IBM Maximo PQI SaaS Visual Insights

用户指南

注

在使用本信息及其支持的产品之前，请先阅读第 113 页的『声明』中的信息。

第一版（2019 年 3 月）

目录

第 1 章 产品概述

IBM® Maximo® PQI SaaS Visual Insights 是一种质量监控和警报解决方案，可以接收在制品、成品和组合件的图像，并将其分类成缺陷类别。

角色

要了解 Maximo PQI SaaS Visual Insights，先了解不同角色是如何与产品进行交互的会很有用。

表 1. *Maximo PQI SaaS Visual Insights* 角色

角色	描述
模型管理者	管理缺陷类型和模型，上载特定缺陷类型的图像集，培训模型，以及将可执行模型分发到 Edge。
检验员	验证产品生成的检查结果，根据需要更改缺陷类型，标记未知缺陷类型，并将其传递给检验员主管以进行进一步评估。
检验员主管	复核检验员的检查结果。复查未知缺陷类型，并对其进行分类。复查 KPI 仪表板，此仪表板包含单位缺陷数和缺陷率。

如何备份和复原数据?

IBM Open Platform 冗余用于保护大数据环境中的客户数据。此外，Tivoli® Storage Manager 用于备份生产环境中的数据，包括 Linux 文件。Linux 文件包含客户上载的文件以及中间件/应用程序配置/日志文件。

下表显示了解决方案各个方面的备份计划。

表 2. 解决方案数据的备份计划

数据	备份类型	频率	时间（中央标准时间）	保留期
文件	完整	双周	每月 1 号和 16 号的 00:00 - 03:00	5 周
文件	递增	每天两次	00:00 - 03:00 和 12:00 - 15:00	14 天

在备份窗口中，可以访问解决方案，但性能会受影响。

如果发生系统故障而导致数据损坏或丢失，IBM 会根据数据备份策略，帮助将数据复原到恢复时间点。

本发行版中的新增内容

IBM Maximo PQI SaaS Visual Insights 中提供了以下新功能。

本发行版中的新增内容

2019 年 3 月

- 支持 TensorFlow 模型上载和图像评分

前发行版中的新增内容

2019 年 1 月

- 产品名称已更改为 IBM Maximo PQI SaaS Visual Insights。
- 修复了并行培训分类模型时的培训曲线问题。
- 修复了重新培训 YOLO 模型时的模型培训问题。
- 修复了对 FRCNN-VGG16 模型进行验证和评分时的问题。
- 更新了连接属性以提高评分并行性。

2018 年 11 月

- 在一个租户中支持多个 Edge 集群。可将模型部署到多个 Edge 集群。
- 支持分类培训集群，可在多个培训作业提交期间进行扩展并减少等待时间。
- 支持 Edge 版本管理。在注册或升级 Edge 时，将维护版本信息。在部署模型时，可检查 Edge 的版本。
- 添加了图像归档策略以指定将归档哪些类型的图像并将其用于重新培训模型。
- 产品会在评分失败时返回错误详细信息。
- 解决了会话截取的安全性问题。
- 更新了定价规则，以计算培训文件数、评分文件数和已用存储量。

2018 年 9 月

- 支持在培训分类模型时将 epoch 作为单元。这可使定义超参数更容易。
- 在培训对象检测模型时启用了培训服务器集群。此集群机制支持在一个培训服务器无法处理多个并行请求时进行扩展。
- 支持 Edge 在 Linux on Power Systems Servers 平台上运行。
- 在 MacOS 上，上载图像 zip 文件和模型 zip 文件时，支持使用打包的 zip 文件。
- 改进了自动标注工具。

2018 年 6 月

- 能够通过小型注释数据集自动标记培训图像，这可降低注释数据所需的成本和时间。
- 基于实地反馈增强了 Edge 的功能，包括从中心除去 NFS 访问，支持主 Edge 高可用性以及支持在特定 Edge 和特定 GPU 上运行模型。
- 我们的数据研究员专家基于预培训网络改进了模型培训，以支持高精度小型迭代。

2018 年 3 月

- 支持浏览模型目录中的模型，以及使用共享模型测试自己的图像。
- 支持使用模型目录中的共享模型，以及使用特定图像来培训自己的模型。
- 支持在创建新模型时，选择不同的模型类型、算法、网络和超参数。
- 支持联机模型培训。模型文件会自动生成，无需其他培训工具。
- 培训仪表板，用于显示模型损失和精确性值的实时图表以及迭代和详细日志文件。
- 在培训过程中列出可用快照。支持将一个快照用作已培训模型。

· 分三步执行的 API 指南，用于帮助集成服务，以注册、部署和测试模型。
· 可用于将图像发送到服务器并复查检查结果的模拟器。

2017 年 12 月

· 用户界面支持数据研究员下载数据集和附加培训的模型。
· 支持重新培训对象检测模型。
· 用户界面支持模型管理者监视重新培训队列。
· 支持 Edge 使用脱机方式。在脱机方式下，Edge 会在本地存储检查结果，并将结果批量发送到中心应用程序。
· 支持 Edge 集群通过负载均衡来处理评分请求。
· 支持对数据集采样以重新培训分类模型。
· 能够导出 CSV 文件，以便与 IBM Prescriptive Quality 相集成。
· 公共 REST API 支持执行多种功能，例如获取所有现有数据组、获取属于一个数据组的所有数据文件和获取所有现有模型。

2017 年 9 月

· 现在，可以通过图像组使用一个或多个压缩图像文件来表示同一类型的图像。
· 添加了对多个模型版本的支持，这些模型版本共享相同的图像组，但使用不同的图像文件培训模型。
· 添加了模型重新培训过程。可以使用不同的图像文件自动或手动重新培训新的模型版本。
· 添加了对模型验证的支持。验证过程会基于验证图像文件来计算并显示模型精确性报告。
· 现在，可以在一个图像上使用多个缺陷框和缺陷类型，为检验员标记缺陷位置。
· 现在，可以在一个图像上显示多个缺陷位置。还可以添加、调整和删除图像上的缺陷框。
· 更新了 KPI 仪表板，以包含单位缺陷数和缺陷率供检验员主管使用。
· 添加了对对象检测模型的支持，可使用 CNN 模型检测一个图像中的多个缺陷。

辅助功能

辅助功能可帮助那些身体残障（例如行动不便或视力有障碍）的用户使用信息技术产品。

有关 IBM 对辅助功能的承诺的信息，请参阅 IBM Accessibility Center (www.ibm.com/able)。

HTML 文档具有辅助功能。PDF 文档是补充性的，因此不包括附加的辅助功能。

第 2 章 创建 Edge 系统

Maximo PQI SaaS Visual Insights 由中心应用程序和 Edge 组成。Edge 为 Linux 系统，用于执行运行时缺陷检测。

Edge 系统使用 Caffe 深度学习框架。Caffe 是一种专用人工神经网络 (ANN) 培训环境。深度学习需要大量处理资源。深度学习可以使用图形处理单元 (GPU) 来有效地执行。虽然大多数深度学习框架还支持 CPU 处理，但 GPU 处理为生产环境提供了恰当的性能。

Edge 可建立集群以进行负载均衡。一个集群由一个主 Edge 和多个从属 Edge 组成。工作站发送图像供处理时，主 Edge 会接收该图像，然后将其发送到具有可用 GPU 资源的从属 Edge。

Edge 集群体系结构可扩展。如果需要更多处理资源，可以添加更多 Edge。

创建 Edge 时，可以指定该 Edge 是主 Edge 还是从属 Edge。创建的第一个 Edge 必须是主 Edge。创建 Edge 后，即不能更改 Edge 类型。对于每个租户，只能创建一个主 Edge。在删除所有从属 Edge 之后，才能删除主 Edge。

主 Edge 可以处于连接或独立状态。连接状态意味着在 Edge 上对图像评分时，检查结果会立即发送到中心应用程序。独立状态意味着检查结果会存储在 Edge 上，直到从“**预览 Edge**”对话框中选择**拉取结果**或调用部署在该 Edge 上的服务时才会发送。

Edge 系统需求

创建 Edge 系统之前，请确保系统满足相应需求。

- 下列其中一个操作系统：
  - Ubuntu 16.04（x86_64 上）
  - Red Hat Enterprise Linux 7.5（x86_64 上）
  - Red Hat Enterprise Linux 7.5（IBM Power System 上）
- CPU 体系结构：x86_64 或 IBM Power System
- 4 核处理器
- 64GB 内存
- 2TB 硬盘驱动器
- 一块或多块 NVIDIA GPU 卡

打开 Edge 端口

使用 Edge 之前，必须打开 Edge 系统使用的防火墙端口。

关于此任务

如果在 Edge 上启用并激活了 Uncomplicated Firewall (UFW)，请使用以下命令打开网络文件系统 (NFS) 服务的 UFW，并在 Edge 上打开以下防火墙端口：

```
sudo ufw enable
sudo ufw allow nfs
sudo ufw allow 22
sudo ufw allow 5005
sudo ufw allow 5070:5090/tcp
sudo ufw allow 6005
sudo ufw allow 8449
```

安装用于 Ubuntu 的 NVIDIA GPU 软件包

使用此任务以安装用于 Ubuntu 系统的 NVIDIA GPU 软件包。要启用 GPU 处理，必须安装必需的 NVIDIA GPU 软件包。

过程

1. 下载并安装 NVIDIA GPU 的驱动程序。以下链接中提供了 Ubuntu 的 NVIDIA 驱动程序列表：Binary Driver How to - Nvidia。下面是命令示例：
   ```
   sudo apt-get install ubuntu-drivers-common
   sudo ubuntu-drivers devices
   sudo apt-get install nvidia-384
   ```
2. 使用以下命令检查 NVIDIA 驱动程序是否安装正确：
   ```
   sudo nvidia-smi
   ```
3. 下载并安装 NVIDIA CUDA 工具箱和对应的 CUDNN 库。支持 CUDA 8.0、CUDA 9.0 和 CUDA 10.0。以下命令是 CUDA 8.0 的示例。
   ```
   wget
   http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/
   cuda-repo-ubuntu1604_8.0.61-1_amd64.deb
   ```
4. 使用以下命令在目标服务器上安装 CUDA 文件：
   ```
   sudo dpkg -i cuda-repo-ubuntu1604_8.0.61-1_amd64.deb
   sudo apt-get update
   sudo apt-get install cuda
   ```
   NVIDIA 升级工具箱时，可使用此命令来获取工具箱的更高版本。建议安装 CUDA 8.0，因为这是经过测试的版本。如果发现安装的是更高版本，可以使用以下命令将其降级到 CUDA 8.0：
   ```
   sudo apt-get remove cuda
   sudo apt-get install cuda-8-0
   sudo ln -s /usr/local/cuda-8.0 /usr/local/cuda
   ```
5. 从以下链接获取 NVIDIA CUDA Deep Neural Network 库 `cudnn-8.0-linux-x64-v6.0.tgz`：https://developer.nvidia.com/compute/machine-learning/cudnn/secure/v6/prod/8.0_20170307/cudnn-8.0-linux-x64-v6.0-tgz。下载该文件之前，您可能需要创建帐户并登录。
6. 使用以下命令将 `cudnn-8.0-linux-x64-v6.0.tgz` 文件解压到 cuda 安装目录：
   ```
   sudo tar -xvf cudnn-8.0-linux-x64-v6.0.tgz -C /usr/local
   ```
7. 使用以下命令设置环境变量：
   ```
   export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
   export PATH=/usr/local/cuda/bin:$PATH
   ```
   此外，将这些命令添加到 `~/.bashrc` 脚本。
8. 使用以下命令安装 NVIDIA NCCL 软件包：
   ```
   git clone https://github.com/NVIDIA/nccl.git
   cd nccl
   sudo make install -j4
   ```

安装用于 Linux on NVIDIA Tesla K80 Power Systems Servers 的 NVIDIA GPU 软件包

使用此任务以安装用于 Linux on NVIDIA Tesla K80 Power Systems Servers 的 NVIDIA GPU 软件包。要启用 GPU 处理，必须安装必需的 NVIDIA GPU 软件包。

关于此任务

对于 Power8 系统，使用 CUDA 8，如以下任务中所述。对于 Power9 系统，在以下任务中用 CUDA 10 替换 CUDA 8。

过程

1. 下载并安装 NVIDIA GPU 的驱动程序。以下链接中提供了 Linux on Power Systems Servers 的 NVIDIA 驱动程序列表：NVIDIA Driver Downloads。遵循下载页面上的安装指示信息进行操作。
2. 使用以下命令下载 CUDA 存储库文件并安装 CUDA 8：
```
wget https://developer.download.nvidia.com/compute/cuda/repos/rhel7/ppc64le/
cuda-repo-rhel7-8.0.61-1.ppc64le.rpm
rpm -i cuda-repo-rhel7-8.0.61-1.ppc64le.rpm
yum clean all
yum install cuda
```
3. 对于 CUDA 8，从以下 URL 下载 NVIDIA CUDA Deep Neural Network 库 `cudnn-8.0-linux-ppc64le-v6.0-tgz`：https://developer.nvidia.com/compute/machine-learning/cudnn/secure/v6/prod/8.0_20170307/cudnn-8.0-linux-ppc64le-v6.0-tgz。对于 CUDA 10，从以下 URL 下载 NVIDIA CUDA Deep Neural Network 库 `cudnn-10.0-linux-ppc64le-v7.3.1.20.tgz`：https://developer.download.nvidia.com/compute/cuda/repos/rhel7/ppc64le/cuda-10.0.130-1.ppc64le.rpm。下载该文件之前，您可能需要创建帐户并登录。
4. 使用以下命令将 `cudnn-8.0-linux-ppc64le-v6.0.tgz` 文件解压到 cuda 安装目录：
```
sudo tar -xvf cudnn-8.0-linux-ppc64le-v6.0.tgz -C /usr/local
```
5. 使用以下命令设置环境变量：
```
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```
此外，将此命令添加到 ~/.bashrc 脚本。
6. 使用以下命令安装 NVIDIA NCCL 软件包：
```
git clone https://github.com/NVIDIA/nccl.git
cd nccl
sudo make install -j4
```

安装用于 Linux on NVIDIA Tesla V100 Power Systems Servers 的 NVIDIA GPU 软件包

使用此任务以安装用于 Linux on NVIDIA Tesla V100 Power Systems Servers 的 NVIDIA GPU 软件包。要启用 GPU 处理，必须安装必需的 NVIDIA GPU 软件包。

过程

1. 从 developer.nvidia.com/cuda-92-download-archive 下载并安装 NVIDIA CUDA 9.2.148。
   a) 选择操作系统：Linux。
   b) 选择体系结构：ppc64le。
   c) 选择分发版：RHEL。
   d) 选择版本：7。
   e) 选择安装程序类型：rpm（本地）。本地 rpm 优先于网络 rpm，因为这可确保安装的版本是下载的版本。如果使用网络 rpm，`yum install cuda` 命令始终会安装最新版本的 CUDA Toolkit。
   f) 单击**下载**以下载基本安装程序。
   g) 单击**下载**以下载补丁 1。
   h) 遵循 CUDA Quick Start Guide 中的 Linux on POWER 安装指示信息进行操作，包括描述如何通过更新 PATH 和 `LD_LIBRARY_PATH` 来设置 CUDA 开发环境的步骤。
2. 从 http://www.nvidia.com/Download/index.aspx 下载 NVIDIA 410.104 驱动程序。
   a) 选择产品类型：Tesla。
   b) 选择产品系列：V-Series。
   c) 选择产品：Tesla V100。
   d) 选择操作系统：Linux POWER LE RHEL 7。
   e) 选择 CUDA Toolkit：10.0。

f) 单击**搜索**以转至下载链接，然后单击**下载**。

3. **注:** 对于 IBM Power® System AC922 系统，在安装最新的 GPU 驱动程序之前，需要执行操作系统和系统固件更新。

   安装 CUDA 和 GPU 驱动程序:

   a) 安装 CUDA 基本存储库 rpm。

   b) 安装 CUDA 补丁 1 存储库 rpm。

   c) 安装 GPU 驱动程序存储库 rpm。

   d) 运行以下命令安装 CUDA、补丁和 GPU 驱动程序:
   ```
   sudo yum install cuda
   ```

   e) 重新启动系统以激活驱动程序。

4. 使用以下 shell 命令启用 NVIDIA 系统持久性服务:
   ```
   systemctl enable nvidia-persistenced
   ```
5. 使用以下 shell 命令检查 NVIDIA 驱动程序:
   ```
   nvidia-smi
   ```
6. 从 developer.nvidia.com/cudnn 下载适用于 CUDA 10.0 的 NVIDIA cuDNN V7.4.2（适用于 Linux (Power8/Power9) 的 cuDNN V7.4.2 库）。需要注册加入 NVIDIA Accelerated Computing Developer Program。
7. 从 developer.nvidia.com/nccl 下载适用于 CUDA 10.0 的 NVIDIA NCCL V2.3.7（与 NCCL 2.3.7 操作系统无关，并且适用于 CUDA 10.0 和 IBM Power）。需要注册加入 NVIDIA Accelerated Computing Developer Program。
8. 使用以下命令安装 cuDNN V7.4.2 和 NCCL V2.3.7 软件包，然后刷新共享库高速缓存:
   ```
   sudo tar -C /usr/local --no-same-owner -xzvf cudnn-9.2-linux-ppc64le-
   v7.4.2.tgz
   sudo tar -C /usr/local --no-same-owner -xzvf nccl_2.3.7+cuda10.0_ppc64le.tgz
   sudo ldconfig
   ```

安装用于 Ubuntu 的 Caffe

必须安装 Caffe 深度学习框架和相关软件包。Caffe 用于模型培训和缺陷分类。

过程

1. 使用以下命令安装 Caffe 所需的软件包:
   ```
   sudo apt-get update
   sudo apt-get upgrade
   sudo apt-get install -y build-essential cmake git pkg-config
   sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev
   libhdf5-serial-dev protobuf-compiler
   sudo apt-get install -y libatlas-base-dev libjasper-dev
   sudo apt-get install -y --no-install-recommends libboost-all-dev
   sudo apt-get install -y libgflags-dev libgoogle-glog-dev liblmdb-dev
   sudo apt-get install -y python-pip
   sudo apt-get install -y python-dev
   sudo apt-get install -y python-numpy python-scipy
   sudo apt-get install -y libopencv-dev
   sudo pip install opencv-python
   sudo pip install flask_httpauth
   sudo pip install gevent
   sudo pip install pyinotify
   sudo pip install tornado
   ```
2. 使用以下命令下载 Caffe 源代码:
   ```
   wget https://github.com/BVLC/caffe/archive/1.0.zip
   ```
3. 使用以下命令解压软件包，然后进入软件包目录:

```
unzip 1.0.zip
cd ./caffe-1.0
```

4. 使用以下命令创建 make 配置文件的副本：
```
cp Makefile.config.example Makefile.config
```
5. 在 Makefile.config 文件中添加以下变量：
```
USE_CUDNN := 1
CUDA_DIR := /usr/local/cuda
PYTHON_INCLUDE := /usr/include/python2.7 \
/usr/lib/python2.7/dist-packages/numpy/core/include
PYTHON_LIB := /usr/lib/x86_64-linux-gnu
WITH_PYTHON_LAYER := 1
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include \
/usr/include/hdf5/serial
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib \
/usr/lib/x86_64-linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial
```
6. 在 caffe-1.0 目录中，运行以下命令：
```
find . -type f -exec sed -i -e 's^"hdf5.h"^"hdf5/serial/hdf5.h"^g' -e
's^"hdf5_hl.h"^"hdf5/serial/hdf5_hl.h"^g' '{}' \;
```
7. 运行以下命令：
```
cd /usr/lib/x86_64-linux-gnu
sudo ln -s libhdf5_serial.so.10.1.0 libhdf5.so
sudo ln -s libhdf5_serial_hl.so.10.0.2 libhdf5_hl.so
```
8. 使用以下命令在 caffe-1.0/python 目录中安装必需的 Python 软件包：
```
cd {caffe-installation-path}/caffe-1.0/python
for req in $(cat requirements.txt); do sudo -H pip install $req --upgrade;
done
```
其中，{caffe-installation-path} 是 Caffe 部署路径。
9. 打开 {caffe-installation-path} 目录中的 Makefile，并将参数 NVCCFLAGS 更改为以下设置：
```
NVCCFLAGS += -D_FORCE_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $
(COMMON_FLAGS)
```
10. 在主 Caffe 目录 caffe-1.0 中，使用以下命令开始 Caffe 构建和安装：
```
make all
make test
make runtest
make pycaffe
make distribute
```
11. 将以下行添加到 ~/.bashrc 脚本：
```
export PYTHONPATH="/usr/lib/python2.7:{caffe-installation-path}/caffe-1.0/
python:$PYTHONPATH"
```
其中，{caffe-installation-path} 是 Caffe 部署路径。

安装用于 Ubuntu 的 TensorFlow

如果要基于 TensorFlow 模型对图像评分，您必须在 Edge 上准备 TensorFlow 环境。

过程

1. 使用以下命令安装 tensorflow-gpu 1.4.0。确保在安装期间未报告任何错误，并且所有从属软件包的版本都正确。
```
pip install tensorflow-gpu==1.4.0
```
2. 将 TensorFlow 导入 Python，以验证 TensorFlow 是否正确安装。如果没有任何错误，说明 TensorFlow 已成功安装。
```
import tensorflow
```

安装用于 Linux on Power Systems Servers 的 Caffe

使用此任务以安装用于 Linux on Power Systems Servers 系统的 Caffe。必须安装 Caffe 深度学习框架和相关软件包。Caffe 用于模型培训和缺陷分类。

过程

1. 使用以下命令安装 Caffe 所需的软件包：
```
sudo yum clean all
sudo yum update
sudo yum install upgrade
sudo yum install -y libboost-*
sudo yum install -y gflags-devel glog-devel lmdb-devel
sudo yum install -y python-pip
sudo yum install -y python-devel
sudo yum install -y opencv-devel
sudo yum makecache
sudo yum install -y protobuf-devel leveldb-devel lmdb-devel snappy-devel
opencv-devel boost-devel hdf5-devel atlas-devel glog-devel gflags-devel
sudo yum install libpng-devel
sudo yum install freetype-devel
sudo yum install libjpeg-turbo-devel
sudo yum install opencv-python
sudo rpm -e --nodeps numpy
sudo pip install numpy
pip install --upgrade pip
sudo pip install flask_httpauth
sudo pip install gevent
sudo pip install pyinotify
ln -s /usr/local/cuda-10.0 /usr/local/cuda
pip install scikit-image
sudo pip install tornado
```
2. 使用以下命令链接 Atlas 库：
```
ln -fs /usr/lib64/atlas/libsatlas.so /usr/lib64/libatlas.so
ln -fs /usr/lib64/atlas/libsatlas.so /usr/lib64/libcblas.so
```
3. 使用以下命令下载 Caffe 源代码：
```
wget https://github.com/BVLC/caffe/archive/1.0.zip
```
4. 使用以下命令解压软件包，然后进入软件包目录：
```
unzip 1.0.zip
cd ./caffe-1.0
```
5. 替换以下 Caffe 文件：
   - 在 `/include/caffe/util/cudnn.hpp` 目录中，将 `cudnn.hpp` 文件替换为 GitHub 上 Caffe 存储库中最新的 `cudnn.hpp` 文件：github.com/BVLC/caffe.git
   - 将 `/src/caffe/layers` 文件夹中的所有 cudnn 文件替换为 GitHub 上 Caffe 存储库中最新的 cudnn 文件：github.com/BVLC/caffe.git

   例如，运行以下命令：
```
cp -rf /root/source/caffe-git/caffe-master/include/caffe/util/
cudnn.hpp /usr/local/caffe-1.0/include/caffe/util/
cp -rf /root/source/caffe-git/caffe-master/src/caffe/layers/cudnn_* /usr/
local/caffe-1.0/src/caffe/layers/
cp -rf /root/source/caffe-git/caffe-master/include/caffe/layers/
cudnn_* /usr/local/p/usr/local/caffe-1.0/include/caffe/layers/
```
6. 使用以下命令创建 make 配置文件的副本：
```
cp Makefile.config.example Makefile.config
```

7. 在 Makefile.config 文件中添加以下变量：

```
USE_CUDNN := 1
CUDA_DIR := /usr/local/cuda
PYTHON_INCLUDE := /usr/include/python2.7 \
/usr/lib64/python2.7/site-packages/numpy/core/include/
PYTHON_LIB := /usr/lib/gcc/ppc64le-redhat-linux/4.8.5/
WITH_PYTHON_LAYER := 1
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
/usr/local/cuda-10.0/targets/ppc64le-linux/include/
LIBRARY_DIRS
:= $(PYTHON_LIB) /usr/local/lib /usr/lib \
/usr/lib64 /usr/local/lib64
```

   将 CUDA_ARCH 更改为以下文本：

```
CUDA_ARCH := -gencode arch=compute_30,code=sm_30
-gencode arch=compute_35,code=sm_35
-gencode arch=compute_50,code=sm_50
-gencode arch=compute_52,code=sm_52
-gencode arch=compute_60,code=sm_60
-gencode arch=compute_61,code=sm_61
-gencode arch=compute_61,code=compute_61
```

8. 使用以下命令在 caffe-1.0/python 目录中安装必需的 Python 软件包：
   `cd` *`caffe-installation-path`*`/caffe-1.0/python`
   `for req in $(cat requirements.txt); do sudo -H pip install $req --upgrade; done`
   其中，*caffe-installation-path* 是 Caffe 部署路径。
9. 打开 *caffe-installation-path* 目录中的 Makefile，并将参数 NVCCFLAGS 更改为以下设置：
   `NVCCFLAGS += -D_FORCE_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)`
10. 在主 Caffe 目录 `caffe-1.0` 中，使用以下命令开始 Caffe 构建和安装：
    ```
    make all
    make test
    make runtest
    make pycaffe
    make distribute
    ```
11. 将以下行添加到 ~/.bashrc 脚本：
    `export PYTHONPATH="/usr/lib/python2.7:`*`caffe-installation-path`*`/caffe-1.0/python:$PYTHONPATH"`
    其中，*caffe-installation-path* 是 Caffe 部署路径。
12. 运行以下安装后测试：
    a) `make runtest | tee -a runtest.out`
    b) `grep -i OK runtest.out | wc –l`
       Caffe 测试输出必须为 2101
    c) `python -c "import caffe"`
       测试 Pycaffe 安装
    d) `tail -n 2 runtest.out`
       `runtest.out` 的内容应该包含以下文本：
       [==========] 2101 tests from 277 test cases ran. (291548 ms total)
       [PASSED] 2101 tests

Caffe 安装故障诊断

如果开始 Caffe 构建和安装时，日志中显示错误消息，那么可以执行相关步骤来尝试解决问题。

症状 1

开始 Caffe 构建和安装时，将显示以下消息：

```
    1.    In file included from ./include/caffe/util/device_alternate.hpp:40:0,
    2.                     from ./include/caffe/common.hpp:19,
    3.                     from src/caffe/common.cpp:7:
    4.    ./include/caffe/util/cudnn.hpp: In function  ‘void
caffe::cudnn::createPoolingDesc(cudnnPoolingStruct**,
        caffe::PoolingParameter_PoolMethod, cudnnPoolingMode_t*, int, int, int, int, int,
int)’ :
    5.    ./include/caffe/util/cudnn.hpp:127:41: error: too few arguments to function
 ‘cudnnStatus_t
        cudnnSetPooling2dDescriptor(cudnnPoolingDescriptor_t, cudnnPoolingMode_t,
cudnnNanPropagation_t, int,
        int, int, int, int, int)’
    6.             pad_h, pad_w, stride_h, stride_w));
    7.                                             ^
    8.    ./include/caffe/util/cudnn.hpp:15:28: note: in definition of macro  ‘CUDNN_CHECK’
    9.         cudnnStatus_t status = condition; \
    10.                             ^
    11.    In file included from ./include/caffe/util/cudnn.hpp:5:0,
    12.                     from ./include/caffe/util/device_alternate.hpp:40,
    13.                     from ./include/caffe/common.hpp:19,
    14.                     from src/caffe/common.cpp:7:
    15.    /usr/local/cuda-7.5//include/cudnn.h:803:27: note: declared here
    16.     cudnnStatus_t CUDNNWINAPI cudnnSetPooling2dDescriptor(
    17.                               ^
    18.    make: *** [.build_release/src/caffe/common.o] Error 1
    19.
```

解决问题 1

要解决此错误，请参阅以下步骤：

1. 在 `/include/caffe/util/cudnn.hpp` 目录中，将 `cudnn.hpp` 文件替换为 GitHub 上 Caffe 存储库中最新的 `cudnn.hpp` 文件。
2. 在 `/src/caffe/layers` 文件夹中，将 `/src/caffe/layers` 文件夹中的所有 `cudnn` 文件替换为 GitHub 上 Caffe 存储库中最新的 `cudnn` 文件。

症状 2

在 `caffe-1.0/python` 目录中安装必需的 Python 软件包时，将显示以下消息：

```
Traceback (most recent call last):
  File "/usr/bin/pip", line 11, in <module>
    sys.exit(main())
  File "/usr/lib/python2.7/dist-packages/pip/__init__.py", line 215, in main
    locale.setlocale(locale.LC_ALL, '')
  File "/usr/lib/python2.7/locale.py", line 581, in setlocale
    return _setlocale(category, locale)
locale.Error: unsupported locale setting
Traceback (most recent call last):
  File "/usr/bin/pip", line 11, in <module>
    sys.exit(main())
  File "/usr/lib/python2.7/dist-packages/pip/__init__.py", line 215, in main
    locale.setlocale(locale.LC_ALL, '')
  File "/usr/lib/python2.7/locale.py", line 581, in setlocale
    return _setlocale(category, locale)
locale.Error: unsupported locale setting
```

解决问题 2

要解决此错误，请运行以下命令：
`export LC_ALL=C`

症状 3

开始 Caffe 构建和安装时，将显示以下消息：

```
nvcc fatal   : Unsupported gpu architecture 'compute_20'
Makefile:595: recipe for target '.build_release/cuda/src/caffe/layers/prelu_layer.o' failed
make: *** [.build_release/cuda/src/caffe/layers/prelu_layer.o] Error 1
```

解决问题 3

注释掉 `Makefile.config` 中的 `-gencode arch=compute_20`。

症状 4

开始 Caffe 构建和安装时，将显示以下消息：

```
PROTOC src/caffe/proto/caffe.proto
make: protoc: Command not found
Makefile:639: recipe for target '.build_release/src/caffe/proto/caffe.pb.cc' failed
make: *** [.build_release/src/caffe/proto/caffe.pb.cc] Error 127
```

解决问题 4

运行以下命令安装 protoc 程序：

`sudo apt install protobuf-compiler`

症状 5

开始 Caffe 构建和安装时，将显示以下消息：

```
src/caffe/layers/hdf5_data_layer.cpp:13:30: fatal error: hdf5/serial/hdf5.
h: No such file or directory
compilation terminated.
Makefile:582: recipe for target '.build_release/src/caffe/layers/hdf5_data_layer.o' failed
make: *** [.build_release/src/caffe/layers/hdf5_data_layer.o] Error 1
```

解决问题 5

您可能需要使用以下命令安装 Caffe 从属软件包：

`sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler`
`sudo apt-get install --no-install-recommends libboost-all-dev`
`sudo apt-get install libopenblas-dev liblapack-dev libatlas-base-dev`
`sudo apt-get install libgflags-dev libgoogle-glog-dev liblmdb-dev`

安装 OpenCV

如果需要使用 Open Source Computer Vision (OpenCV) 3.2 的功能来定制模型，那么可以安装 OpenCV 3.2 库。

过程

1. 从 GitHub 获取 OpenCV 源代码：
   `wget https://github.com/opencv/opencv/archive/3.2.0.zip`

2. 解压下载的软件包，然后切换到软件包目录：
   ```
   unzip 3.2.0.zip
   cd opencv-3.2.0
   ```
3. 创建构建子目录，然后切换到该目录：
   ```
   mkdir build
   cd build
   ```
4. 准备和生成构建配置：
   ```
   cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
   WITH_TBB=ON -D WITH_V4L=ON ..
   ```
5. 编译和构建软件包：
   ```
   make -j $(($(nproc) + 1))
   ```
6. 安装软件包：
   ```
   sudo make install
   ```
7. 将库和模块注册到系统：
   ```
   sudo /bin/bash -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
   sudo ldconfig
   ```
8. 如果需要，请卸载旧版本的 OpenCV 以避免版本冲突：
   ```
   sudo apt-get autoremove libopencv-dev
   ```

安装对象检测库

您可安装对象检测库，以便可以在 Edge 上运行对象检测模型。

关于此任务

IBM Maximo PQI SaaS Visual Insights 支持以下对象检测库：YOLO (you only look once)、Faster R-CNN 和 SSD（单图多框检测器）。

过程

1. 使用以下命令安装相关的 Python 软件包：
   ```
   sudo apt-get install python-numpy
   sudo apt-get install python-scipy
   sudo pip install cython
   sudo pip install easydict
   sudo pip install uuid
   sudo pip install multiprocessing
   ```
2. 安装以下所有库：

库	安装指示信息
YOLO V2 库	a. 运行以下命令获取 YOLO 源代码： **git clone --recursive https://github.com/pjreddie/darknet.git** **cd darknet** **git checkout 691debd** b. 编辑 Makefile 文件以启用 **GPU**，并根据您的机器配置，选择正确的 **GPU ARCH** 参数： **vi Makefile** **GPU=1** c. 运行以下命令来编译 YOLO： **make**
Faster-RCNN Python 库	a. 运行以下命令获取 Faster R-CNN 源代码：

库	安装指示信息

	git clone --recursive https://github.com/rbgirshick/py-faster-rcnn.git
	b. 在 py-faster-rcnn 目录的 lib 文件夹中，运行以下命令编译 Cython：
	make
	c. 使用以下命令转至 py-faster-rcnn 目录下的 caffe-fast-rcnn 目录，并创建 Makefile 配置文件的副本：
	cd caffe-fast-rcnn
	cp Makefile.config.example Makefile.config
	d. 在 Makefile.config 文件中添加以下变量：
	`USE_CUDNN := 1`
	`CUDA_DIR := /usr/local/cuda`
	`PYTHON_INCLUDE := /usr/include/python2.7 \/usr/lib/`
	`python2.7/ dist-packages/numpy/core/includePYTHON_LIB :=`
	`/usr/lib/x86_64-linux-gnuWITH_PYTHON_LAYER := 1`
	`INCLUDE_DIRS :=`
	`$(PYTHON_INCLUDE) /usr/local/include \/usr/include/hdf5/`
	`serialLIBRARY_DIRS`
	`:= $(PYTHON_LIB) /usr/local/lib /usr/lib \/usr/lib/`
	`x86_64-linux-gnu /usr/lib/x86_64-linux-gnu/hdf5/serial`
	e. 运行以下命令编译 Caffe：
	make
	f. 运行以下命令使用 Python 层来编译 pycaffe：
	make pycaffe

SSD 库	a. 运行以下命令来获取 SSD 源代码： **git clone --recursive https://github.com/weiliu89/caffe.git ~/ssd-caffe** **cd ~/ssd-caffe** **git checkout ssd** b. 使用以下命令创建 Makefile 配置文件的副本： **cp Makefile.config.example Makefile.config** c. 编辑 Makefile 配置文件，并根据您的机器配置，更改 **CUDA_ARCH**、**BLAS** 和 **PYTHON_INCLUDE** 参数。 d. 在一行上运行以下命令： **find . -type f -exec sed -i -e 's^"hdf5.h"^"hdf5/serial/hdf5.h"^g' -e 's^"hdf5_hl.h"^"hdf5/serial/hdf5_hl.h"^g' '{}' \;** e. 使用以下命令来编译代码： **make -j8** f. 使用以下命令来编译 Python 层： **make py** g. 使用以下命令来编译测试： **make test -j8**

3. 将以下环境变量添加到 ~/.bashrc 文件：*YOLO_HOME*、*FRCNN_HOME* 和 *SSD_HOME*。以下文本是添加环境变量的示例：YOLO_HOME=~/darknet/ FRCNN_HOME=~/py-faster-rcnn/ SSD_HOME=~/ssd-caffe/。

Faster-RCNN Python 库安装故障诊断

如果安装 Faster-RCNN Python 库时显示错误消息，那么可以执行相关步骤来尝试解决问题。

症状

安装 Faster-RCNN Python 库时，显示以下 CUDNN 编译错误：

```
CXX src/caffe/layers/hdf5_data_layer.cpp
./include/caffe/util/cudnn.hpp: In function 'const char* cudnnGetErrorString(cudnnStatus_t)':
./include/caffe/util/cudnn.hpp:21:10: warning: enumeration value
'CUDNN_STATUS_RUNTIME_PREREQUISITE_MISSING' not handled in switch
./include/caffe/util/cudnn.hpp:15:28: note: in definition of macro 'CUDNN_CHECK'
     cudnnStatus_t status = condition;
Makefile:563: recipe for target '.build_release/src/caffe/layers/hdf5_data_layer.o' failed
make: *** [.build_release/src/caffe/layers/hdf5_data_layer.o] Error 1
```

在具有 CUDA 10 的 Power9 系统上，显示以下 CUDNN 编译错误：

```
./include/caffe/util/cudnn.hpp:127:41: error: too few arguments to function
'cudnnStatus_t cudnnSetPooling2dDescriptor(cudnnPoolingDescriptor_t, cudnnPoolingMode_t,
cudnnNanPropagation_t, int, int, int, int, int, int)' pad_h, pad_w, stride_h, stride_w));
^./include/caffe/util/cudnn.hpp:15:28: note: in definition of macro 'CUDNN_CHECK'
cudnnStatus_t status = condition; \ ^In file included from ./include/caffe/util/
cudnn.hpp:5:0, from ./include/caffe/util/device_alternate.hpp:40, from ./include/
caffe/common.hpp:19, from ./include/caffe/blob.hpp:8, from src/caffe/blob.cpp:4:/
usr/local/cuda/include/cudnn.h:991:1: note: declared here cudnnSetPooling2dDescriptor
(cudnnPoolingDescriptor_t poolingDesc, ^
```

解决问题

要解决此错误，请找到 Caffe 安装并运行以下命令：

cp -rf ~/caffe-1.0/include/caffe/util/cudnn.hpp ~/py-faster-rcnn/caffe-fast-rcnn/include/caffe/util/
cp -rf ~/caffe-1.0/src/caffe/layers/cudnn_* ~/py-faster-rcnn/caffe-fast-rcnn/src/caffe/layers/
cp -rf ~/caffe-1.0/include/caffe/layers/cudnn_* ~/py-faster-rcnn/caffe-fast-rcnn/include/caffe/layers/

对于具有 CUDA 10 的 Power9 系统，通过使用以下命令编辑 `Makefile.config` 文件：
`vim ~/py-faster-rcnn/caffe-fast-rcnn/Makefile.config`
如下所示更改 CUDA_ARCH 值：

```
CUDA_ARCH := -gencode arch=compute_30,code=sm_30 \
  -gencode arch=compute_35,code=sm_35 \
  -gencode arch=compute_50,code=sm_50 \
  -gencode arch=compute_50,code=compute_50
```

配置图像服务器

在主 Edge 机器上配置图像服务器，使该主 Edge 机器可以存储工业摄像机捕获的图像。图像服务器由 Edge 控制器进行监视。添加新图像时，会对该图像评分，并将检查结果发送到中心应用程序，以便检验员可以评估图像和检查结果。

过程

1. 使用以下命令在主 Edge 系统上安装并启动 NFS 服务：
   `sudo apt-get install nfs-kernel-server`
   对于 Linux on Power Systems Servers，使用以下命令：
   `sudo yum install -y nfs-utils`
   `sudo yum install -y rpcbind`
   `systemctl start nfs`
   `systemctl start nfslock`
   `systemctl start rpcbind`

2. 使用以下命令创建名为 imageserver 的目录并更改文件夹所有者：
   ```
   sudo mkdir /imageserver
   sudo chown user /imageserver
   ```
   其中，*user* 是您向中心应用程序注册 Edge 时指定的用户。
3. 可选： 仅当同时具有主 Edge 系统和从属 Edge 系统时，完成以下步骤。
   a. 要导出 imageserver 目录，请使用以下命令编辑 /etc/exports 文件：
      ```
      vi /etc/exports
      ```
   b. 将以下行添加到导出文件。IP 地址网络是可以访问共享文件夹的 IP 子网。格式为：*IP address*/*subnet mask*。所有从属 Edge 系统必须能够访问共享文件夹：
      ```
      /imageserver IP address/network(rw,sync,no_root_squash,no_all_squash)
      ```
      例如：
      ```
      /imageserver 10.173.0.0/29(rw,sync,no_root_squash,no_all_squash)
      ```
   c. 使用以下命令重新启动 NFS 服务：
      ```
      sudo service nfs-server restart
      ```
4. 在每个从属 Edge 机器上，将 /imageserver 目录装载到主 Edge 的共享文件夹：
   a) 在从属 Edge 机器上运行以下命令：
      mkdir /imageserver
   b) 使用以下命令编辑 /etc/fstab 文件：
      vi /etc/fstab
   c) 将以下行添加到该文件，并指定主 Edge 的 IP 地址：
      ```
      IP_address:/imageserver /imageserver nfs defaults 0 0
      ```
   d) 在从属 Edge 机器上运行以下命令使装载生效：
      mount -a

配置模型店铺

必须在主 Edge 机器上配置模型店铺。模型店铺是从中心应用程序分发的可执行模型的存储库。

过程

1. 使用以下命令创建 modelstore 目录，并更改主 Edge 系统上的文件夹所有者：
   ```
   sudo mkdir /modelstore
   sudo chown user /modelstore
   ```
   其中，*user* 是您向中心应用程序注册 Edge 时指定的用户。
2. 可选： 仅当同时具有主 Edge 系统和从属 Edge 系统时，完成以下步骤。
   a. 要导出 modelstore 目录，请使用以下命令编辑 /etc/exports 文件：
      ```
      vi /etc/exports
      ```
   b. 将以下行添加到导出文件。IP 地址网络是可以访问共享文件夹的 IP 子网。格式为：*IP address*/*subnet mask*。所有从属 Edge 系统必须能够访问共享文件夹：
      ```
      /modelstore IP address/network(rw,sync,no_root_squash,no_all_squash)
      ```
      例如：
      ```
      /modelstore 10.173.0.0/29(rw,sync,no_root_squash,no_all_squash)
      ```
   c. 使用以下命令重新启动 NFS 服务：
      ```
      sudo service nfs-server restart
      ```
3. 在每个从属 Edge 机器上，将 /modelstore 目录装载到主 Edge 的共享文件夹：
   a) 在从属 Edge 机器上运行以下命令：
      mkdir /modelstore
   b) 使用以下命令编辑 /etc/fstab 文件：
      vi /etc/fstab
   c) 将以下行添加到该文件，并指定主 Edge 的 IP 地址：

IP_address:/modelstore /modelstore nfs defaults 0 0

d) 在从属 Edge 机器上运行以下命令使装载生效:
mount -a

安装 Python 模块

完成此任务以安装必备的 Python 模块。

过程

在主 Edge 系统和每个从属 Edge 系统上运行以下命令以安装必备软件包:
```
sudo pip install flask
sudo pip install gevent
sudo pip install requests
sudo pip install pyinotify
sudo pip install opencv-python
sudo pip install lmdb
```
对于 Ubuntu: `sudo apt-get install dos2unix`

对于 Linux on Power Systems Servers: `sudo yum install dos2unix`
```
sudo apt install curl
sudo pip install flask_httpauth
sudo pip install paramiko
sudo pip install tornado
```

安装用于 Linux on Power Systems Servers 的 Paramiko

如果在安装用于 Linux on Power Systems Servers 的 Paramiko 时遇到问题，请使用此任务手动安装 Paramiko。

过程

1. 将 `rhel-75-server.repo` 置于 `/etc/yum.repos.d`
2. 运行以下命令:
   ```
   sudo yum install redhat-rpm-config gcc libffi-devel python-devel openssl-devel
   ```
3. 运行以下命令:
   ```
   pip install cryptography --no-binary cryptography
   ```
4. 运行以下命令:
   ```
   sudo pip install bcrypt
   ```
5. 运行以下命令:
   ```
   sudo pip install pynacl
   ```
6. 运行以下命令:
   ```
   sudo pip install paramiko
   ```

向中心应用程序注册 Edge

您必须向中心应用程序注册 Edge。对于已连接的 Edge 和独立 Edge，此过程是不同的。

有两种类型的 Edge：已连接的 Edge 和独立 Edge。已连接的 Edge 具有一个公共 IP 地址，支持中心应用程序直接连接到 Edge。Edge 实时向中心应用程序发送检查结果。独立 Edge 没有公共 IP，并且有时无法连接到中心应用程序。检查结果必须本地存储在独立 Edge 上，并且在 Edge 连接到因特网时，Edge 管理员必须显式调用 Edge 服务以与中心应用程序进行通信。

向中心应用程序注册已连接的 Edge

配置已连接的 Edge 系统后，必须在中心应用程序中注册该系统。可以创建 Edge，也可以编辑现有 Edge。Edge 用于运行评分模型。

过程

1. 在“**模型管理器**”中，选择**数据** > **Edge**。
2. 选择**新建 Edge**，然后输入 Edge 名称。
3. 输入 Edge 系统的 IP 地址、SSH 用户名和密码，然后将 Edge 类型指定为**主控**或**从属**。IP 地址必须可供中心应用程序访问。SSH 用户名和密码用于登录 Edge 系统以部署 Edge 控制器。
4. 如果要创建主 Edge，请将连接方式指定为连接。如果要创建从属 Edge，请指定对应的主 Edge。
5. 如果选择了连接方式，请选择**下一步**，然后指定删除策略。
6. 选择**创建**以创建 Edge。Edge 控制器和评分引擎会部署到 Edge 系统，并且该 Edge 会添加到已注册的列表中。

向中心应用程序注册独立 Edge

配置独立 Edge 系统后，必须在中心应用程序中注册该系统。可以创建 Edge，也可以编辑现有 Edge。Edge 用于运行评分模型。

关于此任务

通过独立 Edge，您可以在与中心应用程序断开连接时执行产品的某些功能，例如对生产线中的图像进行评分，将数据同步到中心以及清理上载的数据。要安装独立 Edge，必须安装 PostgreSQL 和必需的 Python 模块。您还必须在 Edge 系统上手动安装 Edge。

过程

1. 在“**模型管理器**”中，选择**数据** > **Edge**。
2. 选择**新建 Edge**，然后输入 Edge 名称。
3. 输入 Edge 系统的 IP 地址、SSH 用户名和密码，然后将 Edge 类型指定为**主控**或**从属**。IP 地址必须可供中心应用程序访问。SSH 用户名和密码用于登录 Edge 系统以部署 Edge 控制器。
4. 如果要创建主 Edge，请将连接方式指定为独立。
5. 如果选择了连接方式，请选择**下一步**，然后指定删除策略。
6. 选择**创建**以创建 Edge。对于独立 Edge，Edge 管理员必须向 Edge 系统执行 SSH 登录并手动安装该 Edge。有关更多信息，请参阅“安装独立 Edge”主题。

安装独立 Edge

通过独立 Edge，您可以在与中心断开连接时执行产品的某些功能，例如对生产线中的图像进行评分，将数据同步到中心以及清理上载的数据。要安装独立 Edge，必须安装 PostgreSQL 和必需的 python 模块。您还必须手动将 Edge 安装到 Edge 机器上。

要创建独立 Edge，必须在创建 Edge 时选择独立连接方式。

必须在主 Edge 所在的系统上创建独立 Edge。

安装用于 Ubuntu 的 PostgreSQL

使用此任务以安装用于 Ubuntu 系统的 PostgreSQL。培训服务器和独立 Edge 使用 PostgreSQL 数据库。

过程

1. 使用以下命令安装 PostgreSQL、PHPPgadmin 和 Apache2：
   ```
   sudo apt-get -y install postgresql postgresql-contrib phppgadmin
   ```
2. 配置 PostgreSQL 用户。
   a) 使用以下命令以 PostgreSQL 用户身份登录：
   ```
   sudo su
   su - postgres
   ```

```
psql
```

b) 使用以下命令配置用户 postgres 的密码：
```
password postgres
password
\q
```

3. 通过编辑 nano phppgadmin.conf 文件来配置 Apache2：
```
cd /etc/apache2/conf-available/
nano phppgadmin.conf
```
删除以下行：Require local。将以下行添加到该文件中：
```
Require all granted
```

4. 通过编辑 config.inc.php phppgadmin.conf 文件来配置 PHPPgadmin：
```
cd /etc/phppgadmin/
nano config.inc.php
```
在文件中查找以下行：
```
$conf['extra_login_security'] = true
```
将 true 更改为 false。

5. 使用以下命令重新启动 PostgreSQL 和 Apache2：
```
systemctl restart postgresql
systemctl restart apache2
```

6. 通过访问以下 URL，验证是否可以在独立 Edge 上访问用户界面：
http://*standalone_edge_IP*/phppgadmin
其中，*standalone_edge_IP* 是独立 Edge 的 IP 地址。

7. 在 PostgreSQL 中创建数据库模式。

a) 在 PHPPgadmin 上的 SQL 控制台中运行以下命令：
```
create database edge with owner postgres encoding='UTF-8'
lc_collate='en_US.utf8' lc_ctype='en_US.utf8' template template0;
```

b) 在数据库中，创建以下各表：
```
CREATE TABLE vi_tenant_inspectionresult(id text, info jsonb);
CREATE TABLE vi_tenant_notification(id text, info jsonb);
CREATE TABLE vi_tenant_defectsummary(id text, info jsonb);
CREATE TABLE vi_tenant_uploaddataset(id text, info jsonb);
CREATE TABLE vi_tenant_syncprocess(id text, info jsonb);
CREATE TABLE vi_tenant_model(id text, info jsonb);
CREATE TABLE vi_tenant_datagroup(id text, info jsonb);
```
其中，*tenant* 是 Maximo PQI SaaS Visual Insights 中心内操作用户的租户。在中心应用程序的用户界面中，从用户概要文件中获取租户值。

安装用于 Linux on Power Systems Servers 的 PostgreSQL

使用此任务以安装用于 Linux on Power Systems Servers 的 PostgreSQL。培训服务器和独立 Edge 使用 PostgreSQL 数据库。

过程

1. 使用以下命令对 root 用户执行 su 命令：
```
sudo su
```

2. 下载 PostgreSQL 的源代码：
```
wget https://ftp.postgresql.org/pub/source/v9.5.13/postgresql-9.5.13.tar.gz
```

3. 使用以下命令安装 PostgreSQL：
```
tar -zxvf postgresql-9.5.13.tar.gz
cd postgresql-9.5.13/
yum -y install readline-devel
./configure --prefix=/usr/local/postgresql
make
make install
```

4. 创建用户 postgres，并更改 postgres 目录的所有者：

```
useradd postgres
chown -R postgres:postgres /usr/local/postgresql/
```

5. 切换至用户 postgres:
```
su postgres
```
6. 为 postgres 配置系统路径:
```
vi ~/.bashrc
PGHOME=/usr/local/postgresql
export PGHOME
PGDATA=/usr/local/postgresql/data
export PGDATA
PATH=$PATH:$HOME/.local/bin:$HOME/bin:$PGHOME/bin
export PATH
```
7. 获取配置的源:
```
source ~/.bashrc
```
8. 初始化 PostgreSQL 数据库:
```
initdb
```
9. 配置数据库。在 vi 中打开 postgresql.conf:
```
vi /usr/local/postgresql/data/postgresql.conf
```
将以下内容:
```
#listen_address='localhost'
#port = 5432
```
更改为:
```
listen_address='*'
port = 5432
```
在 vi 中打开 pg_hba.conf:
```
vi /usr/local/postgresql/data/pg_hba.conf
```
将以下行添加到该文件中:
```
host all all 0.0.0.0/0 trust
```
10. 重新启动 PostgreSQL:
```
pg_ctl -D /usr/local/postgresql/data -l logfile restart
```
11. 在 PostgreSQL 数据库中更改用户 postgres 的密码:
```
psql
ALTER USER postgres WITH PASSWORD 'password';
\q
```
如果 PostgreSQL 服务未启动，请运行以下命令:
```
su postgres
vi ~/.bashrc
```
将 /usr/local/pgsql/bin/ 添加到文件:
```
export PATH=/usr/local/cuda-8.0/bin:$PATH:/usr/local/pgsql/bin/
```
运行以下命令:
```
source ~/.bashrc
```
12. 在 PostgreSQL 中创建数据库模式。在 psql 控制台上运行以下命令:
```
create database edge with owner postgres encoding='UTF-8'
lc_collate='en_US.utf8' lc_ctype='en_US.utf8' template template0;
```
在数据库中，创建以下各表:
```
CREATE TABLE vi_tenant_inspectionresult(id text, info jsonb);
CREATE TABLE vi_tenant_notification(id text, info jsonb);
CREATE TABLE vi_tenant_defectsummary(id text, info jsonb);
CREATE TABLE vi_tenant_uploaddataset(id text, info jsonb);
CREATE TABLE vi_tenant_syncprocess(id text, info jsonb);
CREATE TABLE vi_tenant_model(id text, info jsonb);
CREATE TABLE vi_tenant_datagroup(id text, info jsonb);
```

其中，*tenant* 是 Maximo PQI SaaS Visual Insights 中心内操作用户的租户。在中心应用程序的用户界面中，可以从用户概要文件中获取租户值。

安装 Python 模块
要创建培训服务器或独立 Edge，必须安装必需的 Python 模块。

过程

使用以下命令安装 Python 模块：
```
sudo apt-get update
sudo apt -y install postgresql
sudo apt -y install libpq-dev
sudo pip install PyGreSQL
sudo pip install DBUtils
```

安装独立 Edge
对于独立 Edge，必须手动将 Edge 安装到 Edge 机器上。

下载构建文件和安装脚本
必须下载独立 Edge 构建文件和安装脚本来安装独立 Edge。

过程

1. 在 Maximo PQI SaaS Visual Insights 应用程序中，选择**数据** > **Edge**。选择主 Edge，并查看 Edge 详细信息。
2. 选择并下载用于必需操作系统的构建文件。支持以下操作系统：Ubuntu、Red Hat Enterprise Linux 和 IBM Power System。
3. 选择并下载安装 shell 脚本文件。

在独立 *Edge* 上安装构建文件
要在 Edge 机器上安装 Edge 系统，必须使用 SSH 用户名并向 Edge 机器执行 SSH 登录。

过程

1. 执行 SSH 登录后，将工作目录切换到 Edge 创建对话框的部署路径。
2. 使用以下命令将执行许可权添加到 shell 脚本文件：
   `chmod +x ./edgeDeployed.sh`
3. 运行该 shell 脚本文件以安装独立 Edge。请使用以下命令：
   `./edgeDeployed.sh`
4. 使用 PostgreSQL 信息更新 *deploy_path*`/vi_edge-bin_vi/vi_edge/postgres/DBConfig.json` 文件以纠正数据库连接。然后，运行以下命令以重新启动 Edge：
   `cd` *deploy-path*`/vi_edge-bin_vi/vi_edge/`
   `./restartController.sh`

检查 Edge 系统上服务的状态

在向中心应用程序注册 Edge 后，验证正确的 Python 进程是否正在 Edge 系统上运行。

过程

1. 在 Edge 系统上安装 Edge 后，向 Edge 系统执行 SSH 登录。
2. 在主 Edge 系统上，运行以下命令以查找主引擎的 Python 进程：
   `ps aux | grep python`
   结果应该包含以下 Python 进程：
   `python` *deployment_folder*`/vi_edge-bin_vi/vi_task_manager/run.py`。
   如果在主 Edge 系统上找不到 Python 进程，请查看日志文件：*deployment_folder*`/vi_edge-bin_vi/vi_task_manager/master.log`。
   尝试运行以下命令来启动主引擎：

deployment_folder/vi_edge-bin_vi/vi_task_manager/restartMaster.sh *IP_of_master_node*

3. 在主 Edge 系统上，运行以下命令以查找 Edge 控制器的 Python 进程：
   ps aux | grep python
   结果应包含以下 Python 进程：
   python *deployment_folder*/vi_edge-bin_vi/vi_edge/runMonitor.py
   python *deployment_folder*/vi_edge-bin_vi/vi_edge/runService.py.
   如果在主 Edge 系统上找不到 runMonitor Python 进程，请查看日志文件：*deployment_folder*/vi_edge-bin_vi/vi_edge/Event.log。
   如果在主 Edge 系统上找不到 runService Python 进程，请查看日志文件：*deployment_folder*/vi_edge-bin_vi/vi_edge/Service.log。
   尝试运行以下命令来启动 Edge 控制器：
   deployment_folder/vi_edge-bin_vi/vi_edge/restartController.sh
4. 在主 Edge 系统和从属 Edge 系统上运行以下命令，以查找分类模型的评分引擎的 Python 进程：
   ps aux | grep python
   结果应包含以下 Python 进程：
   python *deployment_folder*/vi_edge-bin_vi/vi_score_engine_restful/front_run.py 5005 6005
   python *deployment_folder*/vi_edge-bin_vi/vi_score_engine_restful/back_run.py 6005
   如果在 Edge 系统上找不到 Python 进程，请查看日志文件：
   deployment_folder/vi_edge-bin_vi/vi_score_engine_restful/front_log.txt
   deployment_folder/vi_edge-bin_vi/vi_score_engine_restful/back_log.txt
   解决所有问题后，主引擎应该会自动启动评分引擎。要手动启动评分引擎，请运行以下命令：
   source ~/.bashrc
   deployment_folder/vi_edge-bin_vi/vi_score_engine_restful/restartEngine.sh
5. 在主 Edge 系统和从属 Edge 系统上运行以下命令，以查找对象检测模型的评分引擎的 Python 进程：
   ps aux | grep python
   结果应包含以下 Python 进程：
   python deployment_folder/vi_edge-bin_vi/vi_obj_detection/RESTAPI/model/run.py port gpuid FRCNN
   python deployment_folder/vi_edge-bin_vi/vi_obj_detection/RESTAPI/model/run.py port gpuid SSD
   如果在 Edge 系统上找不到 Python 进程，请查看以下日志文件：
   deployment_folder/vi_edge-bin_vi/vi_obj_detection/RESTAPI/model/FRCNN_*gpuid*.log
   deployment_folder/vi_edge-bin_vi/vi_obj_detection/RESTAPI/model/SSD_*gpuid*.log
   如果看到“没有名为 iotmyolo 的模块”错误消息，请将 iotmyolo.so 或 iotmyolo.py 文件从 /home/*user*/vi_edge-bin_vi/vi_obj_detection/model_library/yolo 目录复制到 YOLO_HOME 目录。
   解决所有问题后，主引擎应该会自动启动评分引擎。
6. 使用 Edge 组件重新构建 YOLO 库：
   a. 转至 /home/*user*/vi_edge-bin_vi/vi_obj_detection/model_library/yolo 目录。
   b. 将 detectorobj.c 文件添加到 darknet/examples 文件夹。
   c. 编辑 darknet/Makefile 文件，并指明 EXECOBJA=detectorobj.o。以下代码是 Makefile 文件中的代码示例：

```
EXECOBJA=detectorobj.o captcha.o lsd.o super.o voxel.o art.o tag.o cifar.o go.o rnn.o
rnn_vid.o compare.o segmenter.o regressor.o classifier.o coco.o dice.o yolo.o
detector.o writing.o nightmare.o swag.o darknet.o
```

   d. 在 Makefile 文件中，为 $(SLIB) 和 $(ALIB) 对象添加 $(EXECOBJ)。以下代码是 Makefile 文件中的代码示例：

```
$(ALIB): $(EXECOBJ) $(OBJS)
     $(AR) $(ARFLAGS) $@ $^
$(SLIB): $(EXECOBJ) $(OBJS)
     $(CC) $(CFLAGS) -shared $^ -o $@ $(LDFLAGS)
```

e. 运行以下命令：
make

升级 Edge 系统

升级 Edge（如果是较旧版本）。您可以下载 Edge 构建文件和安装脚本以升级 Edge。

关于此任务

如果升级 Maximo PQI SaaS Visual Insights 中心应用程序，但不升级 Edge，这会导致在中心和 Edge 之间产生不一致，并且模型可能无法部署。您可能会看到类似于以下内容的错误消息：“在部署新模型之前需要升级 Edge。”

过程

1. 运行以下命令以获取 Edge 的标识：
   /ibm/iotm/vi/service/edge?user=*xxx*&solution=vi
2. 对于安装 shell 脚本，调用以下服务：
   /ibm/iotm/vi/service/edgeFile?edgeId=*xxx*&version=shell&user=*xxx*&solution=vi
   将 edgeId=*xxx* 替换为从先前步骤获取的 Edge 标识。将文件另存为 edgeDeployed.sh。下面是样本命令：
   curl -k -H "apikey:*yourapikey*" "https://iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/vi/service/edgeFile?edgeId=1508123458000&version=shell&user=*youruser*&solution=vi" > edgeDeployed.sh
3. 请查看 edgeDeployed.sh 文件。查看 username（这是用于启动 Edge 服务的 SSH 用户）、password（这是 SSH 用户的密码）和 basefolder（这是 Edge 服务的部署路径）的值。输入 <to edit> 的值。
4. 对于 Edge 构建文件，调用以下服务：
   /ibm/iotm/vi/service/edgeFile?edgeId=*xxx*&version=*xxx*&user=*xxx*&solution=vi
   其中，version 是 ubuntu、redhat 或 power（取决于系统）。将文件另存为 vi_edge-bin_vi.zip。下面是样本命令：
   curl -k -H "apikey:*yourapikey*" -o vi_edge-bin_vi.zip "https://iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/vi/service/edgeFile?edgeId=1508123458000&version=ubuntu&user=*youruser*&solution=vi"
5. 将 edgeDeployed.sh 和 vi_edge-bin_vi.zip 文件放在 Edge 系统上。
6. 以在中心应用程序中创建 Edge 时使用的 SSH 用户身份登录到 Edge。
7. 将 edgeDeployed.sh 和 vi_edge-bin_vi.zip 文件放入 deployPath 目录中。
8. 关闭当前 Edge 服务。
9. 运行以下命令：
   dos2unix edgeDeployed.sh
   chmod +x edgeDeployed.sh
   ./edgeDeployed.sh
10. 查看 Edge 的 Python 服务是否已启动。

相关任务

检查 Edge 系统上服务的状态
在向中心应用程序注册 Edge 后，验证正确的 Python 进程是否正在 Edge 系统上运行。

第 3 章 创建和使用模型

可以创建模型来收集历史图像和缺陷信息。这些信息用于培训模型。培训模型后，必须验证该模型，然后再将其部署到 Edge。验证模型将提供模型精确性信息。一个模型可以有多个版本。模型可以共享缺陷信息，但具有来自不同产品系列的不同图像文件。可以重新培训模型以尝试获得更高的模型精确性，使模型版本可以替换为更新的模型版本。有两种类型的模型实现：分类模型和对象检测模型。

压缩图像文件的结构

在为图像组添加历史图像之前，您的文件必须包含分类模型或对象检测模型所需的图像文件。

分类模型

将图像添加到压缩文件中。一个压缩文件必须包含属于同一图像组的所有图像。必须在压缩文件中不带任何子文件夹的平面结构中放入所有图像。支持以下图像类型：PNG、JPEG 和 JPG。图像文件必须都具有大写的文件扩展名（例如 PNG、JPEG、JPG），或者都必须具有小写的文件扩展名（例如 png、jpeg、jpg）。

对象检测模型

对象检测模型的结构必须在一个压缩文件中包含两个文件夹。一个文件夹必须名为 `JPEGImages`，另一个文件夹必须名为 `Annotations`。除了这两个文件夹外，该压缩文件还必须包含一个 `labels.txt` 文件。

将所有图像文件添加到 `JPEGImages` 文件夹。支持以下图像类型：PNG、JPEG 和 JPG。图像文件必须都具有大写的文件扩展名（例如 PNG、JPEG、JPG），或者都必须具有小写的文件扩展名（例如 png、jpeg、jpg）。将所有注释文件添加到 `Annotations` 文件夹。注释文件必须与其图像文件同名。这两个文件必须为 XML 格式。如果在 Maximo PQI SaaS Visual Insights 中使用图像标注工具，那么输出是包含图像文件和注释文件的压缩图像文件。如果使用外部图像标注工具，那么必须确保图像文件和注释文件采用的是所需结构。以下信息是注释文件的示例：

```
<annotation>
  <folder>JPEGImages</folder>
  <filename>000001.jpg</filename>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>864</width>
    <height>1296</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>defect1</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>474</xmin>
      <ymin>368</ymin>
      <xmax>540</xmax>
      <ymax>448</ymax>
    </bndbox>
  </object>
  <object>
    <name>defect2</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>303</xmin>
      <ymin>387</ymin>
      <xmax>369</xmax>
      <ymax>452</ymax>
    </bndbox>
```

```
  </object>
</annotation>
```

`labels.txt` 文件包含注释文件夹中所有缺陷类型的名称。每种缺陷必须独占一行，如以下示例中所示：

```
defect1
defect2
defect3
```

为图像组添加历史图像

模型管理者使用历史图像来培训模型。

关于此任务

您可以将压缩 (.zip) 图像文件上载到图像组。准备具有正确文件大小的压缩图像文件。如果文件太大，可以将该文件拆分为多个文件。根据应用程序的网络上载速度，确保每个压缩图像文件可以在 30 分钟内上传完毕。例如，如果网络上载速度为 100 KB/秒，那么压缩图像文件的最大大小为 200 MB。

过程

1. 选择**数据** > **图像组** > **新建图像组**。如果要使用 Maximo PQI SaaS Visual Insights 中的图像标记工具来标记图像，请选择**新建未标记图像组**。如果图像已经由外部工具标记，请选择**新建已标记图像组**。
2. 添加唯一的图像组名称和描述，选择图像组类型，然后选择**下一步**。

   对于图像组类型：

   - “单特征”表示组中的图像属于一种缺陷类型。单特征使用分类模型。
   - “不是缺陷”表示组中图像不包含缺陷的单特征。“不是缺陷”使用分类模型。
   - “多特征”表示每个图像都包含可能属于相同或不同缺陷类型的一个或多个缺陷。“多特征”使用对象检测模型。

   注: 上载图像集后，即无法更改选择的图像组类型。
3. 在“**图像集**”窗格中，添加图像，然后选择**添加图像组**。

结果

在“**图像组**”窗格中，可以选择图像组，然后选择“编辑”。您可以添加或删除图像集，更新图像组名称或描述，或者更改图像组类型。

注: 如果在模型实例中引用了图像集，那么无法删除该图像集。

图像标记工具

图像标记工具可用于半自动标记图像。您可以上载未标记的图像文件，手动标记某些图像，然后触发自动标记以标记未标记的图像。可以检查自动标记的结果。根据摘要信息，可以接受标记结果。图像标记后，会打包成相应的图像 zip 文件，并自动附加到图像组，以便可以在模型培训过程中使用这些文件。

创建未标记图像组

您可以上载包含未标记图像的压缩 (.zip) 图像文件，以创建新的未标记图像组。

关于此任务

创建未标记图像组时，可以使用不包含已标记图像的压缩文件，也可以使用包含已标记和未标记图像的压缩文件。对于单特征文件，任何已标记图像都需要包含在文件夹名称为缺陷名称的子文件夹中。对于多特征文件，已标记图像文件必须在相同的文件夹中具有相应的注释 XML 文件。

准备具有正确文件大小的压缩图像文件。如果文件太大，可以将该文件拆分为多个文件。根据应用程序的网络上载速度，确保每个压缩图像文件可以在 30 分钟内上传完毕。例如，如果网络上载速度为 100 KB/秒，那么压缩图像文件的最大大小为 200 MB。

过程

1. 选择**数据 > 新建图像组 > 新建未标记图像组**。
2. 在“**新建未标记图像组**”对话框中，指定图像组的名称。
3. 指定图像组类型：
   - 单特征意味着组中的图像只有一种缺陷类型。
   - 多特征意味着组中的图像包含一种或多种缺陷类型。
4. 单击上载链接以上载包含未标记图像的压缩文件。
5. 单击**添加图像组**以创建新的未标记图像组。
6. 在“**采样规则**”对话框中，设置要手动标记的图像的比率，然后单击**确认**。

结果

完成此任务后，即可以开始手动标记样本图像。

手动标记样本图像

您可以对未标记图像组中包含的图像进行手动标记。

过程

1. 在“**所有未标记图像组**”窗口中，单击包含要标记的图像的组旁边的**查看**。
2. 单击“采样规则”图标以设置采样率；采样率指示要手动标记的图像数。单击**确认**。
3. 对于每个样本图像，通过选择相应的标签来手动标记图像，然后保存图像。可以选择现有标签，也可以创建新标签。对于单特征组，一个标签对应于一个数据组。对于多特征组，将创建一个具有相同组名的数据组。未标记组的图像中使用的所有标签都会在相应的数据组中保存为标记。
4. 标记所有样本图像后，可以单击**运行自动标记**以启动自动标记过程。

自动标记样本图像

Maximo PQI SaaS Visual Insights 使用基于卷积神经网络的深度学习方法来自动标记图像。

过程

1. 在“**所有未标记图像组**”窗口中，单击包含要标记的图像的组旁边的**查看**。
2. 单击**运行自动标记**以启动自动标记过程。可以单击**在后台运行**以在后台运行自动标记过程。
3. 自动标记过程完成后，单击**查看**以查看结果。
4. 对于每个自动标记的图像，确认或修改标签。
5. 根据需要，重新运行自动标记过程。
6. 单击**查看摘要**以查看标签和模型精确性信息。
7. 对模型精确性感到满意后，单击**接受**。
   所有已标记图像都会另存为已标记压缩文件。您可以在**数据**窗格中找到该文件。

创建模型

要获得良好的模型精确性，您需要基于图像集和所选模型类型仔细设置超参数。

创建模型时，可以选择建议的模型设置，也可以选择定制这些设置。建议的模型设置并不适用于所有场景。这些设置仅供参考。您可能需要定制模型设置。

培训模型时，您需要选择所需的图像集。一些图像用作培训数据集，另一些用作验证数据集。具体比例取决于培训和验证采样规则。缺省情况下，80% 的图像用于培训，20% 的图像用于验证。

了解以下神经网络术语会很有帮助：

Epoch
一个 epoch 等于所有培训图像上的一个完整培训周期。

批量大小
一个批次中的培训示例数。批量大小越大，需要的内存空间越多。

迭代次数
迭代次数等于批次数，每个批次使用 *batch_size* 个图像。

例如，如果有 1000 个培训图像，批量大小为 50，那么需要 20 次迭代才能完成 1 个 epoch。如果将 epoch 设置为 10，那么需要 200 次迭代才能完成培训。

如果选择分类模型类型，那么网络定义中已预定义批量大小。对于 GoogLeNet，培训批量大小为 32，测试批量大小为 16。对于 AlexNet，培训批量大小为 128，测试批量大小为 32。对于 LeNet，培训批量大小为 64，测试批量大小为 32。可以根据批量大小、图像总数和采样规则来调整超参数。

创建模型

添加了缺陷类型后，模型管理者将创建模型。模型可以立即另存为草稿或培训的模型。

关于此任务

可以在**所有模型**选项卡上查看模型的详细信息。模型详细信息包含模型的版本。不同的模型版本是通过不同的图像集构建的。

模型名称必须唯一。

过程

1. 选择**新建模型 > 新建模型**。
2. 在**常规**选项卡上，更新信息。

   产品类型用于映射模型。您可以根据自己的公司标准选择 Maximo PQI SaaS Visual Insights 外部的产品类型。产品类型是模型和生产线之间的链接。Edge 控制器将基于图像上的产品类型信息来选择模型。
3. 在“**模型类型**”选项卡上，更新模型类型信息。对于**模型类型**，可以指定分类或多对象检测，也可以选择上载现有已培训模型。如果选择分类或多对象检测，那么可以选择建议的模型设置，也可以选择定制这些设置。

   如果选择定制模型设置，请指定检测方法、超参数和采样规则。每个超参数都有有效的值范围。如果输入的值超过该范围，将会显示错误消息。由于资源限制，epoch、迭代次数和批量大小都有最大值。迭代次数的最大值与租户的支付类型相关。对于付费用户，FRCNN 的迭代次数最大值为 40,000，YOLO 的为 20,000，SSD 的为 5,000。对于试用用户，FRCNN 的迭代次数最大值为 3,000，YOLO 的为 2,000，SSD 的为 500。YOLO 和 SSD 的批量大小为 32。对于 CNN，付费用户的最大 epoch 值为 30,000，试用用户的最大 epoch 值为 5,000。

   如果上载现有的已培训模型压缩文件，那么会从该文件中读取模型设置值，并将这些值显示在对话框中。如果 epoch 或迭代次数值和批量大小超过最大值，这些值会替换为最大值。
4. 在“**图像集**”选项卡上，选择要用于培训模型的图像集。
5. 在**全局策略**选项卡上，设置重新培训策略、采样策略和手动检查策略设置。
6. 在“**摘要**”选项卡上，选择**另存为草稿**以保存模型，或者选择**培训**以培训模型。
7. 在“**所有模型**”窗格中，查看该模型。如果模型版本设置为“草稿”，那么可以单击**查看**以查看模型实例详细信息。培训模型之前，可以编辑图像组。

使用模型目录

模型目录是共享模型的库，这些模型可以复用于行业解决方案并由租户共享。

模型目录包含以下模型。

领域	模型名称	描述
电子 - 硬件制造	服务器硬件缺陷检查	此对象检测模型用于检测 IBM 服务器的缺失部件。它支持七种主要缺陷，包括 missingChinLabel、missingPowerLabel、missingIBMLogo、missingDASD、 incorrectDASD、missingCover 和 missingPowerButton。
汽车 - 汽车磨损	汽车磨损缺陷检查	此对象检测模型用于检测汽车磨损情况。汽车制造商会对开合活动执行数以万计的测试，并检测何时磨损达到不可接受的水平。此模型专注于检测油漆划痕。
汽车 - 汽车座椅	汽车座椅缺陷检查	此对象检测模型用于检测汽车座椅中的皱褶缺陷。检查结果可识别汽车座椅上是否存在褶皱缺陷，并定位图像上的缺陷。

测试模型目录中的模型

您可以通过上载自己的图像并检查评分结果，测试模型目录中的模型。可以上载多个图像并逐个检查评分结果。

过程

1. 选择**新建模型 > 浏览模型目录**。
2. 在“**模型目录**”页面上，选择要使用的共享模型。
3. 单击**上载更多**以上载图像。模型会对图像评分，然后显示结果。

使用模型目录创建模型

您可以使用模型目录中的共享模型来创建新模型。要使用共享模型，请导入模型定义并使用自己的数据集来培训模型。

过程

1. 选择**新建模型 > 浏览模型目录**。
2. 在“**共享模型**”页面上，选择要使用的共享模型。
3. 单击**使用模型**。
4. 在**常规**选项卡上，更新信息。产品类型用于映射模型。
5. 在**模型类型**选项卡上，更新模型类型和模型设置信息。
6. 在**图像集**选项卡上，选择要用于培训模型的图像集。
7. 在**全局策略**选项卡上，根据需要更新重新培训策略和手动检查策略。
8. 在**摘要**选项卡上，单击**另存为草稿**。模型状态会设置为“草稿”。

培训模型

Maximo PQI SaaS Visual Insights 使用基于卷积神经网络的深度学习方法来培训模型。培训模型时，模型会学习已标记图像的特征，并构建嵌入这些知识的可执行模型。培训的模型能够对来自产品系列的图像执行分类或对象检测。

关于此任务

可以培训新创建的模型，或培训使用模型目录中的现有模型所创建的模型。可以培训状态为“草稿”、“失败”或“已拒绝”的模型。在培训期间，可以检查培训状态和日志文件。培训完成后，可以检查培训曲线，包括培训损失、测试损失和测试精确性值。

过程

1. 在“**所有模型**”窗格中，选择模型。
2. 单击**查看**以查看模型实例详细信息。
3. 可选：更改模型使用的图像组。
4. 单击**培训**以开始培训。
   在准备数据并且培训作业在服务器上排队时，培训状态会更改为“正在等待”。培训开始后，状态会更改为“正在培训”。培训完成后，状态会更改为“已培训”。
5. 培训完成后，在“**培训仪表板**”中查看培训结果。如果模型支持快照，那么快照也会显示在该仪表板中。可以检查每个快照的信息。您可以通过选择快照，然后单击**使用**来使用快照。可以选择一个中间快照来用作最终的模型文件。还可以查看和下载培训日志文件。

培训的模型

创建或编辑模型后，模型管理者可以培训模型。培训模型后，模型状态会更改为“已培训”。模型管理者可验证培训的模型，并接受或拒绝该模型。模型管理者可使用验证图像集来验证模型。

如果模型管理者在创建新模型时选择上载培训的模型，那么执行上载过程后，模型状态会更改为“已接受”。

模型文件的结构

Maximo PQI SaaS Visual Insights 支持卷积神经网络 (CNN) 分类模型类型和对象检测模型类型。

用于 Caffe 的 CNN 分类模型

CNN 分类模型必须是单个压缩文件，并包含正确的目录结构和文件。

压缩模型文件

压缩模型文件必须包含以下目录和文件。

- `model.config`（文件，必需）
- `sink.config`（文件，必需）
- `parameter.config`（文件，可选）
- `cnet1`（目录）。`cnet1` 目录必须包含以下文件：
  - `labels.txt`（文件，必需）
  - `deploy.prototxt`（文件，必需）
  - `mean.binaryproto`（文件，必需）
  - `info.json`（文件，可选）
  - `snapshot.caffemodel`（文件，必需）
  - `solver.prototxt`（文件，必需）

– train_val.prototxt（文件，必需）

每个文件必须具有正确的结构和关键字。以下各部分中描述了这些文件。

model.config

以下文本是 model.config 文件内容的示例。关键字显示为粗体文本。

```
submodel{
  module {
       type:"ChipROIExtractor"
       ref_file:"parameter.config"
  }
  module {
       type:"ClassificationNet"
       net_name:"cnet1"
  }
}
sink{
    type:"SinkFilter"
    ref_file:"sink.config"
}
```

此文件必须至少有一个模块在子模型中具有 **ClassificationNet** 类型。ref_file 关键字指向 CNN 分类模型压缩文件内的其他配置文件。net_name 关键字是指包含 CNN 模型的文件夹名称。无需更改 sink 信息，除非您将其他名称用于 sink.config 文件。

sink.config

sink.config 文件的内容如下所示。无需编辑其内容。

```
keyword:"position"
keyword:"probableTypes"
```

labels.txt

labels.txt 文件包含用于对此分类模型进行分类的所有类名。每个类名必须独占一行，如以下示例中所示。

```
defect1
defect2
defect3
```

其他文件

所有 *filename*.prototxt 文件均为培训 CNN 模型时必需的模型定义文件。

snapshot.caffemodel 和 mean.binaryproto 文件为模型培训完成后创建的输出文件。

用于 TensorFlow 的 CNN 分类模型

使用 TensorFlow 进行培训的 CNN 分类模型必须是单个压缩文件，并包含正确的目录结构和文件。

压缩模型文件

压缩模型文件必须包含以下目录和文件。

- model.config（文件，必需）
- source.config（文件，必需）
- cnet1.config（文件，必需）
- cnet1（目录）。cnet1 目录必须包含以下文件：
  - labels.txt（文件，必需）
  - checkpoint（文件，必需）

- model.ckpt-10000.data-00000-of-00001（文件，必需）
- model.ckpt-10000.index（文件，可选）
- model.ckpt-10000.meta（文件，必需）

每个文件必须具有正确的结构和关键字。以下各部分中描述了这些文件。

model.config

以下文本是 model.config 文件内容的示例。关键字显示为粗体文本。

```
ipc_buffer_size:80000000
use_ipc_buffer:1
gpu_runnable:1

input_field_mapper {
  entry {
        app_field: "imageFile"
        model_field: "image"
        }
}
source {
       type:"utility.ImageFileSource"
       name:"source"
}
sink{
     type:"utility.Sink"
     name:"sink"
     port_num_config{
        input_num:1
     }
}
module {
       type:"tf_classification.TF_ClassificationNet"
       name:"cnet1"
}
connection{
    from_module:"source"
    from_port:"0"
    to_module:"cnet1"
    to_port:"0"
}
connection{
    from_module:"cnet1"
    from_port:"0"
    to_module:"sink"
    to_port:"0"
}
process{
    module_group {
            module:"cnet1"    }
    gpu_runnable:1
}
```

此文件必须至少有一个模块在 **input_field_mapper** 中具有 **tf_classification.TF_ClassificationNet** 类型。模块名称（例如 cnet1）是指包含 CNN 模型的文件夹名称。可以根据需要添加模块。无需更改 source 信息，除非 source.config 文件具有其他名称。如果更改名称 source，那么需要相应地更改连接中的模块名称。

source.config

source.config 文件的内容如下所示。无需编辑其内容。

```
param {
  name: "image"
  value: "color"
}
param {
  name: "gray_image"
  value: "gray"
}
```

cnet1.config

cnet1.config 文件的内容如下所示。不能更改此配置文件的 **param** 中的 name。value 是培训模型时设置的变量名称和图像大小。

```
param {
  name: "input_name"
  value: 'input'
}
param {
  name: "output_name"
  value: 'InceptionV3/Predictions/Softmax'
}
param {
  name: "resize_width"
  value: '299'
}
param {
  name: "resize_height"
  value: '299'
}
```

labels.txt

labels.txt 文件包含用于对此分类模型进行分类的所有类名。每个类名必须独占一行，如以下示例中所示。

```
defect1
defect2
defect3
```

checkpoint

checkpoint 文件的内容如下所示。培训模型时，会自动生成此文件。

```
model_checkpoint_path: "model.ckpt-10000"
all_model_checkpoint_paths: "model.ckpt-10000"
```

对象检测模型

对象检测模型必须是单个压缩文件，并包含正确的目录结构和文件。支持以下对象检测算法：基于区域的更快卷积神经网络 (Faster R-CNN)、You Only Look Once (YOLO) V2 和单图多框检测器 (SSD)。

Faster R-CNN

Faster R-CNN 对象检测模型必须是单个压缩文件，并包含正确的目录结构和文件。

压缩模型文件

压缩模型文件必须包含以下所有文件：

- labels.txt
- faster_rcnn_final.caffemodel
- model.config
- stage1_fast_rcnn_solver30k40k.pt
- stage1_fast_rcnn_train.pt
- stage1_rpn_solver60k80k.pt
- stage1_rpn_train.pt
- stage2_fast_rcnn_solver30k40k.pt
- stage2_fast_rcnn_train.pt
- stage2_rpn_solver60k80k.pt
- stage2_rpn_train.pt

- faster_rcnn_test.pt
- rpn_test.pt

每个文件必须具有正确的结构和关键字。以下各部分中描述了这些文件。

labels.txt

labels.txt 文件包含此对象检测模型检测的所有对象的名称。每个对象必须独占一行，如以下示例中所示。

```
defect1
defect2
defect3
```

faster_rcnn_final.caffemodel

此文件包含培训 Faster R-CNN 模型后的输出。此文件名必须与 model.config 中 **model** 的定义一致。当前，针对 Faster R-CNN 支持两个网络：ZF 和 VGG16。对于 ZF-net，名称应该以 ZF 开头，例如，ZF_mobile_final.caffemodel。对于 VGG16-net，名称应该以 VGG16 开头，例如，VGG16_mobile_final.caffemodel。

model.config

以下示例中的关键字显示为粗体文本：

```
{
    "modelType": "FRCNN",
    "model": " faster_rcnn_final.caffemodel",
    "solvers": "stage1_rpn_solver60k80k.pt,stage1_fast_rcnn_solver30k40k.pt,
               stage2_rpn_solver60k80k.pt,stage2_fast_rcnn_solver30k40k.pt",
    "net_file": "stage1_rpn_train.pt,stage1_fast_rcnn_train.pt,
               stage2_rpn_train.pt,stage2_fast_rcnn_train.pt",
    "deploy_net": "faster_rcnn_test.pt",
    "parameters": {
        "iteration": "40000,80000,40000,80000",
        "learningRate": 0.001,
        "stepsize": "10000",
        "gamma": "0.1"
    }
}
```

modelType 的值始终为 FRCNN。model 的值为模型文件的名称。solvers 的值为模型培训期间使用的解析器文件的列表。net_file 的值为网络定义文件的列表。deploy_net 的值为评分网络定义的名称。parameters 中的值为 Faster R-CNN 模型的所有超参数。iteration 是由逗号分隔的四个数字组成的字符串值，表示培训过程中四个阶段的迭代次数。

***.pt 文件**

扩展名为 .pt 的文件是模型定义文件。支持 Faster R-CNN 提供的 pascol_voc 模型。模板文件位于 models/pascal_voc/*netname*/faster_rcnn_alt_opt/ Faster R-CNN 安装目录中，其中 *netname* 为 ZF 或 VGG16。

YOLO V2

YOLO V2 对象检测模型必须是单个压缩文件，并包含正确的目录结构和文件。

压缩模型文件

压缩模型文件必须包含以下文件：

- labels.txt
- model.config
- yolo_final.weights
- Yolo.cfg

每个文件必须具有正确的结构和关键字。以下各部分中描述了这些文件。

labels.txt

labels.txt 文件包含此对象检测模型检测的所有对象的名称。每个对象必须独占一行，如以下示例中所示。

```
defect1
defect2
defect3
```

model.config

以下示例中的关键字显示为粗体文本：

```
{
  "modelType": "YOLO",
  "modelCfg": "Yolo.cfg",
  "model": " yolo_final.weights",
  "parameters": {
    "iteration": "40000",
    "batchSize": 16,
    "learningRate": 0.001,
    "subBatchSize": 2,
    "steps": "100,15000,25000,35000",
    "scales": "1,10,0.1,0.1"
  }
}
```

modelType 的值始终为 YOLO。modelCfg 的值为深度学习网络定义文件的名称。model 的值为实际模型文件的名称。parameters 中的值为 YOLO V2 模型的超参数。parameters 中 batchSize 的值不能大于 32。

yolo_final.weights

此文件包含培训 YOLO 模型后的输出。此文件名必须与 model.config 中 **model** 的定义一致。

Yolo.cfg

此模型定义文件包含网络定义、超参数和锚点设置。此文件的模板位于 darknet/cfg/ YOLO 安装目录中。此文件必须与 weights 文件一致。

SSD

SSD（单图多框检测器）对象检测模型必须是单个压缩文件，并包含正确的目录结构和文件。

压缩模型文件

压缩模型文件必须包含以下所有文件：

- labels.txt
- solver.prototxt
- deploy.prototxt
- model.config
- SSD.caffemodel

每个文件必须具有正确的结构和关键字。以下各部分中描述了这些文件。

labels.txt

`labels.txt` 文件包含此对象检测模型检测的所有对象的名称。每个对象必须独占一行，如以下示例中所示。

```
defect1
defect2
defect3
```

solver.prototxt

此文件包含 SSD 模型的所有超参数。

deploy.prototxt

此文件包含已培训模型的网络定义。

model.config

以下示例中的关键字显示为粗体文本：

```
{"modelType": "SSD",
"parameters": {
    "learningRate": 0.01,
    "iteration": 10000,
    "steps": "6000,8000",
    "batchSize": 16,
    "learningRatePolicy": "multistep",
    "display": 10,
    "snapshot": 1000
}
}
```

modelType 的值始终为 SSD。parameters 中的值为 SSD 模型的超参数。

SSD.caffemodel
此文件包含培训 SSD 模型后的输出。

验证模型

培训模型后，模型的状态会更改为“已培训”。模型管理者可以验证模型版本，并接受或拒绝培训的模型。模型管理者可使用验证图像集来验证模型。模型经过验证后，即可使用和部署该模型。

可以验证状态为“已培训”的模型版本。已培训或已重新培训的模型版本可以触发验证过程。验证报告时，必须管理图像集。每个图像组必须至少具有一个验证图像集才可验证模型。必须使用不同的图像集和培训图像集来验证模型版本。要启动验证过程，请选择**验证**。

完成验证过程后，会生成显示模型精确性的报告。可以创建以下类型的报告：

分类模型报告
在分类模型报告中，一个图像最多具有一个缺陷。混淆矩阵用于生成报告；报告中的每列表示验证数据集内的一个实际图像组类型。每行表示预测的图像组类型。图表中的最后一行表示聚集结果。

对象检测模型报告
在对象检测模型报告中，一个图像可具有多个缺陷。在此报告中，会计算精度和重新调用的平均值以指示模型精确性。

要查看报告，请选择**显示报告**。在“**验证报告**”窗口中，可以重新验证模型，拒绝或接受模型，以及部署模型。

如果拒绝了第一个模型版本，那么可以更改培训参数，然后重新培训模型。如果拒绝的模型版本不是第一个版本，那么可以使用新的图像集来重新培训新的模型版本。

将培训的模型分发到 Edge

模型管理者接受培训的模型版本后，会将该模型分发到 Edge，以便可对其进行检查。

过程

1. 选择状态为“已验证”的模型版本。首次部署模型时，单击**查看报告**以查看验证报告。
2. 单击“验证报告”页面上的**接受并发布**。
3. 这会显示“发布并部署”对话框。选择想要将模型部署到的 Edge，然后单击**发布并部署**以部署模型。
4. 如果该模型已部署在某些 Edge 上，请单击**管理部署**以打开部署对话框。选择想要进行部署或取消部署的 Edge，然后单击**部署**或**取消部署**。

重新培训模型

重新培训模型时，将创建新的模型版本。创建模型请求时，可定义重新培训策略。重新培训策略是用于触发自动重新培训的条件。如果模型管理者担心当前的模型精确性，那么可以手动选择图像文件来触发重新培训过程。

过程

1. 选择状态为“已部署”的模型版本，然后选择**重新培训**。
2. 管理图像集。每个图像组必须至少具有一个重新培训图像集才可重新培训模型。

   注: 要重新培训模型版本，最好使用包含更多图像的不同图像集。
3. 选择**重新培训**以开始重新培训过程。

下一步做什么

模型管理者验证是否已接受重新培训的模型。模型管理者可以部署新版本，旧的模型版本不会再处于已部署状态。

将模型用于独立 Edge

您可以将模型用于独立 Edge，以对生产线中的图像进行评分，将数据同步到中心，以及清理上载的数据。要将模型用于独立 Edge，必须发布模型，并将模型部署到独立 Edge。

发布模型

在手动部署独立 Edge 之前，必须在中心应用程序上发布模型。

关于此任务

要发布模型，必须已对其进行验证。

过程

1. 登录到中心应用程序的用户界面，然后选择模型实例。
2. 选择**查看报告**以查看验证报告。
3. 选择**接受并发布**以发布模型实例。

结果

发布模型后，模型的状态将会更改为“准备就绪可供部署”。

部署一个模型实例

使用此 API 在独立 Edge 上部署一个模型实例。

过程

1. 使用以下命令获取中心应用程序上所有发布的模型:
   `GET https://<edge machine host>:8449/api/getAvailableModels`
2. 使用以下命令在独立 Edge 上部署一个模型实例:
   `POST https://<edge machine host>:8449/api/deployModel`
   使用从 getAvailableModels API 返回的主体。

下一步做什么

有关 API 详细信息的更多信息，请转至“独立 Edge 服务”主题。

取消部署模型

使用此 API 在独立 Edge 上取消部署模型。

过程

使用以下命令取消部署模型:
`POST https://<edge machine host>:8449/api/undeployModel`
使用以下主体:

```
{"model_id":"model_id","model_instance_id":"model_instance_id"}
```

下一步做什么

有关 API 详细信息的更多信息，请转至“独立 Edge 服务”主题。

第 4 章 对检查结果进行检查

将检查结果发送到中心应用程序后，检验员和检验员主管可以转至“缺陷检查”选项卡来查看和过滤检查结果，并进行任何必要的更改。

图像

检验员和检验员主管可以查看图像，以确定这些图像是否分类为现有缺陷，并了解是否还有其他人在检查这些图像。查看图像可确定检验员或检验员主管在检查缺陷时必须执行的操作。

检验员可查看未检查和已检查图像。未检查图像是指模型已对图像评分，但检验员尚未对其进行检查。已检查图像是指模型已对图像评分，并且检验员已经对其进行检查。

检验员可查看未确认和已确认图像。未确认图像是指模型已对图像评分，但检验员尚未对其进行确认。已确认图像是指模型已对图像评分，并且检验员已经对其进行确认。

检验员主管可以查看对象和未知对象。未知对象是指检验员标记为未知缺陷的图像，因为检验员未将该图像分类为现有缺陷。这些图像在检验员主管的列表中会突出显示。

过滤缺陷

检验员和检验员主管可以向工作站概述和缺陷列表应用过滤器。

过程

1. 在“**所有工作站**”窗口上，选择一个工作站以查看未确认对象、已确认对象和未知对象的列表。
2. 选择“过滤器”图标。
3. 为条件输入值以设置过滤器，然后选择“添加”图标。过滤器会立即应用于列表。

检查缺陷

检验员和检验员主管会复查检查结果，并进行任何必要的更改。

关于此任务

选择图像时，会显示缺陷候选项和对应的置信度。缺省情况下，会选择第一个缺陷。如果缺陷类型未知，检验员可以为缺陷类型选择“未知”。

过程

1. 选择一个图像以查看图像详细信息和检查结果。
2. 选择**编辑缩放**来放大和缩小该图像或拖动该图像以定位。
3. 选择**设置缩放**来切换回编辑方式。针对缺陷框的详细信息，可以进行添加、移动、查看以及调整其大小。
4. 选择要确认的缺陷框，然后查看缺陷类型和置信度级别的详细信息。可以更改缺陷类型或删除位置。
5. 选择**确认**。
6. 如果图像不属于任何现有缺陷，那么检验员主管可以创建新的图像组。新缺陷会添加到同一模型下的图像的候选列表中。

使用模拟器上载图像

您可以使用模拟器来手动将图像发送到 Edge。此过程模拟从产品系列向 Edge 发送图像。您可以发送自己的图像，或使用服务器上的预定义图像。

关于此任务

过程

1. 从主菜单中选择**模拟器**。
2. 指定产品类型和工作站。
3. 要上载自己的图像，请选择**使用我自己的图像**，然后浏览至您的图像。或者，要使用服务器中的图像，请清除**使用我自己的图像**，然后指定图像总数和发送频率。
4. 单击**启动**。
5. 单击**查看分析结果**。

第 5 章 KPI 仪表板

检验员主管使用 KPI 仪表板来检查图像级别的缺陷率和位置级别的缺陷率。这些度量值可以帮助提供信息，方便您要求 IT 团队重新培训模型或调整制造过程。

KPI 仪表板位于“**KPI**”选项卡上。您可以选择所有工作站，也可以选择特定工作站。此选择会影响您所处理的范围。您还可以在实时视图和历史视图之间切换。在实时视图中，KPI 数据每 5 秒刷新一次。KPI 值包含单位缺陷数和缺陷率。单位缺陷数的计算公式是特定缺陷数除以图像总数。单位缺陷数的值表示一种缺陷类型的发生率。缺陷率的计算公式是具有一个或多个缺陷的图像数除以图像总数。缺陷率表示产品缺陷率。图表中的每一行表示当前 5 秒时间间隔内的 KPI 值。历史视图显示历史 KPI 数据。您可以编辑开始日期和结束日期来确定时间范围。历史视图中的 KPI 包含单位缺陷数和缺陷率。

历史视图有三个粒度：每小时、每天和每月。在每小时图表中，每个点表示 1 小时。例如，等于 24 点的 KPI 值表示 24 小时。在每天图表中，每个点表示 1 天。例如，等于 30 点的 KPI 值表示 30 天。在每月图表中，每个点表示 1 个月。例如，等于 12 点的 KPI 值表示 12 个月。所选时间范围和粒度之间存在一定的关系。如果所选时间范围的粒度中存在太多的点，那么该粒度会被禁用，您需要缩短时间范围才能将其启用。如果想要刷新图表，可以更改时间范围，或者单击**刷新**。服务器会定期计算历史 KPI 数据，因此根据配置会有一些时间延迟。计算历史 KPI 的缺省周期为 1 小时。

缺省情况下，“单位缺陷数”图表中仅显示前五个缺陷类型。如果想要检查其他缺陷类型，可以在图表下选择一个或多个缺陷类型，然后单击**显示趋势**。新的 KPI 图表会显示所选的缺陷类型。如果缺陷类型的 KPI 值为 0，那么将无法选择该类型来显示趋势。

第 6 章 与 Prescriptive Quality 集成

您可以将 Maximo PQI SaaS Visual Insights 中的历史缺陷率数据以 CSV 格式导出，然后在 Prescriptive Quality 中导入这些数据。

在 KPI 仪表板中查看历史缺陷率时可以导出数据。要查看这些数据，请在 KPI 仪表板上选择**历史记录**，然后选择**缺陷率**。选择**导出**时，将根据所选工作站、时间范围和时间粒度来导出数据。

以下各列会包含在 CSV 文件中。

表 *3. CSV* 文件中的列

列	**描述**
ATTRIBUTE_NAME	此值始终为 VI_DEFECT_IMAGE_RATE。
DATE	时间间隔为“每天”或“每月”时的日期（格式为 yyyy-MM-dd）或时间间隔为“每小时”时的日期（格式为 yyyy-MM-dd HH:mm:ss）。
CELL_NAME	发生缺陷的工作站的名称。这是 Prescriptive Quality 中的其中一个维度。
PRODUCED_QTY	所选工作站和时间范围内的图像总数。
TESTED_QTY	已测试的图像总数。
FAILED_QTY	显示缺陷的图像数。

第 7 章 应用程序编程接口

通过应用程序编程接口，可以执行管理数据组、数据文件、模型、检查结果和 Edge 等操作。

API 工作流程

许多 Maximo PQI SaaS Visual Insights API 调用都具有先决条件或与其他 API 调用相关。此外，对于各个 API 调用，只有特定用户角色才有权进行访问。因此，了解每个角色的 API 工作流程非常重要。

模型管理者

模型管理者通常按以下顺序使用 API 调用：

- 创建数据组。请参阅“数据组服务”中的“创建数据组”。
- 将数据文件上载到数据组。请参阅“数据文件服务”中的“将数据文件上载到数据组”。
- 创建模型。请参阅“模型服务”中的“创建模型”。
- 创建模型实例。请参阅“模型实例服务”中的“创建模型实例”。
- 培训模型。请参阅“培训模型实例”。根据图像数量和培训参数，培训过程会需要几分钟到几小时不等。要检查模型实例的状态，请参阅“获取一个模型实例”。
- 联机培训完成后，模型管理者可以验证模型。请参阅“验证模型实例”。
- 创建 Edge。请参阅“Edge 服务”中的“创建 Edge”。
- 将模型部署到 Edge 或拒绝模型。请参阅“模型实例操作服务”中的“部署模型实例”或“拒绝模型实例”。
- 重新培训或取消部署已部署的模型。请参阅“模型实例操作服务”中的“重新培训模型实例”或“取消部署模型实例”。

检验员

检验员通常按以下顺序使用 API 调用：

- 对图像评分。请参阅“评分服务”中的“对图像评分”。
- 获取检查结果工作站概述。请参阅“检查结果服务”中的“获取检查结果工作站概述”。
- 获取检查结果列表。请参阅“检查结果服务”中的“获取检查结果列表”。
- 确认检查结果。请参阅“检查结果服务”中的“确认检查结果”。

主管

主管通常按以下顺序使用 API 调用：

- 对图像评分。请参阅“评分服务”中的“对图像评分”。
- 获取检查结果工作站概述。请参阅“检查结果服务”中的“获取检查结果工作站概述”。
- 获取检查结果列表。请参阅“检查结果服务”中的“获取检查结果列表”。
- 确认检查结果。请参阅“检查结果服务”中的“确认检查结果”。
- 获取图像缺陷率文件。请参阅“QEWS 集成服务”中的“获取图像缺陷率文件”。

独立 Edge 管理员

独立 Edge 管理员必须拥有独立 Edge 机器的凭证。独立 Edge 管理员通常按以下顺序使用 API 调用：

- 获取可用模型。请参阅“独立 Edge 服务”中的“获取可用模型”。
- 部署一个模型。请参阅“独立 Edge 服务”中的“部署模型”。
- 部署所有模型。请参阅“独立 Edge 服务”中的“部署所有模型”。

- 将图像上载到 Edge 并对其评分。请参阅“独立 Edge 服务”中的“将图像上载到 Edge 并对其评分”。生产线或外部服务通常会使用此 API。
- 将检查结果从 Edge 同步到中心应用程序。请参阅“独立 Edge 服务”中的“将检查结果从 Edge 同步到中心应用程序”。
- 清除已同步到中心应用程序的检查结果。请参阅“独立 Edge 服务”中的“清除已同步到中心应用程序的检查结果”。
- 取消部署模型。请参阅“独立 Edge 服务”中的“取消部署模型”。

准备使用 API 调用

在使用任何 API 调用之前，请确保已准备好用户和解决方案并已获取 API 密钥。

执行以下任务以准备用户和解决方案：

- 确保登录的用户有权访问所需的 API 调用。
- 对于需要解决方案标识的 API 调用，请指定 `vi` 作为解决方案。
- 确保用户具有 IBM 为其创建的 API 密钥。

服务响应

API 调用完成后，将提供成功或错误响应。

API 服务提供以下响应。

- 成功响应：
  - 对于创建服务，为状态码 201
  - 对于其他服务，为状态码 200
- 数据库错误响应：
  - 状态码 500
  - 错误消息，例如“创建新数据组时缺少参数或参数格式无效。”
- 授权错误
  - 状态码 401
  - 错误消息，例如“需要在头中设置授权或 APIKEY。”

数据组服务

通过数据组服务，可以执行获取所有数据组、获取一个数据组和创建新数据组等任务。

获取所有数据组

获取所有现有数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup

方法

请求类型 GET

URL 参数

`user`：*`String`*。用于标识用户。必需。

`solution`：*`String`*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
  {
    "id": "892d05cd-84e5-4bae-b57f-edbbc3f8c598",
    "groupName": "MisT",
    "createdBy": "auto_mm",
    "description": "test_description",
    "dataFormat": "jpg",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "updatedTime": "2017-11-29T16:47:54.039+08:00"
  },
  {
    "id": "9bebc890-0237-4a3b-8908-9f5f53fdb3a2",
    "groupName": "NoDefect",
    "createdBy": "auto_mm",
    "description": "test_description",
    "dataFormat": "jpg",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "updatedTime": "2017-11-29T16:47:54.463+08:00"
  },
  {
    "id": "b989aae2-f1ae-4a77-93bb-4e9e4f18264d",
    "groupName": "Defect",
    "createdBy": "auto_mm",
    "description": "test_description",
    "dataFormat": "jpg",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "updatedTime": "2017-11-29T16:47:53.536+08:00"
  }
]
```

响应项

id：*String*。数据组标识。

groupName：*String*。数据组名称。

createdBy：*String*。数据组所有者。

description：*String*。数据组描述。

dataFormat：*String*。数据组格式。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

updatedTime: *Time*。上次更新数据组的时间。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aa
```

```
b2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/dataGroup?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取具有数据文件的所有数据组

获取具有每个组的数据文件的所有现有数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup

方法

请求类型 GET

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

dataFiles: *[]*。用于标识和获取数据文件。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
  {
    "id": "3e420dbc-88fb-463a-83ee-a317688b02e0",
    "groupName": "test",
    "createdBy": "auto_mm",
    "description": "test_description",
    "dataFormat": "jpg",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "dataFiles": [
      {
        "id": "3e420dbc-88fb-463a-83ee-a317688b02e0_539bf24a-afc9
-4f43-9ca9-b7ec0648e781",
        "groupId": "3e420dbc-88fb-463a-83ee-a317688b02e0",
        "createdBy": "auto_mm",
        "fileName": "MisT.zip",
        "fileCount": "3",
        "updatedTime": "1511949371332",
        "fileUrl": "/user/AUTO/VI/datagroup/3e420dbc-88fb-463a-83
ee-a317688b02e0/1511949371332.zip"
      },
      {
        "id": "3e420dbc-88fb-463a-83ee-a317688b02e0_f76b31fb-f803-
4d0e-995e-b597bf3c761d",
        "groupId": "3e420dbc-88fb-463a-83ee-a317688b02e0",
        "createdBy": "auto_mm",
        "fileName": "Defect.zip",
        "fileCount": "3",
        "updatedTime": "1511949368998",
        "fileUrl": "/user/AUTO/VI/datagroup/3e420dbc-88fb-463a-83ee
-a317688b02e0/1511949368998.zip"
      }
```

```
    ],
    "updatedTime": "2017-11-29T16:59:43.873+08:00"
  }
]
```

响应项

id：*String*。数据组标识。

groupName：*String*。数据组名称。

createdBy：*String*。数据组所有者。

description：*String*。数据组描述。

dataFormat：*String*。数据组格式。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

dataFiles: *JSONObject*。包含有关数据文件的信息。

id：*String*。数据文件标识。

groupId：*String*。数据组标识。

createdBy：*String*。数据文件所有者。

fileName：*String*。文件名。

fileCount：*String*。文件计数。

updatedTime: <Time>。上次更新数据文件的时间。

fileUrl:。服务器中存储的文件路径。

updatedTime: *Time*。上次更新数据组的时间。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d
4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.229.91:
9447/ibm/iotm/service/dataGroup?user=auto_mm&solution=vi &dataFiles=[]"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取一个数据组

获取一个数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup/*groupId*

方法

请求类型 GET

URL 参数

groupId：*String*。数据组标识。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
  {
    "tags": "",
    "id": "3e420dbc-88fb-463a-83ee-a317688b02e0",
    "groupName": "test",
    "createdBy": "auto_mm",
    "description": "test_description",
    "dataFormat": "jpg",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "updatedTime": "2017-11-29T16:59:43.873+08:00"
  }
]
```

响应项

tags：*String*。数据组标记。

id：*String*。数据组标识。

groupName：*String*。数据组名称。

createdBy：*String*。数据组所有者。

description：*String*。数据组描述。

dataFormat：*String*。数据组格式。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

updatedTime: *Time*。上次更新数据组的时间。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77
d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/dataGroup/3e420dbc-88fb-463a-8
3ee-a317688b02e0?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

创建数据组

创建数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup

方法

请求类型 POST

URL 参数

user: *String*。用于标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

groupName: *String*。数据组名称。

description: *String*。数据组描述。

dataFormat: *String*。数据组格式。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

样本主体

```
[
  {
    "groupName": "test",
    "description": "test_description",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    }
  }
]
```

成功响应

```
201
[
  {
    "id": "7dac9493-4d8c-4472-a4ca-ae450ccaea5d",
    "groupName": "test",
    "tags": null,
    "dataFormat": "jpg",
    "createdBy": "auto_mm",
    "updatedTime": "2017-11-30T09:59:09.597+08:00",
    "parameters": {
      "isHybrid": "false",
      "isDefect": "true"
    },
    "description": "test_description"
  }
]
```

响应项

id: *String*。数据组标识。

groupName: *String*。数据组名称。

tags: *String*。数据组标记。

createdBy: *String*。数据组所有者。

description: *String*。数据组描述。

dataFormat: *String*。数据组格式。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

updatedTime: *Time*。上次更新数据组的时间。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f05
79e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H "Content-Type:
application/json;charset=UTF-8" "https://9.112.229.91:9447/ibm/iotm/service/
dataGroup?user=auto_mm&solution=vi"  --data '[{"groupName":"test","dataFormat":
"jpg", "description": "test_description","parameters":{"isHybrid":"false","isDefect"
:"true"}}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

更新数据组

更新数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup/*groupId*

方法

请求类型 PUT

URL 参数

groupId：*String*。数据组标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

description：*String*。数据组描述。

isHybrid: *Boolean*。True 表示对象检测。False 表示分类。

isDefect: *Boolean*。True 表示是缺陷。False 表示不是缺陷。

样本主体

```
200
[
    {

    "description":"test",

    "parameters": {

    "isDefect":"true",

    "isHybrid": "true" }
    }
]
```

成功响应

```
[{}]
```

响应项

无

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533d
d34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H "Content-Type:
application/json;charset=UTF-8" -X PUT "https://9.112.229.91:9447/ibm/iotm/service
/dataGroup/7dac9493-4d8c-4472-a4ca-ae450ccaea5d?user=auto_mm&solution=vi"  --data
'[{"description":"test","parameters":{"isHybrid":"true","isDefect":"true"}}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

删除数据组

删除现有数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataGroup/*groupId*

方法

请求类型 DELETE

URL 参数

groupId：*String*。数据组标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

```
[{}]
```

成功响应

```
200
[{}]
```

响应项

无

样本调用

```
curl -k -X DELETE -H "APIKEY:8b796582
25de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f05
79e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/dataGroup/
ed568556-1462-45e9-9319-4bfa8186d2cd?user=auto_mm&solution=
vi”  -d  ‘[{}]’
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

数据文件服务

通过数据文件服务，可以执行获取属于一个数据组的所有数据文件、获取一个数据文件和下载二进制文件等任务。

获取属于一个数据组的所有数据文件

获取属于一个数据组的所有数据文件。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataFile

方法

请求类型 GET

URL 参数

groupId：*String*。数据组标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "id": "06c870e5-1701-419a-8a84-0f40ee82fe33_a7f1772f-07a0-4d33-9efe
-f2e45a2dc904",
        "groupId": "06c870e5-1701-419a-8a84-0f40ee82fe33",
        "createdBy": "modelmanager1",
        "fileName": "NG.zip",
        "updatedTime": "1502185276752",
        "fileCount": "52",
        "fileUrl": "/user/T1/VIQ/datagroup/06c870e5-1701-419a-8a84
-0f40ee82fe33/1502185276752.zip"
    },
    {
        "id": "06c870e5-1701-419a-8a84-0f40ee82fe33_b3af214c-b603-43f4-944d-
40277b9a6fd9",
        "groupId": "06c870e5-1701-419a-8a84-0f40ee82fe33",
        "createdBy": "modelmanager1",
        "fileName": "IoT4M_March2017_VI_Source.zip",
        "updatedTime": "1502184389077",
        "fileCount": "52",
        "fileUrl": "/user/T1/VIQ/datagroup/06c870e5-1701-419a-8a84-0f40ee82fe
33/1502184389077.zip"
    },
    {
        "id": "06c870e5-1701-419a-8a84-0f40ee82fe33_c901e409-d9db-4f02-8d9c
-fa6002929701",
        "groupId": "06c870e5-1701-419a-8a84-0f40ee82fe33",
        "createdBy": "modelmanager1",
        "fileName": "NG.zip",
        "updatedTime": "1502185248433",
        "fileCount": "52",
        "fileUrl": "/user/T1/VIQ/datagroup/06c870e5-1701-419a-8a84-0f40ee82fe
33/1502185248433.zip"
```

```
    }
]
```

响应项

id：*String*。数据文件标识。

groupId：*String*。数据组标识。

createdBy：*String*。数据文件所有者。

fileName：*String*。文件名。

fileCount：*String*。文件计数。

updatedTime: *Time*。上次更新数据文件的时间。

fileUrl:。服务器中存储的文件路径。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0
579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/dataFile?user=auto_mm&solution
=vi&groupId=ed568556-1462-45e9-9319-4bfa8186d2cd"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取一个数据文件

获取一个数据文件。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataFile

方法

请求类型 GET

URL 参数

fileId：*String*。数据文件标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "id": "06c870e5-1701-419a-8a84-0f40ee82fe33_c901e409-d9db-4f02-8d9c
-fa6002929701",
        "groupId": "c901e409-d9db-4f02-8d9c-fa6002929701",
        "createdBy": "modelmanager1",
        "fileName": "NG.zip",
        "updatedTime": "1502185248433",
```

```
        "fileCount": "52",
        "fileUrl": "/user/T1/VIQ/datagroup/06c870e5-1701-419a-8a84
-0f40ee82fe33/1502185248433.zip"
    }
]
```

响应项

id：*String*。数据文件标识。

groupId：*String*。数据组标识。

createdBy：*String*。数据文件所有者。

fileName：*String*。文件名。

updatedTime: *Time*。上次更新数据文件的时间。

fileUrl:。服务器中存储的文件路径。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f05
79e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112
.229.91:9447/ibm/iotm/service/dataFile/06c870e5-1701-419a-8a84-0f40ee82fe33_
c901e409-d9db-4f02-8d9c-fa6002929701?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

下载数据文件的二进制内容

下载数据文件的二进制内容。模型管理者和数据研究员有权使用此 API。

URL

/ibm/iotm/service/dataFileBinary

方法

请求类型 GET

URL 参数

fileId：*String*。数据文件标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

数据文件的二进制内容。

响应项

无

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f05
79e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.
112.229.91:9447/ibm/iotm/service/ dataFileBinary ?fileId=06c870e5-1701-419a
-8a84-0f40ee82fe33_c901e409-d9db-4f02-8d9c-fa6002929701&user=auto_mm&solution
=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

将数据文件上载到数据组

将一个或多个数据文件上载到数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataFileBinary

方法

请求类型 POST

URL 参数

groupId: *String*。数据组标识。必需。

user: *String*。用于标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

Content-type: multipart/form-data。

Content-Disposition: form-data; name="files[]"; filename=*your file name*。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

数据文件的二进制内容。

成功响应

```
200
{"result":{"Defect.zip":{"id":"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-057a-
4904-8fad-bbca93abd80b","count":"3","name":"Defect.zip","errorMsg":"","updatedTime"
:"1512021663498","url":"\/user\/AUTO\/VI\/datagroup\/9f7aa9d6-24d6-4611-8d4d-79a52d2
bab02\/1512021663498.zip"}},"error_message":{}}
```

响应项

id: *String*。数据文件标识。

name: *String*。文件名。

count: *String*。图像计数。

updatedTime: *Time*。上次更新数据文件的时间。

fileUrl:。服务器中存储的文件路径。

error_message:。上载响应的错误消息。

样本调用

```
curl -k -X POST -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e53
3dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
-H "Content-Type:multipart/form-data"  -H "Content-Disposition:
form-data; name=\"files[]\"; filename=\"Defect.zip\"" "https://9.112.229.91:9447
/ibm/iotm/service/dataFileBinary?user=auto_mm&solution=vi&groupId=9f7aa9d6-24d6-
4611-8d4d-79a52d2bab02" --connect-timeout 600 -F file=@C:\\code\\Automation_98\\
API\\vi\\VI_API\\file\\Defect.zip
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

删除一个数据文件

删除一个数据文件。模型管理者有权使用此 API。

URL

/ibm/iotm/service/dataFile/*fileId*

方法

请求类型 DELETE

URL 参数

fileId：*String*。数据文件标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

```
[{}]
```

成功响应

```
200
[{}]
```

响应项

无

样本调用

```
curl -k -X DELETE -H "APIKEY:8b79658
225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f057
9e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/dataFile/
9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_e2cc3d4a-bae9-
4600-9dff-d80fa3ee8832?user=auto_mm&solution=vi”  -d  ‘[{}]’
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

未标记数据组服务

通过未标记数据组服务，可以执行上载未标记压缩图像文件、创建未标记数据组和获取一个未标记数据组等任务。

上载未标记的压缩图像文件

将未标记的压缩文件上载到服务器。模型管理者有权使用此 API。

URL

/ibm/iotm/service/unLabeledImageZipServlet

方法

请求类型 POST

URL 参数

groupType: *String*。数据组类型。值必须为 *classification* 或 *objectdetection*。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: multipart/form-data。

Content-Disposition: form-data; name="files[]"; filename=*your file name*。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

数据文件的二进制内容。

成功响应

```
200 {"message":{},"result":{"imageZip":{"imageZipId":"b76b33b0-26d6-4f85-83ea-e520edbe5ff8",
"labeledImageCount":0,"labelCount":0,"imageZipName":"images.zip","count":85,"updatedTime":
"2019-03-19 00:18:53.304","url":"\/user\/TENANT\/VI\/unlabeledgroup\/b76b33b0-26d6-4f85-
83ea-e520edbe5ff8.zip"}}}
```

响应项

imageZip: *JSON Object*。上载的压缩图像文件的信息。信息包括 imageZipId、LabeledImageCount、labelCount、imageZipName、count、updatedTime 和 URL。

imageZipId: *String*。为上载的压缩图像文件分配的标识。

LabeledImageCount: *Integer*。压缩图像文件中预标记的图像数。

labelCount: *Integer*。分配给压缩图像文件中图像的标签数。

imageZipName: *String*。压缩图像文件的文件名。

count: *Integer*。压缩图像文件中的图像总数。

updatedTime: *Time*。上载压缩图像文件的时间。

Url: *String*。服务器中存储的文件路径。

error_message: *JSON Object*。上载响应的错误消息。

样本调用

```
curl -k -X POST -H"APIKEY:apikey"
  -H "Content-Type:multipart/form-data"  -H "Content-Disposition: form-data;
 name=\"files[]\"; filename=\"image.zip\"" "https://iotm.predictivesolutionsapps
.ibmcloud.com/ibm/iotm/service/unLabeledImageZipServlet?user=auto_mm&solution=vi&
 groupType=classification" --connect-timeout 600
-F file=@C:\\code\\Automation_98\\API\\vi\\VI_API\\file\\image.zip
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

创建未标记数据组

使用上载的压缩图像文件来创建未标记数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/unLabeledGroup

方法

请求类型 POST

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: multipart/form-data。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

groupName：*String*。数据组名称。

groupType: *String*。数据组类型。值必须为 *classification* 或 *objectdetection*。

imageZip: *JSON Array*。上载的压缩图像文件的信息。上载未标记压缩图像文件时，可以从响应中获取相关信息。信息包括 imageZipId、LabeledImageCount、labelCount、imageZipName、count、updatedTime 和 URL。

imageZipId: *String*。为上载的压缩图像文件分配的标识。

LabeledImageCount: *Integer*。压缩图像文件中预标记的图像数。

labelCount: *Integer*。分配给压缩图像文件中图像的标签数。

imageZipName: *String*。压缩图像文件的文件名。

count: *Integer*。压缩图像文件中的图像总数。

updatedTime: *Time*。上载压缩图像文件的时间。

Url: *String*。服务器中存储的文件路径。

样本主体

```
[{"groupName":"DI","groupType":"classification","imageZip":[{"imageZipId":
"b76b33b0-26d6-4f85-83ea-e520edbe5ff8","labeledImageCount":0,"labelCount":
0,"imageZipName":"image.zip","count":85,"updatedTime":
"2019-03-19 00:18:53.304","url":"/user/TENANT/VI/unlabeledgroup/
b76b33b0-26d6-4f85-83ea-e520edbe5ff8.zip"}]}]
```

成功响应

```
200
[{"groupName":"DI","id":"04f4eaf3-dc93-4ad8-b2e2-d3fbd202c758","
groupType":"classification","imageZip":[{"imageZipId":"b76b33b0-
26d6-4f85-83ea-e520edbe5ff8","labeledImageCount":0,"labelCount":
0,"imageZipName":"image.zip","count":85,"updatedTime":"2019-03-
19 00:18:53.304","url":"/user/TENANT/VI/unlabeledgroup/b76b33b0
-26d6-4f85-83ea-e520edbe5ff8.zip"}]}]
```

响应项

id: *String*。数据组标识。

groupName：*String*。数据组名称。

groupType: *String*。数据组类型。

createdBy: *String*。数据组所有者。

groupStatus: *String*。未标记数据组的状态。

updatedTime: *Time*。上次更新数据组的时间。

imageZip: *JSON Array*。关联的压缩图像文件的信息。每个项包含 imageZipId、LabeledImageCount、labelCount、imageZipName、count、updatedTime 和 URL。

imageZipId: *String*。为上载的压缩图像文件分配的标识。

LabeledImageCount: *Integer*。压缩图像文件中预标记的图像数。

labelCount: *Integer*。分配给压缩图像文件中图像的标签数。

imageZipName: *String*。压缩图像文件的文件名。

count: *Integer*。压缩图像文件中的图像总数。

updatedTime: *Time*。上载压缩图像文件的时间。

Url: *String*。服务器中存储的文件路径。

样本调用

```
curl –k –X POST -H "APIKEY:apikey" -H
"Content-Type:application/json;charset=UTF-8"
"https://iotm.predictivesolutionsapps.ibmcloud.com/
ibm/iotm/service/unLabeledGroup?user=auto_mm&solution=vi"
--data '[{"groupName":"DI","groupType":"classification",
"imageZip":[{"imageZipId":"b76b33b0-26d6-4f85-83ea-e520edbe5ff8",
"labeledImageCount":0,"labelCount":0,"imageZipName":"image.zip",
"count":85,"updatedTime":"2019-03-19 00:18:53.304","url":
"/user/TENANT/VI/unlabeledgroup/b76b33b0-26d6-4f85-83ea-
e520edbe5ff8.zip"}]}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取一个未标记数据组

获取一个未标记数据组。模型管理者有权使用此 API。

URL

/ibm/iotm/service/unLabeledGroup/*groupId*

方法

请求类型 GET

URL 参数

groupId: *String*。数据组标识。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200

[{"groupName":"DI","id":"04f4eaf3-dc93-4ad8-b2e2-d3fbd202c758",
"groupType":"classification","imageZip":[{"imageZipId":"b76b33b0-
26d6-4f85-83ea-e520edbe5ff8","labeledImageCount":0,"labelCount":0,
"imageZipName":"image.zip","count":85,"updatedTime":"2019-03-19
00:18:53.304","url":"/user/TENANT/VI/unlabeledgroup/b76b33b0-26d6-
4f85-83ea-e520edbe5ff8.zip"}]}]
```

响应项

id: *String*。数据组标识。

groupName：*String*。数据组名称。

groupType: *String*。数据组类型。

createdBy: *String*。数据组所有者。

groupStatus: *String*。未标记数据组的状态。

updatedTime: *Time*。上次更新数据组的时间。

imageZip: *JSON Array*。关联的压缩图像文件的信息。每个项包含 imageZipId、LabeledImageCount、labelCount、imageZipName、count、updatedTime 和 URL。

imageZipId: *String*。为上载的压缩图像文件分配的标识。

LabeledImageCount: *Integer*。压缩图像文件中预标记的图像数。

labelCount: *Integer*。分配给压缩图像文件中图像的标签数。

imageZipName: *String*。压缩图像文件的文件名。

count: *Integer*。压缩图像文件中的图像总数。

updatedTime: *Time*。上载压缩图像文件的时间。

Url: *String*。服务器中存储的文件路径。

样本调用

```
curl -k -H "APIKEY:apikey"
 "https://iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/service
/unLabeledGroup/04f4eaf3-dc93-4ad8-b2e2-d3fbd202c758?user=auto_mm&
solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

模型服务

通过模型服务，可以执行获取所有模型、获取一个模型和创建模型等任务。

获取所有模型

获取所有现有模型。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model

方法

请求类型 GET

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[{
        "modelId": "a9150695-5dc5-4bef-937d-c71fc186bc14",
        "groupIds": "e83f9faf-2b08-44d1-81b0-7c69e6407329",
        "modelName": "appscan test01",
        "createdBy": "demoadm@cn.ibm.com",
        "modelType": "classification",
        "statusStatistics": {},
        "dataFormat": "png",
        "retrainPolicy": {
            "scheduler": "Weekly,Sunday",
            "ratio": "",
            "reuseDayNumber": "",
            "maxPieceNumber": "",
            "archiveDataType": "0",
            "imageNumber": 1000,
            "accuracy": "70"
        },
        "parameters": {
            "imageSize": "100*100",
            "confidence": 100,
            "trainParam": {
                "epoc": 50,
                "ratioTrain": 80,
                "stepsize": 30,
                "test_iter": 20,
                "snapshot": 100,
                "momentum": 0.9,
                "ratioVal": 20,
                "learningRate": 0.0001,
                "display": 1,
                "learningRatePolicy": "step",
                "weight_decay": 0.00001,
                "test_epoc": 10,
                "network": "GoogLeNet",
                "algorithm": "CNN",
                "test_interval": 5,
                "gamma": 0.1,
                "solver_type": "SGD",
```

```
                "maxIter": 600
            }
        },
        "updatedTime": "2018-11-16T00:16:20.032-06:00",
        "productType": "appscan test01"
    }
……
]
```

响应项

modelId：*String*。模型标识。

modelName：*String*。模型名称。

createdBy：*String*。模型所有者。

updatedTime: *Time*。上次更新模型的时间。

productType：*String*。模型产品类型。

description：*String*。模型描述。

StatusStatistics：*String*。模型状态。

groupIds：*String*。模型数据组标识。

dataFormat：*String*。模型数据格式。

retrainPolicy: *JSON Object*。模型重新培训策略。

scheduler: *Time*。模型重新培训日期，值为周一到周日。

imageNumber: *Integer*。可以重新培训编号大于此数字的图像。

accuracy: *Double*。可以重新培训小于此数字的模型精确性值。

parameters: *JSON Object*。包含图像大小和模型置信度信息以及 trainParam 字段。

imageSize: *Double * Double*。图像大小。

confidence: *Double*。模型定义的置信度值。

trainParam: *JSON Object*。包含培训参数信息。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e
533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112
.229.91:9447/ibm/iotm/service/model?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取一个模型

获取一个现有模型。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model/<modelID>

方法

请求类型 GET

URL 参数

modelId：*String*。模型标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "modelId": "f39df888-7559-4283-ae9d-b3504118dddd",
        "groupIds": "9e744bc4-0a22-4bf3-9e63-fa39135842de",
        "modelName": "arthuryolo22000v2",
        "createdBy": "demoadm@cn.ibm.com",
        "modelType": "objectdetection",
        "description": "arthuryolo22000v2",
        "statusStatistics": {
            "trained": "1"
        },
        "dataFormat": "png",
        "retrainPolicy": {
            "scheduler": "Weekly,Sunday",
            "ratio": "",
            "reuseDayNumber": "",
            "maxPieceNumber": "",
            "archiveDataType": "0",
            "imageNumber": 1000,
            "accuracy": "70"
        },
        "trainedBy": "demoadm@cn.ibm.com",
        "parameters": {
            "JobInstanceMap": {
                "4a9605a2-03e3-43ff-9a47-d7638a0145d9": "f39df888-7559-4283-ae9d-
b3504118dddd_1541744213466"
            },
            "odRetrainUrl": "/user/Q3T1/VI/modellist/f39df888-7559-4283-ae9d-
b3504118dddd_1541761094044.zip",
            "imageSize": "100*100",
            "confidence": 100,
            "trainParam": {
                "ratioTrain": 80,
                "subBatchSize": 2,
                "scales": "1,1",
                "ratioVal": 20,
                "learningRate": 0.0001,
                "batchSize": 16,
                "steps": "6000,10000",
                "recommend": 0,
                "network": "YoloV2",
                "iteration": 10000,
                "algorithm": "YOLO"
            }
        },
        "updatedTime": "2018-11-09T00:24:19.949-06:00",
        "productType": "arthuryolo22000v2"
    }
]
```

响应项

modelId：*String*。模型标识。

modelName：*String*。模型名称。

createdBy：*String*。模型所有者。

updatedTime: *Time*。上次更新模型的时间。

productType：*String*。模型产品类型。

description：*String*。模型描述。

StatusStatistics：*String*。模型状态。

groupIds：*String*。模型数据组标识。

dataFormat：*String*。模型数据格式。

retrainPolicy: *JSON Object*。模型重新培训策略。

scheduler: *Time*。模型重新培训日期，值为周一到周日。

imageNumber: *Integer*。可以重新培训编号大于此数字的图像。

accuracy: *Double*。可以重新培训小于此数字的模型精确性值。

parameters: *JSON Object*。包含图像大小和模型置信度信息以及 odRetrainUrl、JobInstanceMap 和 trainParam 字段。

imageSize: *Double * Double*。图像大小。

confidence: *Double*。模型定义的置信度值。

trainParam: *JSON Object*。包含算法和网络以及超参数信息。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533
dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.229.91:
9447/ibm/iotm/service/model/ab219b13-16c0-4c7b-ae74-721f4719e314?user=auto_mm&so
lution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

创建模型

创建模型。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model

方法

请求类型 POST

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

modelName：*String*。模型名称。

modelType：*String*。模型类型。选项为“classification”和“objectdetection”。

createdBy：*String*。模型所有者。

description：*String*。模型描述。

groupIds：*String*。模型数据组标识。

dataFormat：*String*。模型数据格式。

retrainPolicy: *JSON Object*。模型重新培训策略。

scheduler: *Time*。模型重新培训日期，值为周一到周日。

imageNumber: *Integer*。可以重新培训编号大于此数字的图像。

accuracy: *Double*。可以重新培训小于此数字的模型精确性值。

parameters: *JSON Object*。包含图像大小、模型置信度信息以及培训参数的属性，包括算法、网络和超参数信息。

培训参数、超参数和模型类型值具有对应关系。如果模型类型值为“classification”，那么“algorithm”值为“CNN”，对应的“network”有 3 个选项：“GoogleNet”、“LeNet”或“AlexNet”。对应的超参数包括：“learningRate”（0 到 1 之间的实数）、“maxIter”或“epoch”（正整数）、“stepsize”（正整数）、“gamma”（0 到 1 之间的实数）、“learningRatePolicy”（值为“step”）、“test_iter”或“test_epoch”（正整数）、“test_interval”（正整数）、“snapshot”（等于或大于 maxIter/20 的整数）、“ratioTrain”（用于培训的图像的比例，是 0 到 100 之间的整数）和“ratioVal”（用于验证的图像的比例，是 0 到 100 之间的整数）。

如果模型类型值为“objectdetection”，那么“algorithm”值包括：“FRCNN”、“YOLO”和“SSD”。如果“algorithm”值为“FRCNN”，那么对应的“network”有 2 个选项：“ZfNet”和“VGG16”。对应的超参数包括：“learningRate”（0 到 1 之间的实数）、“iteration”（由逗号分隔的正整数组成，例如：10000, 10000, 10000, 10000）、“stepsize”（正整数）、“gamma”（0 到 1 之间的实数）、“ratioTrain”（0 到 100 之间的整数）和“ratioVal”（0 到 100 之间的整数）。

如果“algorithm”值为“YOLO”，那么对应的“network”有 3 个选项：“YoloV2”、“TinyYolo”和“YoloV1”。对应的超参数包括：“learningRate”(0 到 1 之间的实数）、“iteration”（正整数）、“steps”（逗号分隔的正整数，例如：100, 1000, 5000, 8000）、“batchSize”（正整数）、“scales”（逗号分隔的实数，例如：0.1, 0.1, 0.1, 0.1）、“subBatchSize”（正整数）、“ratioTrain”（0 到 100 之间的整数）、“ratioVal”（0 到 100 之间的整数）。

如果“algorithm”值为“SSD”，那么对应的“network”为“SSD300”。对应的超参数包括“learningRate”（0 到 1 之间的实数）、“iteration”（正整数）、“steps”（逗号分隔的正整数，例如：100, 1000, 5000, 8000）、“batchSize”（值为正整数）、“learningRatePolicy”（值为“multistep”）、“snapshot”（等于或大于 iteration/20 的整数）、“ratioTrain”（0 到 100 之间的整数）和“ratioVal”（0 到 100 之间的整数）。

confidence: *Double*。模型定义的置信度值。

productType: *String*。模型产品类型。

样本主体

```
[{"modelName":"walkermodel1","productType":"walkermodel1","dataFormat":
"png","modelType":"classification","retrainPolicy":{"imageNumber":"1000",
"accuracy":"70","scheduler":"Weekly, Sunday","reuseDayNumber":"",
"maxPieceNumber":"","ratio":""},"description":"this is a test",
"groupIds":"b5164942-370c-418d-9fc2-f6ffd860b79d","parameters":
{"imageSize":"100*100","confidence":100,"trainParam":{"ratioTrain":80,
"ratioVal":20,"gamma":0.1,"stepsize":33,"maxIter":100,"learningRate"
:0.01,"learningRatePolicy":"step","test_iter":1,"test_interval":2,
"snapshot":10,"network":"GoogLeNet","algorithm":"CNN"}}}]
```

成功响应

```
201

[{"modelId":"4d0dfe01-7a26-4071-92fb-ec597e7863ce","modelName":
"walkermodel1","description":"this is a test","productType":
"walkermodel1","groupIds":"b5164942-370c-418d-9fc2-f6ffd860b79d",
"dataFormat":"png","createdBy":"automm@cn.ibm.com","updatedTime":
"2018-02-13T00:44:31.288-06:00","parameters":{"imageSize":"100*100",
"confidence":100,"trainParam":{"ratioTrain":80,"stepsize":33,
```

```
"test_iter":1,"snapshot":10,"momentum":0.9,"ratioVal":20,
"learningRate":0.01,"display":1,"learningRatePolicy":"step",
"weight_decay":1.0E-5,"network":"GoogLeNet","test_interval":
2,"algorithm":"CNN","gamma":0.1,"solver_type":"SGD","maxIter":100}},
"retrainPolicy":{"ratio":"","scheduler":"Weekly, Sunday",
"maxPieceNumber":"","reuseDayNumber":"","imageNumber":"1000",
"accuracy":"70"}}]
```

响应项

modelId：*String*。模型标识。

modelName：*String*。模型名称。

productType：*String*。产品类型。

createdBy：*String*。模型所有者。

updatedTime: *Time*。上次更新模型的时间。

productType：*String*。模型产品类型。

description：*String*。模型描述。

groupIds：*String*。模型数据组标识。

dataFormat：*String*。模型数据格式。

retrainPolicy: *JSON Object*。模型重新培训策略。

scheduler: *Time*。模型重新培训日期，值为周一到周日。

imageNumber: *Integer*。可以重新培训编号大于此数字的图像。

accuracy: *Double*。可以重新培训小于此数字的模型精确性值。

parameters: *JSON Object*。包含图像大小和模型置信度信息。

confidence: *Double*。模型定义的置信度值。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c
77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
-H "Content-Type:application/json;charset=UTF-8" "https://9.112.229.91/
ibm/iotm/service/model?&tenant=VITest&solution=vi&user=auto_mm"
--data ' [{"modelName":"testcreatemodel11","imagegroupList":"[{\"id\":
\"b5164942-370c-418d-9fc2-f6ffd860b79d\",\"dataFiles\":[{\"id\":\
"b5164942-370c-418d-9fc2-f6ffd860b79d--b5164942-370c-418d-9fc2-
f6ffd860b79d_74674410-06df-4a9e-a9e9-12f5038efc22\",\"name\":\
"Good_val.zip\",\"count\":5,\"updatedTime\":\"12/27/2017,
10:38:13 AM\"}],\"name\":\"Good\",\"updatedTime\":\"12/27/2017,
10:36:10 AM\"}]","productType":"testcreatemodel11","dataFormat":
"png","modelType":"classification","retrainPolicy":{"imageNumber":
"1000","accuracy":"70","scheduler":"Weekly, Sunday","reuseDayNumber"
:"","maxPieceNumber":"","ratio":""},"description":"this is a test",
"groupIds":"b5164942-370c-418d-9fc2-f6ffd860b79d","parameters":
{"imageSize":"100*100","confidence":100,"trainParam":{"ratioTrain":
80,"ratioVal":20,"gamma":0.1,"stepsize":33,"maxIter":100,"learningRate"
:0.01,"learningRatePolicy":"step","test_iter":1,"test_interval":2,
"snapshot":10,"network":"GoogLeNet","algorithm":"CNN"}}}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

更新模型

更新一个现有模型。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model/*modelId*

方法

请求类型 PUT

URL 参数

modelId：*String*。模型标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

groupIds：*String*。数据组标识。

样本主体

```
[{"groupIds":"45805b3c-f234-4f18-a55e-61ce203a7db4,678a3f42-f9d4-48a2-bb
29-c226d61a08ea"}]
```

成功响应

```
200
[{}]
```

响应项

无

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd
34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H "Content-Type:application
/json;charset=UTF-8" -X PUT "https://9.112.229.91:9447/ibm/iotm/service/model/106ef
cb7-1338-4ed2-b8ca-27b45e11e865?user=auto_mm&solution=vi"  --data '[{"groupIds":"9f
7aa9d6-24d6-4611-8d4d-79a52d2bab02"}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

删除一个模型

删除一个现有模型及所有相关的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model/*modelId*

方法

请求类型 DELETE

URL 参数

modelId：*String*。模型标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

```
[{}]
```

成功响应

```
200
[{}]
```

响应项

无

样本调用

```
curl -k -X DELETE -H "APIKEY:8b79658225de5
3488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e53
3dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/model/
106efcb7-1338-4ed2-b8ca-27b45e11e865?user=auto_mm&solution=
vi”  -d  ‘[{}]’
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取所有共享模型

获取所有共享模型。模型管理者有权使用此 API。

URL

/ibm/iotm/service/model/

方法

请求类型 GET

URL 参数

category:'shared'。用于标识模型目录。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
[{"modelId":"0f9c8d47-5ccc-4e52-8e0d-0596e792a2cd",
"groupIds":"e5e83a2d-5814-45b3-af40-74096198cd39","modelName":
"Car Seat Defect Inspection","createdBy":"vimodelmanager@163.com",
"modelType":"objectdetection","description":"This is the object
detection model we build to detect the wrinkle defect for car seat.
The inspection result will identify whether there is wrinkle defect
on the car and localize the defect on the image.","statusStatistics":
```

```
{"deployed":"1"},"dataFormat":"jpg","retrainPolicy":{"scheduler":
"Weekly, Sunday","ratio":"","reuseDayNumber":"","maxPieceNumber":
"","imageNumber":"1000","accuracy":"70"},"trainedBy":
"vimodelmanager@163.com","parameters":{"odRetrainUrl":
"\/user\/VI\/VI\/modellist\/carseatfrcnn_1516957839865.zip",
"lastRetrainData":
"[{\"groupId\":\"e5e83a2d-5814-45b3-af40-74096198cd39\"}]",
"imageSize":"100*100","confidence":"100"},"updatedTime":
"2018-01-26T03:11:05.804-06:00","productType":"carseat"}]
```


响应项

modelId：*String*。模型标识。

groupIds：*String*。模型数据组标识。

modelName：*String*。模型名称。

createdBy：*String*。模型所有者。

modelType：*String*。模型类型。

description：*String*。模型描述。

updatedTime: *Time*。上次更新模型的时间。

productType：*String*。模型产品类型。

StatusStatistics：*String*。模型状态。

dataFormat：*String*。模型数据格式。

retrainPolicy: *JSON Object*。模型重新培训策略。

scheduler: *Time*。模型重新培训日期，值为周一到周日。

imageNumber: *Integer*。可以重新培训编号大于此数字的图像。

accuracy: *Double*。可以重新培训小于此数字的模型精确性值。

parameters: *JSON Object*。包含图像大小和模型置信度信息。

imageSize: *Double * Double*。图像大小。

confidence: *Double*。模型定义的置信度值。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb
8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713
bf305a158fdf485cc6f275f5" " https://9.112.229.91:9447/ibm/
iotm/service/model?category=shared&tenant=Q3T1&solution=
vi&user=testuser"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

模型实例服务

通过模型实例服务，可以执行创建模型实例、返回属于模型的模型实例和返回一个模型实例等任务。

创建模型实例

创建模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance

方法

请求类型 POST

URL 参数

`user`：*`String`*。用于标识用户。必需。

`solution`：*`String`*。用于标识解决方案。必需。

`tenant`：*`String`*。用于标识租户。可选。

头

`Content-type: application/json`。

`APIKEY: encrypted key`。用于认证的 API 密钥。必需。

数据参数

`modelId`：*`String`*。模型标识。

`trainData: JSON Object`。模型培训数据。

`groupId`：*`String`*。数据组标识。

`fileIds`：*`String`*。数据文件标识。

样本主体

```
[
  {
    "modelId": "91a21e63-17e6-4e0e-bbe8-5c6b9d599a35",
    "trainData": [
      {
        "groupId": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02",
        "fileIds": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-057a-4904-8
fad-bbca93abd80b"
      },
      {
        "groupId": "d84c028e-235a-4c13-8a77-894500b8a736",
        "fileIds": "d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b
918-ae1b9f7db894"
      },
      {
        "groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb",
        "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-a
2fa-e74c1578eebe"
      }
    ]
  }
]
```

成功响应

```
200

[
    {
        "status": "deployed",
        "validateData": [
            {
                "groupId": "1a7d4c81-36ff-4a3e-a614-9e996fb380ba"
            }
        ],
        "instanceId": "4cc40ab4-d3b3-4911-b7a0-90281248c075_1542340917709",
        "validateResult": {
            "matrix": [
                [ "","scratch", "Chris"],
                ["mAP","-","-" ],
                ["recall","-","-"]
            ],
            "type": "objectdetection",
            "accuracy": "-",
            "fileCount": "0"
        },
        "trainedBy": "demoadm@cn.ibm.com",
```

```
        "validateTime": "2018-11-15T22:01:57.726-06:00",
        "fileCount": "0",
        "modelId": "4cc40ab4-d3b3-4911-b7a0-90281248c075",
        "createdBy": "demoadm@cn.ibm.com",
        "instanceName": "1",
        "parameters": {
            "edges": [
                {"id": "1542102975873" },
                {"id": "1508476898000"}
            ],
            "modelType": "objectdetection",
            "attached": true,
            "edgesDeploying": []
        },
        "isDeletable": false,
        "modelUrl": "/user/Q3T1/VI/modellist/frcnnmodel_1542340914671.zip",
        "trainData": [
            {"groupId": "1a7d4c81-36ff-4a3e-a614-9e996fb380ba"}
        ],
        "accuracy": "-",
        "updatedTime": "2018-11-16T04:21:34.656-06:00"
    }
]
```


响应项

instanceId: *String*。模型实例标识。

modelId: *String*。模型标识。

instanceName: *Integer*。模型实例名称。

createdBy: *String*。模型实例所有者。

updatedTime: *Time*。上次更新模型实例的时间。

status: *String*。模型实例状态。

trainData: *JSON Object*。模型培训数据。

groupId: *String*。数据组标识。

fileIds: *String*。数据文件标识。

parameters: *JSON Object*。包含有关已部署的 Edge 的信息。

edges: *JSON Array*。包含有关已部署的 Edge 的标识信息。

libModelUrl：*String*。模型库中模型实例的 modelURL。

lastAutoTrainTime: *String*。上次自动重新培训的时间，采用 yyyy-mm-dd hh:mm:ss.SSS 格式。

trainType: *String*。培训类型：online_train、retrain_manual 或 retrain_auto。

onDataPreparing: *String*。1 表示数据正在准备。0 表示数据准备已完成。

snapshot: *String*。用于联机培训的快照信息。

unlabeledGroupId: *String*。未标记的组标识。

样本调用


```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd
34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H "Content-Type:application/
json;charset=UTF-8" " https://9.112.229.91:9447/ibm/iotm/service/modelInstance?user
=auto_mm&solution=vi"  --data '[  {    "modelId": "91a21e63-17e6-4e0e-bbe8-5c6b9d59
9a35",    "trainData": [      {        "groupId": "9f7aa9d6-24d6-4611-8d4d-79a52d2ba
b02",        "fileIds": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-057a-4904-8fa
d-bbca93abd80b"      },      {        "groupId": "d84c028e-235a-4c13-8a77-894500b8a73
6",        "fileIds": "d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b918-a
e1b9f7db894"      },      {        "groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb",
        "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-a2fa-e74c1
```

```
578eebe"      }    ]  }]
'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取属于模型的模型实例

获取属于模型的模型实例。模型管理者和数据研究员有权使用此 API。

URL

/ibm/iotm/service/modelInstance

方法

请求类型 GET

URL 参数

modelId：*String*。模型标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
  {
    "modelId": "91a21e63-17e6-4e0e-bbe8-5c6b9d599a35",
    "createdBy": "auto_mm",
    "status": "draft",
    "instanceName": "1",
    "instanceId": "91a21e63-17e6-4e0e-bbe8-5c6b9d599a35_1512029557869",
    "trainData": [
      {
        "fileIds": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-057a-4904-
8fad-bbca93abd80b",
        "groupId": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"
      },
      {
        "fileIds": "d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-
b918-ae1b9f7db894",
        "groupId": "d84c028e-235a-4c13-8a77-894500b8a736"
      },
      {
        "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-
a2fa-e74c1578eebe",
        "groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"
      }
    ],
    "fileCount": "9",
    "updatedTime": "2017-11-30T16:12:37.869+08:00"
  }
]
```

响应项

modelId：*String*。模型标识。

createdBy：*String*。模型实例所有者。

status：*String*。模型实例状态。

instanceId：*String*。模型实例标识。

instanceName: *Integer*。模型实例名称。

trainData: *JSON Object*。模型培训数据。

groupId：*String*。数据组标识。

fileIds：*String*。数据文件标识。

fileCount: *Integer*。模型图像总计数。

updatedTime: *Time*。上次更新模型实例的时间。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533d
d34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.229.91:9447
/ibm/iotm/service/modelInstance?modelId=91a21e63-17e6-4e0e-bbe8-5c6b9d599a35&user=
auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取一个模型实例

获取一个模型实例。模型管理者和数据研究员有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 GET

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
  {
    "modelId": "91a21e63-17e6-4e0e-bbe8-5c6b9d599a35",
    "createdBy": "auto_mm",
    "status": "draft",
    "instanceName": "1",
    "instanceId": "91a21e63-17e6-4e0e-bbe8-5c6b9d599a35_1512029557869",
    "trainData": [
```

```
      {
        "fileIds": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-057a-
4904-8fad-bbca93abd80b",
        "groupId": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"
      },
      {
        "fileIds": "d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-
4dca-b918-ae1b9f7db894",
        "groupId": "d84c028e-235a-4c13-8a77-894500b8a736"
      },
      {
        "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-
44d8-a2fa-e74c1578eebe",
        "groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"
      }
    ],
    "updatedTime": "2017-11-30T16:12:37.869+08:00"
  }
]
```

响应项

instanceId：*String*。模型实例标识。

modelId：*String*。模型标识。

instanceName: *Integer*。模型实例名称。

createdBy：*String*。模型实例所有者。

updatedTime: *Time*。上次更新模型实例的时间。

status：*String*。模型实例状态。

trainData: *JSON Object*。模型培训数据。

groupId：*String*。数据组标识。

fileIds：*String*。数据文件标识。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c
77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/modelInstance/91a21e63
-17e6-4e0e-bbe8-5c6b9d599a35_1512029557869?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取模型实例验证结果

获取报告的模型实例验证结果。模型管理者和数据研究员有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*/validateResult

方法

请求类型 GET

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "instanceName": "FirstBlood_2017-08-08 19:21:53.888",
        "validateResult": {
            "matrix": [
                [
                    "",
                    "type1",
                    "type2",
                    "type3"
                ],
                [
                    "type1",
                    "77,35,45",
                    "0,0,40",
                    "6,3,50"
                ],
                [
                    "type2",
                    "17,8,45",
                    "100,40,40",
                    "4,2,50"
                ],
                [
                    "type3",
                    "4,2,45",
                    "0,0,40",
                    "90,45,50"
                ],
                [
                    "Total",
                    "45",
                    "40",
                    "50"
                ]
            ],
            "accuracy": "84.56"
        },
        "validateTime": "2017-08-09T22:58:46.416+08:00",
        "fileCount": "200"
    }
]
```

响应项

instanceName: *Integer*。模型实例名称。

validateResults: *JSON Object* 。模型实例验证结果。

validateTime: *Time*。模型实例验证时间。

fileCount: *Integer*。模型实例图像数量。

accuracy: *Double*。模型实例精确性值。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e53
3dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.229.91:9447/
ibm/iotm/service/modelInstance/91a21e63-17e6-4e0e-bbe8-5c6b9d599a35_1512029557869/
validateResult?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

检查结果服务

通过检查结果服务，可以执行获取检查结果列表、获取检查结果详细信息和获取检查结果工作站概述等任务。

获取检查结果列表

获取检查结果列表。检验员和主管有权使用此 API。

URL

/ibm/iotm/service/inspectResult

方法

请求类型 GET

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

isConfirmed: *Integer*。0 表示检查结果未确认。1 表示检查结果已确认。2 表示检查结果未知。可选。

cell：*String*。用于标识工作站。可选。

inspectTime: *Time*。用于标识检查的时间。

reverse: *Boolean*。用于标识检查结果列表的顺序。

pageCount: *Integer*。页面计数。

sampleRate: *Integer*。样本检查率。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[{"id":"plant1,line1,cell1_1511768220935_AUTO|auto_1511768206483.jpg",
"isConfirmed":"0","inspectFileUrl":"imageserver\/911222954\/\/AUTO|
auto_plant1_line1_cell1_1511768206483.jpg","isUpdated":"0",
"defectTypeCount":"1","cell":"plant1,line1,cell1","instanceId":"ebf9
4e98-958c-48b0-8377-826241ffccd8_1511765289053","defectCount":"1",
"confidence":"99.87","inspectId":"auto_1511768206483.jpg",
"inspectTime":"1511768220935"},{"id":"plant1,line1,
cell1_1511761780903_AUTO|auto_1511761766470.jpg","isConfirmed":"0","
inspectFileUrl":"imageserver\/911222954\/\/AUTO|
auto_plant1_line1_cell1_1511761766470.jpg","isUpdated":"0",
"defectTypeCount":"1","cell":"plant1,line1,cell1","instanceId":
"9ab990d6-465a-48b0-838c-416c8e4a9879_1511761328538","defectCount":
"1","confidence":"99.79","inspectId":"auto_1511761766470.jpg",
"inspectTime":"1511761780903"},
{"id":"plant1,line1,cell1_1511761559536_AUTO|auto_1511761541254.jpg",
"isConfirmed":"0","i
```

```
nspectFileUrl":"imageserver\/911222954\/\/AUTO|
auto_plant1_line1_cell1_1511761541254.jpg","isUpdated":"0",
"defectTypeCount":"1","cell":"plant1,line1,cell1","instanceId":
"9ab990d6-465a-48b0-838c-416c8e4a9879_1511761328538","defectCount":
"1","confidence":"99.87","inspectId":"auto_1511761541254.jpg",
"inspectTime":"1511761559536"},{"id":"plant1,line1,
cell1_1511421414687_AUTO|auto_1511421410914.jpg","isConfirmed":"0",
"inspectFileUrl":"imageserver\/911222954\/\/AUTO|
auto_plant1_line1_cell1_1511421410914.jpg","isUpdated":"0",
"defectTypeCount":"1","cell":"plant1,line1,cell1","instanceId":
"ae509ec0-e1d6-4827-afd1-a8af598eb28c_1511419442550",
"defectCount":"1","confidence":"99.99","inspectId":
"auto_1511421410914.jpg","inspectTime":"1511421414687"}]
```


响应项

id：*String*。检查标识。

isConfirmed: *Integer*。0 表示检查结果未确认。1 表示检查结果已确认。2 表示检查结果未知。

cell：*String*。工作站信息。

inspectfileUrl：*String*。检查文件 URL。

defectTypeCount: *Integer*。缺陷类型数量。

defectCount: *Integer*。缺陷数量。

confidence: *Double*。缺陷的置信度级别。

inspectId：*String*。检查标识。

inspectTime: *Time*。检查时间。

样本调用

```
curl -k -H
"APIKEY: 8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579
e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.22
9.91:9447/ibm/iotm/service/inspectResult?user=auto_inspector&solution=vi&isCon
firmed=0&cell=plant1,line1,cell1&sampleRate=10&pageCount=1&reverse=true"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取检查结果详细信息

获取检查结果详细信息。检验员和主管有权使用此 API。

URL

/ibm/iotm/service/inspectResult/*inspectResultId*

方法

请求类型 GET

URL 参数

inspectResultId: *String*。用于标识检查标识。

user: *String*。用于标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "isConfirmed": "0",
        "inspectFileUrl": "imageserver/edge1/image1.jpg",
        "isUpdated": "0",
        "cell": "plant1,line1,cell1",
        "instanceId": "6e4bfc6b-0ddb-4c85-81e7-0ea789681ae8_775e39b9-fedd-4418-a5b3
-70de6b86fbec",
        "inspectResult": [
            {
                "position": {
                    "height": 331,
                    "width": 467,
                    "y": 1653,
                    "x": 897
                },
                "probableTypes": [
                    {
                        "confidence": 100,
                        "type": "NoDefect"
                    },
                    {
                        "confidence": 0,
                        "type": "Defect"
                    },
                    {
                        "confidence": 0,
                        "type": "MisT"
                    }
                ]
            }
        ]
    }
]
```

响应项

isConfirmed: *Integer*。0 表示检查结果未确认。1 表示检查结果已确认。2 表示检查结果未知。

inspectfileUrl: *String*。检查文件 URL。

inspectId: *String*。检查标识。

inspectResult: *JSON Object*。检查结果。

position: *JSON Array*。检查的高度、宽度以及 x 和 y 坐标信息。

probableTypes: *JSON Object*。每种缺陷类型及其置信度级别。

样本调用

```
curl -k -H
"APIKEY: 8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2
c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/service/inspectResult/plant1,line1,
cell1_1511166988402_AUTO|auto_1511166972283.jpg?user=auto_inspector&
solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取检查结果工作站概述

获取检查结果工作站概述。检验员和主管有权使用此 API。

URL

/ibm/iotm/service/inspectResultCell

方法

请求类型 GET

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

无

成功响应

```
200
[
    {
        "confirmed": 1,
        "cell": "plant1,line1,cell1",
        "unknown": 0,
        "unconfirmed": 10
    }
]
```

响应项

confirmed: *Integer*。确认的检验员编号。

unconfirmed: *Integer*。未确认的检验员编号。

unknown: *Integer* 。未知检验员。

cell: *Strong*。图像工作站。

样本调用

```
curl -k -H
"APIKEY: 8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c7
7d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" "
https://9.112.229.91:9447/ibm/iotm/service/inspectResultCell?user=auto_
inspector&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

确认检查结果

确认检查结果。检验员和主管有权使用此 API。

URL

/ibm/iotm/service/inspectResult/*inspectResultId*

方法

请求类型 PUT

URL 参数

inspectResultId: *String*。用于标识检验员标识。

user: *String*。用于标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

position: *JSON Array*。检查的高度、宽度以及 x 和 y 坐标信息。

probableTypes: *JSON Object*。每种缺陷类型及其置信度级别。

logs: *JSON Object*。包含确认操作、操作员和更新时间。

description: *String*。确认检验员描述。

样本主体

```
[
    {
        "confirmedResult": [
            {
                "position": {
                    "height": 3231,
                    "width": 467,
                    "y": 1653,
                    "x": 897
                },
                "probableTypes": [
                    {
                        "confidence": 100,
                        "type": "NoDefect"
                    },
                    {
                        "confidence": 0,
                        "type": "Defect"
                    },
                    {
                        "confidence": 0,
                        "type": "MisT"
                    }
                ],
                "logs":[{"action":"confirm",
                "operator":"inspector1",
                "updateTime":"1503657054321"}]
            }
        ],
        "description":"VI result is OK"
    }
]
```

成功响应

```
200
[{}]
```

响应项

无

样本调用

```
curl -k -H"APIKEY: 8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8
c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H
"Content-Type:application/json;charset=UTF-8" -X PUT "https://9.112.229.91:9447/ibm
/iotm/service/inspectResult/plant1,line1,cell1_1511507025284_AUTO|auto_1511507012702
.jpg?user=auto_inspector&solution=vi"  --data ' [    {  "confirmedResult": [  {
"position": {  "height": 430,    "width": 552,  "x": 1254,          "y": 2123  },
"probableTypes": [  {  "confidence": 99.87,   "type": "Defect"   },     {
"confidence": 0.1,    "type": "MisT"  },   {   "confidence": 0.03,    "type":"NoDefect"
```

```
   }      ],    "logs":[{"action":"confirm",   "operator":"auto_super",
"updateTime":"1503657054321"}]   } ],   "description":"VI result is OK"  }]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

删除检查结果

删除检查结果。检验员和主管有权使用此 API。

URL

/ibm/iotm/service/inspectResult

方法

请求类型 DELETE

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

cell：*String*。用于标识工作站。必需。

inspectTime: *Time*。用于标识检查的时间。将删除在此时间之前发生的检查结果。可选。

instanceId：*String*。用于标识实例标识。将删除此实例标识生成的检查结果。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

无。

成功响应

```
200
[{}]
```

响应项

无。

样本调用

```
curl -k -H "APIKEY: 8b79658225de53488321fb
7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd34
ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
-H "Content-Type:application/json;charset=UTF-8" -X DELETE
"https:// 9.112.229.91:9447/ibm/iotm/service/inspectResult?instanceId=
04b61494-99ec-49f4-9a0f-0125650c40dd_1520316716077&
tenant=Q3T1&solution=vi&user=demoadm@cn.ibm.com&cell=plant2,line2,cell2&inspectTime=
1520324778324"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

模型实例操作服务

通过模型实例操作服务，可以执行提交模型实例、附加模型文件和验证模型实例等任务。

培训模型实例

使用深度学习模型来培训模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

action：*String*。操作。设置为 train。必需。

user：*String*。标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

```
 [{}]
```

成功响应

```
200

[{"status":"waiting","instanceId":
"1e760614-ab0e-4c2a-be2e-435505087fe4_1518060853002",
"parameters":{"modelType":"classification","trainType":
"online_train"}}]
```

响应项

status：*String*。模型实例的重新培训的状态。

instanceId：*String*。模型实例标识。

updatedTime: *Time*。上次更新模型实例的时间。

parameters: *JSON object*。有关模型类型和培训类型的信息。

样本调用

```
curl -k -X PUT -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5af
b8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc37
13bf305a158fdf485cc6f275f5" -H "Content-Type:
application/json;charset=UTF-8"
"https://int_iotm.predictivesolutionsapps.ibmcloud.com/
ibm/iotm/service/modelInstance/1e760614-ab0e-4c2a-be2e-
435505087fe4_1518060853002?action=train&tenant=Q3T1&
```

```
solution=vi&user=admin@example.com
" --data ' [{}]'
```

注释

此服务准备图像并执行联机培训。可以使用“获取一个模型实例”API 调用来检查模型实例的状态。

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

下载培训日志文件

下载模型培训的日志文件。模型管理者有权使用此 API。

URL

/ibm/iotm/vi/service/logFile

方法

请求类型 GET

URL 参数

instanceId: *String*。模型实例标识。必需。

user: *String*。标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

无。

成功响应

日志文件的二进制内容。

响应项

无。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d
4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.91:9447/ibm/iotm/vi/service/logFile?instanceId=06c87
0e5-1701-419a-8a84-0f40ee82fe33_11260029297&user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

验证模型实例

验证培训的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

action：*String*。操作。设置为 validate。必需。

user：*String*。标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

Content-Disposition: form-data; name="files[]"; filename="model.zip"。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

validateData: *JSON object*。模型验证数据。

groupId：*String*。数据组标识。

fileIds：*String*。数据文件标识。

样本主体

```
[{"validateData": [    {"fileIds": "9f7aa9d6-24d6-4611-8d4d-
79a52d2bab02_76e8b2cb-057a-4904-8fad-bbca93abd80b",    "groupId":
"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"  },      { "fileIds":
"d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b918-ae1b9f7db894",
"groupId": "d84c028e-235a-4c13-8a77-894500b8a736" },
{  "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-a2fa-
e74c1578eebe",   "groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"   }  ]   }]
```

成功响应

```
200
[{"validateData":[{"fileIds":"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02_76e8b2cb-
057a-4904-8fad-bbca93abd80b","groupId":"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"},
{"fileIds":"d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b918-
ae1b9f7db894","groupId":"d84c028e-235a-4c13-8a77-894500b8a736"},
{"fileIds":"c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-a2fa-
e74c1578eebe","groupId":"c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"}],"status":
"validating","instanceId":"ef12294e-2d09-4acd-9946-00a09bdeeba2_1512092239371"}]
```

响应项

validateData: *JSON Object*。模型验证数据。

groupId：*String*。数据组标识。

fileIds：*String*。数据文件标识。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488321fb7bb657f9f161ac
d2ea830a5afb8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf30
5a158fdf485cc6f275f5" -H "Content-Type:application/json;charset=UTF-8" -X PUT
"https://9.112.229.91:9447/ibm/iotm/service/modelInstance/ef12294e-2d09-
4acd-9946-00a09bdeeba2_1512092239371?action=validate&user=auto_mm&solution=vi"
--data ' [{"validateData": [    {"fileIds": "9f7aa9d6-24d6-4611-8d4d-
79a52d2bab02_76e8b2cb-057a-4904-8fad-bbca93abd80b",    "groupId":
"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"  },      { "fileIds":
"d84c028e-235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b918-ae1b9f7db894",
"groupId": "d84c028e-235a-4c13-8a77-894500b8a736" },
{  "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_
bd0c8246-7208-44d8-a2fa-e74c1578eebe",
"groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"   }  ]   }]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

拒绝模型实例

拒绝已验证的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId: *String*。模型实例标识。必需。

action: *String*。操作。设置为 reject。必需。

user: *String*。标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

```
[{}]
```

成功响应

```
200
[
{
"status": "rejected",
"instanceId": "e23b6796-4f4a-4c5d-8fb7-fa3f729f591c_1510812725969"
}
]
```

响应项

status：*String*。模型实例状态。

instanceId：*String*。模型实例标识。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488321fb7bb657f9f
161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886
fc3713bf305a158fdf485cc6f275f5" -H "Content-Type:application/json;
charset=UTF-8" -X PUT "https://9.112.229.91:9447/ibm/iotm/service/
modelInstance/ef12294e-2d09-4acd-9946-00a09bdeeba2_1512092239371?
action=reject&user=auto_mm&solution=vi"  --data ' [{}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

部署模型实例

部署已接受的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId: *String*。模型实例标识。必需。

action: *String*。操作。设置为 publish。必需。

user: *String*。标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

Parameters: *JSON Object*。将部署模型的 Edge 的标识。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

```
[{"parameters": {"edges":[{"id":"1508476898000"},{"id":"1542102975873"},...]}}]
```

成功响应

```
[{"status":"deployed","instanceId":"96b651a9-2ba0-4392-
aca4-1463320a19c2_1542354262146"}]
```

响应项

status：*String*。模型实例状态。

instanceId：*String*。模型实例标识。

parameters: *JSON Object*。Edge 的名称和 IP 地址。

modelType：*String*。模型类型。

样本调用

```
curl -k -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab
2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
-H "Content-Type:application/json;charset=UTF-8" -X PUT
"https://9.112.229.91:9447/ibm/iotm/service/modelInstance/ef12294e-
2d09-4acd-9946-00a09bdeeba2_1512092239371?action=publish&user=
auto_mm&solution=vi" --data '[{ “parameters": {"edges":[{"id":
"1508476898000"},{"id":"1542102975873"},...]}}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

重新培训模型实例

重新培训部署的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

action：*String*。操作。设置为 retrain。必需。

user：*String*。标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

retrainData: *JSON_object*。模型的重新培训数据。

groupId：*String*。数据组标识。

fileIds：*String*。数据文件标识。

样本主体

```
 [{"trainData": [    {"fileIds": "9f7aa9d6-24d6-4611-
8d4d-79a52d2bab02_76e8b2cb-057a-4904-8fad-bbca93abd80b",
"groupId": "9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"  },
{ "fileIds": "d84c028e-235a-4c13-8a77-894500b8a736_
fe201c83-b138-4dca-b918-ae1b9f7db894",
"groupId": "d84c028e-235a-4c13-8a77-894500b8a736" },
{  "fileIds": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_
bd0c8246-7208-44d8-a2fa-e74c1578eebe",
"groupId": "c5f11ed9-db49-403a-972c-cd6cb1d0f8eb"
   }  ]   }]
```

成功响应

```
200
[{"modelId":"ef12294e-2d09-4acd-9946-00a09bdeeba2","status":
"waiting","instanceId":"ef12294e-2d09-4acd-9946-00a09bdeeba2_
1512095854903","trainData":[{"fileIds":"9f7aa9d6-24d6-4611-
8d4d-79a52d2bab02_76e8b2cb-057a-4904-8fad-bbca93abd80b",
"groupId":"9f7aa9d6-24d6-4611-8d4d-79a52d2bab02"},
{"fileIds":"d84c028e-235a-4c13-8a77-894500b8a736_
fe201c83-b138-4dca-b918-ae1b9f7db894","groupId":
"d84c028e-235a-4c13-8a77-894500b8a736"},{"fileIds":
"c5f11ed9-db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-
44d8-a2fa-e74c1578eebe","groupId":"c5f11ed9-db49-403a-
972c-cd6cb1d0f8eb"}]}]
```

响应项

modelId：*String*。模型标识。

status：*String*。模型实例的重新培训的状态。

instanceId：*String*。模型实例标识。

trainData: *JSON Object*。模型培训数据。

groupId：*String*。数据组的标识。

fileIds：*String*。数据文件的标识。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488321fb
7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd
34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H
"Content-Type:application/json;charset=UTF-8" -X PUT
```

```
"https://9.112.229.91:9447/ibm/iotm/service/modelInstance/
ef12294e-2d09-4acd-9946-00a09bdeeba2_1512092239371?
action=retrain&user=auto_mm&solution=vi"  --data
'[{"trainData": [    {"fileIds": "9f7aa9d6-24d6-
4611-8d4d-79a52d2bab02_76e8b2cb-057a-4904-8fad-
bbca93abd80b",    "groupId": "9f7aa9d6-24d6-4611-
8d4d-79a52d2bab02"  },      { "fileIds": "d84c028e-
235a-4c13-8a77-894500b8a736_fe201c83-b138-4dca-b918-
ae1b9f7db894",         "groupId": "d84c028e-235a-4c13-
8a77-894500b8a736" },      {  "fileIds": "c5f11ed9-
db49-403a-972c-cd6cb1d0f8eb_bd0c8246-7208-44d8-a2fa-
e74c1578eebe",   "groupId": "c5f11ed9-db49-403a-972c-
cd6cb1d0f8eb"   }  ]   }]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

取消部署模型实例

取消部署所部署的模型实例。模型管理者有权使用此 API。

URL

/ibm/iotm/service/modelInstance/*modelInstanceId*

方法

请求类型 PUT

URL 参数

modelInstanceId：*String*。模型实例标识。必需。

action：*String*。操作。设置为 undeploy。必需。

user：*String*。标识用户。必需。

solution：*String*。用于标识解决方案。必需。

tenant：*String*。用于标识租户。可选。

Parameters:*JSON Object*。将从中取消模型实例部署的 Edge 的标识。

头

Content-type: application/json。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

```
[{"parameters": {"edges":[{"id":"1508476898000"}]}}]
```

成功响应

```
200
[{"status":"undeployed","instanceId":"ef12294e-2d09-4acd-9946-
00a09bdeeba2_1512092239371"}]
```

响应项

status：*String*。模型实例的重新培训的状态。

instanceId：*String*。模型实例标识。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830
a5afb8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6
f275f5" -H "Content-Type:application/json;charset=UTF-8"
-X PUT "https://9.112.229.91:9447/ibm/iotm/service/modelInstance/ef12294e-2d09-
```

```
4acd-9946-00a09bdeeba2_1512092239371?action=undeploy&user=auto_mm&solution=vi"
--data '[{ “parameters": {"edges":[{"id":"1508476898000"}]}}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

Edge 服务

通过 Edge 服务，可以执行创建 Edge、获取 Edge 和更新 Edge 等任务。

创建 Edge

创建 Edge。模型管理者有权使用此 API。

URL

/ibm/iotm/vi/service/edge

方法

请求类型 POST

URL 参数

`user`: *`String`*。标识用户。必需。

`solution`: *`String`*。用于标识解决方案。必需。

`tenant`: *`String`*。用于标识租户。可选。

头

`Content-type`: `application/json`。

`APIKEY`: *`encrypted key`*。用于认证的 API 密钥。必需。

数据参数

`name`: *`String`*。Edge 的名称。必需。

`ip`: *`IP`*。Edge 的 IP 地址。必需。

`port`: *`integer`*。Edge 的端口。必需。

`userName`: *`String`*。Edge 虚拟机的用户名。必需。

`passWord`: *`String`*。Edge 虚拟机的密码。必需。

`clusterType`: *`String`*。Edge 的集群类型。指定 `slave` 或 `master`。必需。

`online`: *`String`*。指示 Edge 是否处于联机状态。如果 Edge 处于联机状态，请指定 `true`；如果 Edge 处于脱机状态，请指定 `false`。必需。

样本主体

```
[{"name":"edge_auto","ip":"9.112.229.54",
"port":"22","userName":"root","passWord":"U2FtcGxlQDY2NiE=",
"clusterType":"mater","online":"true"}]
```

成功响应

```
 [{"port":"22","clusterType":"master","deployPath":"\/home\/pmqopsadmin","hierarchyList":
["plant
1,line1,cell1","plant2,line2,cell2"],"servicePass":"passw0rd@","deletePolicy":
{"schedule":false,"rate
":0.2,"onDay":7,"afterDay":"1","frequency":"weekly"},"version":"2018Q4.01","ip":"viedge1","o
nline
":"true","id":"1508476898000","upgradeFlag":"false","createdBy":"demoadm@cn.ibm.com","servic
eUser":
```

```
"admin","name":"master","sshUser":"pmqopsadmin","updatedTime":"2018-11-16T03:54:13.382-06:00
"}]
```

响应项

id: *String*。Edge 的标识。

deletePolicy: *JSON Object*。用于删除 Edge 上的检查结果的策略。

version: *String*。Edge 版本信息。

upgradeFlag: *String*。指示 Edge 是否需要升级。

样本调用

```
curl -k -H "APIKEY:8b79658225d
e53488321fb7bb657f9f161acd2ea830a5afb8c149
f6aab2c77d4f0579e533dd34ggb959317ac69ff
73f886fc3713bf305a158fdf485cc6f275f5" -H
"Content-Type:application/json;charset=UTF-8"
"https://9.112.229.91:9447/ibm/iotm/vi/service/edge?
user=auto_mm&solution=vi "  --data '[{"name":
"edge_auto","ip":"9.112.229.54","port":"22",
"userName":"root","passWord":"U2FtcGxlQDY2NiE=",
"clusterType":"master","online":"true"}]'
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

获取 Edge

获取 Edge。模型管理者有权使用此 API。

URL

/ibm/iotm/vi/service/edge

方法

请求类型 GET

URL 参数

user: *String*。标识用户。必需。

solution: *String*。用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

hierarchyList:。工作站层次结构信息，例如 plant1, line1, cell1。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

无。

成功响应

```
200
[{"port":"22","id":"1512098793912","clusterType":"slave","hierarchyList":
["plant1, line1, cell1"],"createdBy":"auto_mm","description":"test edge",
"name":"edge_auto","updatedTime":"2017-12-01T10:58:26.559+08:00","online":
"false","ip":"9.112.229.54"}]
```

响应项

name: *String*。Edge 的名称。

ip: *IP*。Edge 的 IP 地址。

port: *Integer*。Edge 的端口。

clusterType: *String*。Edge 的集群类型。指定 slave 或 master。

online: *String*。Edge 的联机状态：true 表示 Edge 处于联机状态，false 表示 Edge 处于脱机状态。

hierarchyList: *String*。Edge 的层次结构。

description: *String*。Edge 的描述。

createdBy: *String*。Edge 的所有者。

updatedTime: *Time*。上次更新 Edge 的时间。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488321fb7bb657
f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff
73f886fc3713bf305a158fdf485cc6f275f5" "https://9.112.229.91:9447/ibm/iotm/vi/
service/edge?user=auto_mm&solution=vi"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

删除 Edge

删除 Edge。模型管理者有权使用此 API。

URL

/ibm/iotm/vi/service/edge/*edgeId*

方法

请求类型 DELETE

URL 参数

edgeId: 。Edge 标识。必需。

user: *String*。标识用户。必需。

solution: *String*. 用于标识解决方案。必需。

tenant: *String*。用于标识租户。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

[{}]

成功响应

```
200
[{}]
```

响应项

无。

样本调用

```
curl -k -X DELETE -H "APIKEY:8b7965
8225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab
2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a
```

```
158fdf485cc6f275f5" "https://9.112.229.91:9447/ibm/iotm/
vi/service/edge/1512098793912?user=auto_mm&solution=
vi”  -d  ‘[{}]’
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

升级 Edge

升级 Edge。有权创建 Edge 的用户有权使用此 API。

要升级 Edge，您必须使用 API 调用以获取所需的 Edge 文件，包括 `edgeDeployed.sh` 和 `edge-vi.zip`。

要升级 Edge Side 上的 Edge，创建 Edge 时指定的 SSH 用户必须在 Edge 系统上执行 shell 脚本。

URL

/ibm/iotm/vi/service/edgeFile

方法

请求类型 GET

URL 参数

`edgeID`：Edge 的标识。必需。

`user`：*`String`*。标识用户。必需。

`solution`：*`String`*。用于标识解决方案。必需。

`tenant`：*`String`*。用于标识租户。可选。

`Version`：*`String`*。该值可以为 `shell`（针对 shell 脚本下载）或 `ubuntu`、`redhat` 或 `power`（针对 Edge 系统上的 Linux 类型）。

头

`APIKEY: encrypted key`。用于认证的 API 密钥。必需。

数据参数

无

成功响应

```
200
```

`edgeDeployed.sh` 文件或 `vi_edge-bin_vi.zip` 文件。

响应项

无

样本调用

```
curl -k -X GET -H "APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb
8c149f6aab2c77d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://int_iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/vi/service/edgeFile?
tenant=Q3T1&solution=vi&user=modelmanager1&version=shell&edgeId=1508476898000”
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

评分服务

通过评分服务，可以对图像评分。

对图像评分

对图像评分。检验员和主管有权使用此 API。

URL

/ibm/iotm/vi/service/uploadScoreImage

方法

请求类型 POST

URL 参数

`productType:`*`String`*。模型的产品类型。必需。

`cell:`*`String`*。图像的工作站。必需。

`user:`*`String`*。标识用户。必需。

`solution:`*`String`*。用于标识解决方案。必需。

`tenant:`*`String`*。用于标识租户。可选。

头

`Content-Type: application/binary`

`APIKEY:` *`encrypted key`*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

图像文件的二进制内容。

成功响应

```
200
 //for object detection example:
{
  "detections": [
    {
      "position": {
        "height": 60,
        "width": 253,
        "x": 1594,
        "y": 773
      },
      "probableTypes": [
        {
          "confidence": 99.798703193664551,
          "type": "scratch"
        }
      ],
      "properties": []
    }
  ],
  "timestamp": "2017-09-28 10:32:13.528415"
}

//for classification example:
{
  "Timestamp": "2017-09-12 18:43:52",
  "detections": [
    {
      "position": {
        "height": 436,
        "width": 537,
        "x": 1574,
        "y": 1588
```

```
      },
      "probableTypes": [
        {
          "confidence": 100.0,
          "type": "Defect"
        },
        {
          "confidence": 0.0,
          "type": "MisT"
        },
        {
          "confidence": 0.0,
          "type": "NoDefect"
        }
      ]
    }
  ]
}
```

响应项

`position`: *JSON Array*。检查高度、宽度以及 x 和 y 坐标的信息。

`probableTypes`: *JSON Object*。每个缺陷的类型和置信度级别。

`Timestamp`: *Time*。对图像评分的时间。

样本调用

```
curl -k -X POST -H "APIKEY:8b79658225de53488321
fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533dd34ggb9593
17ac69ff73f886fc3713bf305a158fdf485cc6f275f5"  -H
"Content-Type: application/binary" "https://9.112.229.91:9447/
ibm/iotm/vi/service/uploadScoreImage?user=auto_super&productType=
auto&cell=plant1_line1_cell1&solution=vi"  --connect-timeout 600
--data-binary @C:\\code\\Automation_98\\API\\vi\\VI_API\\file\\test.JPG
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

QEWS 集成服务

通过 QEWS 集成服务，可以获取图像缺陷率文件。

获取图像缺陷率文件

获取包含图像缺陷率相关数据的文件。主管有权使用此 API。

URL

/ibm/iotm/service/defectImageRateFile

方法

请求类型 GET

URL 参数

`user`: *String*。标识用户。必需。

`solution`: *String*。用于标识解决方案。必需。

`tenant`: *String*。用于标识租户。可选。

`date from`。开始日期，格式为 yyyy-MM-dd 或 yyyy-MM-dd HH:mm:ss。可选。

`date to`。结束日期，格式为 yyyy-MM-dd 或 yyyy-MM-dd HH:mm:ss。可选。

`cell name`。工作站的名称。可选。

`interval`。图像缺陷率的时间间隔。指定 `daily`、`hourly` 或 `monthly`。可选。

头

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无。

样本主体

无。

成功响应

200。数据文件的二进制内容。数据文件是逗号分隔值 (CSV) 文件。

响应项

ATTRIBUTE_NAME：VI_DEFECT_IMAGE_RATE，固定值。

DATE：当 interval 为 daily 和 monthly 时，为 yyyy-MM-dd 格式的日期；当 interval 为 hourly 时，为 yyyy-MM-dd HH:mm:ss 格式的日期。

CELL_NAME：工作站的名称。在此响应项中，逗号字符 (,) 会替换为下划线字符 (_)，因为不支持逗号。

样本调用

```
curl -k -H "APIKEY:8b79658225de53488
321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533
dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.89:9447/ibm/iotm/service/
defectImageRateFile?user=supervisor1&solution=
vi&tenant=T1" --output "C:\Users\IBM_ADMIN\Downloads\a.csv"
```

```
curl -k -H "APIKEY:8b79658225de53488
321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d4f0579e533
dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"
"https://9.112.229.89:9447/ibm/iotm/service/
defectImageRateFile?user=supervisor1&solution=
vi&tenant=T1&dateFrom=2017-10-23%2011:00:00&dateTo=
2017-11-14%2018:40:00&cellName=plant1,line1,cell1&
interval=hourly" --output "C:\Users\IBM_ADMIN\
Downloads\cell1_hourly.csv"
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

组合服务

通过组合服务，可以获取包含已培训模型的压缩文件，然后将该模型注册到 Maximo PQI SaaS Visual Insights。

注册模型

获取包含已培训模型的压缩文件，然后将该模型注册到 Maximo PQI SaaS Visual Insights。模型管理者有权使用此 API。

URL

/ibm/iotm/vi/service/registerModel

方法

请求类型 POST

URL 参数

user：*String*。用于标识用户。必需。

solution：*String*。用于标识解决方案。必需。

name：*String*。模型名称。

productType：*String*。与模型关联的产品类型。

头

Content-Type: multipart/form-data。

Content-Disposition: form-data; name="files[]"; filename="model.zip"。

APIKEY: *encrypted key*。用于认证的 API 密钥。必需。

数据参数

无

样本主体

模型文件的二进制内容。

成功响应

```
200

{"errorMessage":"","modelId":"3914ad2c-3b7b-47ad-974c-e5b6bc2075ef",
"createdBy":"modelmanager1","status":"validated","instanceId":
"3914ad2c-3b7b-47ad-974c-e5b6bc2075ef_1517796906688",
"updatedTime":"2018-02-05 10:15:06.608"}
```

响应项

modelId：*String*。模型标识。

createdBy：*String*。模型所有者。

status：*String*。模型实例的状态。

instanceId：*String*。模型实例的标识。

updatedTime: *Time*。上次更新模型实例的时间。

样本调用

```
curl -k -X POST -H
"APIKEY:8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77
d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5" -H
"Content-Type:multipart/form-data" -H "Content-Disposition: form-data;
name="files[]"; filename="walkermodel1.zip""
"https://int_iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/vi/
service/registerModel?user=demoadm@cn.ibm.com&solution=vi&name=
samplename&productType=sampletype" --connect-timeout 6000 -F
file=@walkermodel.zip
```

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

独立 Edge 服务

通过独立 Edge 服务，您可以执行获取可用模型、部署模型、上载图像分数、同步检查结果以及清理检查结果等任务。

独立 Edge 服务在独立 Edge 机器上运行，而不是在 Maximo PQI SaaS Visual Insights 中心上运行。您必须使用不同的主机、端口和用户凭证。

获取可用模型

获取所有可用模型。独立 Edge 管理员有权使用此 API。

URL

https://<edge machine host>:8449/api/getAvailableModels

方法

请求类型 GET

URL 参数

无。

头

Authorization: *user_name*/*password*

数据参数

无。

样本主体

无。

成功响应

```
{"model_list": [
        {
            "coviacenter": {
                "apikey": "8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77d

4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5",
                "credential": "ZGVtb2FkbUBjbi5pYm0uY29tOlEzVDE=",
                "url": "https://int_iotm.predictivesolutionsapps.ibmcloud.com/ibm/iotm/
service
                 /apiWrapper?apiName=addInspectResult",
                "username": "demoadm@cn.ibm.com"
            },
            "groupIds": "0c4f11d7-b255-4c05-9c87-694e9f912fa9",
            "model_id": "049dabe5-d865-4f09-862f-fb7430afef6d",
            "model_instance_id": "049dabe5-d865-4f09-862f-fb7430afef6d_1518401501376",
            "model_type": "classification",
            "model_url": "049dabe5-d865-4f09-862f-fb7430afef6d_1518401599389.zip",
            "product_type": "Q3T1|jianyuan45",
            "status": "deployed"
        },
        ......
]}
```

响应项

model_list: *String*。已发布模型的列表。

model_id: *String*。已发布模型的模型标识。

model_instance_id: *String*。已发布模型的模型实例标识。

model_type: *String*。已发布模型的模型类型。

model_url: *String*。已发布模型的模型 URL。

product_type: *String*。已发布模型的产品类型。

coviacenter: *JSONObject*。Edge 的必需中心信息。

样本调用

```
curl -k -H 'Authorization: Basic YWRtaW46cGFzc3cwcmRA' -X GET https://{localhost}
:8449/api/getAvailableModel
```

部署一个模型

部署一个模型。独立 Edge 管理员有权使用此 API。

URL

https://<edge machine host>:8449/api/deployModel

方法

请求类型 POST

URL 参数

无。

头

`Authorization: user_name/password`

数据参数

`model_list`: *String*。已发布模型的列表。

`model_id`: *String*。已发布模型的模型标识。

`model_instance_id`: *String*。已发布模型的模型实例标识。

`model_type`: *String*。已发布模型的模型类型。

`model_url`: *String*。已发布模型的模型 URL。

`product_type`: *String*。已发布模型的产品类型。

`coviacenter`: *JSONObject*。Edge 的必需中心信息。

`edgeId`: *String*。当前 Edge 的标识。

样本主体

```
{"model_id": "56608a3d-df64-4ef0-85a6-f88778cf583b","model_url": "56608a3d
-df64-4ef0-85a6-f88778cf583b_1540279487176.zip","edgeId": "1540275842203",
"coviacenter": {"url": "https://int_iotm.predictivesolutionsapps.ibmcloud.com/
ibm/iotm/service/apiWrapper?apiName=addInspectResult","credential":
"c3VwZXJ2aXNvcjE6UTNUMQ==","username": "supervisor1","apikey":
"8b79658225de53488321fb7bb657f9f161acd2ea830a5afb8c149f6aab2c77
d4f0579e533dd34ggb959317ac69ff73f886fc3713bf305a158fdf485cc6f275f5"},"groupIds":
"b18a8d56-7a4c-4266-9280-49258c3fd798","model_type": "classification",
"product_type": "Q3T1|qwtest12","model_instance_id": "56608a3d-df64
-4ef0-85a6-f88778cf583b_1540279416752","model_name": "qwtest12"}
```

成功响应

```
{ “status” : “success” }
```

响应项

`Status`:*String*。success 或 failed。

样本调用

```
curl -k -H 'Authorization: Basic YWRtaW46cGFzc3cwcmRA'
-H "Content-Type:application/json;charset=UTF-8" -X POST --data
'{"model_id": "56608a3d-df64-4ef0-85a6-f88778cf583b", "model_url":
"56608a3d-df64-4ef0-85a6-f88778cf583b_1540279487176.zip", "edgeId":
"1540275842203", "coviacenter": { "url": "https://int_iotm.
predictivesolutionsapps.ibmcloud.com/ibm/iotm/service/apiWrapper?
apiName=addInspectResult", "credential": "c3VwZXJ2aXNvcjE6UTNUMQ==",
"username": "supervisor1", "apikey": "8b79658225de53488321fb7bb657f9f161
acd2ea830a5afb8c149f6aab2c77d4f0579e533dd34ggb959317ac
69ff73f886fc3713bf305a158fdf485cc6f275f5" }, "groupIds": "b18a8d56-7a4c-
4266-9280-49258c3fd798", "model_type": "classification", "product_type":
"Q3T1|qwtest12", "model_instance_id": "56608a3d-df64-4ef0-85a6-f88778cf5
83b_1540279416752", "model_name": "qwtest12"} '   https://{localhost}:
8449/api/deployModel
```

取消部署模型

取消部署模型。独立 Edge 管理员有权使用此 API。

URL

https://<edge machine host>:8449/api/undeployModel

方法

请求类型 POST

URL 参数

无。

头

`Authorization: user_name/password`

数据参数

`model_id: String`。已取消部署模型的模型标识。

`model_instance_id: String`。已取消部署模型的模型实例标识。

样本主体

```
{"model_id":"5868dcd7-7032-4101-9ba8-797dea9005aa","model_instance_id":"5868dcd7-
7032-4101-9ba8-797dea9005aa_1523253620387"}
```

成功响应

```
{ “status” : “success” }
```

响应项

`Status:String`。success 或 failed。

样本调用

```
curl -k -H 'Authorization: Basic YWRtaW46cGFzc3cwcmRA'
 -H "Content-Type:application/json;charset=UTF-8" -X POST --data
 '{"model_id": "8e474fb5-97dd-4f5d-8415-14c72855c6a5","model_instance_id":
 "8e474fb5-97dd-4f5d-8415-14c72855c6a5_1532404443311"}'  https://{local
host}:8449/api/undeployModel
```

将图像上载到 Edge 并对其评分

将图像上载到 Edge 并对其评分。生产线或外部服务通常会使用此 API。

URL

/api/uploadScoreImage

方法

请求类型 POST

URL 参数

`productType: String`。模型的产品类型。必需。

`cell: String`。图像的工作站。必需。

头

`Content-Type: application/binary`。

`Authorization:`。Edge 服务的基本授权代码。 必需。

数据参数

无。

样本主体

图像文件的二进制内容。

成功响应

对象检测示例:

```
200
{
  "detections": [
    {
```

```
      "position": {
        "height": 60,
        "width": 253,
        "x": 1594,
        "y": 773
      },
      "probableTypes": [
        {
          "confidence": 99.798703193664551,
          "type": "scratch"
        }
      ],
      "properties": []
    }
  ],
  "timestamp": "2017-09-28 10:32:13.528415"
}
```

分类示例:

```
200
{
  "Timestamp": "2017-09-12 18:43:52",
  "detections": [
    {
      "position": {
        "height": 436,
        "width": 537,
        "x": 1574,
        "y": 1588
      },
      "probableTypes": [
        {
          "confidence": 100.0,
          "type": "Defect"
        },
        {
          "confidence": 0.0,
          "type": "MisT"
        },
        {
          "confidence": 0.0,
          "type": "NoDefect"
        }
      ]
    }
  ]
}
```

样本调用

```
curl -k --cert /home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/cert.pem --key
 /home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/key.pem   -H 'Authorization:
 Basic auth' -T 'filePath' -X POST https://localhost
:8449/api/uploadScoreImage?productType=productType\&cell=cell
```

auth 值是访问 Edge 服务的基本授权。*filePath* 值是要上载的文件的路径。*localhost* 值是部署 Edge 的系统的 IP 地址。*productType* 值是用于对上载的图像进行评分的模型的产品类型。*cell* 值是您希望将分数结果发送到的单元格。

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

将检查结果从 Edge 同步到中心应用程序

将检查结果从 Edge 同步到中心应用程序。独立 Edge 管理员有权使用此 API。

URL

/api/syncInspectResult

方法

请求类型 POST

URL 参数

无

头

`Content-type: application/json`。

`Authorization:`。Edge 服务的基本授权代码。必需。

数据参数

无。

样本主体

```
{
“percentage” :100
}
```

成功响应

```
200
{
“message” : “success”
}
```

样本调用

```
curl -k --cert /home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/cert.pem --key
/home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/key.pem -H "Content-Type: application/json"
-H 'Authorization: Basic YWRtaW46cGFzc3cwcmRA' -X POST --data '{"percentage":percentage}'
https://localhost:8449/api/syncInspectResult
```

percentage 值是要发送的检查结果的百分比。*localhost* 值是 Edge 部署到的 IP 地址。

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

清除已同步到中心应用程序的检查结果

清除已同步到中心应用程序的检查结果。独立 Edge 管理员有权使用此 API。

URL

/api/cleanInspectResult

方法

请求类型 POST

URL 参数

无

头

`Content-type: application/json`。

`Authorization:`。Edge 服务的基本授权代码。必需。

数据参数

无。

样本主体

```
{
“fromDayBefore” :1
}
```

成功响应

```
{
 “message” : “success”
}
```

样本调用

```
curl -k --cert /home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/cert.pem
--key /home/pmqopsadmin/vi_edge-bin_vi/vi_edge/https/key.pem -H "Content-Type:
application/json"  -H 'Authorization: Basic YWRtaW46cGFzc3cwcmRA' -X POST --data
'{"fromDayBefore":dayNumber}' https://localhost:8449/api/cleanInspectResult
```

dayNumber 值是要保留数据的天数。例如：如果将 *dayNumber* 设置为 0，将除去今天之前的所有数据。如果将 *dayNumber* 设置为 1，将除去昨天之前的所有数据。*localhost* 值是 Edge 部署到的 IP 地址。

注释

系统使用 APIKEY 执行认证。如果未提供 APIKEY，系统会拒绝您的请求。

第 8 章 使用 API 注册、部署和测试模型

“API 指南”功能是一个 Web 界面，用于引导您完成一个常见的 API 用例：注册、部署和测试已培训模型的过程。“API 指南”功能可自动执行该过程的许多步骤，以便更轻松地完成各项任务。

关于此任务

第一步是注册模型。您可以导入包含已培训模型的压缩文件，然后将其注册到产品。

第二步是部署模型。您可以将模型部署到 Edge 系统，使模型可用于检查图像。

第三步是测试模型。您可以使用自己的图像来测试部署的模型并查看评分结果。

过程

1. 从下拉菜单中选择 **API 指南**。
2. 单击**尝试注册模型**以进入“**端到端 API 指南**”页面。
3. 输入模型名称和产品类型。
4. 单击**选择**以选择包含已培训模型的压缩文件。
5. 单击**注册**以将模型注册到产品。
   注册结果会显示在“**结果**”框中。
6. 单击**下一步**。
7. 根据需要，输入实例标识。
8. 单击**部署**以部署模型。
   部署结果会显示在“**结果**”框中。
9. 单击**下一步**。
10. 输入产品类型和工作站信息，选择图像，然后单击**评分**以启动评分过程。
    评分结果会显示在“**结果**”框中。

第 9 章 故障诊断

查看日志文件和错误消息可帮助对产品进行故障诊断。

日志文件

请参阅产品日志文件，以获取有关错误状况的更多信息。

日志文件	描述
/*Liberty_path*/usr/servers/VICenterServer/logs/messages.log	WebSphere® Application Server Liberty 消息
/home/*user_name*/edgeDeploy_*tenant name*.log	Edge 消息
/home/*user_name*/vi_edge-bin_vi/vi_task_manager/master.log	任务管理器消息
/home/*user_name*/vi_edge-bin_vi/vi_edge/Service.log	服务消息

消息

Edge 消息

以下消息会在 Edge 系统发生错误状况期间显示。

表 *4. Edge* 系统的错误消息

消息	操作
主 Edge 已存在，只支持一个主 Edge。	指定未使用的 Edge 名称。
必填字段不能为空。	在所有必填字段中包含值。
无法连接到机器。请使用正确的 IP 地址、SSH 用户名和密码。	验证 IP 地址、SSH 用户名和密码是否为有效值。
要管理 Edge，您必须具有正确的许可权。	确保您以具有模型管理者许可权的用户身份登录。如果无法解决问题，请与系统管理员联系。
IP 地址无效。请输入正确的 IP 地址，例如 10.172.0.23。	验证 IP 地址的值。
由于数据库错误而未创建 Edge。	在 Ambari 控制台中检查 Apache HBase 服务。
具有相同 IP 地址的 Edge 已存在。	无法在同一台计算机上创建两个 Edge 系统。请使用另一台计算机的 IP 地址来创建新 Edge。
将脚本上载到 Edge 机器失败。请检查部署路径是否存在并检查路径许可权。	验证部署路径是否存在以及是否具有正确的许可权。
在 Edge 机器上启动服务失败。	确保服务在主 Edge 上运行。请查看 Edge 系统上的日志文件。
在 Edge 机器上执行脚本失败。	请查看 Edge 系统的日志文件。
Edge 名称重复。请使用其他 Edge 名称。	使用其他名称来创建新 Edge。
如果主 Edge 不存在，那么无法创建从属 Edge。	必须创建主 Edge 后，才可创建从属 Edge。

表 *4. Edge* 系统的错误消息 *(续)*	
消息	**操作**
删除 Edge 失败，因为 *model_name* 模型在使用该 Edge。	如果 Edge 在模型版本中使用，那么无法删除该 Edge。请停止 Edge 上运行的所有模型版本后，再删除该 Edge。
由于数据库错误而未删除 Edge。	在 Ambari 控制台中检查 Apache HBase 服务。
无法删除主 Edge，因为存在现有的从属 Edge。	无法删除仍具有从属 Edge 的主 Edge。请删除所有从属 Edge 后，再删除主 Edge。
从 *repository_name* 主存储库中删除 Edge 失败。	确保服务在主 Edge 上运行。请查看服务日志文件。

压缩图像文件消息

以下消息会在压缩图像文件发生错误状况期间显示。

表 *5.* 压缩图像文件的错误消息	
消息	**操作**
上载的压缩文件在预期位置不包含图像。	上载的压缩图像文件必须包含图像。压缩图像文件中的所有图像必须采用无子目录的扁平结构。
文件无效。必须仅上载 .zip 文件。	唯一支持的文件格式是 .zip。
上载的压缩文件中缺少注释 XML 文件。	将注释 XML 文件添加到压缩图像文件，然后重新上载该文件。
上载的压缩文件中缺少 labels.txt。	将 `labels.txt` 文件添加到压缩图像文件，然后重新上载该文件。
输入无效。必须指定标识。	API 请求中没有提供文件标识。请提供文件标识。
输入无效。可以删除的最大实体数为 100 个。	一次删除的压缩文件不能超过 100 个。
删除数据文件失败。	无法删除数据文件。
model_name 模型中引用了指定的数据文件。无法将其删除。	压缩图像文件被模型引用。无法将其删除。

图像组消息

以下消息会在图像组发生错误状况期间显示。

表 *6.* 图像组的错误消息	
消息	**操作**
创建新数据组时，组名不能为空。	为新组指定名称。
所选数据组中已存在数据文件。更新失败。请确保所选数据组不包含任何数据文件。	如果图像组是使用数据文件创建的，那么无法更新图像组类型。如果图像组不是使用数据文件创建的，那么导致此错误的原因可能是 Apache HBase 操作错误。
创建数据组失败，因为缺少参数或参数格式无效。	确保为新数据组指定参数，并且参数格式有效。
组名重复。请使用其他组名。	为新组指定唯一名称。
创建数据组时出错。	正在创建数据组时发生 Apache HBase 错误。请在 Ambari 控制台中检查 Apache HBase 服务。
数据组不存在。	请求中缺少组标识或组标识无效。

表 6. 图像组的错误消息 (续)	
消息	**操作**
删除指定数据组中的数据文件时出错。	正在删除数据文件时，发生 Hadoop 分布式文件系统 (HDFS) 错误。
模型中引用了指定的数据组。无法将其删除。	如果组在模型中使用，那么无法删除该组。
删除数据组时出错。	正在删除数据组时发生 Apache HBase 错误。请在 Ambari 控制台中检查 Apache HBase 服务。

模型消息

以下消息会在模型发生错误状况期间显示。

表 7. 模型的错误消息	
消息	**操作**
创建新模型时，模型名称不能为空。	为模型指定名称。
模型名称必须少于 128 个字符，并且只能包含字母、数字、空格和下划线。	指定有效的型号名称。
创建新模型时，产品类型不能为空。	指定产品类型。
产品类型必须少于 128 个字符，并且只能包含字母、数字和空格。	指定有效的产品类型。
创建新模型时，数据组标识不能为空。	在 API 调用中指定 `groupIds` 的值。
创建新模型时，数据格式不能为空。	在 API 调用中指定 `dataFormat` 的值。
创建新模型时，重新培训策略的格式必须有效。	在 API 调用中为 `retrainPolicy` 指定有效值。
创建新模型时，模型类型不能为空。	在 API 调用中指定 `modelType` 的值。
模型类型必须为 classification 或 objectdetection。	在 API 调用中为 `modelType` 指定值 `classification` 或 `objectdetection`。
创建新模型时，参数的格式必须有效。	在 API 调用中为 `parameters` 指定有效值。
创建模型失败，因为参数中的 trainParam 值无效。	在 API 调用中为 `trainParam` 指定有效值。
数据组的标识无效。	在 API 调用中为 `groupIds` 指定有效值。
发生错误，因为数据组必须全部是多特征类型，或者全部是单特征类型。	在 API 调用中，`groupIds` 中的组必须具有相同的组类型。
创建模型失败。	创建模型期间发生 Apache HBase 错误。请在 Ambari 控制台中检查 Apache HBase 服务。
输入无效。可以删除的最大实体数为 100 个。	指定数量少于 100 的模型进行删除。
输入无效。必须指定标识。	在请求中包含模型标识。
无法更新模型，因为模型标识为空。请输入模型标识。	在请求中包含模型标识。
更新模型组失败，因为模型版本不处于草稿状态。	确保要更新的模型处于草稿状态。
更新模型组失败，因为各数据组的 isHybrid 值不同。	创建模型后，无法更改模型类型。

模型实例消息

以下消息会在模型实例发生错误状况期间显示。

表 *8.* 模型实例的错误消息

消息	**操作**
创建新模型实例时，模型标识不能为空。	在创建相应模型后，再创建模型实例。
创建新模型实例时，培训数据不能为空。	在 API 调用中为 `trainData` 指定有效值。
向培训服务器注册模型失败。	检查服务是否在培训服务器上运行。
创建模型失败，因为参数中的 trainParam 值无效。	在 API 调用中为 `trainParam` 指定有效值。
培训数据无效。必须至少存在一个数据组。	确保属于实例的每个数据组都有有效的数据文件。
培训数据无效。请检查培训和验证比例，以确保每个数据组的培训和测试集内至少存在一个图像。	确保每个数据组的培训和测试集内至少存在一个图像。
培训队列中有 *n* 个模型。在完成或取消这些模型之前，不能提交新的培训请求。	取消作业或等待当前正在运行的作业完成。
取消培训失败。	在 Ambari 控制台中检查 Apache HBase 服务。
输入主体无效。	检查请求输入，然后重试操作。
由于验证超时，*model_name* 模型进行了状态回滚。	请查看验证服务器上的日志文件。对于分类模型，日志为 `/home/pmqopsadmin/vi_edge-bin_vi/vi_score_engine_restful/back.log`。对于对象检测模型，日志为 `/home/pmqopsadmin/vi_obj_detection_retrain/RESTAPI/model/frcnn_log.txt` 或 `ssd_log.txt`。请解决任何问题，然后重试操作。
验证模型版本失败。	请查看验证服务器上的日志文件。对于分类模型，日志文件为 `/home/pmqopsadmin/vi_edge-bin_vi/vi_score_engine_restful/back.log`。对于对象检测模型，日志文件为 `/home/pmqopsadmin/vi_obj_detection_retrain/RESTAPI/model/frcnn_log.txt` 或 `ssd_log.txt`。请解决任何问题，然后重试操作。
模型不存在。	找不到具有此模型标识的模型。请检查模型标识。
拒绝模型版本失败。	在 Ambari 控制台中检查 Apache HBase 服务。
部署模型版本失败，因为模型文件为空。	创建模型时发生错误。请创建新模型。
没有主 Edge 可部署模型。	创建主 Edge 后，再部署模型。
Edge IP *IP_address* 无效，只能使用主 Edge 部署模型。	使用主 Edge 的正确 IP 地址来部署模型。
无法将模型部署到 Edge，因为已部署相同产品类型的 *model_name* 模型。具有相同产品类型的其他模型在 Edge 上运行时，不能将模型部署到该 Edge。必须先取消部署其他模型。	具有相同产品类型的其他模型在 Edge 上运行时，不能将模型部署到该 Edge。
创建数据文件时发生数据库错误。	在 Ambari 控制台中检查 Apache HBase 服务。
file_name 文件未上载到存储器。	检查 Hadoop 分布式文件系统。

表 *8.* 模型实例的错误消息 *(续)*	
消息	**操作**
由于数据库错误而导致更新数据组失败。	在 Ambari 控制台中检查 Apache HBase 服务。
模型实例的状态不支持此操作。	确保模型实例处于正确的状态，然后重试操作。
培训数据无效。每个数据组中必须至少存在一个数据文件。	确保属于模型实例的每个数据组都有有效的数据文件。
培训数据无效。请检查培训和验证比例，以确保每个数据组的培训和测试集内至少存在一个图像。	确保每个数据组的培训和测试集内至少存在一个图像。
由于数据库错误而导致重新培训模型失败。	在 Ambari 控制台中检查 Apache HBase 服务。
将模型版本部署到 Edge *edge_name* 失败。	确保服务在主 Edge 上运行。请查看主 Edge 上的 `Service.log` 文件。
从 Edge *edge_name* 取消部署模型版本失败。	确保服务在主 Edge 上运行。请查看主 Edge 上的 `Service.log` 文件。
由于数据库错误而导致取消部署模型版本失败。	在 Ambari 控制台中检查 Apache HBase 服务。
取消培训失败。	在 Ambari 控制台中检查 Apache HBase 服务。
snapshot 值无效。无法使用快照。	snapshot 值不存在或无效。
快照文件在培训服务器上不存在。	请跳过快照或重新培训模型。
无法删除模型实例，因为实例标识为空。请输入实例标识。	输入有效的实例标识。
由于数据库错误而导致删除模型实例失败。	在 Ambari 控制台中检查 Apache HBase 服务。
无法删除状态为 *status* 的模型实例。	无法删除具有指定状态的模型实例。

API 指南消息

以下消息会在 API 指南发生错误状况期间显示。

表 *9. API* 指南的错误消息	
消息	**操作**
找不到指定产品类型的模型。	如果因 API 调用而发生此错误，请确认发送的产品类型是否正确。如果在用户界面中发生此错误，请尝试使用其他产品类型。
没有关于模型部署位置的 Edge 信息。	验证是否部署了相应的模型。
将图像传输到 Edge *edge_name* 失败。	确保服务在主 Edge 上运行。请查看主 Edge 上的 `Service.log` 文件。
从 Edge *edge_name* 获取图像评分结果失败。	确保服务在主 Edge 上运行。请查看主 Edge 上的 `Service.log` 文件。
发生超时问题。无法从 Edge *edge_name* 获取图像评分结果。	确保服务在主 Edge 上运行。请查看主 Edge 上的 `Service.log` 文件。
创建新模型时，模型名称不能为空。	为模型指定名称。
模型名称必须少于 128 个字符，并且只能包含字母、数字、空格和下划线。	指定有效的型号名称。
创建新模型时，产品类型不能为空。	指定产品类型。

表 *9. API* 指南的错误消息 *(续)*	
消息	**操作**
产品类型必须少于 128 个字符，并且只能包含字母、数字和空格。	指定有效的产品类型。
模型名称重复。请使用其他模型名称。	为模型指定唯一名称。
file_name 文件未上载到存储器。	检查 Hadoop 分布式文件系统。
上载模型文件失败。	检查网络连接，然后重试上载。

模拟器消息

以下消息会在模拟器发生错误状况期间显示。

表 *10.* 模拟器的错误消息	
消息	**操作**
发送模拟图像失败。	确保服务在主 Edge 上运行。请查看 `Service.log` 文件。

声明

本信息是为在美国提供的产品和服务编写的。IBM 可能提供了本资料的其他语言版本。但是，您可能需要拥有该语言的产品副本或产品版本，才能对其进行访问。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以以书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

有关双字节字符集 (DBCS) 信息的许可查询，请联系您所在国家或地区的 IBM 知识产权部门，或用书面方式将查询寄往：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销或适用于某种特定用途的保证。某些管辖区域在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是此 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可协议或任何同等协议中的条款提供。

所引用的性能数据和客户示例只用于阐述说明。实际性能结果可能因特定的配置和操作条件而有所不同。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

本信息仅用于规划目的。在所描述的产品上市之前，此处的信息会有更改。

本信息包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，这些示例中包括个人、公司、品牌和产品的名称。所有这些名字都是虚构的，若现实生活中实际人员或业务企业与此相似，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。如果是为按照在编写样本程序的操作平台上的应用程序编程接口 (API) 进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例尚未在所有条件下经过全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。这些实例程序“按现状”提供，不附有任何种类的保证。对于因使用样本程序所引起的任何损害，IBM 概不负责。

商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp. 在全球许多管辖区域内注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。IBM 商标的最新列表可以在 Web 上的“Copyright and trademark information”中获取，地址为：www.ibm.com/legal/copytrade.shtml。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其关联公司的商标或注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

产品条款和条件文档

只有遵守以下条款和条件才会授予使用这些出版物的许可权。

适用性

这些条款和条件是对 IBM Web 站点的任何使用条款的补充。

个人用途

您可以为了个人使用而非商业性使用复制这些出版物，但前提是保留所有专有权声明。未经 IBM 的明确许可，您不得分发、显示这些出版物或其中部分出版物，也不得制作其演绎作品。

商业用途

您仅可在贵公司内部复制、分发和显示这些出版物，但前提是保留所有专有权声明。未经 IBM 的明确许可，您不得制作这些出版物的演绎作品，也不得在贵公司外部复制、分发或显示这些出版物或其部分出版物。

权利

除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论明示的还是暗含的。

只要 IBM 认为这些出版物的使用会损害其利益或者 IBM 判定未正确遵守上述指示信息，IBM 将保留撤销本文所授予许可权的权力。

只有您完全遵循所有适用的法律和法规，包括所有的美国出口法律和法规，您才可以下载、出口或再出口该信息。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。

IBM 网上隐私声明

IBM 软件产品（包括软件即服务解决方案，以下统称“软件产品”）可能会使用 cookie 或其他技术来收集产品使用信息，以便帮助改善最终用户体验，定制与最终用户的交互或者满足其他用途。在许多情况下，软件产品不会收集个人可标识信息。我们的一些软件产品可以帮助您收集个人可标识信息。如果此软件产品使用 cookie 来收集个人可标识信息，那么有关此产品对 cookie 的使用的特定信息将如下所述。

根据部署的配置，此软件产品可以使用会话和持久性 cookie 来收集每个用户的名称、用户名、密码或其他个人可识别信息，以实现会话管理、认证、单点登录配置或其他使用情况跟踪或功能目的。可禁用这些 cookie，但是禁用它们也可能消除它们支持的功能。

如果为此软件产品部署的配置使您作为客户能够通过 cookie 和其他技术从最终用户处收集个人可标识信息，那么您应自行寻求有关适用于此类数据收集的任何法律（包括对于声明和同意的任何要求）的法律意见。

有关将各种技术（包括 Cookie）用于这些用途的更多信息，请参阅 IBM 的隐私策略 (http://www.ibm.com/privacy) 和 IBM 的网上隐私声明中标题为“Cookie、Web Beacon 和其他技术”部分 (http://www.ibm.com/privacy/details) 以及 IBM 软件产品和软件即服务隐私声明 (http://www.ibm.com/software/info/product-privacy)。

部件号

(1P) P/N: