN(深度神经网络)进行目标与行为识别。

**ZoneServer**

逻辑处理模块，利用推理数据结合业务进行逻辑判断，发送报警数据;包含ZoneServer与ZoneEngine两个子模块。

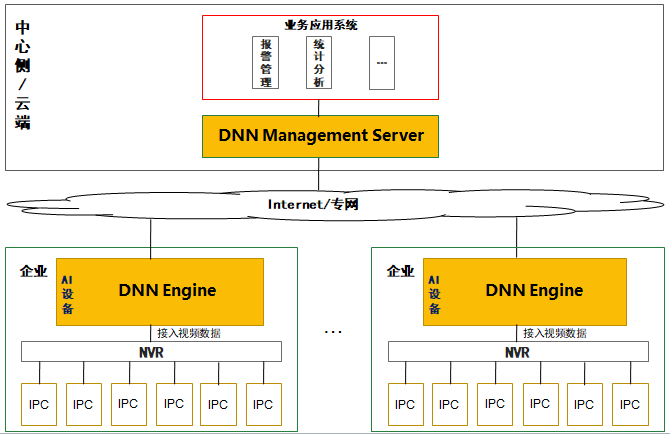
**Mosquitto MQTT**

报警数据接口模块，从ZoneServer接收后路由到RabbitMQ。

## 部署拓扑

### 边缘端部署（推荐）

推荐将AI设备部署在边缘端，软件的DNN Engine部分，部署在AI设备中：

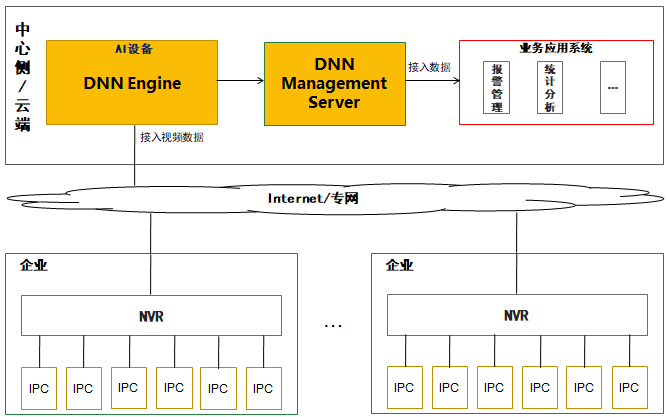


DNN Management Server 部署在中心侧/云端，涵盖AI智能设备运维系统与业务接口系统。

DNN Engine 部署在企业边缘端的AI智能设备上。

### 中心侧/云端部署

AI设备部署在中心侧/云端：



DNN Management Server 部署在中心侧/云端，涵盖AI智能设备运维系统与业务接口系统。

DNN Engine 部署在中心侧/云端端的AI智能设备上。

# DNN Management Server软件安装

## 准备工具

Docker（建议版本18.09.0.40）

rabbitmq



## 安装要求

Docker主流版本即可。

## 安装步骤

yum update

yum install docker #在线安装docker

systemctl enable docker #设为开机自启动

docker pull rabbitmq #获取rabbitmq镜像

#启动rabbitmq容器

docker run -it --restart always --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq

#启动rabbitmq\_web\_mqtt插件，需要浏览器通过websocket连接到rabbitmq

docker exec <容器ID> rabbitmq-plugins enable rabbitmq\_web\_mqtt

#浏览器登陆rabbitmq，默认用户名和密码guest/guest

本机ip:15672

添加用户密码，并设置权限(参照官方文档即可)

#此处设置的账户密码，会使用在AI设备端配置。

# DNN Engine软件安装

## 准备工具

1. AI设备准备：AI设备升级固件，部署到客户现场；
2. 网络环境准备：中心侧/云端网络环境权限，以及网络AI设备联通；
3. 容器准备：镜像文件存放到PC；
4. 授权准备：获取AI设备的MAC地址；
5. 管理环境准备：管理服务器端，根下创建目录/atlasdata，在目录/atlasdata下针对每个AI设备创建一个目录，目录名称为客户企业简拼+mac地址(AI盒子-eth0网口mac去掉冒号)；
6. Graph准备：针对每个AI设备，配置一个graph(IP，视频流列表)，存放进管理服务器端/atlasdata下对应目录；
7. 端口准备：见下表：



## 安装前要求

1. 固件版本与视频分析软件版本匹配；
2. AI设备与管理服务网络互通；
3. 根据项目约定，开放必要端口；

## 安装步骤

### 安装方式1：FD安装

参考以下步骤：

a，PC存放镜像文件，配置文件，脚本文件；

b，PC配置可访问FD服务的IP与网关；

c，安装软件“secoclient”，设置IP信息，账号密码；

d，连接中心侧/云端上的FD服务器，浏览器输入FD服务页面的地址，如："https://172.18.135.4",输入账户密码；

e，上传本地镜像文件到FD；

f，操作FD管理AI设备，基于镜像文件配置容器，添加参数与相关命令，下发容器到AI设备；(容器内后台运行sshd服务)

g，PC端传送配置文件与脚本文件到管理服务DNN Management Server；

h，AI设备端容器初始化，从本地或DNN Management Server获取最新配置文件；

i，AI设备端容器初始化完成，触发业务启动；

j，以上操作不涉及容器内部操作；

### 安装方式1：本地安装

参考以下步骤：

a，PC存放镜像文件，配置文件，脚本文件；

b，上传本地镜像文件到AI设备；

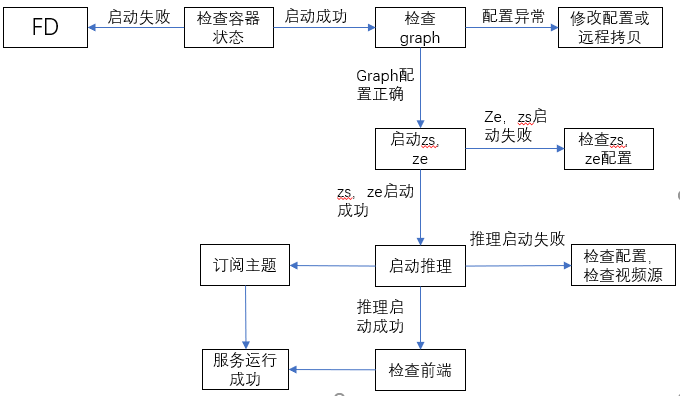
c，PC端传送配置文件与脚本文件到管理服务DNN Management Server；

d，AI设备端容器初始化，从本地或DNN Management Server获取最新配置文件；

e，AI设备端容器初始化完成，触发业务启动；

f，以上操作不涉及容器内部操作；

## 业务流程图



# 软件配置

## 算法参数配置

这些参数位于ZoneEngine运行目录的tmp/app\_name/channel\_name/算法名.json内。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **算法名** | **中文名** | **配置参数名** | **参数说明** | **示例及说明** |
| absence | 离岗检测 | attendance\_time | 上班时间 | 示例：[[1, 5], ['080000', '120000'], ['140000', '180000']]  说明：周一到周五的8:00-12:00和14:00-18:00是上班时间，这个时间内才会检测离岗，配置时需按照该格式 |
| absence\_time\_threshold | 离岗时间阈 | 示例：10  说明：上班时间内，工作人员离岗时间超过该举止就属于离岗 |
| polygon\_vertex | 工作区域 | 示例：[[10,10],[50,20],[100,200],[70,150],[35,100]]  说明：工作区域，多边形顶点依次记录在这个参数中 |
| alarm\_interval\_time | 报警间隔时间阈 | 示例：30  说明：报警间隔时长，避免重复报警 |
| helmet\_suit | 工作服安全帽 | helmet\_confidence | 安全帽报警阈值 | 示例：0.9  说明：系统检测出某个人戴安全帽的概率低于该阈值，则认为该人未带安全帽 |
| suit\_confidence | 工作服报警阈值 | 示例：0.9  说明：系统检测出某个人穿工作服的概率低于该阈值，则认为该人未穿工作服 |
| alarm\_interval\_time | 报警间隔时间阈 | 示例：30  说明：报警间隔时长，避免重复报警 |
| gas\_station\_ped | 抽烟打电话 | cellphone\_alarm\_confidence | 打电话报警阈值 | 示例：0.8  说明：系统检测出某个人打电话的概率高于该阈值，则认为该人在打电话 |
| smoking\_alarm\_confidence | 抽烟报警阈值 | 示例：0.85  说明：系统检测出某个人抽烟的概率高于该阈值，则认为该人在抽烟 |
| alarm\_interval\_time | 报警间隔时间阈 | 示例：30  说明：报警间隔时长，避免重复报警 |
| fire\_smog | 烟雾明火 | alarm\_interval\_time | 报警间隔时间阈 | 示例：15  说明：报警间隔时长，避免重复报警 |
| boundary\_intrude | 区域入侵 | polygon\_vertex | 报警设定的多边形区域 | 示例：[[10,10],[50,20],[100,200],[70,150],[35,100]]  说明：报警区域，多边形顶点依次记录在这个参数中 |

## **ZoneEngine环境配置参数**

4.2.1算法文件配置参数：

该文件位于ZoneEngine运行目录下的tmp/app\_name/channel\_type.json内，配置规则如下：

{"channel1":"fire\_smog"}:表示该app\_name下的channel1运行fire\_smog，即烟雾明火检测算法，如果该app\_name下有多路视频，则只需在后面按照该格式添加即可。

4.2.2 config配置参数：

该文件位于ZoneEngine运行目录的config文件夹下，名为：config.conf，配置内容如下：

|  |  |  |
| --- | --- | --- |
| **配置参数名** | **参数含义** | **说明** |
| storage\_dir | 存储路径 | 通常是ZoneEngine运行目录下的tmp绝对路径 |
| mqtt\_ip\_sub | ip地址 | Mqtt订阅的ip地址 |
| mqtt\_port\_sub | 端口号 | Mqtt订阅的端口号 |
| mqtt\_qos\_sub | Qos | Mqtt的qos值 |
| mqtt\_timeout\_sub | 重连时间阈 | Mqtt重连的时间阈值 |
| mqtt\_ip\_pub | ip地址 | Mqtt发布的ip地址 |
| mqtt\_port\_pub | 端口号 | Mqtt发布的端口号 |
| mqtt\_qos\_pub | Qos | Mqtt的qos值 |
| mqtt\_timeout\_pub | 重连时间阈 | Mqtt重连的时间阈值 |
| send\_no\_alarming\_data | 标志位 | 如果该值为0，则值发送报警数据，如果为1，则发送所有数据 |

## **ZoneServer环境配置参数**

4.3.1 config文件配置参数：

该文件位于ZoneServer运行目录的config文件夹下，名为：config.conf，配置内容如下：

|  |  |  |
| --- | --- | --- |
| **配置参数名** | **参数含义** | **说明** |
| presenter\_server\_ip | 主机ip | Socket服务器地址 |
| presenter\_server\_port | 端口号 | 对应的端口号 |
| web\_server\_ip | 服务器ip | http服务器地址 |
| web\_server\_port | 端口号 | 对应的端口号 |
| storage\_dir | 存储路径 | 通常是ZoneServer运行目录的tmp绝对路径 |
| max\_face\_num | 人脸识别最大数 | 人脸识别的最大数，范围：0-100 |
| face\_match\_threshold | 面部匹配阈值 | 范围：0-1 |
| max\_app\_num | 对大应用数 | 范围：1-10 |
| reserved\_space | 最小可用空间 | 最小可用空间不低于该值，单位是MByte |
| mqtt\_ip | Mqtt服务ip | Mqtt服务器ip |
| mqtt\_port | Mqtt端口 | Mqtt端口号 |
| process\_num | 进程数 | 系统运行ZoneServer开启的进程数，如为0，则开启跟机器cpu数量相同的进程 |

## **Graph文件配置**

Graph文件位置，/root/HIAI\_PROJECTS/ascend\_workspace/videoanalysispersonapp/7rc/目录下

例如 mix\_sc.config

**Server-推理ip、端口信息配置：**

ai\_config {

items { name: "presenter\_server\_ip"

value:"172.17.0.2" }

items { name: "presenter\_server\_port"

value:"7058" }

**app\_name信息配置：**

items { name: "app\_name"

value:"mix" } }

**视频源信息配置：**

items { name: "channel1"

value: "rtsp://admin:12345678@192.168.1.64:554"}

配置时注意事项：

1）推理设备IP与容器IP一致；

2）端口默认7058，建议使用默认值；

3）app\_name 必须与ze-tmp目录下的算法文件目录名称一致；

4）视频信息根据现场网络填写，554端口为默认流服务端口；

# 软件接口

## 接口系统

接口系统有AI 网关的Mosquitto MQTT及DNN Management Server的RabbitMQ组成；ZoneEngine分析的数据结果发送给Mosquitto MQTT，MQTT路由给RabbitMQ。

接口规范如下：

1）以webservice技术实现的接口技术要求

2）时间类字段的数据格式均为：“YYYY-MM-DD H24:MI:SS”。

3）接口实现使用SOAP协议，接口描述使用WSDL语言。

4）接口实现符合SOAP协议V1.1版本规范，兼顾高版本SOAP协议与1.1版本的兼容性。

5）接口实现至少支持SOAP 在HTTP V1.0协议上的传输。

6）接口实现符合WSDL语言V1.1版本规范，高版本WSDL语言与1.1版本的保持兼容。

## 接口数据格式

### 接口方法

* **方法名称**

**string AIDataRecv(String** ImageData, **String** StructData**)**

### 入参格式说明

ImageData，二进制文件流。该文件流是当前这个场景的帧图（帧图采用Base64等标准编码格式转换字符串）。

### StructData，类型为JSON字符串，具体说明如下：

StructData为JOSN格式字符串。

例子：

|  |
| --- |
| {  "Supplier": "XX工厂",  "SupCode": "工厂国标编码",  "VideoUrl": "rtsp://admin:123456@192.168.1.223/PSIA/streaming/channels/101",  "Code": "1123141110319812609 ",  "Position": "西信大门口",  "Alarm": "发生非法入侵事件",  "GpsPos": "24.471713,54.385398",  "AlarmDate": "2019-09-18 14:00:02",  "Matter": [  {  "x": 285,  "y": 470,  "height": 211,  "width": 212,  "type ": 3  }  ]  },  {  "Supplier": "XX工厂",  "SupCode": "工厂国标编码",  "VideoUrl": "rtsp://admin:123456@192.168.1.223/PSIA/streaming/channels/101",  "Code": "1123144006839046145",  "Position": "西信B区2楼楼梯口",  "Alarm": "检测到疑似火灾",  "GpsPos": "24.471713,54.385398",  "AlarmDate": "2019-09-18 14:00:02",  "Matter": [  {  "x": 1,  "y": 200,  "width": 50,  "height": 200,  "type ": 6  },  {  "x": 100,  "y": 50,  "width": 50,  "height": 130,  "type ": 6  }  ]  } |

节点属性描述：

|  |  |  |  |
| --- | --- | --- | --- |
| **节点属性描述** | | | |
| **层次** | **属性字段** | **描述** | |
| 2 | Supplier(必须字段) | **企业名称** |
| 2 | SupCode(必须字段) | **企业代码**（国标编码） |
| 2 | VideoUrl(必须字段) | **视频源URL地址** 用于接入的视频源地址，通过这个地址接入视频源，如: RTSP流（注：AI厂家提供） |
| 2 | Code(必须字段) | **摄像头编码**（AI厂家提供） |
| 2 | Position（必须字段） | **摄像头位置信息描述** 例如：西信大门口 |
| 2 | Alarm(必须字段) | **告警信息描述** 例如：发生非法入侵事件（描述信息长度不超过512字节） |
| 2 | AlarmDate(必须字段) | **告警时间** 格式如：2019-09-18 14:00:02 |
| 2 | Matter（必须字段） | **告警事项** 该层可包含多个告警具体**对象信息** |
| 3 | x（必须字段） | **对象信息** 物体对象所在场景图像的X位置 |
| 3 | y（必须字段） | **对象信息** 物体对象所在场景图像的Y位置 |
| 3 | width（必须字段） | **对象信息** 物体对象宽度 |
| 3 | height（必须字段） | **对象信息** 物体对象高度 |
| 3 | type (必须字段) | **告警类型**（01：工作服、02：打手机、03：离岗、04：打瞌睡、05：人脸、06：烟火、07：抽烟、08：安全帽、09：越界、999：其他。目前类型为这几种，后期需要补充则后期再做添加说明） |
| 2 | GpsPos | **GPS位置信息** 该场景所在GPS经纬度信息，用于定位当前位置。 |

### 出参说明

示列：

{

"RltCode": "200",

"RltMsg": "接收成功"

}

|  |  |  |
| --- | --- | --- |
| **属性字段** | **取值** | **说明** |
| RltCode | 200/500 | 200：成功；500：失败 |
| RltMsg | 字符串 | 结果说明 |

### 应用举例

**Java 抽象类：**

package com.example.snt.common;  
  
import org.eclipse.paho.client.mqttv3.\*;  
  
*/\*\*  
 \* 异步订阅MQTT消息抽象类定义  
 \*/*public abstract class MqttAsyncSubscriber {  
 public static MqttAsyncClient mqttAsyncClientSubscription(String serverUri, String clientId , String[] topic, int[] qos){  
 MqttAsyncClient mqttAsyncClient = null;//引用异步非阻塞订阅MQTT消息客户端类  
 try {  
 mqttAsyncClient = new MqttAsyncClient(serverUri, clientId);  
 MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();//引用MQTT连接操作对象  
 mqttConnectOptions.setCleanSession(true);//清除session  
 MqttAsyncClient finalMqttAsyncClient = mqttAsyncClient;//把mqttAsyncClient变量由finalMqttAsyncClient变量替代  
 mqttAsyncClient.connect(mqttConnectOptions,null, new IMqttActionListener() {  
 @Override  
 public void onSuccess(IMqttToken iMqttToken) {  
 //连接成功打印  
 System.*out*.println("connection successfull with client id : "+clientId+" , on server url : "+serverUri+" , qos level :"+qos + ",连接成功" + " " + "订阅主题：" + topic);  
 try {  
 finalMqttAsyncClient.subscribe(topic,qos);  
 } catch (MqttException e) {  
 System.*out*.println("Error : "+e.getMessage());  
 }  
 }  
 public MqttConnectOptions get\_options()  
 {  
 return mqttConnectOptions;  
 }  
 @Override  
 public void onFailure(IMqttToken iMqttToken, Throwable throwable) {  
 //连接失败打印  
 System.*out*.println("connection fail with client id : "+clientId+" , on server url : "+serverUri+" , qos level :"+qos + ",连接失败");  
 }  
 });  
 } catch (MqttException e) {  
 e.printStackTrace();  
 }  
 return mqttAsyncClient;  
 }  
}

**订阅类：**

package com.example.snt.controller;  
  
import com.example.snt.common.MqttAsyncSubscriber;  
import com.example.snt.util.FileUtil;  
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;  
import org.eclipse.paho.client.mqttv3.MqttAsyncClient;  
import org.eclipse.paho.client.mqttv3.MqttCallback;  
import org.eclipse.paho.client.mqttv3.MqttMessage;  
import org.springframework.context.annotation.Configuration;  
  
import javax.annotation.PostConstruct;  
import javax.imageio.ImageIO;  
  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.math.BigDecimal;  
import java.sql.Timestamp;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
import static com.example.snt.common.Constant.*mult\_topic*;  
  
@Configuration  
public class SntTest implements MqttCallback {  
 private MqttAsyncClient mqttAsyncClient;  
 String time = new Timestamp(System.*currentTimeMillis*()).toString();  
 private final SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd\_HH-mm-ss");  
 //自定义订阅消息数据写文件路径  
 String channel1 = "channel1";  
 String slash = "/";  
 String recognitionType = "face921";  
 private final String SuitAlarmOne = recognitionType + slash + channel1 + slash + "suit\_alarm";  
 private final String HelmetAlarmOne = recognitionType + slash + channel1 + slash + "helmet\_alarm";  
 private final String BoundaryAlarmOne = recognitionType + slash + channel1 + slash + "boundary\_alarm";  
  
 @PostConstruct  
 public void setCallback() {  
 mqttAsyncClient = MqttAsyncSubscriber.*mqttAsyncClientSubscription*("tcp://120.78.179.214:3883", java.util.UUID.*randomUUID*().toString(), *mult\_topic*, new int[]{0, 0, 0});  
 mqttAsyncClient.setCallback(this);  
 }  
  
 @Override  
 public void connectionLost(Throwable throwable) {  
 while (true) {  
 System.*out*.println("3秒后重新连接================");  
 try {  
 Thread.*sleep*(3000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("重新连接成功");  
 mqttAsyncClient = MqttAsyncSubscriber.*mqttAsyncClientSubscription*("tcp://120.78.179.214:3883", java.util.UUID.*randomUUID*().toString(), *mult\_topic*, new int[]{0, 0, 0});  
 // 重连成功后,mqttAsyncClient 不为null  
 if (mqttAsyncClient != null) {  
 mqttAsyncClient.setCallback(this);  
 break;  
 }  
 }  
 }  
  
 @Override  
 public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception {  
 System.*out*.println("当前接收消息的主题是 : " + topic);  
 System.*out*.println("接收消息的质量Qos : " + mqttMessage.getQos());  
 if (topic.endsWith("zs/appsntby/systemctl")) {  
 System.*out*.println("当前时间" + time);  
 System.*out*.println("接收心跳主题 : " + topic);  
 System.*out*.println("接收消息Qos : " + mqttMessage.getQos());  
 System.*out*.println("接收消息内容1 : " + new String(mqttMessage.getPayload()).length());  
 }  
  
 */\*\*  
 \* 不带图片的工装安全帽----不正常---原因是v2主题是没有图片所以存图片就会导致重连  
 \*/* if (topic.endsWith("zs/appsntby/channel1/helmet\_suit/v2/out")) {  
 String hat\_no = "model" + slash + channel1 + slash + "hat\_alarm\_no";  
 String suit\_no = "model" + slash + channel1 + slash + "suit\_alarm\_no";  
 chipeak.zoneserver.proto.ZoneServerMessage.HelSuiDetectionData helSuiDetectionData = chipeak.zoneserver.proto.ZoneServerMessage.HelSuiDetectionData.*parseFrom*(mqttMessage.getPayload());  
 System.*out*.println("接收工作服安全帽结构体订阅主题消息 : " + topic);  
 System.*out*.println(helSuiDetectionData.getAppId() + " " + "当前应用的ID");  
 System.*out*.println(helSuiDetectionData.getChannelId() + " " + "当前视频源对应的ID");  
 System.*out*.println(helSuiDetectionData.getFrameId() + " " + "获取到的图片帧号");  
 System.*out*.println(helSuiDetectionData.getFps() + " " + "当前channel的帧速");  
 //时间  
 double achieveTime = helSuiDetectionData.getTimestamp();  
 System.*out*.println(achieveTime);  
 BigDecimal conversionTime = new BigDecimal(achieveTime);//把时间转换成秒并减去8个小时  
 System.*out*.println(conversionTime);  
 //全局坐标点设置  
 int ltX\_alarm = 0;  
 int ltY\_alarm = 0;  
 int rbX\_alarm = 0;  
 int rbY\_alarm = 0;  
 for (chipeak.zoneserver.proto.ZoneServerMessage.SingleHelSuiObjectData singleHelSuiObjectData : helSuiDetectionData.getSingleDataListList()) {  
 ltX\_alarm = singleHelSuiObjectData.getPedestrianRectangle().getLtX();  
 ltY\_alarm = singleHelSuiObjectData.getPedestrianRectangle().getLtY();  
 rbX\_alarm = singleHelSuiObjectData.getPedestrianRectangle().getRbX();  
 rbY\_alarm = singleHelSuiObjectData.getPedestrianRectangle().getRbY();  
 }  
  
 System.*out*.println(helSuiDetectionData.getAlarmFlag() + " " + "如果当前图片中有任何一种报警的存在，该标志位就设置为1，否则为0");  
 System.*out*.println(helSuiDetectionData.getAlarmType() + " " + "在该模式下，该值为：" + helSuiDetectionData.getAlarmType());  
 System.*out*.println(helSuiDetectionData.getImg().toByteArray().length + " " + "当前的图片");  
 if (helSuiDetectionData.getAlarmFlag() == 1) {  
 System.*out*.println(helSuiDetectionData.getAlarmFlag() + " " + "如果当前图片中有任何一种报警的存在，该标志位就设置为1，否则为0");  
 boolean suit\_alarm = false, helmet\_alarm = false;  
 String suit\_content = "", helmet\_content = "";  
// String dateStr = format.format(new Date());  
 byte[] suit\_img = helSuiDetectionData.getImg().toByteArray();  
 byte[] helmet\_img = helSuiDetectionData.getImg().toByteArray();  
 for (chipeak.zoneserver.proto.ZoneServerMessage.SingleHelSuiObjectData singleHelSuiObjectData : helSuiDetectionData.getSingleDataListList()) {  
 System.*out*.println(singleHelSuiObjectData.getHelmetAlarmFlag() + " " + "安全帽报警标志位");  
 System.*out*.println(singleHelSuiObjectData.getSuitAlarmFlag() + " " + "工作服报警标志位");  
 System.*out*.println(singleHelSuiObjectData.getHelmetAlarmConfidence() + " " + "戴安全帽的概率值");  
 System.*out*.println(singleHelSuiObjectData.getSuitAlarmConfidence() + " " + "穿工作服的概率");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianConfidence() + " " + "检测到人的概率");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle() + " " + "人体框");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtX() + " " + "矩形左上角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtY() + " " + "矩形左上角纵坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbX() + " " + "矩形右下角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbY() + " " + "矩形右下角纵坐标");  
 //安全帽标志位等于1（未戴安全帽报警接收图片）  
 if (singleHelSuiObjectData.getHelmetAlarmFlag() == 1) {  
 helmet\_alarm = true;  
 int helmet = singleHelSuiObjectData.getHelmetAlarmFlag();//安全帽报警标志位  
 float helmetConfidence = singleHelSuiObjectData.getHelmetAlarmConfidence();//戴安全帽的概率  
 String alarmType = helSuiDetectionData.getAlarmType();//报警类型  
 //人体框获取  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtX() + " " + "矩形左上角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtY() + " " + "矩形左上角纵坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbX() + " " + "矩形右下角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbY() + " " + "矩形右下角纵坐标");  
 String singleDataList = "[" + ltX\_alarm + "," + ltY\_alarm + "," + rbX\_alarm + "," + rbY\_alarm + "," + "]";//人体框坐标  
 System.*out*.println(singleDataList + "组合的人体框坐标");  
 helmet\_content += "single\_data\_list" + ":" + singleDataList + "\n" + "alarm\_type" + ":" + alarmType + "\n" + "helmet\_alarm" + ":" + helmet + "\n" + "helmet\_alarm\_confidence" + ":" + helmetConfidence + "\n";  
 //工装报警标志位等于1(未穿工装报警接收图片)  
 System.*out*.println("helmet\_alarm变量为假进入");  
 }  
 if (singleHelSuiObjectData.getSuitAlarmFlag() == 1) {  
 suit\_alarm = true;  
 int suit = singleHelSuiObjectData.getSuitAlarmFlag();//工作服报警标志位  
 float suitConfidence = singleHelSuiObjectData.getSuitAlarmConfidence();//穿工作服的概率  
 String alarmType = helSuiDetectionData.getAlarmType();//报警类型  
 //人体框获取  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtX() + " " + "矩形左上角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getLtY() + " " + "矩形左上角纵坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbX() + " " + "矩形右下角横坐标");  
 System.*out*.println(singleHelSuiObjectData.getPedestrianRectangle().getRbY() + " " + "矩形右下角纵坐标");  
 String singleDataList = "[" + ltX\_alarm + "," + ltY\_alarm + "," + rbX\_alarm + "," + rbY\_alarm + "," + "]";//人体框坐标  
 System.*out*.println(singleDataList + "组合的人体框坐标");  
 suit\_content += "single\_data\_list" + ":" + singleDataList + "\n" + "alarm\_type" + ":" + alarmType + "\n" + "suit\_alarm" + ":" + suit + "\n" + "suit\_alarm\_confidence" + ":" + suitConfidence + "\n";  
 }  
 }  
 //v2主题是没有图片所以存图片就会导致重连  
 if (helmet\_alarm) {  
 System.*out*.println("helmet\_alarm变量为假进入");  
 String saveTxt = FileUtil.*saveTxt*(hat\_no, "helmetAlarm" + conversionTime + ".txt", helmet\_content);  
 System.*out*.println("saved success : " + saveTxt);  
 }  
  
 if (suit\_alarm) {  
 System.*out*.println("suit\_alarm变量为假进入");  
 String saveTxt = FileUtil.*saveTxt*(suit\_no, "suitAlarmOne" + conversionTime + ".txt", suit\_content);  
 System.*out*.println("saved success : " + saveTxt);  
 }  
 }  
 }  
 }  
  
 @Override  
 public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {  
 System.*out*.println("deliveryComplete driver........" + iMqttDeliveryToken);  
 }  
}

**主题类：**

package com.example.snt.common;  
  
*/\*\*  
 \* 主题常量抽象类定义  
 \*/*public abstract class Constant {  
 //主题传参方式  
 public static String *mult\_topic*[] = {"zs/+/+/+/out","zs/+/systemctl","zs/+/+/+/v2/out"};  
}

**启动类：**

package com.example.snt;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class SntApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SntApplication.class, args);  
 }  
  
}

# 软件授权

## 授权限制

ChiPeakTEC DNN 是授权使用软件, 软件授权安装在DNN Engine

## 初始化配置

### 申请并导入License

#### 申请License

ChiPeakTEC DNN 是授权使用软件，软件授权安装在DNN Engine目录下，具体位置：/root/HIAI\_PROJECTS/ascend\_workspace/videoanalysispersonapp/7rc/ (7rc 为推理版本跟随版本变化，如7rc1，8rc)

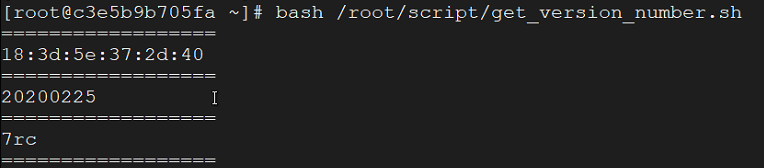
申请授权需要提供：盒子eth0网口mac地址，zoneserver，zoneengine版本号，DNN Engine版本号。

一键获取信息：

ssh登录盒子，进入容器内部docker exec -it 容器ID bash；

执行脚本 bash /root/script/get\_version\_number.sh

输出如下图：

请将脚本输出信息，以文本形式发送给售后人员。

#### 导入License

前提条件

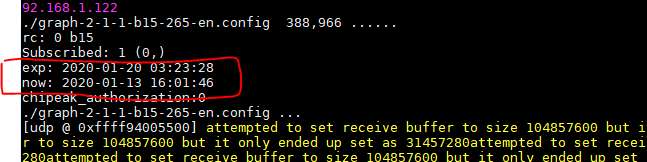
已申请好License，并准备好License文件。

操作步骤

copy license 文件到推理引擎启动目录， 进入启动目录，比如：

cd /opt/HIAI\_PROJECTS/ascend\_workspace/videoanalysispersonapp/7rc2

软件启动后窗口有到期显示：



# 检查ChiPeakTEC DNN版本

## 检查ChiPeakTEC DNN 软件版本

软件版本号定义：

ChiPeakTEC DNN 1.8.major.minor

1.8 代表当前软件版本适配硬件系列-如华为atlas系列；

major 代表功能性更新，比如从支持边缘设备到支持服务器设备；

minor 代表普通更新，包括 zs，ze，engine；

获取软件版本号方法：



## 检查AI设备固件版本

检查与ChiPeakTEC DNN软件配套的AI设备固件版本，在安装软件前请确认当前AI设备固件版本是否匹配，若版本不匹配，请执行升级操作。

检查方法一

依次单击“菜单 > 设备 > 设备列表 > 边缘设备”，进入边缘设备管理界面。

单击指定边缘设备的名称，进入边缘设备节点详细介绍界面。

查看“设备信息”里的“软件版本”或者选择“固件”查看固件页签下的“当前固件版本”。

* 固件版本为XXX及以上，支持HA特性，配置HA特性具体操作请参见zh-cn\_topic\_0191620319.xml。
* 固件版本为XX以下，请升级边缘设备固件版本，具体操作请参见10.1 （推荐）使用ChiPeakTEC DNN升级ChiPeakTEC DNN

检查方法二

依次单击“菜单 > 配置 > 升级管理 > 设备版本状态”，进入设备版本状态界面。

单击指定边缘设备前的。

打开详细信息区域，可查看到相关固件的名称、型号、厂商、当前版本、目标版本等信息。

确认边缘设备“当前版本”。

* 当前版本为XXX及以上，支持HA特性，配置HA特性具体操作请参见zh-cn\_topic\_0191620319.xml。
* 当前版本为XX以下，请升级边缘设备固件版本，具体操作请参见10.1 （推荐）使用ChiPeakTEC DNN升级ChiPeakTEC DNN。

----结束

# 升级ChiPeakTEC DNN

## 升级ChiPeakTEC DNN容器镜像

采用FD升级更新：

使用FD下发镜像，镜像文件格式要求为tgz、gz格式，下发完成后容器按事先约定的模版启动；

本地升级更新：

采用scp协议下发镜像，镜像文件格式无限制，不能自动启动容器，连接盒子执行命令启动容器，灵活添加参数；

## 升级业务软件

**更新准备：**

在管理服务器端/atlasdata/企业AI设备/ 目录下存放要更新数据（算法目录及算法定义文件，om文件，graph文件等）；

**更新操作：**

在客户端(AI盒子)上重启容器或重启AI盒子，工具软件自动监测服务端数据，实现更新；

**可更新的内容：**

可更新的数据包括，算法目录及算法定义文件，om文件，graph文件；

**数据更新要求：**

存放要更新的新数据，移除旧数据；

# 卸载ChiPeakTEC DNN

卸载ChiPeakTEC DNN前请将ChiPeakTEC DNN上纳管的设备移除。

AI盒子上删除容器与镜像

#查看容器

docker ps -a

#停止容器

docker stop CONTAINER ID

#删除容器

docker rm CONTAINER ID

#查看镜像

docker images

#删除镜像

docker rmi REPOSITORY

服务端删除rabbitmq服务与rsync服务删除数据存储目录

#删除数据存储目录

~]# rm -rf /atlasdata/ #使用此命令请确认当前目录与需要删除的目录

#删除软件

~]# yum remove rabbitmq rsync -y

# 其他

## 摄像头配置

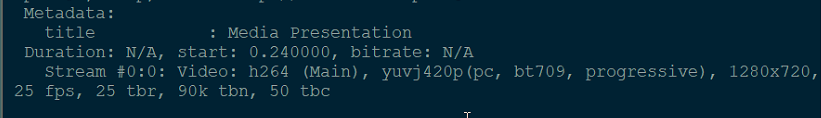
注意事项

完成设备上电，请等待1分钟后，再将以太网线的一端插入ChiPeakTEC DNN的网口，另一端插入PC端口。

## 摄像头配置检查

ffprobe “rtsp://admin:admin@192.168.1.64:554”

下图为摄像头开启流服务的输出：



## 带宽控制

由于网路带宽的限制（参照网络拓扑图），报警短视频的传输不需要实时性，而且要控制带宽以保障系统实时报警数据。

rsync --bwlimit=KBPS [options] src dst

rsync --bwlimit=1000 /var/www/html/ \  
backups@server1.cyberciti.biz:~/mysite.backups/

1. FAQ
   1. 系统重启
      1. 配置的IP地址已存在，请重新配置后重试

问题描述

ChiPeakTEC DNN配置完成后，单击“立即体验”，提示“配置的IP地址已存在，请重新配置后重试！”。

解决方案

检查已有ChiPeakTEC DNN的所有IP地址。单击“上一步”，直到返回配置网络界面，重新配置IP地址，确保新配置的IP地址未被使用。

1. 如何获取帮助

日常维护或故障处理过程中遇到难以解决或者重大问题时，请寻求芯峰科技（广州）有限公司的技术支持。

* 1. 收集必要的故障信息

在进行故障处理前，需要收集必要的故障信息。

收集的信息主要包括：

* 客户的详细名称、地址
* 联系人姓名、电话号码
* 故障发生的具体时间
* 故障现象的详细描述
* 设备类型及软件版本
* 故障后已采取的措施和结果
* 问题的级别及希望解决的时间
  1. 如何使用文档

芯峰科技（广州）有限公司提供全面的随设备发货的指导文档。指导文档能解决您在日常维护或故障处理过程中遇到的常见问题。

为了更好的解决故障，在寻求芯峰技术支持前，建议充分使用指导文档。

* 1. 获取技术支持

芯峰科技（广州）有限公司通过办事处、公司二级技术支持体系、电话技术指导、远程支持及现场技术支持等方式向用户提供及时有效的技术支持。

技术支持邮箱

查阅技术支持网站上的技术资料：[http://support.chipeak.com](http://support.huawei.com)。

获取芯峰技术支持

如果在设备维护或故障处理过程中，遇到难以确定或难以解决的问题，通过文档的指导仍然不能解决，请通过如下方式获取技术支持：

* 联系芯峰科技（广州）有限公司客户服务中心。

中国区企业用户请通过以下方式联系我们：

* 客户服务电话：400-8778-638
* 客户服务邮箱：[support@chipeak.com](mailto:support@chipeak.com)
* 联系芯峰科技（广州）有限公司驻当地办事处的技术支持人员。

公司总部：广州市南沙区进港大道8号南沙城二期1205

成都办事处：四川省成都市锦江区锦兴路1号1栋7层711号