Reference 文档

2019-09-18

百度智能云

目录

企业组织

产品介绍

简介

企业组织是百度智能云为企业客户提供的用于集中管理成员、资源、财务的账户管理服务，是多用户访问控制的功能扩展。使用企业组织服务，您可以将相关账户加入企业组织，组织成员具有层级关系的结构，上层可以管理下层的财务和资源、控制下层账户的权限和策略。

在一个企业组织内，仅存在一个主账户、可以存在1个以上的子账户。

主账户： 是资金、资源管理的主体，负责管理组织，主账户可以创建子账户，或邀请其他账号成为组织中的成员，也可以将成员账户从企业组织中移除，将服务策略应用到组织层级中，从而控制成员账户的访问权限。主账号是付款账户，负责支付成员账户产生的所有费用。主账户有且只有一个。

子账户： 又称成员账户，可以被主账户邀请或创建。被邀请的账户，本身可作为独立账户使用，可以主动选择离开企业组织；被主账户创建的成员账户，无法主动离开企业组织。

主要功能

成员管理

在企业组织中，主账户可以邀请其他账户（或快速新建账户）共同组建多级账户体系。

权限控制

主账户对其成员账户有绝对地控制权限，可以对成员添加服务策略，使成员账户符合合规性要求，或限制成员账户不能随意修改某些公共服务等。

资源管理

主账户和成员账户间资源隔离，主账户拥有成员账户的资源管理员权限，可集中管理资源。

财务管理

- 多账户统一财务结算。
    - 主账户全权负责打款、付账单、开票，可获得跟踪每个子账户的费用和支付明细。
    - 子账户负责使用资源，不可打款、付账单、开票，可查看自己账户的费用便于同主账户对账。
- 多账户统一同步主账户的优惠等财务特权。

财务管理功能在公测阶段，如您有财务管理的需求，请您在工单系统中提交申请/或直接联系您的客户经理提交申请。

申请和使用财务管理需满足：

- 实名认证类型必须为企业客户。
- 主账户的实名认证信息和子账户的主体信息必须保持一致。
- 签署线下合同。

企业组织vs多用户访问控制

企业组织

企业组织：适用于企业内部多部门间、代理商与客户等多层级结构的管理，各个主体都是独立账户（具有独立账单），这些独立账户之间有组织层级关系，上层可以管理下层的财务和资源、控制下层账户的操作权限。

场景一：A公司拥有多个子公司，同一主体，A公司和子公司之间既是相互关联的一个整体，也相对独立运行。

场景说明：

- A公司和子公司都希望具有单独的使用账户，且可在百度智能云上单独使用。
- A公司和子公司相互独立管理资源，A公司拥有对子公司的监管权，可管理子公司的资源。
- A公司和子公司同一主体，A公司来统一支付A公司和子公司的账单。

场景二：代理商B拥有多个客户，代理商有管理这些客户的需求，同时代理商和每个客户之间有独立管理资源的需求；

场景说明：

- 出于安全考虑，代理商B希望客户有单独的使用账户。
- 代理商B拥有对客户的资源使用监管权限。
- 如果代理商B和客户合同终止，代理商B可以随时解除客户的授权。

多用户访问控制

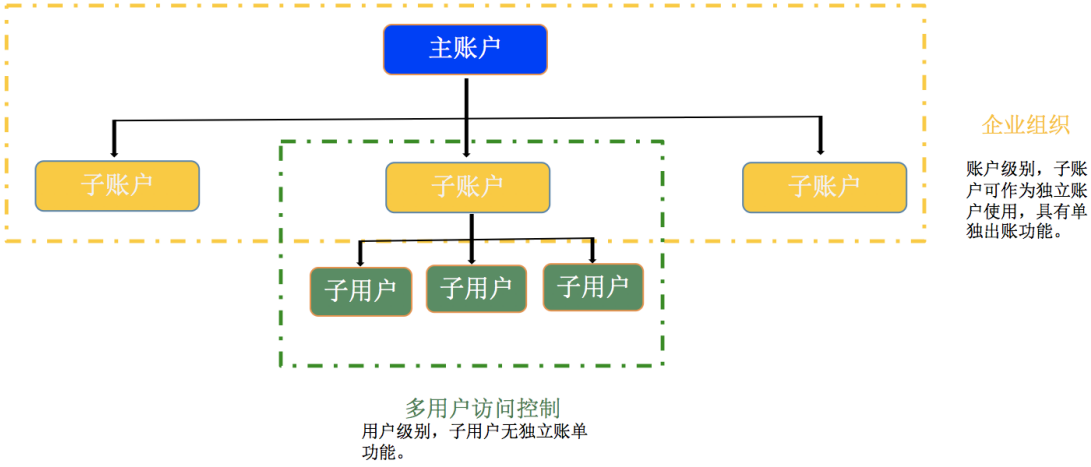
多用户访问控制：适用于企业组织内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，例如，只读，运维，管理，也可细化到资源级别，且子用户不需要对操作产生的费用单独支付。当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。

应用场景：某企业的A账户购买了多种云资源（例如：云服务器BCC,对象存储BOS,内容分发网络CDN等）。该企业拥有许多员工，包括开发人员、测试人员、运维人员等，由于每个员工的工作职责不同，所以需要的权限也不同，且员工不需要对操作产生的费用单独支付。

场景说明：

- 某企业的A账户可以为不同的工作人员赋予使用产品的不同权限，例如，只读，运维，管理，同时可以将控制权限细化到资源层级，例如，BCC某个instance的操作权限。
- 企业员工使用子用户账号进行登录使用，且不需要对操作产生的费用单独支付。

区别



区别	企业组织	多用户访问控制
资源归属	资源归属于各个账户，哪个账户开通/购买，就属于哪个账户。	资源归属于主账户，不属于子用户。
资金和账单归属	企业组织中的每个账户为资金的拥有者，可以单独出账单，同时主账户可以申请开通财务管理权限，通过资金划拨的方式统一支付组织中所有子账户的账单。	账户为资金和计费的载体，子用户不会单独出账单，账户下所有子用户产生的资源费用都记在主账户下。
使用场景	适用于企业组织间，各个主体都是独立账户（具有独立账单），这些独立账户之间有组织层级关系，上层可以管理下层的财务和资源、控制下层账户的操作权限。	适用于企业组织内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，例如，只读，运维，管理，也可细化到资源级别，且子用户不需要对操作产生的费用单独支付。当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。
区别	A公司拥有多个子公司，同一主体，A公司和子公司之间既是相互关联的一个整体，也相对独立运行。 场景说明： 1.A公司和子公司都希望具有单独的使用账户，且可在百度智能云上单独使用。 2.A公司和子公司相互独立管理资源，A公司拥有对子公司的监管权，可管理子公司的资源。 3.A公司和子公司同一主体，A公司拥有整体的财务结算权，A公司来统一支付A公司和子公司的账单。	某企业的A账户购买了多种云资源（例如：云服务器BCC,对象存储BOS,内容分发网络CDN等）。该企业拥有许多员工，包括开发人员、测试人员、运维人员等，由于每个员工的工作职责不同，所以需要的权限也不同，且员工不需要对操作产生的费用单独支付。 场景说明： 1.某企业的A账户可以为不同的工作人员赋予使用产品的不同权限，例如，只读，运维，管理，同时可以将控制权限细化到资源层级，例如，BCC某个instance的操作权限。 企业员工使用子用户账号进行登录使用，且不需要对操作产生的费用单独支付。

创建企业组织

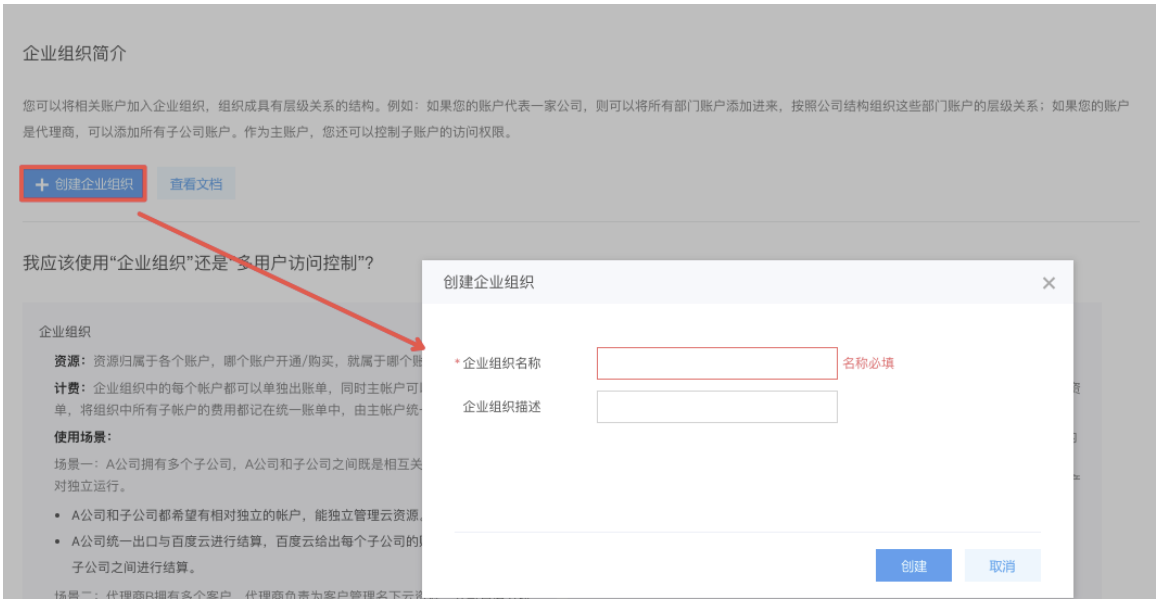
在使用企业组织服务前，您需要创建一个百度智能云账户，请按照下述步骤进行注册和登录。

1. 注册并登录百度智能云平台，请参考注册和登录。

2. 如果未进行实名认证，请参考实名认证操作方法完成认证。

3. 登录成功后，右上角选择“企业组织”，进入企业组织首页，即可开始创建企业组织。

4.点击【创建企业组织】，填写企业组织名称和企业组织描述，此时当前账户为主账户。



邀请-新建账户

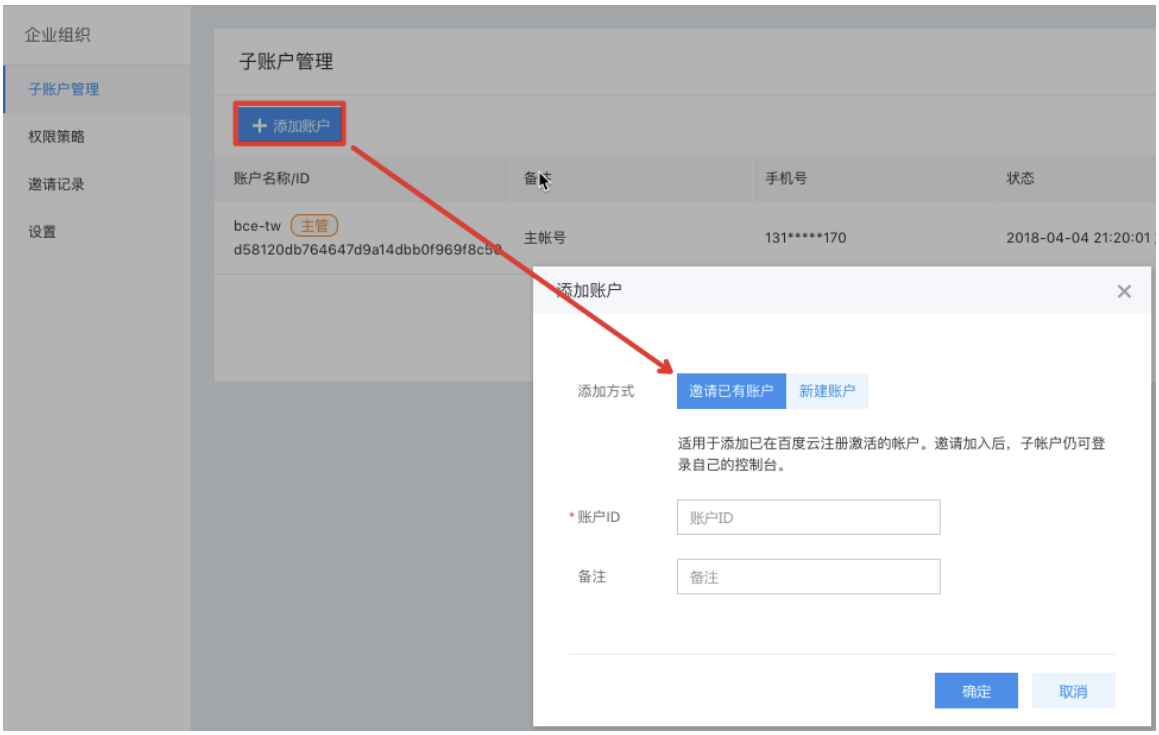
开通企业组织后，需要为企业组织添加新成员，主账户可以选择邀请已有账户，也可以选择新建账户，具体操作方法如下：

在“企业组织“列表中，选择“子账户管理“，点击【添加账户】。

方式一：邀请已有账户

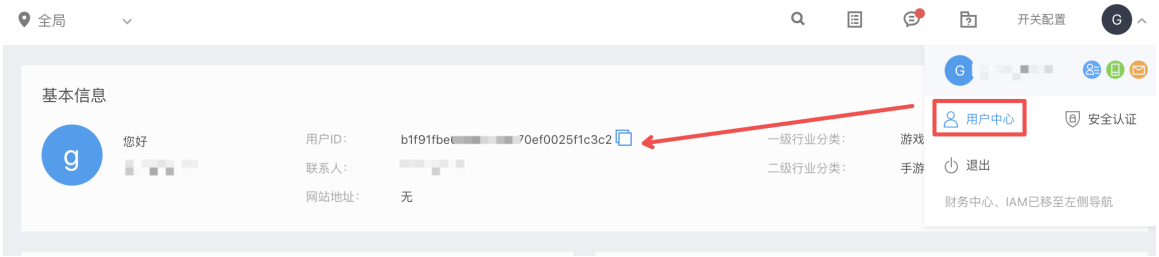
主账户邀请在百度智能云登录已激活的独立账户加入企业组织，新账户加入企业组织后，依然可独立登录控制台，管理资源及账单。

1.点击"添加账户"，选择邀请已有账户。 2.添加子账户的账户ID。



主账户获取子账户ID方法

登录被邀请账户（子账户），进入用户的管理控制台，点击“用户中心”，来到用户信息页。如图所示：复制用户ID，此ID号是子账户提供给主账户进行邀请时使用。



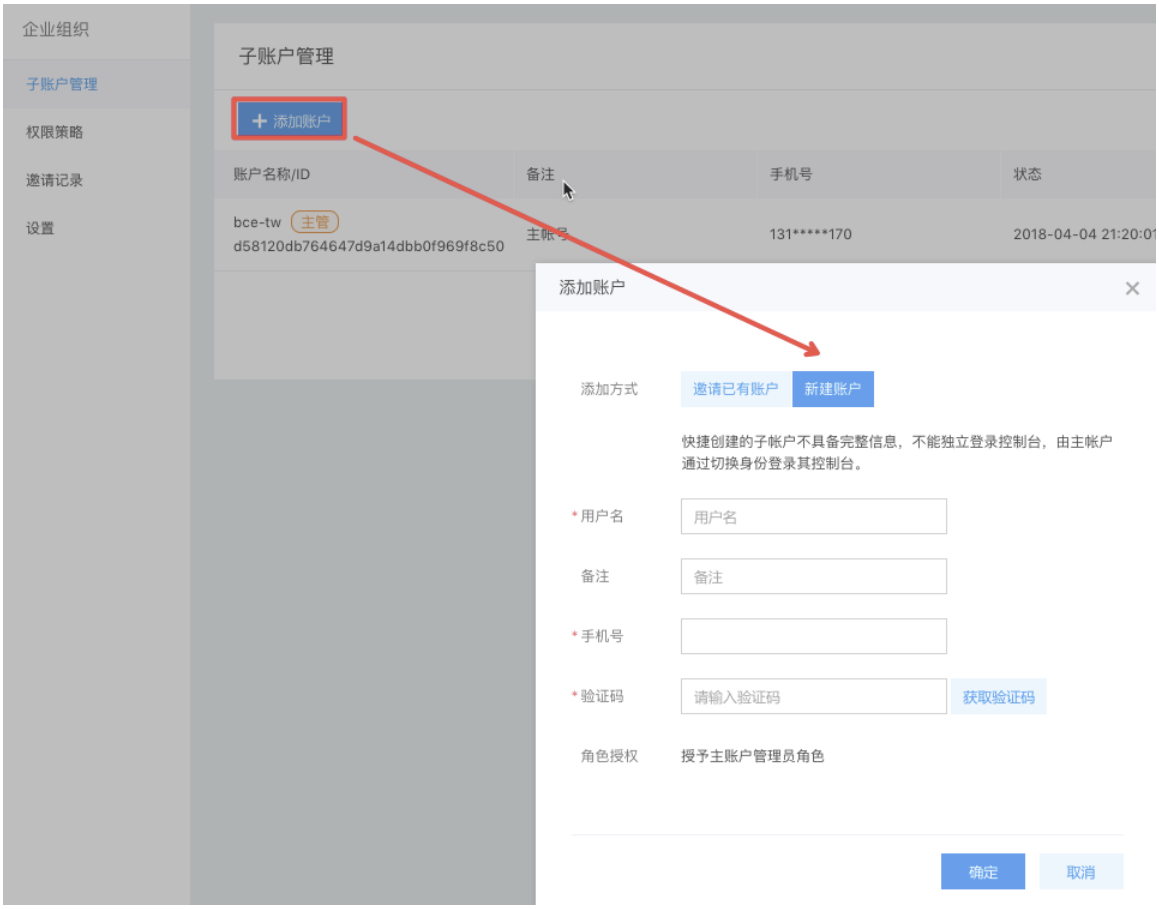
子账户接受邀请

如下图所示，子账户登录百度智能云控制台，点击企业组织图标，可以看到来自主账户的邀请。

点击【查看】，选择【接受】，即子账户成功加入主账户所建立的企业组织。

方式二：新建账户

主账户可以通过新建账户快捷创建子账户，但新建账户默认不能单独登录控制台，需要由主账户通过切换身份登录控制台。



区别

邀请制

邀请机制是组织中的主账户邀请其他账户加入组织的过程。只有主账户可以发出邀请。受邀账户必须已经是百度智能云的激活账户，受邀账户接受邀请后，将成为组织中的成员账户。主管账户可以删除成员账户，成员账户也可以在自己的控制台中主动选择离开组织。

新建账户

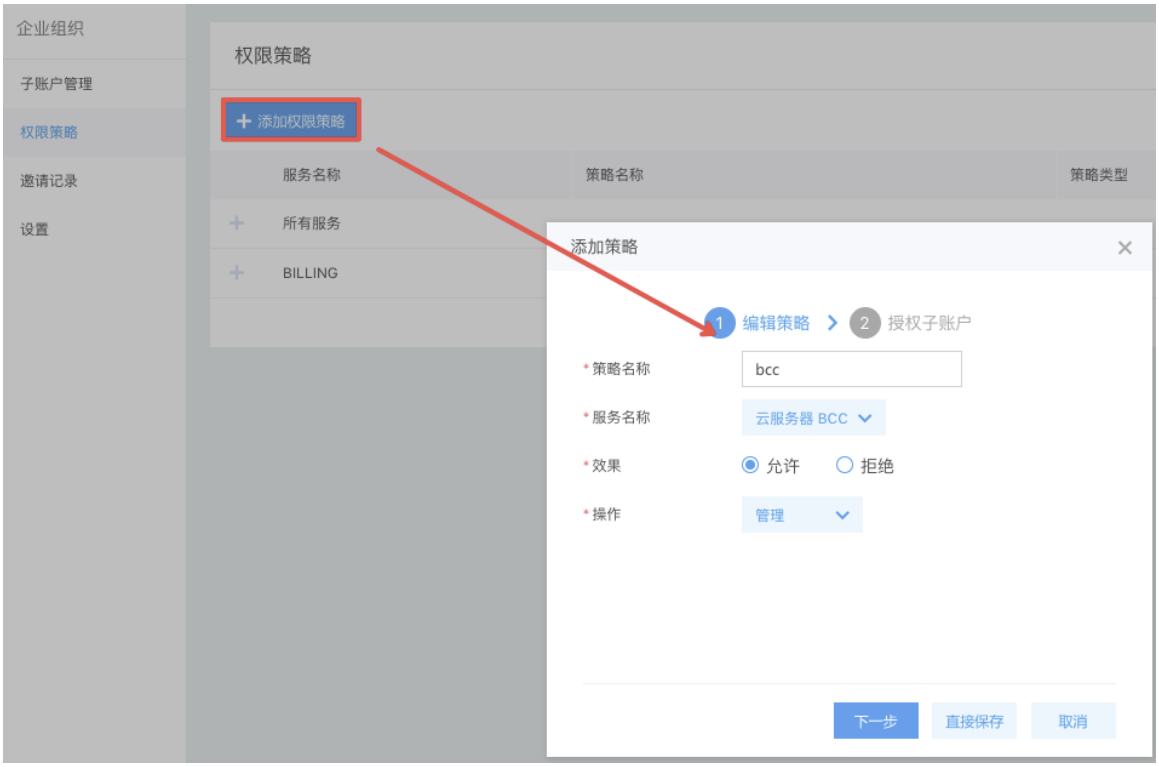
新建账户为非完整信息账户，不能登录控制台(如果需要对成员账户进行资源管理和访问，可以使用IAM子用户功能实现)，需要由主账户进行全面管理账户的信息。该账户为快捷创建账户，如果想要从组织中删除，需要提交工单联系客户经理申请换绑一个完整的账户，然后再进行删除。

账户授权

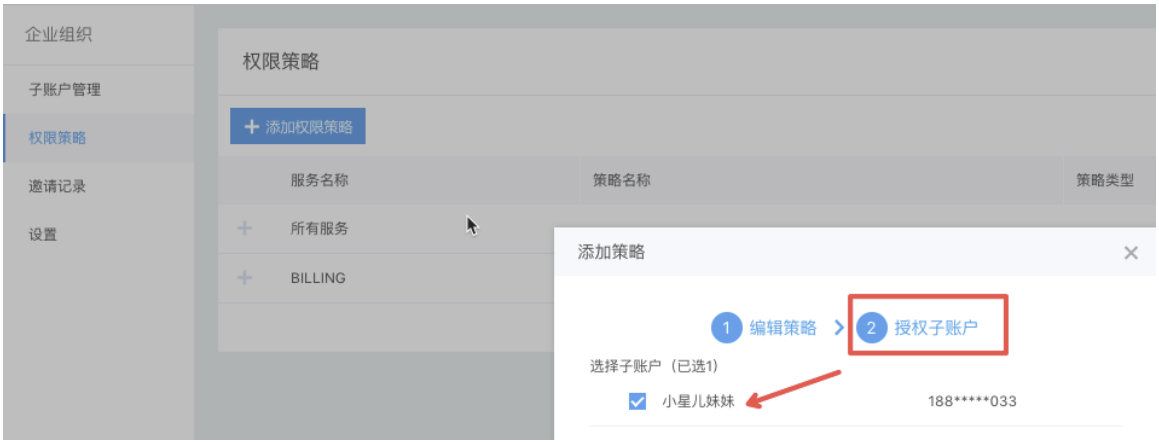
账户授权主要是为组织成员账户设置权限策略，具体操作如下：

1.在企业组织列表中，选择"权限策略"，点击【添加权限策略】,填写以下信息。

- 主账户需要自定义权限策略的名称，选择相应的产品服务。
- 允许：可直接授权子账户，允许其操作该产品服务。
- 拒绝：主账户拒绝子账户使用该产品服务。
- 管理：即子账户具备该产品服务管理权限。

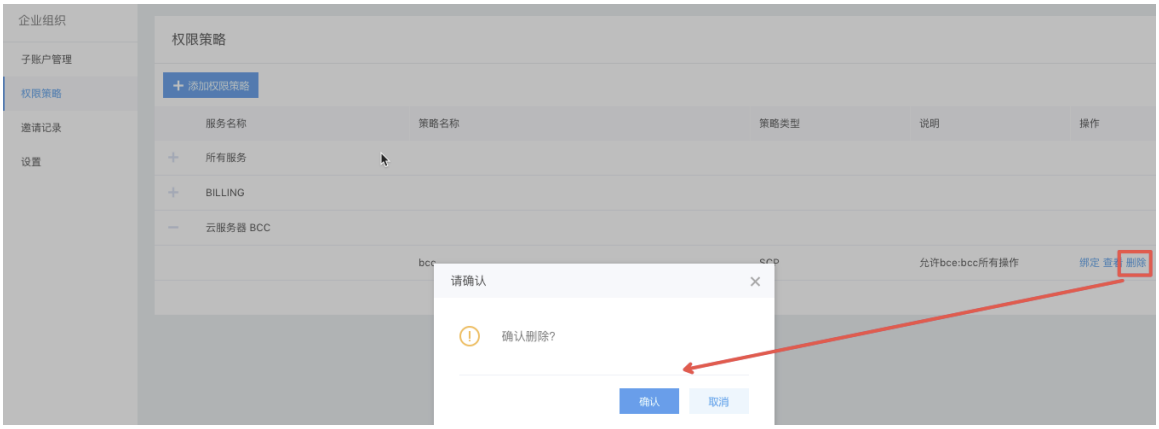


2.如上图所示，选择【下一步】，则可将刚编辑好的策略直接授权给子账户。完成授权后，该子账户即拥有BCC的管理权限。

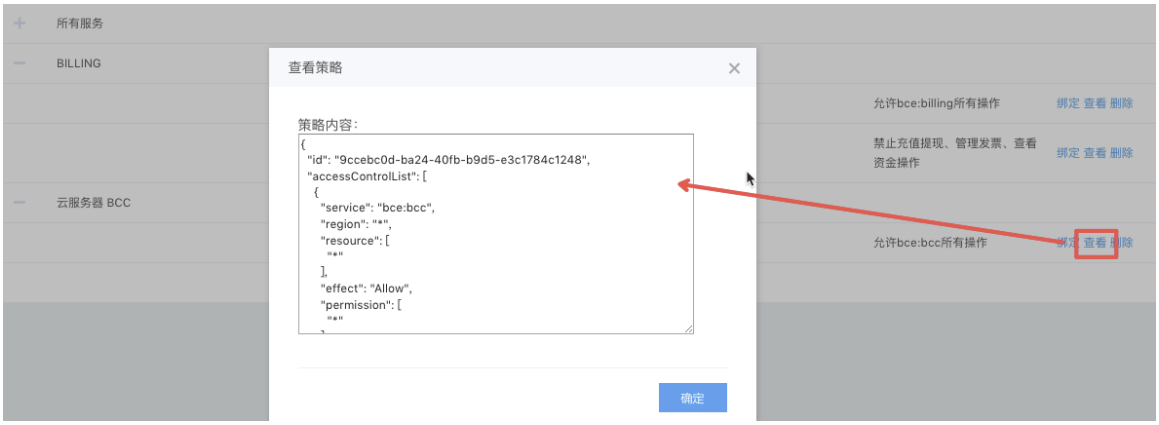


3.可在权限策略列表中，为已经创建好的权限进行子账户绑定，同时也可以进行解除。

4.点击【删除】，则可以删除已经创建好的产品权限。



5.点击【查看】，即可以查看该权限所对应的ACL策略。



组织单元

概述

组织单元是基于企业内的组织架构，或是企业间的服务关系构建的树状管理结构,可以帮助你提升账户管理的效率。组织单元是企业账户和服务控制策略的容器，根节点为**root**。你可以为组织单元 附加服务控制策略，也可以添加下级子单元，将企业组织内的成员账户加入到单元中，下级子单元和账户会自动继承父单元的权限策略，以实现对成员账户的权限、财务等分组管控。

典型场景

当前使用百度智能云企业组织服务的企业客户中，分为如下2种场景：

- **企业内基于组织架构的账户管理**
  客户按照企业组织架构，负责IT管理的信息部门负责和维护主账户，进行统一的账户、权限和财务的管理；企业内其他事业部单独申请使用账户，并加入企业组织中，IT管理员将不同的事业部的成员账户划分到不同的组织单元中，这些组织单元可访问的资源受到特定的业务限制，通过服务控制策略的配置实现。

- **代理商企业与其代理的客户间的账户管理**

代理商企业代表百度智能云为其客户提供云服务，每个项目或客户需要独立的账户，进行单独的财务结算和管理，此时，代理商客户管理员会基于当前账户建立企业组织，并为其客户创建成员账户。将这些客户的成员账户以组织单元的形式进行分组管理，并通过服务控制策略限制客户成员账户对部分系统功能操作、财务信息的查看等，详细可参考财务管理。

产品限制

- 你最多可以创建**10**个组织单元；
- 组织单元嵌套不超过**5**层，不含root层级。

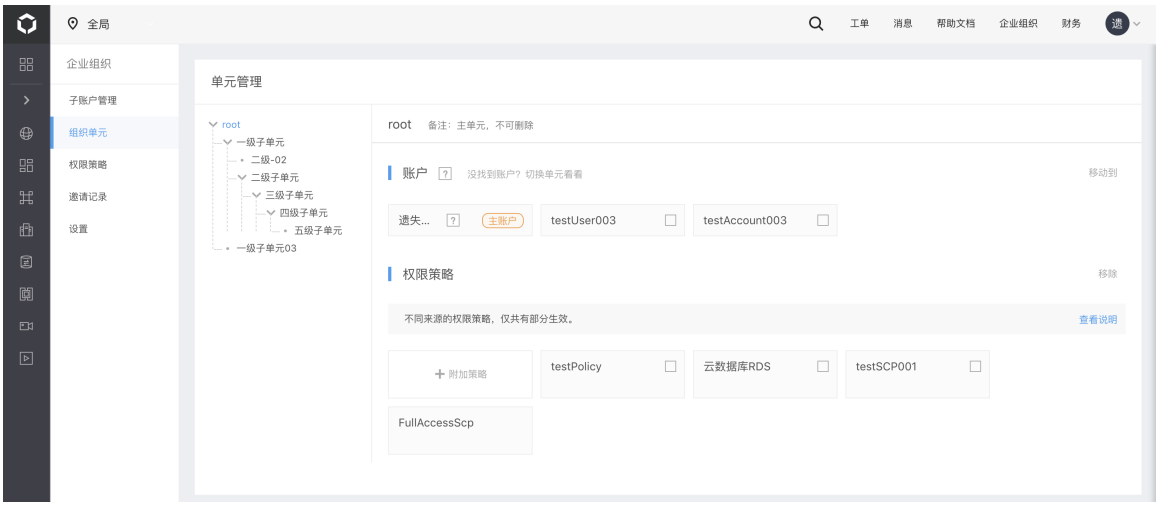
先决条件

- 账户已经开通企业组织功能；
- 拥有账户系统管理员权限；

查看组织单元树

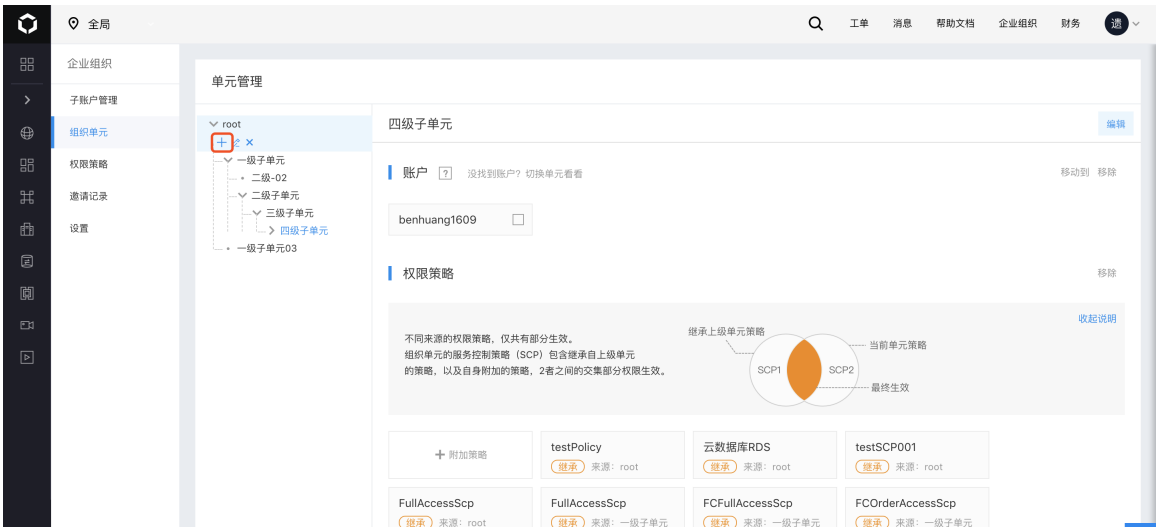
1. 登录百度智能云控制台，在顶部导航栏中找到**企业组织**；
2. 左侧导航栏中进入**组织单元**，在页面左侧会显示组织单元树；
3. 选中某一组织单元节点，可查看当前单元内包含的账户和权限策略。

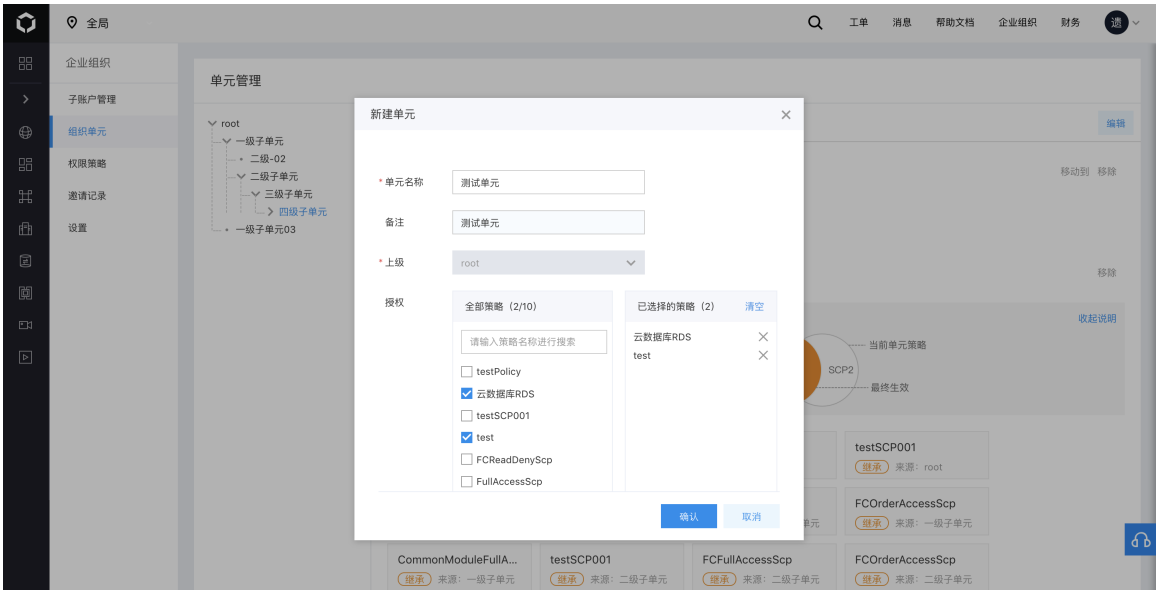
> 注意：组织根节点为**root**,不可编辑及删除。



创建组织单元

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；
2. 展开组织单元树，鼠标移动到目标组织，在单元名称后点击**添加**按钮；
3. 填写弹窗中的单元名称(同一组织内需保持唯一)、备注，选择所需附加的策略；
4. 点击**确认**，即可为所选单元创建一个子单元。





管理组织单元

你可以编辑已存在的组织单元、在单元间移动成员账户、管理组织单元的权限策略等，本节将介绍如何管理组织单元。

编辑组织单元

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；
2. 展开组织单元树，鼠标移动到目标组织，在单元名称后点击**编辑**按钮；
3. 编辑弹窗中的内容：单元名称、备注、切换上级单元、修改权限策略等；
4. 点击**确认**，完成对当前单元的编辑。

移动账户

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；
2. 展开组织单元树，点击进入一个组织单元；
3. 在中间内容区域的**账户**模块中，勾选需要移动的账户(可以勾选多个)，点击右上角的**移动到**按钮；你也可以勾选账户，直接点击**移除**，所选账户将被直接移动到**root**单元中；
4. 在弹窗中选择需要移动到的目标组织单元后，点击**确认**，所选账户将被移动到目标的组织单元；

附加策略

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；
2. 展开组织单元树，点击进入一个组织单元；
3. 在中间内容区域的**权限策略**模块中，点击**附加策略**；
4. 弹窗中选择需要附加的服务控制策略，点击**确认**，将策略附加到当前单元；

移除策略

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；
2. 展开组织单元树，点击进入一个组织单元；
3. 在中间内容区域的**权限策略**模块中，勾选需要移除的策略(可以勾选多个)，点击右上方的**移除**按钮，完成移除策略操作；

> 权限策略中默认展示来自父级组织单元的策略，并标识**继承**标签和来源单元，继承自父级单元的策略不可移除。

删除组织单元

在一些业务场景下，如业务组拆分或重组，需要对重新构建已有的组织单元树，此时涉及到删除已有组织

单元，本节将介绍如何删除组织单元。

1. 登录百度智能云控制台，导航到**企业组织>组织单元**；

2. 展开组织单元树，鼠标移动到目标组织，在单元名称后点击**删除**按钮；

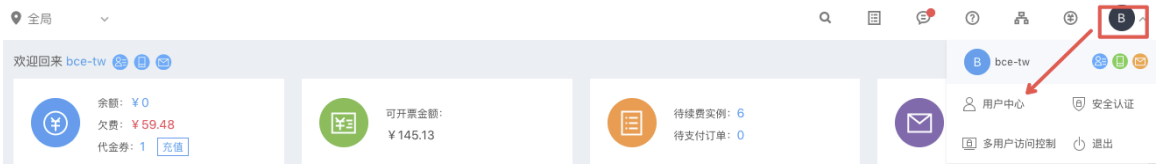
3. 弹窗中**确认**删除单元。

> 注意：删除单元前，请确认单元中的所有账户已经被移出单元，以防止误删，影响到你的业务。

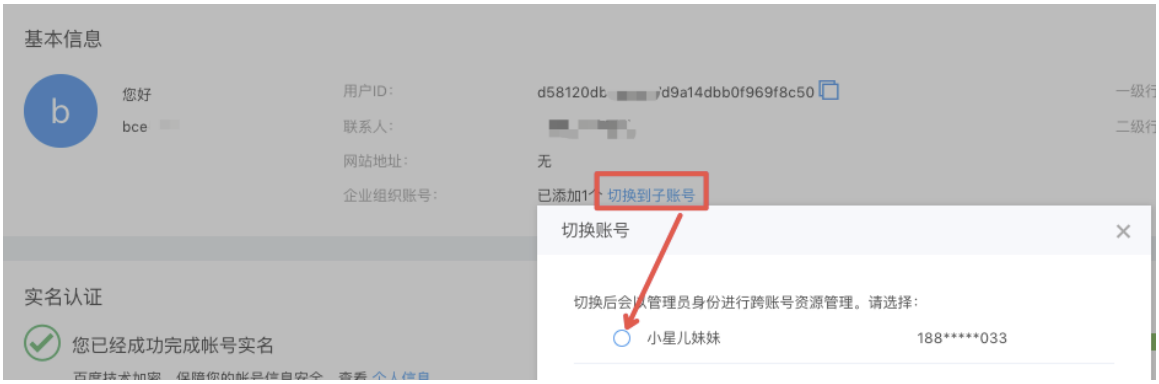
资源管理

主账户对子账户具有资源查看权限，即主账户可以登录子账户控制台，查看子账户资源情况。具体操作如下：

1.点击右上角图标，选择"用户中心"。



2.在"用户中心->企业组织账号"处选择"切换到子账号"，选择对应的子账户进行切换，则可以实现主账户跨账户资源管理。



3.例如，主账户切换到子账户，即登录子账户的控制台，查看子账户资源使用情况。

财务管理

概要说明

企业组织中财务管理功能，通过主账户邀请成员建立财务关系并对其进行财务管理，主账户和子账户统一主体、统一优惠等财务特权，达到财务统一结算的目的。

功能说明

- 多账户统一财务结算。
    - 主账户全权负责打款、付账单、开票，可获得跟踪每个子账户的费用和支付明细。
    - 子账户负责使用资源，不可打款、付账单、开票，可查看自己账户的费用便于同主账户对账。
- 多账户统一同步主账户的优惠等财务特权。

申请说明

财务管理功能在公测阶段，如您有财务管理的需求，请您在工单系统中提交申请/或直接联系您的客户经理提交申请。

申请和使用财务管理需满足：

- 实名认证类型必须为企业客户。
- 主账户的实名认证信息和子账户的主体信息必须保持一致。
- 签署线下合同。

功能入口

开通主子账户财务管理功能后，主账户登录控制台 - 财务中心，点击右下角“主子账户财务”。

关联企业组织成员建立财务关系

您可以关联企业组织中的已有成员建立财务关系，即财务授权，目的为对其进行财务管理。

> 请您注意：本期暂不支持子账户退组并恢复自己独立的财务结算身份，关联成员建立财务关系请谨慎操作。

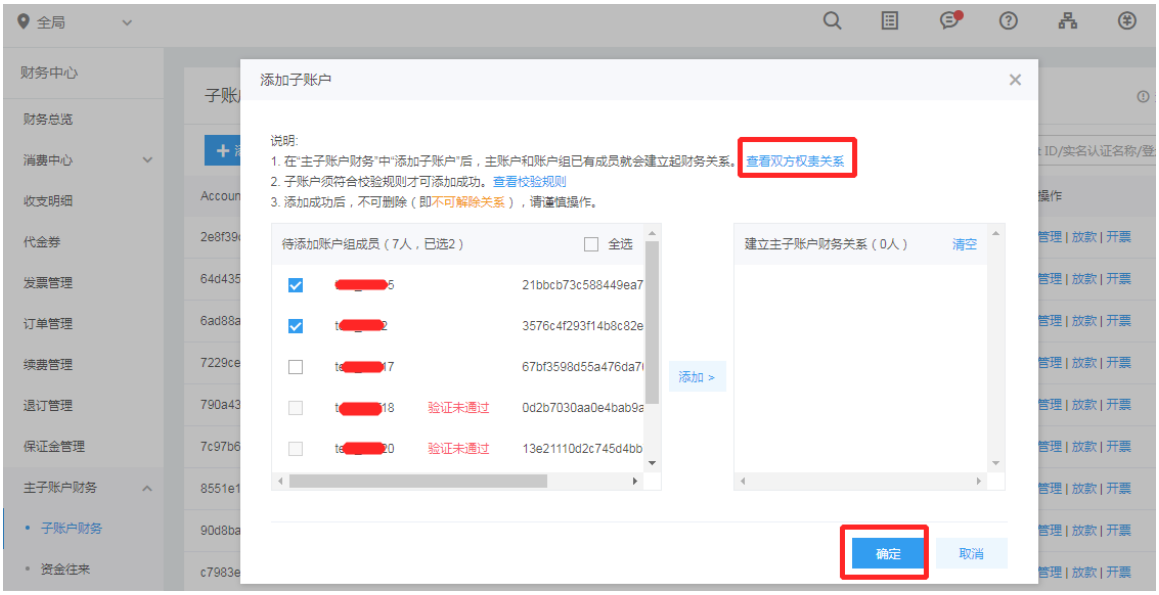
关联企业组织成员建立财务关系，具体操作如下：

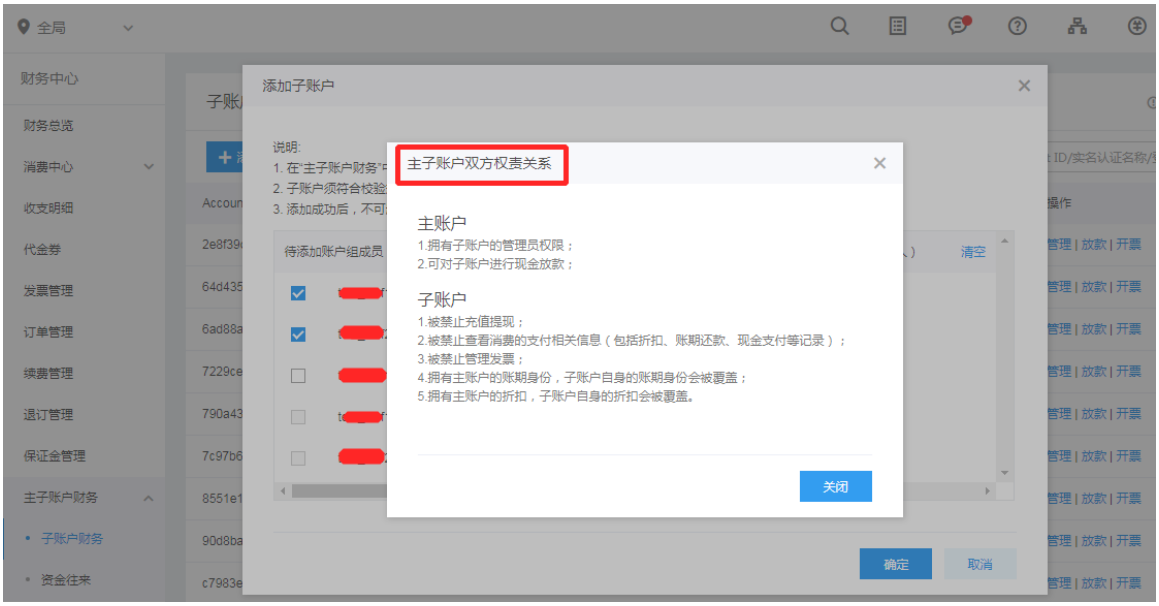
1. 前提为您已有企业组织和成员，若还没有企业组织请参考建立企业组织。

2. 在财务中心 - 主子账户财务中，点击【添加子账户】，进入操作对话框。

3.选择成员后，点击【添加】即可建立财务关系，建立财务关系成功即代表财务授权成功，您可在子账户财务列表中查看。

> 请您注意：操作前请务必点击查看双方权责关系。



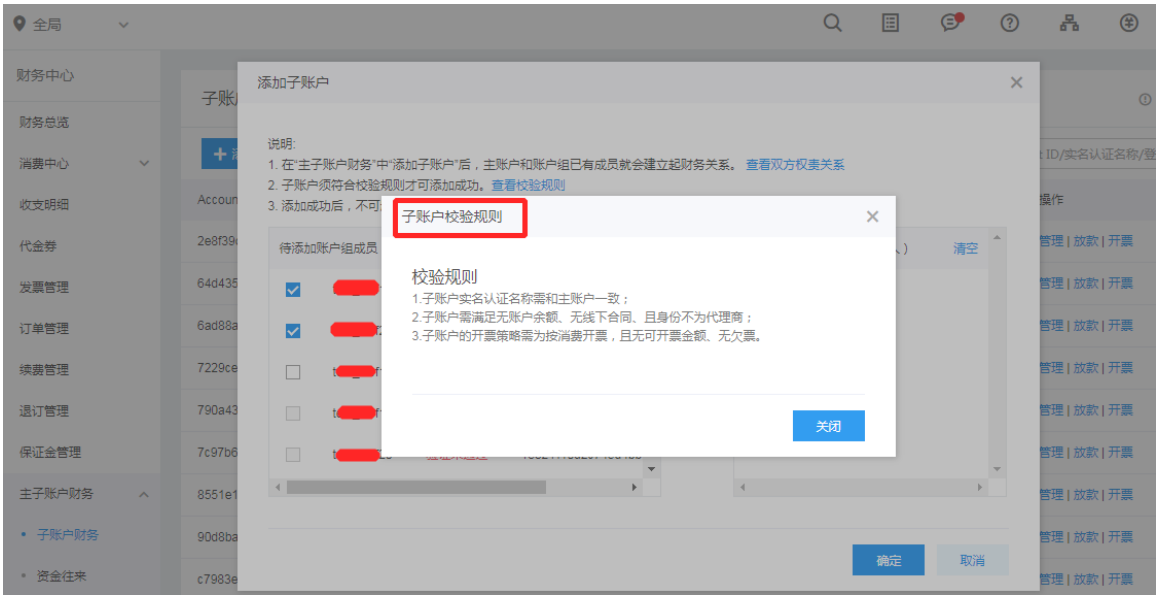


注意事项一：

子账户须符合校验规则才可添加成功。校验规则如下：

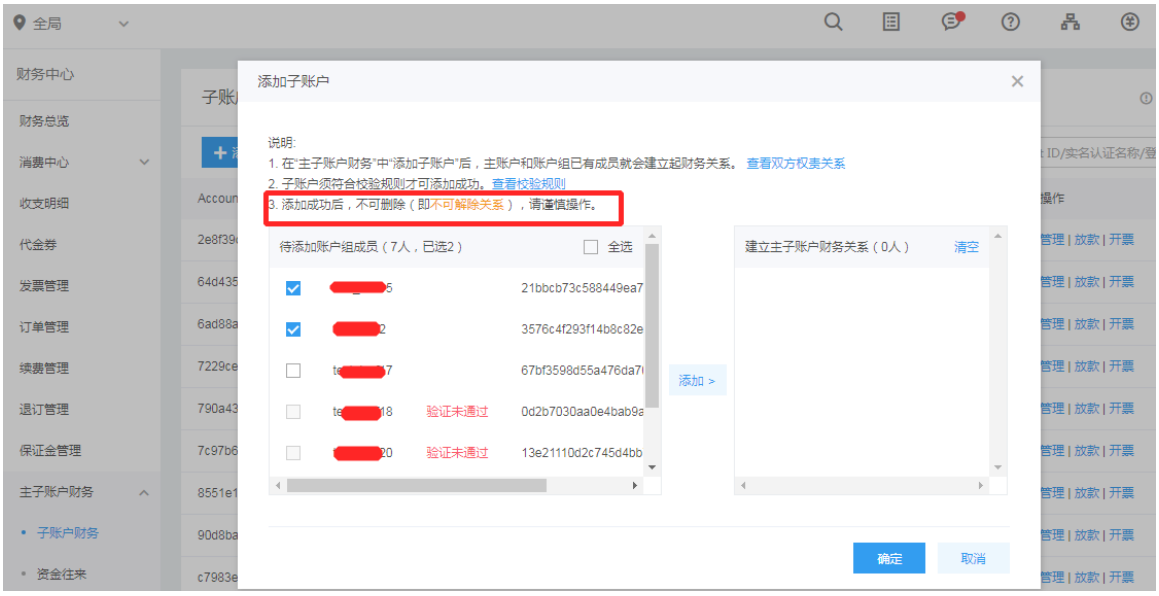
- 子账户实名认证名称需和主账户一致。

- 子账户需满足无账户余额、无线下合同、且身份不为代理商。
- 子账户的开票策略需为按消费开票，且无可开票金额、无欠票。



注意事项二：

本期暂不支持子账户删除并恢复自己独立的财务结算身份，请谨慎操作。



管理子账户财资和账单

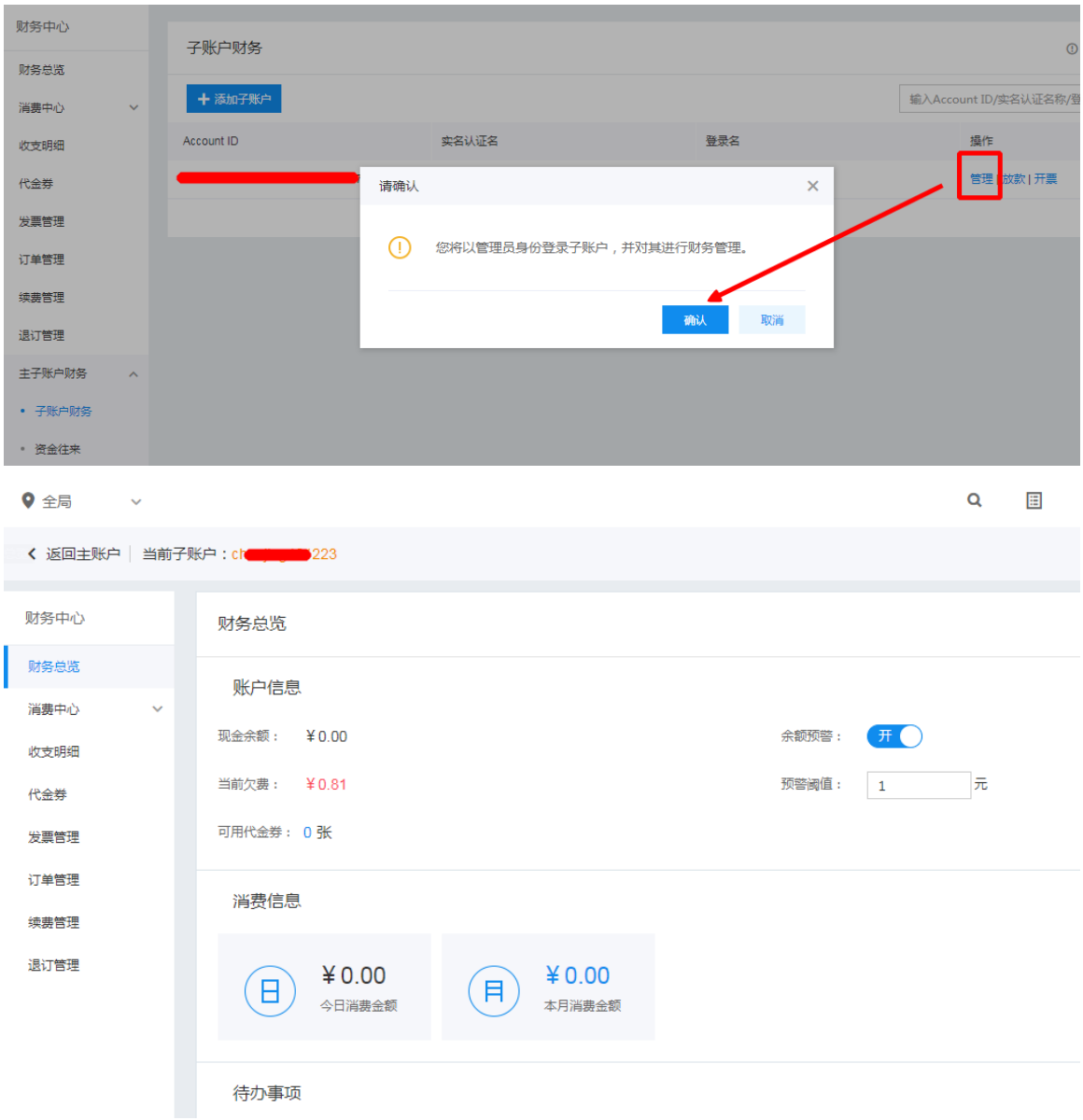
建立主子账户财务关系后，主账户即可管理子账户的财资和账单。

功能点：

- 主账户可查看子账户的财资和账单。
- 主账户可通过对子账户现金划拨的方式，实现对子账户消费账单的支付。

主账户查看子账户的财资和账单

1. 在“子账户财务”中，选择要查看的子账户，点击【管理】后，【确认】即可进入子账户的“财务总览”，可查看子账户的账户信息、消费信息、待办事项。



2. 切换到“消费中心”，可查看子账户的消费总览和消费明细。

3. 切换到“收支明细”，可查看子账户的现金收支和代金券收支情况。

4. 切换到“代金券”，可查看子账户的代金券。

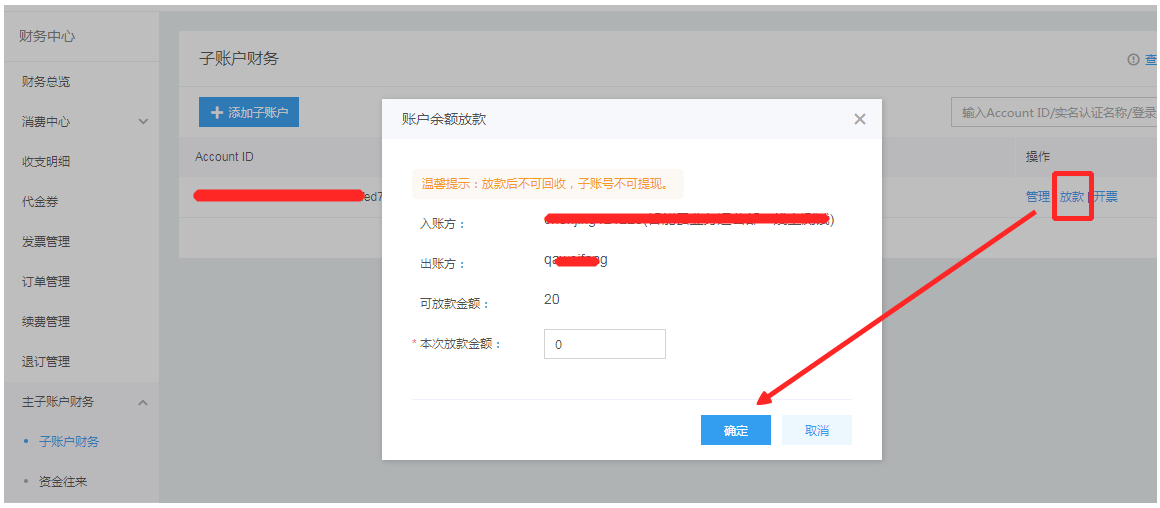
5. 点击【返回主账户】，可回到主账户的“主子账户财务”管理。

主账户现金放款

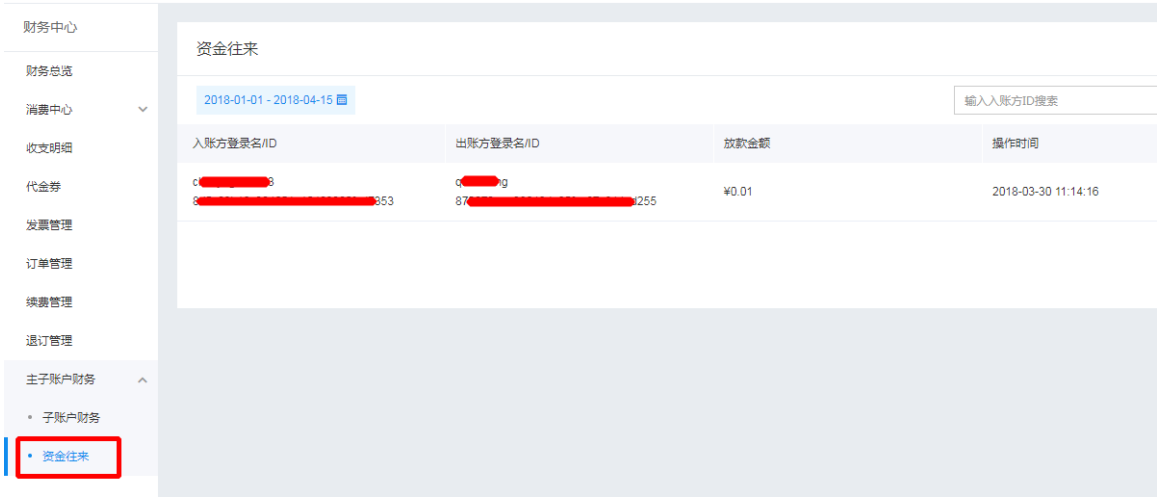
1. 在“子账户财务”中，选择要查看的子账户，点击【放款】，输入放款金额，点击【确定】。

    - 入账方：子账户，即接受放款账户。
    - 出账方：主账户，即放款账户。

> 注意：子账户不可提现；目前不支持主账户对已放款的金额回收。请谨慎操作放款金额。



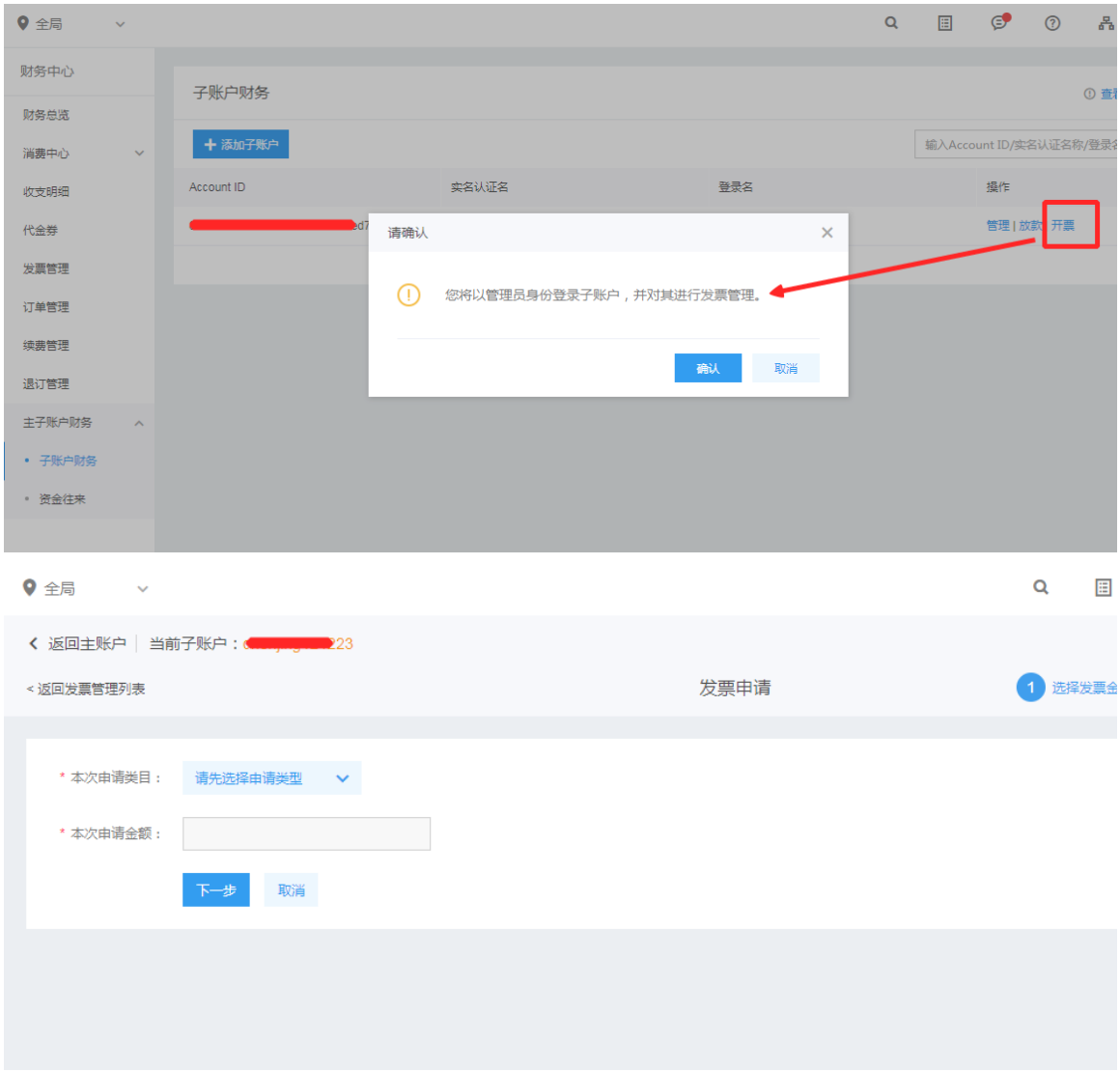
2. 可在“资金往来”中，查看到放款记录。



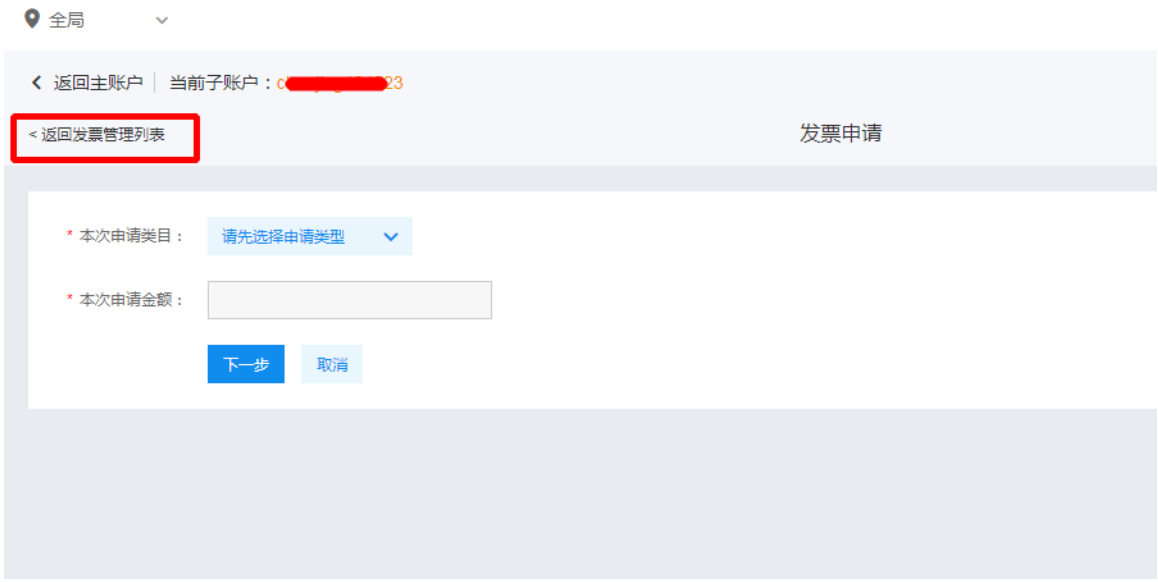
管理子账户发票

建立主子账户财务关系后，主账户即可管理子账户的发票。

1. 在“子账户财务”中，选择要查看的子账户，点击【开票】后，【确认】即可开具发票。



2. 点击【返回发票管理列表】，可操作子账户的全部发票管理功能。



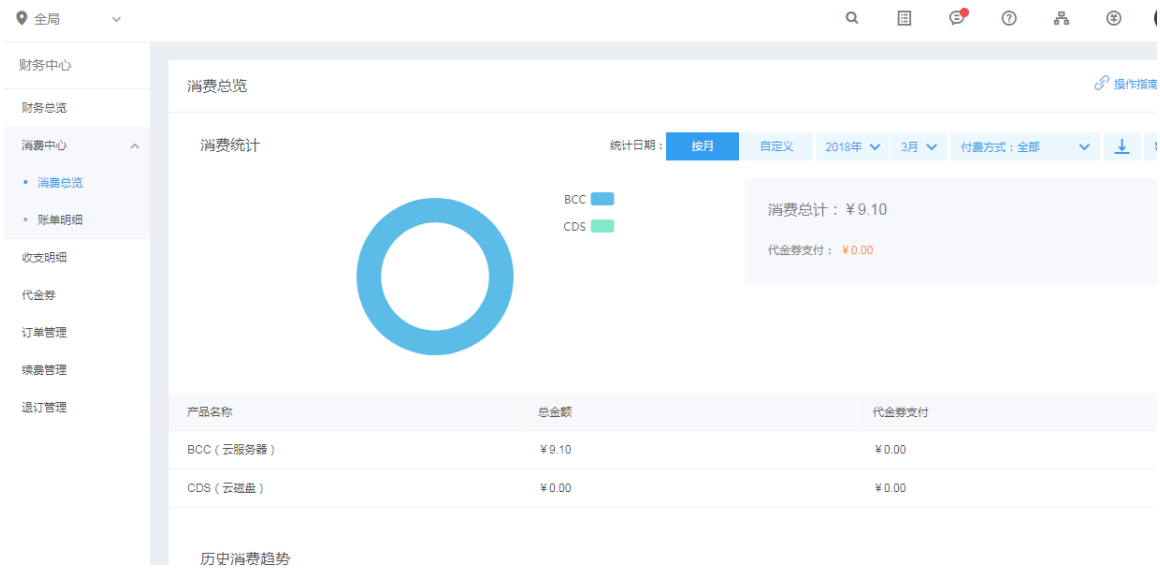
子账户功能与限制

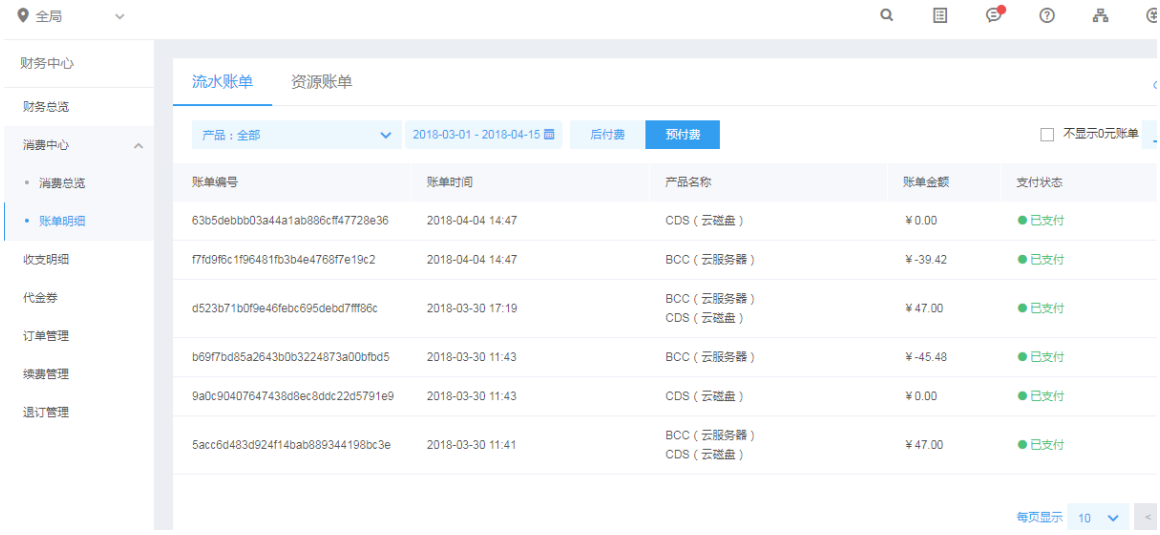
在主子账户财务关系中：

子账户负责使用资源，不可打款来付账单、不可开票。子账户可查看自己账户的费用便于同主账户对账，不可查看账单的支付相关信息（包括折扣、现金支付等记录）。

1. 子账户可以下单购买资源，使用主账户划拨过来的资金支付账单。
2. 子账户打款、管理资金、管理发票功能被禁止。如下图，子账户充值、提现、查看现金收支、管理发票功能不可操作。

3. “消费中心”中，保留子账户账单消费信息，子账户查看账单支付相关信息（包括折扣、现金支付等记录）被禁止。





退出组织

解除子账户

进入企业组织，点击子账户管理，在对应的账户下，点击【删除】，可解除对子账户的管理。

删除组织

在企业组织列表中，选择"设置"，点击【删除组织】，主账户即可删除该组织服务

注意

1.当需要从组织中删除新建账户时，请提交工单联系客户经理补充信息，构成完整账户才能从组织中删除，成为一个独立的账户。

2.已经加入财务管理的主子账户暂不支持删除，详情请参考财务管理。

常见问题

企业组织里子账号为什么无法管理CDN业务?

企业组织的功能是，主账户可以以子账户的身份查看和管理任何子账户的资源，但是，子账号无权查看和管理主账户的资源。如您需要子账号管理主账号中的CDN资源，您可以使用多用户访问控制功能。

如何加入财务管理?

财务管理功能目前在公测阶段，如您有财务管理的需求，请您在工单系统中提交申请/或直接联系您的客户经理提交申请。申请和使用财务管理需满足:

- 实名认证类型必须为企业客户。
- 主账户的实名认证信息和子账户的主体信息必须保持一致。
- 签署线下合同。

加入企业组织的成员账户数量是否有限制?

目前没有。

是否有方式可以限制通过邀请加入企业组织的账户离开组织?

不能。只有主账户新建的账户不可以主动离开组织。

为什么我无法删除快速创建的子账户?

快速创建的子账户因为不信息不全，无法形成一个独立运行的账户，所以如果需要删除，需要提工单补齐账户的信息。

鉴权认证机制

简介

本文档主要针对RESTful API调用者。

API认证简介

用户可以通过认证方式和匿名方式两种方式与百度智能云进行交互。对于认证方式，需要通过使用Access Key Id / Secret Access Key加密的方法来验证某个请求的发送者身份。Access Key Id（AK）用于标示用户，Secret Access Key（SK）是用户用于加密认证字符串和百度智能云用来验证认证字符串的密钥，其中SK必须保密，只有用户和百度智能云知道。

当百度智能云接收到用户的请求后，系统将使用相同的SK和同样的认证机制生成认证字符串，并与用户请求中包含的认证字符串进行比对。如果认证字符串相同，系统认为用户拥有指定的操作权限，并执行相关操作；如果认证字符串不同，系统将忽略该操作并返回错误码。

> **说明：**
>
> 本文档主要针对RESTful API调用者，SDK使用者无需关注。在SDK中已经封装了完整的签名算法，使用者无需自己实现。

API认证优势

API认证将为用户带来以下优势：

- 对于请求者的身份进行验证。
  认证字符串使用指定用户的AK/SK对HTTP请求进行签名，可以起到验证用户身份的作用。关于AK/SK的获取，请参看获取AK/SK。
- 对被传输内容进行保护，防止非法篡改。
  用户基于HTTP请求的指定内容生成认证字符串，如果在传输过程中遭到非法篡改，将导致系统生成的认证字符串与用户生成的认证字符串不匹配，最终导致认证失败。
- 防止重放攻击。
  认证字符串都具有指定的生效时间，一个请求必须要在指定时间内到达百度智能云，否则系统将拒绝该请求。

为了保护用户的SK信息，百度智能云不直接使用SK信息，而是使用SK生成SigningKey，同时在SigningKey中包含有效时间范围。这样可以减少用户因SigningKey丢失带来的安全隐患。

API认证方式

百度智能云通过认证算法对HTTP请求的指定内容进行计算并输出认证字符串用于认证。开发者需要首先将HTTP请求的指定内容连接成字符串，结合百度智能云分配的SK，通过HMAC算法计算密文摘要，这个过程也就是对HTTP请求进行签名过程。百度智能云API使用基于认证字符串的HTTP请求签名机制来验证用户身份。对于每个HTTP请求，都需要携带一个认证字符串然后通过以下两种方式将这个认证字符串包含在请求中：

- 在HTTP Header中包含认证字符串

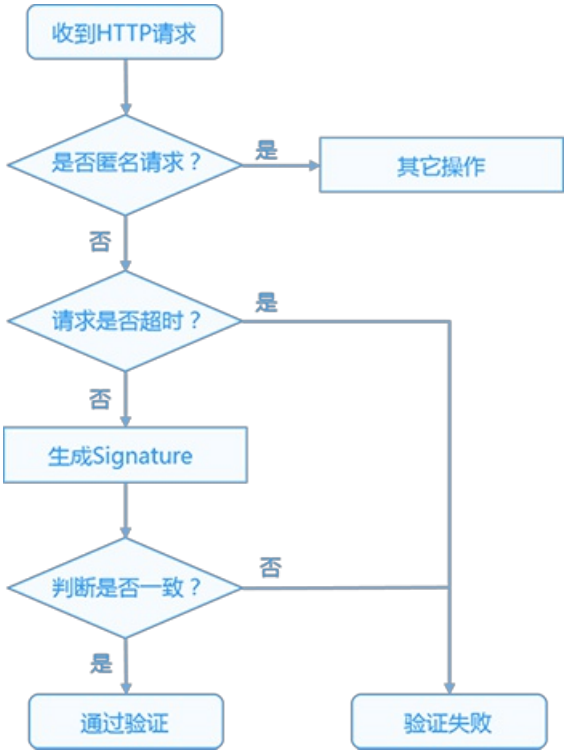
  通常使用的方法是在HTTP请求的Authorization头域中包含认证字符串，除了匿名请求之外，所有与百度智能云的交互都应该包含该字段。更多关于在HTTP Header中包含认证字符串的方法，请参考在Header中包含认证字符串。

- 在URL中包含认证字符串

  用户也可以认证字符串放在HTTP请求Query String的authorization参数中。常用于生成URL给第三方使用的场景，例如要临时把某个数据开放给他人下载。关于如何在URL中包含认证字符串，请参考在URL中包含认证字符串。

用户请求的验证过程

百度智能云API使用基于认证字符串的HTTP请求签名机制来验证用户身份，对于非匿名方式的HTTP请求，都应携带一个认证字符串。当百度智能云收到用户的HTTP请求后，系统将按照下图所示流程进行处理。



1. 判断用户的HTTP请求是否为匿名请求，即用户的请求中是否包含Authorization认证字符串。如果不包含认证字符串，则需要根据不同业务情况，参考其他相关流程进行处理；如果包含认证字符串，则执行下一步操作。

2. 判断用户的请求是否超时，即服务器收到请求的时间需要符合以下要求：

   `{timestamp} - 5分钟 < 服务器接收到请求时间 < {timestamp} + {expirationPeriodInSeconds} + 5分钟`

   其中`timestamp`代表签名生效UTC时间，`expirationPeriodInSeconds`代表签名有效期限。

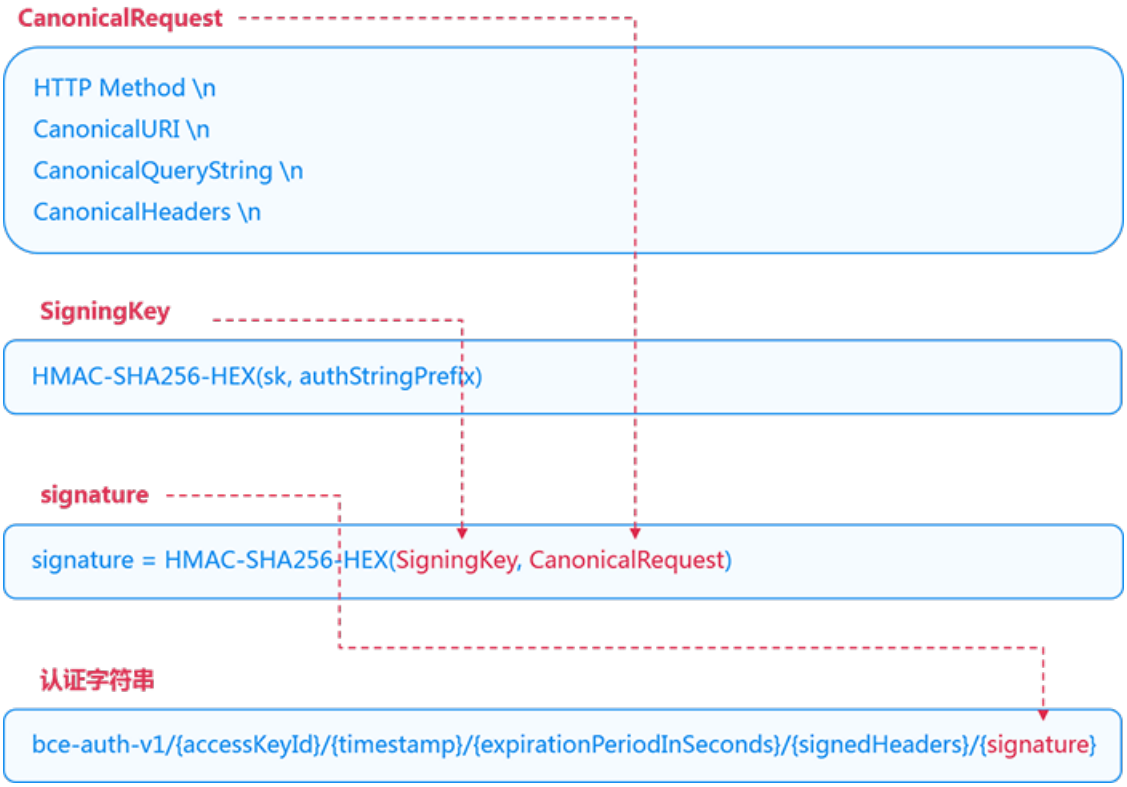
   为了防止用户时钟与服务器时钟不同步而导致的认证失败，此处引入5分钟的宽松系数。如果服务器收到请求的时间不符合以上时间要求，则认为请求超时，拒绝该请求；如果符合上述要求，则执行下一步操作。

3. 基于HTTP请求信息，使用相同的算法，生成Signature字符串。

4. 使用服务器生成的Signature字符串与用户提供的字符串进行比对，如果内容不一致，则认为认证失败，拒绝该请求；如果内容一致，则表示认证成功，系统将按照用户的请求内容进行操作。

认证字符串生成方式

百度智能云认证字符串的生成机制如下图所示：



关于认证字符串的详细生成方案，请参看生成认证字符串。

部分百度智能云产品支持最新的V2认证字符串，其生成方式参看生成V2认证字符串。当前支持V2认证字符串的产品有表格服务BTS。

名词解释

本章节涉及的核心名词解释如下：

- 认证字符串：非匿名请求中必须携带的认证信息。包含生成待签名串CanonicalRequest所必须的信息以及签名摘要signature。
- authStringPrefix：认证字符串的前缀部分。
- canonicalRequest：待签名串。携带经过规范化处理后的请求信息。
- signingKey：签名Key。百度智能云不直接使用SK对待签名串生成摘要。相反的，百度智能云首先使用SK和认证字符串前缀生成signingKey，然后用signingKey对待签名串生成摘要。
- signature：签名摘要。百度智能云使用signingKey对canonicalRequest使用HMAC算法计算签名。

生成认证字符串

概述

在API认证方式中，认证字符串的生成公式如下：

```
bce-auth-v1/{accessKeyId}/{timestamp}/{expirationPeriondInSeconds
}/{signedHeaders}/{signature}
```

其中：

- `accessKeyId`： 是Access Key ID，请参看获取AK/SK来获取。
- `timestamp`：签名生效UTC时间，格式为yyyy-mm-ddThh:mm:ssZ，例如：2015-04-27T08:23:49Z，默认值为当前时间，请注意签名生效时间和发送请求时间务必一致。

- `expirationPeriodInSeconds`：签名有效期限，从`timestamp`所指定的时间开始计算，时间为秒，默认值为1800秒（30）分钟。
- `signedHeaders`：签名算法中涉及到的HTTP头域列表。HTTP头域名字一律要求小写且头域名字之间用分号（;）分隔，如host;range;x-bce-date。列表按照字典序排列。当`signedHeaders`为空时表示取默认值。可以参考示例编写。
- `signature`： 签名。其计算过程较复杂，会在下面进行详细介绍。

生成方式

你可以使用在线签名工具、Postman脚本或是线下编程的方式生成v1版本的认证字符串。

在线签名工具生成

百度智能云提供了在线生成签名工具，用户仅需填写必要头域及AKSK字段，可快速生成认证字符串。

Postman脚本直接生成

百度智能云同时提供了基于Postman脚本的方式，用户仅需按照手册中的步骤，进行简单的配置，就能自动计算百度智能云的签名和认证字符串。

线下编程生成

你也可以通过下面的文字了解认证字符串的具体生成过程， 然后自己编写代码生成相应的认证字符串。

生成认证字符串步骤说明

从认证字符串生成公式可以看出，除了参数signature外，其他参数都是已知的。 下面详细介绍签名的生成过程，进而得出认证字符串。

签名的计算公式为`signature = HMAC-SHA256-HEX(SigningKey,CanonicalRequest)`，从公式可以看出，想要获得签名需要得到`SigningKey`和`CanonicalRequest`两个参数，首先介绍如何获取这两个参数。

生成CanonicalRequest

CanonicalRequest的计算公式为： CanonicalRequest = `HTTP Method + "\n" + CanonicalURI + "\n" + CanonicalQueryString + "\n" + CanonicalHeaders`。

从公式可以看出`CanonicalRequest`由`HTTP Method`、`CanonicalURI`、`CanonicalQueryString`和`CanonicalHeaders`四部分组成，它们的具体含义及获取方式如下。

1. HTTP Method

指HTTP协议中定义的GET、PUT、POST等请求，必须使用全大写的形式。百度智能云API所涉及的HTTP Method有如下五种。

```
GET
POST
PUT
DELETE
HEAD
```

2. CanonicalURI

CanonicalURI是对URL中的绝对路径进行编码后的结果，即`CanonicalURI = UriEncodeExceptSlash(Path)`。要求绝对路径Path必须以“/”开头，不以“/”开头的需要补充上，空路径为“/”。函数UriEncodeExceptSlash的具体含义及功能请参考相关函数。

- 示例：

若URL为`https://bos.cn-n1.baidubce.com/example/测试`，则其URL Path为`/example/测试`，将之规范化得到`CanonicalURI =UriEncodeExceptSlash(/example/测试)= /example/%E6%B5%8B%E8%AF%95`。

3. CanonicalQueryString

`CanonicalQueryString`是对于URL中的Query String（Query String即URL中“？”后面的“key1 = valve1 & key2 = valve2 ”字符串）进行编码后的结果。

编码步骤如下：

1. 提取URL中的Query String项，即URL中“？”后面的“key1 = valve1 & key2 = valve2 ”字符串。
2. 将Query String根据`&`拆开成若干项，每一项是`key=value`或者只有`key`的形式。
3. 对拆开后的每一项进行编码处理，分以下三种情况。
    - 当该项的`key`是`authorization`时，直接忽略该项。
    - 当该项只有`key`时，转换公式为`UriEncode(key) + "="`的形式。
    - 当该项是`key=value`的形式时，转换公式为 `UriEncode(key) + "=" + UriEncode(value)` 的形式。这里`value`可以是空字符串。
4. 将每一项转换后的字符串按照字典顺序（ASCII码由小到大）排序，并使用`&` 符号连接起来，生成相应的`CanonicalQueryString`。

编码示例：

获取URL为`https://bos.cn-n1.baidubce.com/example?text&text1=测试&text10=test`的CanonicalQueryString。

1. 提取URL中的Query String，得到 `text&text1=测试&text10=test`。
2. 根据`&`对Query String进行拆分，得到`text` 、`text1=测试` 、 `text10=test`三项。
3. 对拆分的每一项进行编码。
    - 对`text`项进行编码得到 `UriEncode("text") + "="` = `text=`
    - 对`text1=测试`项进行编码得到 `UriEncode("text1") + "=" + UriEncode("测试")` = `text1=%E6%B5%8B%E8%AF%95`
    - 对`text10=test`项进行编码得到 `UriEncode("text10") + "=" + UriEncode("test")`=> `text10=test`
4. 对上面三项编码后的字符串进行（按照ASCII码由小到大）排序，得到结果是`text10=test` 、`text1=%E6%B5%8B%E8%AF%95` 、`text=` ，然后用`&`连接起来，得到CanonicalQueryString为`text10=test&text1=%E6%B5%8B%E8%AF%95&text=`。

> **说明：**
>
> 1. 函数UriEncode的含义及功能请参考相关解释 。
> 2. 示例中展示了如何处理只有key的项，非英文的value，以及数字和=进行排序。在实际的BCE API中，因为参数起名是规范的，基本不会遇到这样的排序。正常的排序结果和只按照key进行排序是完全一致的。算法中有这个约束主要是出于定义严密性的考虑。

4. CanonicalHeaders

`CanonicalHeaders`是对HTTP请求中的Header部分进行选择性编码的结果。

编码步骤如下：

1. 选择编码的Header。 您可以自行决定哪些Header需要编码。百度智能云API的唯一要求是Host域必须被编码。大多数情况下，我们推荐您对以下Header进行编码：
    - Host
    - Content-Length
    - Content-Type
    - Content-MD5
    - 所有以 `x-bce-` 开头的Header

    > 说明
    >
    > 1. 如果上述Header没有全部出现在您的HTTP请求里面，那么没有出现的部分无需进行编码。**如果发送的请求里包含以上header，出现的header必须签名。**
    > 2. 如果您使用上述推荐范围的Header进行编码，那么认证字符串中的 `{signedHeaders}` 可以直接留空，无需填写。如果您传入了signedHeaders，此时会根据signedHeaders内容进行签名。
    > 3. 您也可以自己选择想要编码的Header。如果您选择了不在推荐范围内的Header进行编码，或者您的HTTP请求包含了推荐范围内的Header但是您选择不对它进行编码，那么您必须在认证字符串中填写 `{signedHeaders}` 。填写方法为，把所有在这一阶段进行了编码的Header名字转换成全小写之后按照字典序排列，然后用分号（`;`）连接。
    > 4. 选择哪些Header进行编码不会影响API的功能，但是如果选择太少则可能遭到中间人攻击，百度智能云要求至少要包含Host域。

2. 对Header进行编码获取CanonicalHeaders，编码步骤如下。
    1. 将Header的名字变成全小写，注意仅改名字。
    2. 将Header的值去掉开头和结尾的空白字符。
    3. 经过上一步之后值为空字符串的Header忽略，其余的转换为 `UriEncode(name) + ":" + UriEncode(value)` 的形式。
    4. 把上面转换后的所有字符串按照字典序进行排序。
    5. 将排序后的字符串按顺序用`\n`符号连接起来得到最终的`CanonicalHeaders`。

> 说明：
> 1.UriEncode的含义及功能请参考相关函数说明。
>
> 2. 很多发送HTTP请求的第三方库，会添加或者删除你指定的header（例如：某些库会删除content-length:0这个header），如果签名错误，请检查您真实发出的http请求的header，看看是否与签名时的header一样。

编码示例1：

该示例演示使用百度智能云推荐范围之外的Header进行编码，此时signedHeaders不能为空（默认值）。在下面的示例中选择对 `Date` 进行编码，忽略 `x-bce-date`。

如下是发送请求的Header:

```
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z
```

1. 选择需要编码的Header，然后把所有名字都改为小写。

```
host: bj.bcebos.com
date: Mon, 27 Apr 2015 16:23:49 +0800
content-type: text/plain
content-length: 8
content-md5: NFzcPqhviddjRNnSOGo4rw==
```

2. 将Header的值去掉开头和结尾的空白字符。

```
host:bj.bcebos.com
date:Mon, 27 Apr 2015 16:23:49 +0800
content-type:text/plain
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw==
```

3. 对每个Header进行UriEncode转换。

```
host:bj.bcebos.com
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
content-type:text%2Fplain
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
```

4. 将步骤3中转换后的所有字符串按照字典序进行排序。

```
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
content-type:text%2Fplain
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
host:bj.bcebos.com
```

5. 将排序后的字符串按顺序用`\n`符号连接起来得到最终结果。

```
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
content-type:text%2Fplain
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
host:bj.bcebos.com
```

同时获得认证字符串的`signedHeaders`内容应该是`content-length;content-md5;content-type;date;host`。

编码示例2:

该示例演示了`CanonicalHeaders`的排序和`signedHeaders`排序不一致的情况。因为BCE API存在允许用户自定义Header的情况，所以这里需要特别注意。

如在BOS的PutObject中允许用户上传自定义meta，为了简明介绍，我们省略大部分Header，假设要编码的Headers如下：

```
Host: bj.bcebos.com
x-bce-meta-data: my meta data
x-bce-meta-data-tag: description
```

按照上面的编码步骤，最终得到的`CanonicalHeaders`是：

```
host:bj.bcebos.com
x-bce-meta-data-tag:description
x-bce-meta-data:my%20meta%20data
```

此时获取的`signedHeaders`是`host;x-bce-meta-data;x-bce-meta-data-tag`。

可以看出`CanonicalHeaders`和`signedHeaders`的排序不一样，这是因为`signedHeaders`是根据Header的name进行排序的， `x-bce-meta-data` 放在 `x-bce-meta-data-tag` 之前。但是在`CanonicalHeaders`中是按照name和value合成的整个字符串进行排序的，因为在`name`和`value`之间还有一个冒号（`:`），而冒号的ASCII码值要大于连字号（`-`）的ASCII码值，因此`x-bce-meta-data`反而放在了`x-bce-meta-data-tag`之后。

生成SigningKey

SigningKey = `HMAC-SHA256-HEX(sk, authStringPrefix)`，其中：

- HMAC-SHA256-HEX是HMAC SHA256算法函数，具体功能及描述参见相关函数说明。
- `sk`为用户的Secret Access Key，可以通过在控制台中进行查询，关于SK的获取方法，请参看获取AK/SK。
- `authStringPrefix`代表认证字符串的前缀部分，即：`bce-auth-v1/{accessKeyId}/{timestamp}/{expirationPeriodInSeconds}`。

生成签名及认证字符串

通过上面的计算得到的SigningKey和CanonicalRequest按照下面公式可以得到签名。
Signature = `HMAC-SHA256-HEX(SigningKey, CanonicalRequest)`
其中：

- HMAC-SHA256-HEX是HMAC SHA256算法函数，具体功能及描述参见相关函数说明。

最后由公式`bce-auth-v1/{accessKeyId}/{timestamp}/{expirationPeriondInSeconds}/{signedHeaders}/{signature}`得到认证字符串。

认证字符串示例

上面介绍如何生成字符串后，在下面示例中详细的演示生成的每一个步骤。示例中假设用户向北京的BOS集群使用UploadPart接口上传一个文件的最后一个Part，内容为`Example`。 用户信息如下

```
Bucket name：test
Object key：myfolder/readme.txt
uploadId：a44cc9bab11cbd156984767aad637851
partNumber：9
Access Key ID：aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Secret Access Key：bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
时间：北京时间2015年4月27日16点23分49秒（转换为UTC时间是2015年4月27日8点23分49秒）
```

则用户发送的HTTP请求为：

```
PUT /v1/test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851 HTTP/1.1
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z

Example
```

1. 生成CanonicalRequest。

   CanonicalRequest由 HTTP Method 、 CanonicalURI 、CanonicalQueryString 、CanonicalHeaders四部分构成。

   ```
   HTTP Method ="PUT"
   CanonicalURI=
   UriEncodeExceptSlash(/v1/test/myfolder/readme.txt)="/v1/test/myfolder/readme.txt
   "
   CanonicalQueryString="partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851"
   CanonicalHeaders="content-length:8
    content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D
    content-type:text%2Fplain
    host:bj.bcebos.com
    x-bce-date:2015-04-27T08%3A23%3A49Z"
   ```

   最后通过公式CanonicalRequest = HTTP Method + "\n" + CanonicalURI + "\n" + CanonicalQueryString + "\n" + CanonicalHeaders得到CanonicalRequest。

   ```
   PUT
   /v1/test/myfolder/readme.txt
   partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851
   content-length:8
   content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D
   content-type:text%2Fplain
   host:bj.bcebos.com
   x-bce-date:2015-04-27T08%3A23%3A49Z
   ```

2. 生成SigningKey

   通过公式SigningKey = HMAC-SHA256-HEX(sk, authStringPrefix)计算SigningKey。

   SigningKey= HMAC-SHA256-HEX("bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb", "bce-auth-v1/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/2015-04-27T08:23:49Z/1800")=1d5ce5f464064cbee060330d973218821825ac6952368a482a592e6615aef479

3. 生成Signature

由公式Signature = HMAC-SHA256-HEX(SigningKey, CanonicalRequest)得到 Signature= d74a04362e6a848f5b39b15421cb449427f419c95a480fd6b8cf9fc783e2999e

4. 生成认证字符串

   Authorization = bce-auth-v1/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/2015-04-27T08:23:49Z/1800//d74a04362e6a848f5b39b15421cb449427f419c95a480fd6b8cf9fc783e2999e

5. 用户最终发送的HTTP请求中包含认证信息

```
PUT /v1/test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851 HTTP/1.1
Authorization:bce-auth-v1/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/2015-04-
27T08:23:49Z/1800//d74a04362e6a848f5b39b15421cb449427f419c95a480fd6b8cf9fc783e299
e
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z


Example
```

注意：

超时时间1800与签名结果之间为两个"/"，含义是使用默认签名方式，signedHeaders内容留空。

相关函数说明

函数名	功能描述
HMAC-SHA256-HEX()	调用HMAC SHA256算法，根据开发者提供的密钥（key）和密文（message）输出密文摘要，并把结果转换为小写形式的十六进制字符串。
Lowercase()	将字符串全部变成小写。
Trim()	去掉字符串开头和结尾的空白字符。
UriEncode()	RFC 3986规定，"URI非保留字符"包括以下字符：字母（A-Z，a-z）、数字（0-9）、连字号（-）、点号（.）、下划线（_）、波浪线（~），算法实现如下： 1. 将字符串转换成UTF-8编码的字节流 2. 保留所有“URI非保留字符”原样不变 3. 对其余字节做一次RFC 3986中规定的百分号编码（Percent-encoding），即一个“%”后面跟着两个表示该字节值的十六进制字母，字母一律采用大写形式。
UriEncodeExceptSlash()	与UriEncode() 类似，区别是斜杠（/）不做编码。一个简单的实现方式是先调用UriEncode()，然后把结果中所有的`%2F`都替换为`/`

UriEncode()函数参考代码如下：

```
public static String uri-encode(CharSequence input, boolean encodeSlash) {
    StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch >=
'0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') {
                result.append(ch);
            } else if (ch == '/') {
                result.append(encodeSlash ? "%2F" : ch);
            } else {
                result.append(toHexUTF8(ch));
            }
        }
          return result.toString();
}
```

生成V2认证字符串

概述

V2认证字符串是百度智能云最新的API认证协议方式。基于V2协议，服务可提供更高的请求响应性能，用户也可更大程度保证自己的密钥安全。目前表格服务BTS已经支持v2协议的认证字符串，具体参见BTS API参考。

V2版本认证字符串的生成公式如下：

`bce-auth-v2/{accessKeyId}/{date}/{region}/{service}/{signedHeaders}/{signature}`

其中：

- `accessKeyId`： 是Access Key ID，请参看获取AK/SK来获取。
- `date`：签名的UTC日期，格式为yyyymmdd，例如：20150427。
- `region`：所请求服务资源所在的区域，小写格式。例如：bj。
- `service`：所请求的服务名称，小写格式。例如：bos。
- `signedHeaders`：签名算法中涉及到的HTTP头域列表。HTTP头域名字一律要求小写且头域名字之间用分号（;）分隔，如host;range;x-bce-date。列表按照字典序排列。当`signedHeaders`为空时表示取默认值。
- `signature`： 签名，是认证字符串的重要组成部分，其计算过程较复杂，会在文中详细介绍签名的生成过程。

签名的生成

V2认证签名的计算公式为`signature = HMAC-SHA256-HEX(SigningKey,CanonicalRequest)`，从公式可以看出，想要获得签名需要得到`SigningKey`和`CanonicalRequest`两个参数，接下来讲解如何获取这两个参数。

生成CanonicalRequest

CanonicalRequest的计算公式为： CanonicalRequest = `HTTP Method + "\n" + CanonicalURI + "\n" + CanonicalQueryString + "\n" + CanonicalHeaders`，

HTTP Method

指HTTP协议中定义的GET、PUT、POST等请求，必须使用全大写的形式。百度智能云API所涉及的HTTP Method有如下五种。

- GET
- POST
- PUT
- DELETE
- HEAD

CanonicalURI

`CanonicalURI`是对URL中的绝对路径进行编码后的结果，即`CanonicalURI` = `UriEncodeExceptSlash(Path)`。要求绝对路径`Path`必须以“/”开头，不以“/”开头的需要补充上，空路径为“/”，。

- 相关举例：

若URL为`https://bos.cn-n1.baidubce.com/example/测试`，则其URL Path为`/example/测试`，将之规范化得到`CanonicalURI = /example/%E6%B5%8B%E8%AF%95`。

CanonicalQueryString

`CanonicalQueryString`对于URL中的Query String（Query String即URL中“？”后面的“key1 = valve1 & key2 = valve2 ”字符串）进行编码后的结果。

编码步骤如下：

1. 提取URL中的Query String项，即URL中“？”后面的“key1 = valve1 & key2 = valve2 ”字符串
2. 将Query String根据`&`拆开成若干项，每一项是`key=value`或者只有`key`的形式。
3. 对拆开后的每一项进行编码处理，分以下三种情况。
    - 当该项的`key`是`authorization`时，直接忽略该项。
    - 当该项只有`key`时，转换公式为`UriEncode(key) + "="`的形式。
    - 当该项是`key=value`的形式时，转换公式为 `UriEncode(key) + "=" + UriEncode(value)` 的形式。这里`value`可以是空字符串。
4. 将每一项转换后的字符串按照字典顺序（ASCII码由小到大）排序，并使用`&` 符号连接起来，生成相应的`CanonicalQueryString`。

编码示例

获取URL为`https://bos.cn-n1.baidubce.com/example?text&text1=测试&text10=test`的CanonicalQueryString

1. 提取URL中的Query String，得到 `text&text1=测试&text10=test`。
2. 根据`&`对Query String进行拆分，得到`text` 、`text1=测试` 、 `text10=test`三项。
3. 对拆分的每一项进行编码。
    - 对`text`项进行编码得到 `UriEncode("text") + "="` = `text=`
    - 对`text1=测试`项进行编码得到 `UriEncode("text1") + "=" + UriEncode("测试")` = `text1=%E6%B5%8B%E8%AF%95`
    - 对`text10=test`项进行编码得到 `UriEncode("text10") + "=" + UriEncode("test")`=> `text10=test`

4. 对上面三项编码后的字符串进行（按照ASCII码由小到大）排序，得到结果是`text10=test`、`text1=%E6%B5%8B%E8%AF%95`、`text=`，然后用&连接起来，得到CanonicalQueryString为`text10=test&text1=%E6%B5%8B%E8%AF%95&text=`。

> **说明：**
>
> 1. 函数UriEncode的含义及功能请参考相关解释 。
> 2. 示例中展示了如何处理只有key的项，非英文的value，以及数字和=进行排序。在实际的BCE API中，因为参数起名是规范的，基本不会遇到这样的排序。正常的排序结果和只按照key进行排序是完全一致的。算法中有这个约束主要是出于定义严密性的考虑。

CanonicalHeaders

`CanonicalHeaders`是对HTTP请求中的Header部分进行选择性编码的结果。

编码步骤如下：

1. 选择编码的Header。 您可以自行决定哪些Header需要编码。百度智能云API的唯一要求是Host域必须被编码。大多数情况下，我们推荐您对以下Header进行编码：
   - Host
   - Content-Length
   - Content-Type
   - Content-MD5
   - 所有以 `x-bce-` 开头的Header

   > 说明
   >
   > 1. 如果上述Header没有全部出现在您的HTTP请求里面，那么没有出现的部分无需进行编码。**如果发送的请求里包含以上header，出现的header必须签名。**
   > 2. 如果您使用上述推荐范围的Header进行编码，那么认证字符串中的 `{signedHeaders}` 可以直接留空，无需填写。如果您传入了signedHeaders，此时会根据signedHeaders内容进行签名。
   > 3. 您也可以自己选择想要编码的Header。如果您选择了不在推荐范围内的Header进行编码，或者您的HTTP请求包含了推荐范围内的Header但是您选择不对它进行编码，那么您必须在认证字符串中填写 `{signedHeaders}` 。填写方法为，把所有在这一阶段进行了编码的Header名字转换成全小写之后按照字典序排列，然后用分号（`;`）连接。
   > 4. 选择哪些Header进行编码不会影响API的功能，但是如果选择太少则可能遭到中间人攻击。

2. 对Header进行编码获取CanonicalHeaders，编码步骤如下。
   1. 将Header的名字变成全小写，注意仅改名字。
   2. 将Header的值去掉开头和结尾的空白字符。
   3. 经过上一步之后值为空字符串的Header忽略，其余的转换为 `UriEncode(name) + ":" + UriEncode(value)` 的形式。
   4. 把上面转换后的所有字符串按照字典序进行排序。

5. 将排序后的字符串按顺序用`\n`符号连接起来得到最终的`CanonicalHeaders`。

> 说明：
> 1.UriEncode的含义及功能请参考相关函数说明。
> 2. 很多发送HTTP请求的第三方库，会添加或者删除你指定的header（例如：某些库会删除content-length:0这个header），如果签名错误，请检查您真实发出的http请求的header，看看是否与签名时的header一样。

编码示例1

该示例演示使用百度智能云推荐范围之外的Header进行编码，此时signedHeaders不能为空（默认值）。在下面的示例中选择对 `Date` 进行编码，忽略 `x-bce-date`。

如下是发送请求的Header:

```
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z
```

1. 选择需要编码的Header，然后把所有名字都改为小写。

```
host: bj.bcebos.com
date: Mon, 27 Apr 2015 16:23:49 +0800
content-type: text/plain
content-length: 8
content-md5: NFzcPqhviddjRNnSOGo4rw==
```

2. 将Header的值去掉开头和结尾的空白字符。

```
host:bj.bcebos.com
date:Mon, 27 Apr 2015 16:23:49 +0800
content-type:text/plain
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw==
```

3. 对每个Header进行UriEncode转换。

```
host:bj.bcebos.com
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
content-type:text%2Fplain
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
```

4. 将步骤3中转换后的所有字符串按照字典序进行排序。

```
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
content-type:text%2Fplain
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
host:bj.bcebos.com
```

5. 将排序后的字符串按顺序用\n符号连接起来得到最终结果。

```
content-length:8
content-md5:NFzcPqhviddjRNnSOGo4rw%3D%3D?
content-type:text%2Fplain
date:Mon%2C%2027%20Apr%202015%2016%3A23%3A49%20%2B0800
host:bj.bcebos.com
```

同时获得认证字符串的`signedHeaders`内容应该是`content-length;content-md5;content-type;date;host`。

编码示例2

该示例演示了`CanonicalHeaders`的排序和`signedHeaders`排序不一致的情况。因为BCE API存在允许用户自定义Header的情况，所以这里需要特别注意。

如在BOS的PutObject中允许用户上传自定义meta，为了简明介绍，我们省略大部分Header，假设要编码的Headers如下：

```
Host: bj.bcebos.com
x-bce-meta-data: my meta data
x-bce-meta-data-tag: description
```

按照上面的编码步骤，最终得到的`CanonicalHeaders`是：

```
host:bj.bcebos.com
x-bce-meta-data-tag:description
x-bce-meta-data:my%20meta%20data
```

此时获取的`signedHeaders`是`host;x-bce-meta-data;x-bce-meta-data-tag`。

可以看出`CanonicalHeaders`和`signedHeaders`的排序不一样，这是因为`signedHeaders`是根据Header的name进行排序的， `x-bce-meta-data` 放在 `x-bce-meta-data-tag` 之前。但是在`CanonicalHeaders`中是按照name和value合成的整个字符串进行排序的，因为在`name`和`value`之间还有一个冒号（`:`），而冒号的ASCII码值要大于连字号（`-`）的ASCII码值，因此`x-bce-meta-data`反而放在了`x-bce-meta-data-tag`之后。

生成SigningKey

SigningKey = `HMAC-SHA256-HEX(sk, authStringPrefix)`，其中：

- `sk`为用户的Secret Access Key，可以通过在控制台中进行查询，关于SK的获取方法，请参看获取AK/SK。
- `authStringPrefix`代表认证字符串的前缀部分，即：`bce-auth-v2/{accessKeyId}/{date}/{region}/{service}`。

生成Signature及认证字符串

通过上面的计算得到的SigningKey和CanonicalRequest按照下面公式可以得到签名。 Signature = `HMAC-SHA256-HEX(SigningKey, CanonicalRequest)`

其中：

- HMAC-SHA256-HEX是HMAC SHA256算法函数，具体功能及描述参见相关函数说明。

最后根据公式`bce-auth-`

`v2/{accessKeyId}/{date}/{region}/{service}/{signedHeaders}/{signature}`获得认证字符串。

相关函数说明

函数名	功能描述
HMAC-SHA256-HEX()	调用HMAC SHA256算法，根据开发者提供的密钥（key）和密文（message）输出密文摘要，并把结果转换为小写形式的十六进制字符串。
Lowercase()	将字符串全部变成小写。
Trim()	去掉字符串开头和结尾的空白字符。
UriEncode()	RFC 3986规定，"URI非保留字符"包括以下字符：字母（A-Z，a-z）、数字（0-9）、连字号（-）、点号（.）、下划线（_）、波浪线（~），算法实现如下： 1. 将字符串转换成UTF-8编码的字节流 2. 保留所有“URI非保留字符”原样不变 3. 对其余字节做一次RFC 3986中规定的百分号编码（Percent-encoding），即一个“%”后面跟着两个表示该字节值的十六进制字母，字母一律采用大写形式。
UriEncodeExceptSlash()	与UriEncode() 类似，区别是斜杠（/）不做编码。一个简单的实现方式是先调用UriEncode()，然后把结果中所有的`%2F`都替换为`/`

UriEncode()函数参考代码如下：

```
public static String uri-encode(CharSequence input, boolean encodeSlash) {
    StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || (ch >=
'0' && ch <= '9') || ch == '_' || ch == '-' || ch == '~' || ch == '.') {
                result.append(ch);
            } else if (ch == '/') {
                result.append(encodeSlash ? "%2F" : ch);
            } else {
                result.append(toHexUTF8(ch));
            }
        }
          return result.toString();
}
```

V2版本与V1版本认证字符串对比

V2认证字符串相比V1版本认证字符串，主要有以下变化：

- 认证字符串的格式调整。V2认证字符串中比V1认证字符串减少了签名时间戳和过期时间，增加了签名日期和生效的服务区域限定。具体格式为：`bce-auth-v2/{accessKeyId}/{date}/{region}/{service}/{signedHeaders}/{signature}`。
- signingKey生成方式变为：`HMAC-SHA256-HEX(sk, "bce-auth-v2/{accessKeyId}/{date}/{region}/{service}")`
- 必须签名的头域/请求参数。认证字符串中减少的签名时间戳，要求必须以`x-bce-date`请求头或QueryString参数形式，计入签名；请求中如通过`x-bce-expiration`设置过期时间，则也必须计入签名。请求中未指定过期时间的，则默认请求15分钟过期。

相同点： V2中签名的计算过程，同样包含CanonicalRequest、SigningKey、Signature的生成过程，这些与V1版本基本一致。

在Header中包含认证字符串

用户可以在HTTP请求的Header中包含Authorization信息，表明这个消息已被授权，即：在HTTP Header中加入`Authorization: <认证字符串>`。关于认证字符串的生成方法请参看生成认证字符串。

实现方式

基于生成认证字符串中的“认证字符串生成举例”的结果，可以构建如下HTTP Header用于验证。

```
PUT /v1/test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851 HTTP/1.1
Authorization: bce-auth-v1/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/2015-04-
27T08:23:49Z/1800//d74a04362e6a848f5b39b15421cb449427f419c95a480fd6b8cf9fc783e2999
e
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z

Example
```

在URL中包含认证字符串

除了使用Authorization Header，用户还可以在URL中加入认证字符串，具体方法是在URL的Query String中加入`authorization = <认证字符串>`，关于认证字符串的生成方法请参看生成认证字符串。

实现方式

基于生成认证字符串中的“认证字符串生成举例”的结果，可以构建如下URL用于验证。

```
https://bj.bcebos.com/test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851&authorization=bce-auth-
v1/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa/2015-04-
27T08:23:49Z/1800//d74a04362e6a848f5b39b15421cb449427f419c95a480fd6b8cf9fc783e2999
e
```

在URL中包含认证字符串常用于生成URL给第三方使用的场景，例如要临时把某个数据开放给他人下载。

Sample-Code

Python示例

假设用户向北京的BOS集群使用UploadPart接口上传一个文件的最后一个Part，内容为`Example`。

- Bucket name：test
- Object key：myfolder/readme.txt
- uploadId：a44cc9bab11cbd156984767aad637851
- partNumber：9
- Access Key ID：aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

- Secret Access Key：bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
- 时间：北京时间2015年4月27日16点23分49秒（转换为UTC时间是2015年4月27日8点23分49秒）

则其HTTP请求如下：

```
PUT /test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851 HTTP/1.1
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z

Example
```

用户可根据上述HTTP请求填写以下函数中的各个字段。

```
 if __name__ == "__main__":
     credentials =
BceCredentials("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa","bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
)
     http_method = "PUT"
     path = "/test/myfolder/readme.txt"
     headers = {"host": "bj.bcebos.com",
                "content-length": 8,
                "content-md5": "NFzcPqhviddjRNnSOGo4rw==",
                "content-type":"text/plain",
                "x-bce-date": "2015-04-27T08:23:49Z"}
     params = {"partNumber": 9,
               "uploadId": "a44cc9bab11cbd156984767aad637851"}
     timestamp = 1430123029
     result = sign(credentials, http_method, path, headers, params, timestamp)
     print result
```

请点击此处获取完整代码

完整代码内容如下：

```
# -*- coding: UTF-8 -*-
import hashlib
import hmac
import string
import datetime


AUTHORIZATION = "authorization"
BCE_PREFIX = "x-bce-"
DEFAULT_ENCODING = 'UTF-8'


# 保存AK/SK的类
class BceCredentials(object):
    def __init__(self, access_key_id, secret_access_key):
        self.access_key_id = access_key_id
        self.secret_access_key = secret_access_key


# 根据RFC 3986，除了：
```

```
# 根据RFC 3986，除了：
#   1.大小写英文字符
#   2.阿拉伯数字
#   3.点'.'、波浪线'~'、减号'-'以及下划线'_'
# 以外都要编码
RESERVED_CHAR_SET = set(string.ascii_letters + string.digits + '.~-_')
def get_normalized_char(i):
    char = chr(i)
    if char in RESERVED_CHAR_SET:
        return char
    else:
        return '%%%02X' % i
NORMALIZED_CHAR_LIST = [get_normalized_char(i) for i in range(256)]


# 正规化字符串
def normalize_string(in_str, encoding_slash=True):
    if in_str is None:
        return ''

    # 如果输入是unicode，则先使用UTF8编码之后再编码
    in_str = in_str.encode(DEFAULT_ENCODING) if isinstance(in_str, unicode) else
str(in_str)

    # 在生成规范URI时。不需要对斜杠'/'进行编码，其他情况下都需要
    if encoding_slash:
        encode_f = lambda c: NORMALIZED_CHAR_LIST[ord(c)]
    else:
        # 仅仅在生成规范URI时。不需要对斜杠'/'进行编码
        encode_f = lambda c: NORMALIZED_CHAR_LIST[ord(c)] if c != '/' else c

    # 按照RFC 3986进行编码
    return ''.join([encode_f(ch) for ch in in_str])


# 生成规范时间戳
def get_canonical_time(timestamp=0):
    # 不使用任何参数调用的时候返回当前时间
    if timestamp == 0:
        utctime = datetime.datetime.utcnow()
    else:
        utctime = datetime.datetime.utcfromtimestamp(timestamp)

    # 时间戳格式：[year]-[month]-[day]T[hour]:[minute]:[second]Z
    return "%04d-%02d-%02dT%02d:%02d:%02dZ" % (
        utctime.year, utctime.month, utctime.day,
        utctime.hour, utctime.minute, utctime.second)


# 生成规范URI
def get_canonical_uri(path):
    # 规范化URI的格式为：/{bucket}/{object}，并且要对除了斜杠"/"之外的所有字符编码
    return normalize_string(path, False)


# 生成规范query string
def get_canonical_querystring(params):
    if params is None:
        return ''

    # 除了authorization之外，所有的query string全部加入编码
    result = ['%s=%s' % (normalize_string(k), normalize_string(v)) for k, v in
params.items() if k.lower != AUTHORIZATION]
```

```
    # 按字典序排序
    result.sort()

    # 使用&符号连接所有字符串并返回
    return '&'.join(result)


# 生成规范header
def get_canonical_headers(headers, headers_to_sign=None):
    headers = headers or {}

    # 没有指定header_to_sign的情况下，默认使用：
    #   1.host
    #   2.content-md5
    #   3.content-length
    #   4.content-type
    #   5.所有以x-bce-开头的header项
    # 生成规范header
    if headers_to_sign is None or len(headers_to_sign) == 0:
        headers_to_sign = {"host", "content-md5", "content-length", "content-
type"}

    # 对于header中的key，去掉前后的空白之后需要转化为小写
    # 对于header中的value，转化为str之后去掉前后的空白
    f = lambda (key, value): (key.strip().lower(), str(value).strip())

    result = []
    for k, v in map(f, headers.iteritems()):
        # 无论何种情况，以x-bce-开头的header项都需要被添加到规范header中
        if k.startswith(BCE_PREFIX) or k in headers_to_sign:
            result.append("%s:%s" % (normalize_string(k), normalize_string(v)))

    # 按照字典序排序
    result.sort()

    # 使用\n符号连接所有字符串并返回
    return '\n'.join(result)


# 签名主算法
def sign(credentials, http_method, path, headers, params,
         timestamp=0, expiration_in_seconds=1800, headers_to_sign=None):
    headers = headers or {}
    params = params or {}

    # 1.生成sign key
    # 1.1.生成auth-string，格式为：bce-auth-
v1/{accessKeyId}/{timestamp}/{expirationPeriodInSeconds}
    sign_key_info = 'bce-auth-v1/%s/%s/%d' % (
        credentials.access_key_id,
        get_canonical_time(timestamp),
        expiration_in_seconds)
    # 1.2.使用auth-string加上SK，用SHA-256生成sign key
    sign_key = hmac.new(
        credentials.secret_access_key,
        sign_key_info,
        hashlib.sha256).hexdigest()

    # 2.生成规范化uri
    canonical_uri = get_canonical_uri(path)
```

```
    # 3.生成规范化query string
    canonical_querystring = get_canonical_querystring(params)

    # 4.生成规范化header
    canonical_headers = get_canonical_headers(headers, headers_to_sign)

    # 5.使用'\n'将HTTP METHOD和2、3、4中的结果连接起来，成为一个大字符串
    string_to_sign = '\n'.join(
        [http_method, canonical_uri, canonical_querystring, canonical_headers])

    # 6.使用5中生成的签名串和1中生成的sign key，用SHA-256算法生成签名结果
    sign_result = hmac.new(sign_key, string_to_sign, hashlib.sha256).hexdigest()

    # 7.拼接最终签名结果串
    if headers_to_sign:
        # 指定header to sign
        result = '%s/%s/%s' % (sign_key_info, ';'.join(headers_to_sign),
sign_result)
    else:
        # 不指定header to sign情况下的默认签名结果串
        result = '%s//%s' % (sign_key_info, sign_result)

    return result

 if __name__ == "__main__":
    credentials =
BceCredentials("aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa","bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
)
    http_method = "PUT"
    path = "/test/myfolder/readme.txt"
    headers = {"host": "bj.bcebos.com",
               "content-length": 8,
               "content-md5": "NFzcPqhviddjRNnSOGo4rw==",
               "content-type":"text/plain",
               "x-bce-date": "2015-04-27T08:23:49Z"}
    params = {"partNumber": 9,
              "uploadId": "a44cc9bab11cbd156984767aad637851"}
    timestamp = 1430123029
    result = sign(credentials, http_method, path, headers, params, timestamp)
    print result
```

Php示例

假设用户向北京的BOS集群使用UploadPart接口上传一个文件的最后一个Part，内容为`Example`。

- Bucket name：test
- Object key：myfolder/readme.txt
- uploadId：a44cc9bab11cbd156984767aad637851
- partNumber：9
- Access Key ID：aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
- Secret Access Key：bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
- 时间：北京时间2015年4月27日16点23分49秒（转换为UTC时间是2015年4月27日8点23分49秒）

则其HTTP请求如下：

```
PUT /test/myfolder/readme.txt?
partNumber=9&uploadId=a44cc9bab11cbd156984767aad637851 HTTP/1.1
Host: bj.bcebos.com
Date: Mon, 27 Apr 2015 16:23:49 +0800
Content-Type: text/plain
Content-Length: 8
Content-Md5: NFzcPqhviddjRNnSOGo4rw==
x-bce-date: 2015-04-27T08:23:49Z

Example
```

签名示范代码

```
$signer = new SampleSigner();
$credentials = array("ak" => "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa","sk" =>
"bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb");
$httpMethod = "PUT";
$path = "/v1/test/myfolder/readme.txt";
$headers = array("Host" => "bj.bcebos.com",
                "Content-Length" => 8,
                "Content-MD5" => "NFzcPqhviddjRNnSOGo4rw==",
                "Content-Type" => "text/plain",
                "x-bce-date" => "2015-04-27T08:23:49Z");
$params = array("partNumber" => 9, "uploadId" =>
"a44cc9bab11cbd156984767aad637851");
$timestamp = new \DateTime();
$timestamp->setTimestamp(1430123029);
$options = array(SignOption::TIMESTAMP => $timestamp);
$ret = $signer->sign($credentials, $httpMethod, $path, $headers, $params,
$options);
print $ret;
```

请点击此处获取完整代码

完整代码内容如下：

```
<?php
/*
* Copyright (c) 2014 Baidu.com, Inc. All Rights Reserved
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not
* use this file except in compliance with the License. You may obtain a copy of
* the License at
*
* Http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
* distributed under the License is distributed on an "AS IS" BASIS, WITHOUT
* WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the
* License for the specific language governing permissions and limitations under
* the License.
*/

namespace BaiduBce\Auth;

class SignOption
{
    const EXPIRATION_IN_SECONDS = 'expirationInSeconds';

    const HEADERS_TO_SIGN = 'headersToSign';
```

```
    const TIMESTAMP = 'timestamp';

    const DEFAULT_EXPIRATION_IN_SECONDS = 1800;

    const MIN_EXPIRATION_IN_SECONDS = 300;

    const MAX_EXPIRATION_IN_SECONDS = 129600;
}

class HttpUtil
{
    // 根据RFC 3986，除了：
    //   1.大小写英文字符
    //   2.阿拉伯数字
    //   3.点'.'、波浪线'~'、减号'-'以及下划线'_'
    // 以外都要编码
    public static $PERCENT_ENCODED_STRINGS;

    //填充编码数组
    public static function __init()
    {
        HttpUtil::$PERCENT_ENCODED_STRINGS = array();
        for ($i = 0; $i < 256; ++$i) {
            HttpUtil::$PERCENT_ENCODED_STRINGS[$i] = sprintf("%%%02X", $i);
        }

        //a-z不编码
        foreach (range('a', 'z') as $ch) {
            HttpUtil::$PERCENT_ENCODED_STRINGS[ord($ch)] = $ch;
        }

        //A-Z不编码
        foreach (range('A', 'Z') as $ch) {
            HttpUtil::$PERCENT_ENCODED_STRINGS[ord($ch)] = $ch;
        }

        //0-9不编码
        foreach (range('0', '9') as $ch) {
            HttpUtil::$PERCENT_ENCODED_STRINGS[ord($ch)] = $ch;
        }

        //以下4个字符不编码
        HttpUtil::$PERCENT_ENCODED_STRINGS[ord('-')] = '-';
        HttpUtil::$PERCENT_ENCODED_STRINGS[ord('.')] = '.';
        HttpUtil::$PERCENT_ENCODED_STRINGS[ord('_')] = '_';
        HttpUtil::$PERCENT_ENCODED_STRINGS[ord('~')] = '~';
    }

    //在uri编码中不能对'/'编码
    public static function urlEncodeExceptSlash($path)
    {
        return str_replace("%2F", "/", HttpUtil::urlEncode($path));
    }

    //使用编码数组编码
    public static function urlEncode($value)
    {
        $result = '';
        for ($i = 0; $i < strlen($value); ++$i) {
            $result .= HttpUtil::$PERCENT_ENCODED_STRINGS[ord($value[$i])];
        }
```

```
        return $result;
    }

    //生成标准化QueryString
    public static function getCanonicalQueryString(array $parameters)
    {
        //没有参数，直接返回空串
        if (count($parameters) == 0) {
            return '';
        }

        $parameterStrings = array();
        foreach ($parameters as $k => $v) {
            //跳过Authorization字段
            if (strcasecmp('Authorization', $k) == 0) {
                continue;
            }
            if (!isset($k)) {
                throw new \InvalidArgumentException(
                    "parameter key should not be null"
                );
            }
            if (isset($v)) {
                //对于有值的，编码后放在=号两边
                $parameterStrings[] = HttpUtil::urlEncode($k)
                    . '=' . HttpUtil::urlEncode((string) $v);
            } else {
                //对于没有值的，只将key编码后放在=号的左边，右边留空
                $parameterStrings[] = HttpUtil::urlEncode($k) . '=';
            }
        }
        //按照字典序排序
        sort($parameterStrings);

        //使用'&'符号连接它们
        return implode('&', $parameterStrings);
    }

    //生成标准化uri
    public static function getCanonicalURIPath($path)
    {
        //空路径设置为'/'
        if (empty($path)) {
            return '/';
        } else {
            //所有的uri必须以'/'开头
            if ($path[0] == '/') {
                return HttpUtil::urlEncodeExceptSlash($path);
            } else {
                return '/' . HttpUtil::urlEncodeExceptSlash($path);
            }
        }
    }

    //生成标准化http请求头串
    public static function getCanonicalHeaders($headers)
    {
        //如果没有headers，则返回空串
        if (count($headers) == 0) {
            return '';
        }

        $headerStrings = array();
```

```
        $headerStrings = array();
        foreach ($headers as $k => $v) {
            //跳过key为null的
            if ($k === null) {
                continue;
            }
            //如果value为null，则赋值为空串
            if ($v === null) {
                $v = '';
            }
            //trim后再encode，之后使用':'号连接起来
            $headerStrings[] = HttpUtil::urlEncode(strtolower(trim($k))) . ':' .
HttpUtil::urlEncode(trim($v));
        }
        //字典序排序
        sort($headerStrings);

        //用'\n'把它们连接起来
        return implode("\n", $headerStrings);
    }

}
HttpUtil::__init();


class SampleSigner
{

    const BCE_AUTH_VERSION = "bce-auth-v1";
    const BCE_PREFIX = 'x-bce-';

    //不指定headersToSign情况下，默认签名http头，包括：
    //    1.host
    //    2.content-length
    //    3.content-type
    //    4.content-md5
    public static $defaultHeadersToSign;

    public static function  __init()
    {
        SampleSigner::$defaultHeadersToSign = array(
            "host",
            "content-length",
            "content-type",
            "content-md5",
        );
    }

    //签名函数
    public function sign(
        array $credentials,
        $httpMethod,
        $path,
        $headers,
        $params,
        $options = array()
    ) {
        //设定签名有效时间
        if (!isset($options[SignOption::EXPIRATION_IN_SECONDS])) {
            //默认值1800秒
            $expirationInSeconds = SignOption::DEFAULT_EXPIRATION_IN_SECONDS;
        } else {
            $expirationInSeconds = $options[SignOption::EXPIRATION_IN_SECONDS];
```

```
        }

        //解析ak sk
        $accessKeyId = $credentials['ak'];
        $secretAccessKey = $credentials['sk'];

        //设定时间戳，注意：如果自行指定时间戳需要为UTC时间
        if (!isset($options[SignOption::TIMESTAMP])) {
            //默认值当前时间
            $timestamp = new \DateTime();
        } else {
            $timestamp = $options[SignOption::TIMESTAMP];
        }
        $timestamp->setTimezone(new \DateTimeZone("UTC"));

        //生成authString
        $authString = SampleSigner::BCE_AUTH_VERSION . '/' . $accessKeyId . '/'
            . $timestamp->format("Y-m-d\TH:i:s\Z") . '/' . $expirationInSeconds;

        //使用sk和authString生成signKey
        $signingKey = hash_hmac('sha256', $authString, $secretAccessKey);

        //生成标准化URI
        $canonicalURI = HttpUtil::getCanonicalURIPath($path);

        //生成标准化QueryString
        $canonicalQueryString = HttpUtil::getCanonicalQueryString($params);

        //填充headersToSign，也就是指明哪些header参与签名
        $headersToSignOption = null;
        if (isset($options[SignOption::HEADERS_TO_SIGN])) {
            $headersToSignOption = $options[SignOption::HEADERS_TO_SIGN];
        }

        $headersToSign = SampleSigner::getHeadersToSign($headers,
$headersToSignOption);

        //生成标准化header
        $canonicalHeader = HttpUtil::getCanonicalHeaders($headersToSign);

        $headersToSign = array_keys($headersToSign);
        sort($headersToSign);
        //整理headersToSign，以';'号连接
        $signedHeaders = '';
        if ($headersToSignOption !== null) {
            $signedHeaders = strtolower(
                trim(implode(";", $headersToSign))
            );
        }

        //组成标准请求串
        $canonicalRequest = "$httpMethod\n$canonicalURI\n"
            . "$canonicalQueryString\n$canonicalHeader";

        //使用signKey和标准请求串完成签名
        $signature = hash_hmac('sha256', $canonicalRequest, $signingKey);

        //组成最终签名串
        $authorizationHeader = "$authString/$signedHeaders/$signature";

        return $authorizationHeader;
    }
```

```
    /** 根据headsToSign过滤应该参与签名的header
     *
     * @param $headers array
     * @param $headersToSign array
     * @return array
     */
    public static function getHeadersToSign($headers, $headersToSign)
    {

        $ret = array();
        if ($headersToSign !== null) {
            $tmp = array();

            //处理headers的key：去掉前后的空白并转化成小写
            foreach ($headersToSign as $header) {
                $tmp[] = strtolower(trim($header));
            }
            $headersToSign = $tmp;
        }
        foreach ($headers as $k => $v) {
            if (trim((string) $v) !== '') {
                if ($headersToSign !== null) {
                    //预处理headersToSign：去掉前后的空白并转化成小写
                    if (in_array(strtolower(trim($k)), $headersToSign)) {
                        $ret[$k] = $v;
                    }
                } else {
                    //如果没有headersToSign，则根据默认规则来选取headers
                    if (SampleSigner::isDefaultHeaderToSign($k, $headersToSign))
{
                        $ret[$k] = $v;
                    }
                }
            }
        }
        return $ret;
    }

    /**
     * 检查header是不是默认参加签名的：
     * 1.是host、content-type、content-md5、content-length之一
     * 2.以x-bce开头
     *
     * @param $header string
     * @return bool
     */
    public static function isDefaultHeaderToSign($header)
    {
        $header = strtolower(trim($header));
        if (in_array($header, SampleSigner::$defaultHeadersToSign)) {
            return true;
        }
        $prefix = substr($header, 0, strlen(SampleSigner::BCE_PREFIX));
        if ($prefix === SampleSigner::BCE_PREFIX) {
            return true;
        } else {
            return false;
        }
    }
}
```

```
SampleSigner::__init();



//签名示范代码
$signer = new SampleSigner();
$credentials = array("ak" => "0b0f67dfb88244b289b72b142befad0c","sk" =>
"bad522c2126a4618a8125f4b6cf6356f");
$httpMethod = "PUT";
$path = "/v1/test/myfolder/readme.txt";
$headers = array("Host" => "bj.bcebos.com",
                "Content-Length" => 8,
                "Content-MD5" => "0a52730597fb4ffa01fc117d9e71e3a9",
                "Content-Type" => "text/plain",
                "x-bce-date" => "2015-04-27T08:23:49Z");
$params = array("partNumber" => 9, "uploadId" =>
"VXBsb2FkIElpZS5tMnRzIHVwbG9hZA");
date_default_timezone_set("PRC");
$timestamp = new \DateTime();
$timestamp->setTimestamp(1430123029);
$options = array(SignOption::TIMESTAMP => $timestamp);
// $options = array(SignOption::TIMESTAMP => $timestamp,
SignOption::HEADERS_TO_SIGN => array("Content-Type", "Host", "x-bce-date"));
$ret = $signer->sign($credentials, $httpMethod, $path, $headers, $params,
$options);
print $ret;
```

Java示例

用户可参考以下代码，进一步了解百度智能云API认证机制。

代码下载路径：https://github.com/baidubce/bce-sdk-java/blob/master/src/main/java/com/baidubce/auth/BceV1Signer.java

> **说明**：Android语言的API认证示例代码和Java示例代码一致。

```
/*
 * Copyright (c) 2014 Baidu.com, Inc. All Rights Reserved
 *
 * Licensed under the Apache License, Version 2.0 (the "License"); you may not
use this file except in compliance with
 * the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on
 * an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied. See the License for the
 * specific language governing permissions and limitations under the License.
 */

package com.baidubce.auth;

import com.baidubce.BceClientException;
import com.baidubce.http.Headers;
import com.baidubce.internal.InternalRequest;
import com.baidubce.util.DateUtils;
import com.baidubce.util.HttpUtils;
import com.google.common.base.Joiner;
```

```
import com.google.common.base.Joiner;
import com.google.common.collect.Lists;
import com.google.common.collect.Maps;
import com.google.common.collect.Sets;

import org.apache.commons.codec.binary.Hex;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import java.nio.charset.Charset;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.SortedMap;

import static com.google.common.base.Preconditions.checkNotNull;

/**
 * The V1 implementation of Signer with the BCE signing protocol.
 */
public class BceV1Signer implements Signer {

    private static final Logger logger =
LoggerFactory.getLogger(BceV1Signer.class);

    private static final String BCE_AUTH_VERSION = "bce-auth-v1";
    private static final String DEFAULT_ENCODING = "UTF-8";
    private static final Charset UTF8 = Charset.forName(DEFAULT_ENCODING);

    // Default headers to sign with the BCE signing protocol.
    private static final Set<String> defaultHeadersToSign = Sets.newHashSet();
    private static final Joiner headerJoiner = Joiner.on('\n');
    private static final Joiner signedHeaderStringJoiner = Joiner.on(';');

    static {
        BceV1Signer.defaultHeadersToSign.add(Headers.HOST.toLowerCase());

BceV1Signer.defaultHeadersToSign.add(Headers.CONTENT_LENGTH.toLowerCase());
        BceV1Signer.defaultHeadersToSign.add(Headers.CONTENT_TYPE.toLowerCase());
        BceV1Signer.defaultHeadersToSign.add(Headers.CONTENT_MD5.toLowerCase());
    }

    /**
     * @see com.baidubce.auth.Signer#sign(InternalRequest, BceCredentials)
     */
    @Override
    public void sign(InternalRequest request, BceCredentials credentials) {
        this.sign(request, credentials, null);
    }

    /**
     * Sign the given request with the given set of credentials. Modifies the
passed-in request to apply the signature.
     *
     * @param request     the request to sign.
     * @param credentials the credentials to sign the request with.
     * @param options     the options for signing.
     */
```

```
    @Override
    public void sign(InternalRequest request, BceCredentials credentials,
SignOptions options) {
        checkNotNull(request, "request should not be null.");

        if (credentials == null) {
            return;
        }

        if (options == null) {
            if (request.getSignOptions() != null) {
                options = request.getSignOptions();
            } else {
                options = SignOptions.DEFAULT;
            }
        }

        String accessKeyId = credentials.getAccessKeyId();
        String secretAccessKey = credentials.getSecretKey();

        request.addHeader(Headers.HOST,
HttpUtils.generateHostHeader(request.getUri()));

        Date timestamp = options.getTimestamp();
        if (timestamp == null) {
            timestamp = new Date();
        }

        String authString =
                BceV1Signer.BCE_AUTH_VERSION + "/" + accessKeyId + "/"
                        + DateUtils.formatAlternateIso8601Date(timestamp) + "/"
+ options.getExpirationInSeconds();

        String signingKey = this.sha256Hex(secretAccessKey, authString);
        // Formatting the URL with signing protocol.
        String canonicalURI =
this.getCanonicalURIPath(request.getUri().getPath());
        // Formatting the query string with signing protocol.
        String canonicalQueryString =
HttpUtils.getCanonicalQueryString(request.getParameters(), true);
        // Sorted the headers should be signed from the request.
        SortedMap<String, String> headersToSign =
                this.getHeadersToSign(request.getHeaders(),
options.getHeadersToSign());
        // Formatting the headers from the request based on signing protocol.
        String canonicalHeader = this.getCanonicalHeaders(headersToSign);
        String signedHeaders = "";
        if (options.getHeadersToSign() != null) {
            signedHeaders =
BceV1Signer.signedHeaderStringJoiner.join(headersToSign.keySet());
            signedHeaders = signedHeaders.trim().toLowerCase();
        }

        String canonicalRequest =
                request.getHttpMethod() + "\n" + canonicalURI + "\n" +
canonicalQueryString + "\n" + canonicalHeader;

        // Signing the canonical request using key with sha-256 algorithm.
        String signature = this.sha256Hex(signingKey, canonicalRequest);

        String authorizationHeader = authString + "/" + signedHeaders + "/" +
signature;
```

```
        logger.debug("CanonicalRequest:{}\tAuthorization:{}",
canonicalRequest.replace("\n", "[\\n]"),
                authorizationHeader);

        request.addHeader(Headers.AUTHORIZATION, authorizationHeader);
    }

    private String getCanonicalURIPath(String path) {
        if (path == null) {
            return "/";
        } else if (path.startsWith("/")) {
            return HttpUtils.normalizePath(path);
        } else {
            return "/" + HttpUtils.normalizePath(path);
        }
    }

    private String getCanonicalHeaders(SortedMap<String, String> headers) {
        if (headers.isEmpty()) {
            return "";
        }

        List<String> headerStrings = Lists.newArrayList();
        for (Map.Entry<String, String> entry : headers.entrySet()) {
            String key = entry.getKey();
            if (key == null) {
                continue;
            }
            String value = entry.getValue();
            if (value == null) {
                value = "";
            }
            headerStrings.add(HttpUtils.normalize(key.trim().toLowerCase()) + ':'
+ HttpUtils.normalize(value.trim()));
        }
        Collections.sort(headerStrings);

        return headerJoiner.join(headerStrings);
    }

    private SortedMap<String, String> getHeadersToSign(Map<String, String>
headers, Set<String> headersToSign) {
        SortedMap<String, String> ret = Maps.newTreeMap();
        if (headersToSign != null) {
            Set<String> tempSet = Sets.newHashSet();
            for (String header : headersToSign) {
                tempSet.add(header.trim().toLowerCase());
            }
            headersToSign = tempSet;
        }
        for (Map.Entry<String, String> entry : headers.entrySet()) {
            String key = entry.getKey();
            if (entry.getValue() != null && !entry.getValue().isEmpty()) {
                if ((headersToSign == null && this.isDefaultHeaderToSign(key))
                        || (headersToSign != null &&
headersToSign.contains(key.toLowerCase())
                                &&
!Headers.AUTHORIZATION.equalsIgnoreCase(key))) {
                    ret.put(key, entry.getValue());
                }
            }
        }
```

```
        }
        return ret;
    }

    private boolean isDefaultHeaderToSign(String header) {
        header = header.trim().toLowerCase();
        return header.startsWith(Headers.BCE_PREFIX) ||
defaultHeadersToSign.contains(header);
    }

    private String sha256Hex(String signingKey, String stringToSign) {
        try {
            Mac mac = Mac.getInstance("HmacSHA256");
            mac.init(new SecretKeySpec(signingKey.getBytes(UTF8), "HmacSHA256"));
            return new
String(Hex.encodeHex(mac.doFinal(stringToSign.getBytes(UTF8))));
        } catch (Exception e) {
            throw new BceClientException("Fail to generate the signature", e);
        }
    }

}
```

Javascript示例

用户可参考以下代码，进一步了解Javascript语言的百度智能云API认证机制。

代码下载路径：https://github.com/baidubce/bce-sdk-js/blob/master/test/sdk/auth.spec.js

```
var Auth = require('@baiducloud/sdk').Auth;


    var auth = new Auth('my_ak', 'my_sk');

    var method = 'PUT';
    var uri = '/v1/bucket/object1';
    var params = {
        A: null,
        b: '',
        C: 'd'
    };
    var headers = {
        'Host': 'bce.baidu.com',
        'abc': '123',
        'x-bce-meta-key1': 'ABC'
    };

    var signature = auth.generateAuthorization(method, uri, params, headers,
1402639056);
    expect(signature).to.eql('bce-auth-v1/my_ak/2014-06-13T05:57:36Z/1800/host;x-
bce-meta-key1/'
                             +
'80c9672aca2ea9af4bb40b9a8ff458d72df94e97d550840727f3a929af271d25');

    signature = auth.generateAuthorization(method, uri, params, headers,
1402639056, 1800);
    expect(signature).to.eql('bce-auth-v1/my_ak/2014-06-13T05:57:36Z/1800/host;'
                             + 'x-bce-meta-
key1/80c9672aca2ea9af4bb40b9a8ff458d72'
                             + 'df94e97d550840727f3a929af271d25');

    method = 'DELETE';
    uri = '/v1/test-bucket1361199862';
    params = {};
    headers = {
        'Content-Type': 'application/json; charset=utf-8',
        'Content-Length': 0,
        'User-Agent': 'This is the user-agent'
    };
    signature = auth.generateAuthorization(method, uri, params, headers,
1402639056, 1800);
    expect(signature).to.eql('bce-auth-v1/my_ak/2014-06-13T05:57:36Z/1800/'
                             + 'content-length;content-type/'
                             +
'c9386b15d585960ae5e6972f73ed92a9a682dc81025480ba5b41206d3e489822');
```

C#示例

用户可参考以下代码，进一步了解C#语言的百度智能云API认证机制。完整示例代码：

```
using BaiduBce;
using BaiduBce.Auth;
using BaiduBce.Services.Bos;
using BaiduBce.Services.Bos.Model;
using BaiduBce.Services.Sts;
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.Diagnostics;
using System.Globalization;
```

```
using System.IO;
using System.Net;
using System.Security.Cryptography;
using System.Text;
using System.Web;

namespace BOSTest
{
    class Program
    {
        static string UriEncode(string input, bool encodeSlash = false)
        {
            StringBuilder builder = new StringBuilder();
            foreach (byte b in Encoding.UTF8.GetBytes(input))
            {
                if ((b >= 'a' && b <= 'z') || (b >= 'A' && b <= 'Z') || (b >= '0'
&& b <= '9') || b == '_' || b == '-' || b == '~' || b == '.')
                {
                    builder.Append((char)b);
                }
                else if (b == '/')
                {
                    if (encodeSlash)
                    {
                        builder.Append("%2F");
                    }
                    else
                    {
                        builder.Append((char)b);
                    }
                }
                else
                {
                    builder.Append('%').Append(b.ToString("X2"));
                }
            }
            return builder.ToString();
        }

        static string Hex(byte[] data)
        {
            var sb = new StringBuilder();
            foreach (var b in data)
            {
                sb.Append(b.ToString("x2"));
            }
            return sb.ToString();
        }

        static string CanonicalRequest(HttpWebRequest req)
        {
            Uri uri = req.RequestUri;
            StringBuilder canonicalReq = new StringBuilder();

canonicalReq.Append(req.Method).Append("\n").Append(UriEncode(Uri.UnescapeDataString
;

            var parameters = HttpUtility.ParseQueryString(uri.Query);
            List<string> parameterStrings = new List<string>();
            foreach (KeyValuePair<string, string> entry in parameters)
            {
                parameterStrings.Add(UriEncode(entry.Key) + '=' +
```

```
UriEncode(entry.Value));
            }
            parameterStrings.Sort();
            canonicalReq.Append(string.Join("&",
parameterStrings.ToArray())).Append("\n");

            string host = uri.Host;
            if (!(uri.Scheme == "https" && uri.Port == 443) && !(uri.Scheme ==
"http" && uri.Port == 80))
            {
                host += ":" + uri.Port;
            }
            canonicalReq.Append("host:" + UriEncode(host));
            return canonicalReq.ToString();
        }

        static void Main(string[] args)
        {
            string bucket = "mybucket";
            string key = "我的文件";
            string ak = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
            string sk = "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb";
            DateTime now = DateTime.Now;
            int expirationInSeconds = 1200;

            HttpWebRequest req = WebRequest.Create("http://bj.bcebos.com/" +
bucket + "/" + key) as HttpWebRequest;
            Uri uri = req.RequestUri;
            req.Method = "GET";

            string signDate = now.ToUniversalTime().ToString("yyyy'-'MM'-
'dd'T'HH':'mm':'ssK");
            Console.WriteLine(signDate);
            string authString = "bce-auth-v1/" + ak + "/" + signDate + "/" +
expirationInSeconds;
            string signingKey = Hex(new
HMACSHA256(Encoding.UTF8.GetBytes(sk)).ComputeHash(Encoding.UTF8.GetBytes(authString
;
            Console.WriteLine(signingKey);

            string canonicalRequestString = CanonicalRequest(req);
            Console.WriteLine(canonicalRequestString);

            string signature = Hex(new
HMACSHA256(Encoding.UTF8.GetBytes(signingKey)).ComputeHash(Encoding.UTF8.GetBytes(ca
;
            string authorization = authString + "/host/" + signature;
            Console.WriteLine(authorization);

            req.Headers.Add("x-bce-date", signDate);
            req.Headers.Add(HttpRequestHeader.Authorization, authorization);

            HttpWebResponse res;
            string message = "";
            try
            {
                res = req.GetResponse() as HttpWebResponse;
            }
            catch (WebException e)
            {
                res = e.Response as HttpWebResponse;
                message = new StreamReader(res.GetResponseStream()).ReadToEnd();
            }
```

```
            }
            Console.WriteLine((int)res.StatusCode);
            Console.WriteLine(res.Headers);
            Console.WriteLine(message);
            Console.ReadLine();
        }
    }
}
```

iOS示例

用户可参考以下代码，进一步了解iOS的百度智能云API认证机制。代码包含类代码和调用代码两部分，完整代码地址：代码。调用代码请参考：

```
#import <XCTest/XCTest.h>
#import "BDCloudSigner.h"

@interface UT : XCTestCase
@end

@implementation UT

id<BDCloudSigner> createSigner() {
    BDCloudCredentials* credentials = [BDCloudCredentials new];
    credentials.accessKey = @"<access key>";
    credentials.secretKey = @"<secret key>";

    id<BDCloudSigner> signer = [[BDCloudAKSKSigner alloc]
initWithCredentials:credentials];
    signer.expiredTimeInSeconds = 3600;

    return signer;
}

NSMutableURLRequest* createRequest() {
    // create url directly, or use NSURLComponents.
    NSURL* url = [NSURL URLWithString:@"http://bj.bcebos.com/v1/bucket/object?
append"];

    // create request.
    NSMutableURLRequest* request = [NSMutableURLRequest requestWithURL:url];
    request.HTTPMethod = @"POST";
    [request setValue:@"<length>" forHTTPHeaderField:@"Content-Length"];
    [request setValue:@"<md5>" forHTTPHeaderField:@"Content-MD5"];
    [request setValue:@"text/plain" forHTTPHeaderField:@"Content-Type"];

    // custom metadata key should begin with lower case prefix 'x-bce-'.
    [request setValue:@"2017-01-08T21:42:30Z" forHTTPHeaderField:@"x-bce-user-
metadata-createtime"];

    // Host will be set when call sign.
    //[request setValue:@"bj.bcebos.com" forHTTPHeaderField:@"Host"];

    return request;
}

void sign() {
    id<BDCloudSigner> signer = createSigner();
    NSMutableURLRequest* request = createRequest();
    if (![signer sign:request]) {
        return;
```

```
    }

    // url
    NSURL* fileURL = [NSURL fileURLWithPath:@"<file path>"];

    // send request
    // sample purpose, don't care task will running correctly.
    [[NSURLSession sharedSession] uploadTaskWithRequest:request
                                                fromFile:fileURL];
}

- (void)testAKSKSigner {
    sign();
}

@end
```

常见签名认证错误排查

签名认证中常见以下问题，你可以配合签名排查工具在线排查签名问题，或者通过如下的排查说明进行一一排查：

- 不能正确区分URL中的URI部分和QueryString部分
- URI尾部的"/"不一致
- Host头域端口不一致
- x-bce-date头的问题
- 用错了ak/sk
- 客户基于StsCredential来请求但服务未正确传递security_token
- 生成签名过程中注意事项

不能正确区分URL中的URI部分和QueryString部分

假设请求的URL是`/v1/settings/region/list?type=public`，此处URI应该为`/v1/settings/region/list`,`type=public`是querystring。但是客户端并没有正确区分它们，在计算签名时把`/v1/settings/region/list?type=public`都当成了PATH，进而得到的CanonicalRequest，因此传送的签名也有问题。 而服务端在验证签名使用的

```
# 客户端请求的CanonicalRequest：
GET
/v1/settings/region/list%3ftype%3dpublic
host:settings.bce-internal.baidu.com
x-bce-date:2017-02-15T08%3A52%3A48Z
```

服务端向IAM构造验签请求时，传递的是正确的URI和QueryString部分：

```
 {
   "auth" : {
     "authorization" : "bce-auth-v1/f3e1f5c2ef9c4d25b7f8b936861cd175/2017-04-
05T08:29:50Z/3600/host;x-bce-
date/9a48ec83ffb6c486892c4985df7dc6bc20f7c7bebb483f8f355592b8503a3f59",
     "request" : {
       "method" : "GET",
       "uri" : "/v1/settings/region/list",
       "headers" : {
         "host" : "settings.bce-internal.baidu.com",
         "x-bce-date": "2017-02-15T08:52:48Z"
       },
       "params": {
         "type": "public"
       }
     }
   }
```

进而，IAM基于服务提供的请求信息，演算签名时URI与客户端不一致，得到的签名必然与请求值不一致。

URI尾部的"/"不一致

客户端请求：

```
 # 客户端请求的CanonicalRequest：
 GET
 /v1/settings/region/list
 host:settings.bce-internal.baidu.com
 x-bce-date:2017-02-15T08%3A52%3A48Z
```

服务端向IAM构造验签请求时，误将URI多写了个尾部的"/"：

```
 {
   "auth" : {
     "authorization" : "bce-auth-v1/f3e1f5c2ef9c4d25b7f8b936861cd175/2017-04-
05T08:29:50Z/3600/host;x-bce-
date/9a48ec83ffb6c486892c4985df7dc6bc20f7c7bebb483f8f355592b8503a3f59",
     "request" : {
       "method" : "GET",
       "uri" : "/v1/settings/region/list/", #尾部多了个'/'
       "headers" : {
         "host" : "settings.bce-internal.baidu.com",
         "x-bce-date": "2017-02-15T08:52:48Z"
       }
     }
   }
```

进而，IAM基于服务提供的请求信息，演算签名时URI与客户端不一致，得到的签名必然与请求值不一致。

Host头域端口不一致

客户端请求：

```
# 客户端请求的CanonicalRequest：
GET
/v1/settings/region/list
host:settings.bce-internal.baidu.com%3A80
x-bce-date:2017-02-15T08%3A52%3A48Z
```

服务端向IAM构造验签请求时，host字段并没有加上端口：

```
{
  "auth" : {
    "authorization" : "bce-auth-v1/f3e1f5c2ef9c4d25b7f8b936861cd175/2017-04-
05T08:29:50Z/3600/host;x-bce-
date/2671990529b9c46c95e688f1a2809db185c608ee3716a731a8fdde6bb2028d84",
    "request" : {
      "method" : "GET",
      "uri" : "/v1/settings/region/list/",
      "headers" : {
        "host" : "settings.bce-internal.baidu.com", #host并没带端口
        "x-bce-date": "2017-02-15T08:52:48Z"
      }
    }
  }
```

host不一致，得到的签名必然与请求值不一致。

x-bce-date头的问题

默认情况（authorization中未指明SignedHeaders时）x-bce-date会参与签名。客户端需要保证：计算签名时的此域，与最终发出请求时的此域，二者保持一致。

如，计算签名时使用的时间是 2017-02-15T08:52:48Z

```
GET
/v1/settings/region/list
host:settings.bce-internal.baidu.com
x-bce-date:2017-02-15T08%3A52%3A48Z
```

但发出请求时，x-bce-date 变成了 2017-02-15T08:52:49Z

```
GET /v1/settings/region/list HTTP/1.1
User-Agent: curl/7.33.0
Host: settings.bce-internal.baidu.com
Accept: */*
authorization: bce-auth-v1/f3e1f5c2ef9c4d25b7f8b936861cd175/2017-04-
05T08:29:50Z/3600/host;x-bce-date;x-bce-request-
id/9a48ec83ffb6c486892c4985df7dc6bc20f7c7bebb483f8f355592b8503a3f59
x-bce-date: 2017-02-15T08:52:49Z
```

IAM演算签名时x-bce-date域与客户端不一致，得到的签名必然与请求值不一致。

用错了ak/sk

以下两种情况容易出现低级的错误导致签名不通过：

- 基于配置的程序，配置中ak对了，但sk却存在错误。这种情况IAM并不会返回ak不存在，而是签名错误。一个特点是signningKey与IAM演算的不一致。

- 自动从API拿到的ak/sk，但是构造业务client时却传错了，比如构造参数可能分别是ak和sk，但代码调用时两个参数都使用了ak。

客户基于StsCredential来请求但服务未正确传递security_token

客户端请求：

```
# 客户端请求的CanonicalRequest：
GET
/v1/settings/region/list
host:settings.bce-internal.baidu.com
x-bce-date:2017-02-15T08%3A52%3A48Z
x-bce-security-token: ZGZiM2M3MmU4Mjk4NGQ2MGEzYTNhYTAyMDE3NTZmZmV8AAAA...
```

服务端向IAM构造验签请求时，没有将security_token提取出来，放到专门的字段，用于周知IAM客户使用了临时credential：

```
{
  "auth" : {
    "authorization" : "bce-auth-v1/f3e1f5c2ef9c4d25b7f8b936861cd175/2017-04-
05T08:29:50Z/3600/host;x-bce-
date/9a48ec83ffb6c486892c4985df7dc6bc20f7c7bebb483f8f355592b8503a3f59",
    "request" : {
      "method" : "GET",
      "uri" : "/v1/settings/region/list",
      "headers" : {
        "host" : "settings.bce-internal.baidu.com",
        "x-bce-date": "2017-02-15T08:52:48Z",
        "x-bce-security-token":
"ZGZiM2M3MmU4Mjk4NGQ2MGEzYTNhYTAyMDE3NTZmZmV8AAAA..."
      }
    }
  }
```

IAM不知道签名为临时ak/sk签出，从长效ak/sk表中查找不到相应的ak，于是会报"Could not find credential"。

生成签名过程中注意事项

1. **检查HTTP Method是否正确**
2. **CanonicalURI部分**
    - 注意前面不需要加host，必须以“/”开头，不以“/”开头的需要补充上，空路径为“/”；
    - 注意结尾不该有“/”；
    - 注意要做uriEncodeExceptSlash编码。
3. **CanonicalQueryString部分**
    - 注意要按字典序进行排序；
    - 没有value的key，是否也保留了“=”；
    - 没有queryString时，也要在最终的CanonicalRequest中拼接一个\n。
4. **CanonicalHeaders部分**
    - 百度智能云默认的signedHeaders：Host、Content-Length、Content-Type、Content-MD5、所

有以 x-bce- 开头的Header；

- 注意这里的header内容，要与实际请求中的header内容完全一致，尤其注意host是否有端口号，x-bce-date是否有变化；
- 实际发出请求中，是否有header被第三方库删除了？
- 如果请求中{signedHeaders}字段留空，那么是否完全按照百度智能云默认的signedHeaders计算的？

5. **签名前缀信息**

- timestamp是认证字符串创建时间，不能过早或过晚；
- 输入的SK与AK要对应

未命名文件

在认证鉴权过程中常见的错误码有：

错误返回码	错误消息	状态码	说明
AccessDenied	Access denied.	403 Forbidden	无权限访问对应的资源。
InappropriateJSON	The JSON you provided was well-formed and valid, but not appropriate forthis operation.	400 Bad Request	请求中的JSON格式正确，但语义上不符合要求。如缺少某个必需项，或值类型不匹配等。出于兼容性考虑，对于所有无法识别的项应直接忽略，不应该返回这个错误。
InternalError	We encountered an internal error Please try again.	500 Internal Server Error	所有未定义的其他错误。在有明确对应的其他类型的错误时（包括通用的和服务自定义的）不应该使用。
InvalidAccessKeyId	The Access Key ID you provided doesnot exist in our records.	403 Forbidden	Access Key ID不存在。
InvalidHTTPAuthHeader	The Access Key ID you provided does notexist in our records.	400 BadRequest	Authorization头域格式错误。

错误返回码	错误消息	状态码	说明
InvalidHTTPRequest	There was an error in the body of your HTTP request.	400 Bad Request	HTTP body格式错误。例如不符合指定的Encoding等。
InvalidURI	Could not parse the specified URI.	400 Bad Request	URI形式不正确。例如一些服务定义的关键词不匹配等。对于ID不匹配的问题，应定义更加具体的错误码，如NoSuchKey。
MalformedJSON	The JSON you provided was not well-formed.	400 BadRequest	JSON格式不合法。
InvalidVersion	The API version specified was invalid.	404 NotFound	URI的版本号不合法。
OptInRequired	A subscription for the service is required.	403 Forbidden	没有开通对应的服务。
PreconditionFailed	The specified If-Match header doesn’tmatch the ETag header.	412 PreconditionFailed	详见Etag。
RequestExpired	Request has expired. Timestamp date is <Data>.	400 BadRequest	请求超时。要改成x-bce-date。若请求中只有Date，需将Date转成datetime。
IdempotentParameterMismatch	The request uses the same client token asa previous, but non-identical request.	403 Forbidden	clientToken对应的API参数不一样。
SignatureDoesNotMatch	The request signature we calculated does not match the signature you provided. Check yourSecret Access Key and signing method. Consultthe service documentation for details.	400 Bad Request	Authorization头域中附带的签名和服务端验证不一致。

获取AKSK

简介

AK（Access Key ID）/SK（Secret Access Key），主要用于对用户的调用行为进行鉴权和认证，相当于

百度智能云API专用的用户名及密码。有关API认证的详细介绍，请参看API认证机制。

如何获取AKSK

成功登陆Web控制台后，系统会自动分配一对AK / SK，可通控制台申请并管理自己的访问密钥，操作方法如下：

1. 登陆Web控制台，点击“用户账号->安全认证”进入Access Key管理界面。

2. 点击Access Key ID右侧的“显示”，可查看其对应的Secret Access Key，点击“隐藏”可隐藏对应的Secret Access Key。
3. 在密钥列表页中，根据需要可点击“创建Access Key”来创建新的Access Key ID / Secret Access Key密钥对。
4. 一个用户最多创建20对Access Key，可通过点击“删除”来删除多余的Access Key。

证书管理

简介

概述

证书管理模块主要用于管理用户的SSL证书，方便用户录入、查看及应用SSL证书。

- 如果用户尚未申请证书，可以通过百度智能云或第三方机构申请证书，推荐使用百度智能云SSL证书服务申请证书。通过SSL证书服务可实现一键式证书申请，申请的证书将自动导入证书管理模块。
- 如果用户已有证书，可以直接执行上传证书；用户可在【百度智能云管理控制台】->【用户账号】->【安全认证】->【证书管理】中进行添加除百度智能云外的其他证书。

相关概念

SSL证书：在应用HTTPS之前，用户应先配置SSL证书。SSL证书是由证书授权中心（CA）签发的对用户公钥的认证。证书的内容包括：电子签证机关的信息、公钥用户信息、公钥、权威机构的签字和有效期等。目前，证书的格式和验证方法普遍遵循X.509 国际标准。

HTTPS：Hyper Text Transfer Protocol over Secure Socket Layer。HTTPS是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL。

SSL：Secure Sockets Layer。SSL是一个安全协议，它提供使用 TCP/IP 的通信应用程序间的隐私与完整性。因特网的超文本传输协议（HTTP）使用 SSL 来实现安全的通信。

TLS：Transport Layer Security Protocol。TLS是IETF制定的一种新的协议，它建立在SSL 3.0协议规范之上，是SSL 3.0的后续版本。在TLS与SSL3.0之间存在着显著的差别，主要是它们所支持的加密算法不同，所以TLS与SSL3.0不能互操作。TLS也经常用SSL来指代。

CA：即证书授权中心（CA, Certificate Authority）。CA是负责签发证书、认证证书、管理已颁发证书的机关。用户向CA提出申请后，CA负责审核用户信息，然后对关键信息利用私钥进行”签名”，并公开对应的公

钥。客户端可以利用公钥验证签名。

非对称加密：即常见的RSA、DH、EC等算法，算法特点是密钥成对出现，一般称为公钥（公开）和私钥（保密），公钥加密的信息只能私钥解开，私钥加密的信息只能公钥解开。公钥对外公开，私钥由服务器保存。因此掌握公钥的不同客户端之间不能互相解密信息，只能和掌握私钥的服务器进行加密通信，服务器可以实现1对多的通信，客户端也可以用来验证掌握私钥的服务器身份。

CSR：CSR是Certificate Signing Request的英文缩写，即证书请求文件，也就是证书申请者在申请数字证书时由CSP（加密服务提供者）在生成私钥的同时所生成的证书请求文件。证书申请者只要把CSR文件提交给证书颁发机构后，证书颁发机构使用其根证书私钥签名就生成了证书公钥文件，也就是颁发给用户的证书。

PEM：Openssl所使用的文档格式。RFC 1421-1424。

证书和私钥

概述

通过百度智能云申请证书的具体操作方法请参看SSL证书服务。

推荐您使用百度智能云证书申请服务，证书申请成功后将**自动**以高度加密形式导入到“证书管理”服务。您可通过不同服务的证书选择功能选择对应证书，快速部署服务。

如果是在“证书管理”添加第三方证书，请阅读下文，保证证书和私钥内容与格式正确。

证书

上传证书前，用户需要确保证书格式：

证书扩展名一般为“.pem”，“.crt”或“.cer”，在文本编辑器中打开证书文件。

- 证书以`-----BEGIN CERTIFICATE-----`开头，`-----END CERTIFICATE-----`结尾；
- 每行64字符，最后一行不超过64字符。
- 证书内容不包含空格。
- 证书需要在有效期内（证书开始时间 <= 有效期 <= 证书失效时间）

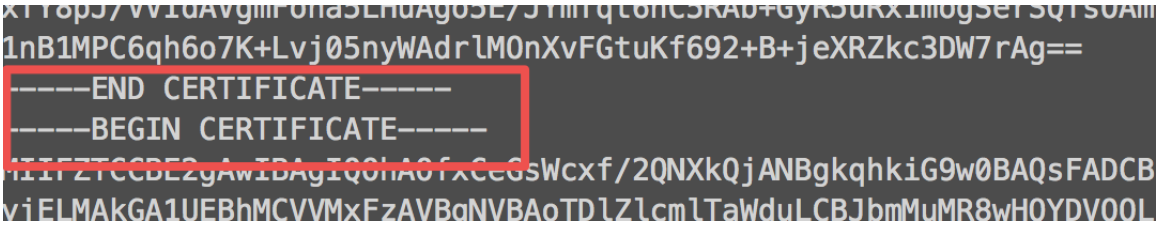
如图所示为PEM格式的证书示例：

如果是通过中级CA机构颁发的证书，您拿到的证书文件包含多份证书，需要人为的将服务器证书与中间证书拼接在一起后在上传。

- 拼接规则为：服务器证书放第一份，中间证书放第二份，中间换行符隔开，不要有空行。
- 一般情况下，机构在颁发证书的时候会有对应说明，请注意查阅规则说明。

拼接格式如下：

```
-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----
```



私钥

私钥扩展名一般为“.pem”或“.key”,在文本编辑器中可以打开私钥文件。

- 私钥格式以`-----BEGIN RSA PRIVATE KEY-----`作为开头，`-----END RSA PRIVATE KEY-----`作

为结尾。

- 私钥不能加密，即执行生成私钥命令时不能添加密码参数。
- 中间的内容每行64字符，最后一行长度可以不足64字符。

如图所示为PEM格式的私钥示例：

> 注意：
>
> 如果您的私钥是加密的，比如私钥的开头和结尾是-----BEGIN PRIVATE KEY-----, -----END PRIVATE KEY-----或-----BEGIN ENCRYPTED PRIVATE KEY-----, -----END ENCRYPTED PRIVATE KEY-----，或者私钥中包含Proc-Type: 4,ENCRYPTED，需要先运行以下命令进行转换：
>
> ```
> openssl rsa -in old_server_key.pem -out new_server_key.pem
> ```

格式转换

如果用户从CA收到的证书不是PEM格式的，需要转换成PEM格式，几种常见的证书格式转换为PEM格式的方法如下表所示：

证书格式	转换方法
DER -> PEM	证书：openssl x509 -inform der -in my-certificate.cer -out my-certificate.pem 私钥：openssl rsa -inform DER -outform PEM -in my-private-key.der -out my-private-key.pem
P7B -> PEM	证书：openssl pkcs7 -print_certs -in my-certificate.p7b -out my-certificate.pem。从my-certificate.pem中获取“-----BEGIN CERTIFICATE-----”开头，“-----END CERTIFICATE-----”结尾的内容作为证书。 私钥：无私钥
PFX -> PEM	证书：openssl pkcs12 -in certname.pfx -nokeys -out my-certificate.pem 私钥：openssl pkcs12 -in certname.pfx -nocerts -out my-private-key.pem -nodes

注意

- 如果您的私钥是加密的，比如私钥的开头和结尾是-----BEGIN PRIVATE KEY-----, -----END PRIVATE KEY-----或-----BEGIN ENCRYPTED PRIVATE KEY-----, -----END ENCRYPTED PRIVATE KEY-----，或者私钥中包含Proc-Type: 4,ENCRYPTED，需要先运行以下命令进行转换：

```
openssl rsa -in old_server_key.pem -out new_server_key.pem
```

- 私钥需要和服务器证书里的公钥匹配。可以通过以下两条命令生成私钥和证书的MD5值，如果二者一致，则可认为私钥与证书里的公钥匹配。

```
openssl x509 -noout -modulus -in my-certificate.pem | openssl md5
openssl rsa -noout -modulus -in my-private-key.pem | openssl md5
```

上传证书

上传证书

1、进入控制台，点击右上角“帐户名称”，在下拉菜单中选择“安全认证”，进入安全认证操作界面。

2、点击左侧“证书”，进入证书管理界面。点击“创建证书”，在弹出的窗口中输入以下信息并点击“确定”完成证书创建。填写参考证书和私钥

- 证书名称： 必填，用户自定义的证书名称。要求：长度限制为1-65个字符，以字母开头，只允许包含字母、数字、‘-’、‘/’、‘.’、‘_’。
- 证书：必填，CA机构所颁发的证书文件，证书中包含了用于认证的公钥。PEM编码。
- 私钥：必填，用户在申请证书过程中的第一步所创建的文件。PEM编码。

管理证书

证书详情

完成证书创建后可返回证书管理界面查看或管理已经创建的证书。

点击详情进入详情页，可在详情页查看证书基本信息，使用信息。

基本信息	描述
证书ID	证书标识
证书名称	用户自定义
通用名称	公司/组织的完全限定域名，用户在生成CRS时进行设置。
证书有效期	CA颁发证书的有效期。

证书状态	关联产品
未使用	未被使用，即未关联产品
已过期（未使用）	未被使用，即未关联产品
使用中	被使用，可在关联产品中查询
已过期（使用中）	被使用，可在关联产品中查询

删除证书

证书状态	是否支持删除
未使用	支持删除
已过期（未使用）	支持删除
使用中	不支持删除
已过期（使用中）	不支持删除

使用中证书不支持删除操作，可在证书详情中进行查询，查看被哪些产品资源使用，并在相应的产品资源中进行替换。

替换参考证书替换,替换后方可进行删除。

多用户访问控制

介绍

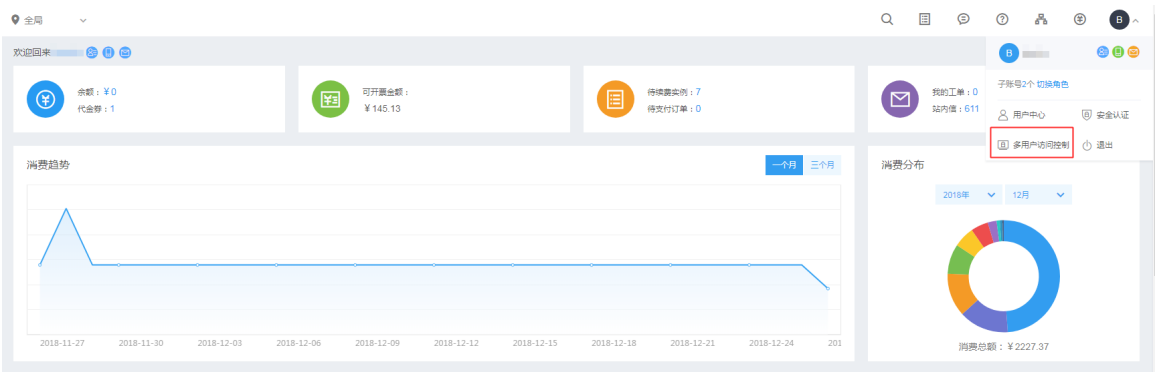
多用户访问控制，主要用于帮助用户管理云账户下资源的访问权限，适用于企业内的不同角色，可以对不同的工作人员赋予使用产品的不同权限，当您的企业存在多用户协同操作资源时，推荐您使用多用户访问控制。

适用于下列使用场景：

- 中大型企业客户：对公司内多个员工授权管理；
- 偏技术型vendor或SAAS的平台商：对代理客户进行资源和权限管理；
- 中小开发者或小企业：添加项目成员或协作者，进行资源管理。

创建用户

1. 主账号用户登录后在控制台选择“多用户访问控制”进入用户管理页面。



2. 在左侧导航栏点击“用户管理”，在“子用户管理列表”页，点击“新建用户”。
3. 在弹出的“新建用户”对话框中，完成填写“用户名”和确认，返回“子用户管理列表”区可以查看到刚刚创建的子用户。

配置策略

证书管理支持系统策略和用户自定义策略两种，分别实现证书的产品级权限和实例级权限控制。

- 系统策略：百度智能云系统为管理资源而预定义的权限集，有产品级别的只读和运维权限，这类策略可直接为子用户授权。用户只能使用而不能修改。
- 自定义策略：由用户自己创建，更细化的管理资源的权限集，可以针对单个证书进行配置权限，更加灵活的满足账户对不同用户的差异化权限管理。

系统策略

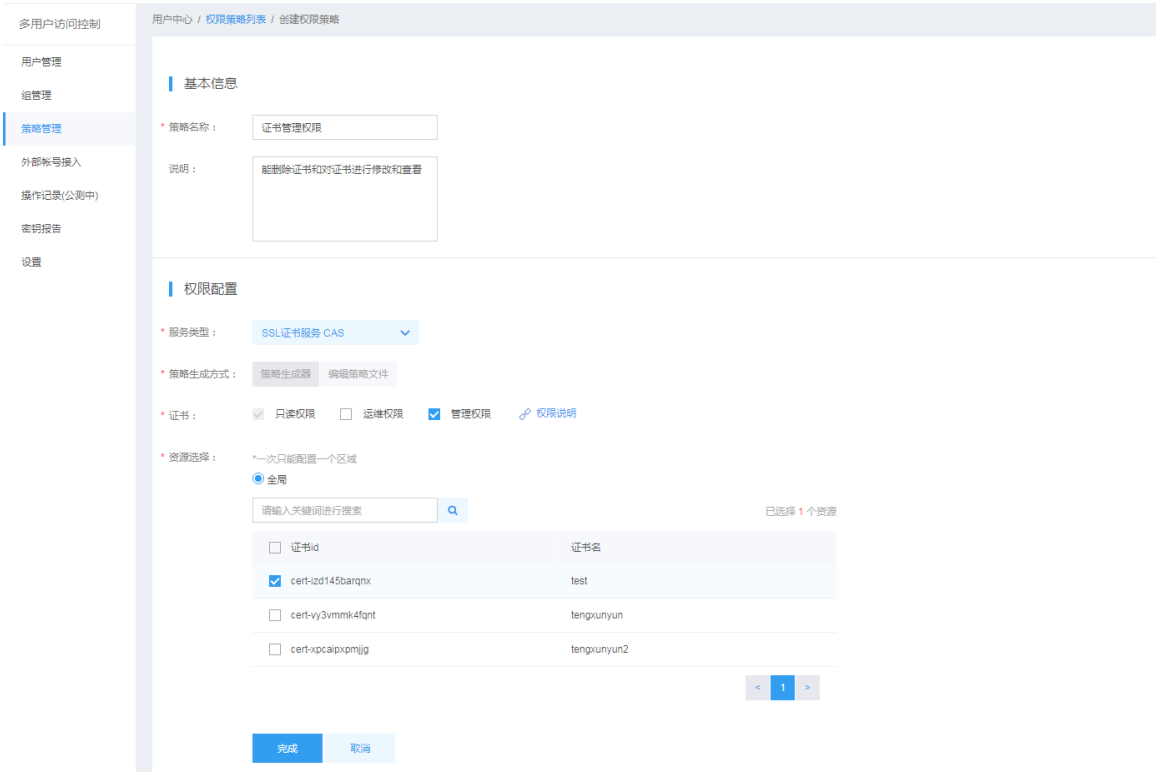
系统策略为证书产品级权限，包含产品级运维权限和产品级只读权限两种系统策略，权限范围详细如下：

策略名称	权限说明	权限范围
CASWritePolicy	产品级运维权限	包括上传证书、修改证书名、获取证书详情列表、获取证书列表、获取证书信息、获取证书详情、获取证书关联的服务列表、获取证书关联的资源数量、获取证书域名列表的权限。
CASReadPolicy	产品级只读权限	包括获取证书详情列表、获取证书列表、获取证书信息、获取证书详情、获取证书关联的服务列表、获取证书关联的资源数量、获取证书域名列表的权限。

自定义策略

自定义策略是从实例维度进行授权，与系统策略不同，只对选定的实例生效。

子用户通过点击“创建策略”，填写策略名称并选择服务类型为“SSL证书服务 CAS”添加自定义策略来进行实例级权限控制，其中策略生成方式默认为策略生成器，不需要修改，用户可以根据具体的权限设置修改策略内容；选择完权限后，勾选可操作的证书即可创建完成。



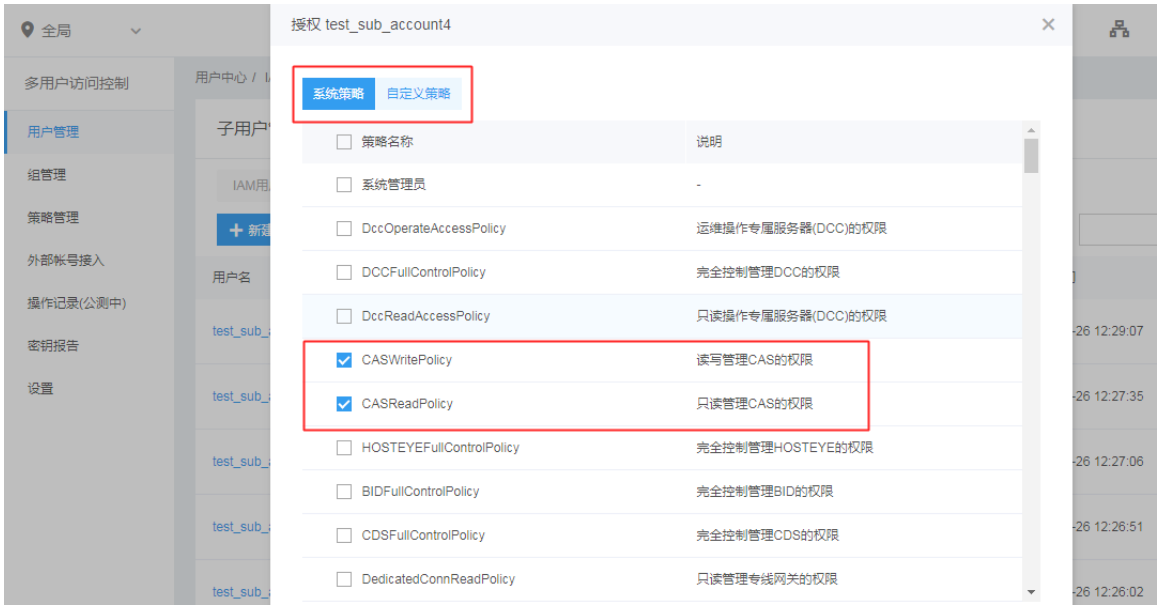
自定义权限范围详细如下：

权限说明	权限范围
实例级管理权限	包括删除证书、修改证书名、获取证书列表、获取证书信息、获取证书详情、获取证书关联的服务列表、获取证书关联的资源数量和获取证书域名列表的权限。
实例级运维权限	包括修改证书名、获取证书列表、获取证书信息、获取证书详情、获取证书关联的服务列表、获取证书关联的资源数量和获取证书域名列表的权限。
实例级只读权限	包括获取证书列表、获取证书信息、获取证书详情、获取证书关联的服务列表、获取证书关联的资源数量和获取证书域名列表的权限。

> **说明：**
>
> - 实例级权限由于是针对特定证书的权限，所以无法上传证书。
> - 由于部分用户证书数量过多（上千），所以获取证书列表接口不批量鉴权，而是每次列出全量。
> - 只有拥有产品级权限的人，才能访问获取证书详情列表；拥有实例级权限的人，只能访问证书列表和单个证书详情。

用户授权

1. 在“用户管理->子用户管理列表页”的对应子用户的“操作”列选择“添加权限”，并为用户选择系统权限或自定义策略进行授权。



> **说明：**如果在不修改已有策略规则的情况下修改某子用户的权限，只能通过删除已有的策略并添加新的策略来实现，不能取消勾选已经添加过的策略权限。

子用户登录

主账号完成对子用户的授权后，可以将链接发送给子用户；子用户可以通过IAM用户登录链接登录主账号的管理控制台，根据被授权的策略对主账户资源进行操作和查看。

其他详细操作参考：多用户访问控制。

证书替换

为了避免影响正常业务，在证书到期即将到期前，请重新申请并更换证书。有关单产品证书的更换方法，请查看相应文档：

- BLB证书替换
- CDN证书替换
- BCH证书替换

> 注意： 为避免影响其他产品方使用，在证书管理中’使用中‘的证书不支持删除操作，’使用中‘的证书支持在产品方进行替换后删除。

第三方机构创建证书

用户可以使用OpenSSL来创建CSR，并通过相关机构（例如：沃通）申请证书，具体操作方法如下文所示。

前置任务

在申请证书前，用户应先获取并安装相关工具。

Linux： 建议使用OpenSSL。OpenSSL是一个强大的SLL密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

Windows： 可供选择的工具包括：IIS Manager，SelfSSL，OpenSSL和Windows PowerShell cmdlets等。

使用OpenSSL创建证书

提交证书申请前，用户需在本地生成私钥和证书请求文件（CSR）；然后用户可以将CSR提交至CA进行签名认证，也可以通过自签名的方式进行本地测试。具体操作方法如以下示例所示（本示例中使用的工具为OpenSSL）。

1、执行命令`openssl genrsa -out my-private-key.pem 2048`，通过RSA算法生成私钥，并保存在*my-private-key.pem*文件中。百度智能云支持1024、2048和4096 bits长度的私钥，建议用户使用2048 bits长度。

注意：请妥善保管私钥，避免遗失和泄露。

查看私钥文件的具体内容如下：

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAuRn81tg5aVRtTLAb+96qQPLzgoVzPEJvUCATvOZDoUcfZqx6
A+PJ33SBZ6wUUMwKwDiAbi3yG2cQ1zfVe68PvtkFcPjwnTFRJbRObZUJXVA8LIMy
LDC9cq71qvptYCMUVdmsrWeRUzZfJLLo8/+KRV0kAoykLyOVYHVXVKa+XChrEEcD
......
PMKG1QKBgCElb8qQlSPzN7If+4+xY/z6iDZ80H/ktJahcD3QHeFUsaXaYQGokudC
4ZWHW1JiLwIqsiJ4iLaTKU4JOn5LyFFR/6O/NvhWuMXHmqLtHwhrKtZX2TBwbfdQ
wRakf1vOSHXHDG1Zym5t5s8UWDtkMcHZ7lIasjgaWB3lgFMwu+0u
-----END RSA PRIVATE KEY-----'
```

2、执行命令`openssl req -sha256 -new -key my-private-key.pem -out csr.pem`，根据系统提示输入相关配置信息，生成CSR文件。其中，`-key my-private-key.pem`用来指定步骤1中生成的私钥文件；`-out csr.pem`用来指定CSR文件的名称。

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:Beijing
Locality Name (eg, city) []:Beijing
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:www.mycompany.com
Email Address []:example@mycompany.c
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:.
An optional company name []:.
```

查看CSR文件具体内容如下：

```
-----BEGIN CERTIFICATE REQUEST-----
MIIC5TCCAc0CAQAwgYgxCzAJBgNVBAYTAkNOMRAwDgYDVQQIDAdCZWlqaW5nMRAw
DgYDVQQHDAdCZWlqaW5nMRMwEQYDVQQKDApNeSBDb21wYW55MRowGAYDVQQDDBF3
d3cubXljb21wYW55LmNvbTEkMCIGCSqGSIb3DQEJARYVZXhhbXBsZUBteWNvbXBh
......
QIdb7HyrK9Xly3dhTVNpkn/675drBf/0V23RyrXusoJMw+IOtPd9EtyRCvue8b2Z
niekGvbI+giNyc45OBJwnuVo83tU528UyoZIpHjfNaw15NYCbhSecrxZYmY3xWnE
uPIZsMavDocgoiGpPN8TyX2XMZUkLtLFCFJ+yU2yD4OycgPJTJSP5zJSCv+JV0mh
1SPr6Vir4rnT7s+EL+ZkubfWrTvVSmkjyg==
-----END CERTIFICATE REQUEST-----
```

- 执行以上命令时所需填写的内容解释如下表所示：

字段	描述
Country Name	公司/组织所在国家ISO代码，中国代码为CN，其它国家代码可参考ISO Code。
State or Province Name	公司/组织所在的省/州名称。
Locality Name	公司/组织所在城市名称。
Organization Name	公司/组织的名称，此处应该出完整全称。
Organizational Unit Name	产品或分支机构名称，可选。
Common Name	公司/组织的完全限定域名（FQDN，Fully-Qualified Domain Name），例如：www.mycompany.com。
Email Address	管理员的Email地址。
A challenge password	加密证书请求的密码。
An optional company name	可选公司/组织名称。

3、向CA中心提交数字证书认证申请。用户可选择VeriSign、GeoTrust等第三方认证机构。

- 用户也可以执行命令`openssl x509 -req -days 365 -in csr.pem -signkey my-private-key.pem -out my-certificate.pem`，对证书进行自签名，用于网站的测试。其中，`-days 365`代表证书有效期为365天；`-in csr.pem`用来指定步骤2生成CSR文件；`-signkey my-private-key.pem`用来指定用于对CSR进行签名的私钥，本示例中使用步骤1生成的私钥进行签名；`-out my-certificate.pem`用来指定生成的证书名称。
- 查看生成的证书文件内容如下：

```
-----BEGIN CERTIFICATE-----
MIIDjjCCAnYCCQC4xa7g5APX/jANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
Q04xEDAOBgNVBAgMB0JlaWppbmcxEDAOBgNVBAcMB0JlaWppbmcxEzARBgNVBAoM
Ck15IENvbXBhbnkxGjAYBgNVBAMMEXd3dy5teWNvbXBhbnkuY29tMSQwIgYJKoZI

....
 Q1PBQWanUPWbZ2+AIudPWpDkDtq6uZkTTSKNd+6E1f5bIlGGvD0eu/gdYFaJN8Ut
 aUSjls8bToQhXs7EAjzEABM9M8BmaQEkEmPutc/y8KqND1Dv8hox/z6olppmwMn5
 9hgcZecsoJ0qAUJC7kqfpSkpitXLSAsE/1OGJ8MhTtpELQsCO0N5m/h2wtIGaI08
 sLk=
-----END CERTIFICATE-----
```

API参考

1 简介

证书管理模块主要用于管理用户的SSL证书，方便用户录入以及查看SSL证书。

本文档适用于开发人员，主要提供接口为创建证书，查看证书列表等。

1.1 名词解释

名词	说明
certId	证书ID, 全局唯一
certName	证书自定义名称
certServerData	服务器证书
certPrivateData	证书私钥
certLinkData	证书链数据
certCommonName	证书通用名称
certDNSNames	证书包含的域名
certStartTime	证书生效时间
certStopTime	证书到期时间
uploadPublicKey	上传证书公钥，用来加密数据。之后需要使用对应的私钥机密数据
certType	证书类型，1表示服务端证书，私钥为必填项；2表示客户端证书，私钥为非必填项

1.2 格式规范

参数	说明
certId	格式：“cert-xxxxxxxxxxxx”（12位随机字符串），例子：“cert-5atue8m3sxsv”。

2.调用方式

证书服务 API以Restful API的形式提供。

2.1 请求结构

2.1.1 通信协议

目前支持HTTP，暂不支持HTTPS。

2.1.2 请求方法

不同类型的API使用不同的请求方法，如下所示：

API类型	请求方法
读取资源	GET
修改资源	PUT
批量查询/创建资源	POST
删除资源	DELETE
获取资源状态	HEAD

2.1.3 字符编码

请求和返回结果都使用UTF-8编码。

2.2 公共参数

2.2.1 公共请求头

头域	是否必须	说明
Authorization	必须	见认证机制
Content-Type	必须	应该总是`application/json; charset=utf-8`
x-bce-date	必须	请求时间，格式见[时间格式](#6.2 时间格式)。服务器会比较该时间与当前服务器时间，如果两者相差超过30分钟，则返回RequestExpired错误

2.2.2 公共响应头

头域	说明
Content-Type	总是`application/json; charset=utf-8`
x-bce-request-id	对应请求的[requestId](#6.3 RequestId)

2.2.3 认证机制

见公有云API规范-认证机制

3.api列表

3.1 创建证书

方法	API	说明
POST	/v1/certificate	创建证书

请求参数

参数名称	参数类型	是否必须	说明
certName	String	必须	证书的名称。长度限制为1-65个字符，以字母开头，只允许包含字母、数字、'-'、'/'、'.'、''，*Java正则表达式* `^[a-zA-Z]a-zA-Z0-9\-/\.]{2,64}$`
certServerData	String	必须	服务器证书的数据内容 (Base64编码)
certPrivateData	String	必须	证书的私钥数据内容 (Base64编码)
certLinkData	String	可选	证书链数据内容 (Base64编码)
certType	Integer	可选	证书类型，非必填，默认为1

返回参数

返回值为一个certificate对象（只包含certId和certName）。

可能异常

异常code	说明
CertExceedLimit (409)	超过用户最大证书数
UnmatchedPairParameterInvalidException (400)	证书有效时间不包含当前时间
PrivateKeyParameterInvalid (400)	私钥解析异常
CertificateParameterInvalid (400)	证书解析异常
CertChainParameterInvalid (400)	证书链解析异常
UnmatchedPairParameterInvalid (400)	公钥私钥不匹配

请求示例

```
    POST /v1/certificate HTTP/1.1
    HOST: certificate.baidubce.com
    Authorization: {authorization}
    Content-Type: application/json; charset=utf-8
    x-bce-date: 2014-06-01T23:00:10Z

    {
        "certName": "TestCert",
        "certServerData": "-----BEGIN CERTIFICATE-----
\ngBs4mWchJjzl0IM3B+TrAD...\n-----END CERTIFICATE-----",
        "certPrivateData": "-----BEGIN RSA PRIVATE KEY-----
\n6JCfAxrrh7AoCg0jhqjgN/by0U2jwG/xFe...\n-----END RSA PRIVATE KEY-----"
    }
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4

{
    "certId": "cert-5atue8m3sxsv",
    "certName": "TestCert"
}
```

3.2 修改证书名称

方法	API	说明
PUT	/v1/certificate/{certId}?certName	修改证书名称

可能异常

异常code	说明
AccessDeniedException	无权限访问
ResourceNotFoundException	证书不存在

请求参数

参数名称	参数类型	是否必须	说明
certName	String	必须	证书名称

请求示例

```
PUT /v1/certificate/cert-5atue8m3sxsv?certName HTTP/1.1
HOST: certificate.bj.bce-internal.baidu.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
x-bce-date: 2014-06-01T23:00:10Z

{
    "certName": "TestCert"
}
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4
```

3.3 查看证书列表

方法	API	说明
GET	/v1/certificate	查看用户的证书列表

返回参数

参数名称	参数类型	说明
certs	List<certificate>	由certificate组成的数组

请求示例

```
GET /v1/certificate HTTP/1.1
HOST: certificate.baidubce.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
x-bce-date: 2014-06-01T23:00:10Z
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4

{
    "certs": [
        {
            "certId": "cert-5atue8m3sxsv",
            "certName": "TestCert",
            "certCommonName": "httpstest.baidu.com",
            "certStartTime": "2014-06-01T23:00:10Z",
            "certStopTime": "2015-06-01T23:00:10Z",
            "certCreateTime": "2014-06-01T23:00:10Z",
            "certUpdateTime": "2014-06-01T23:00:10Z"
        },
        {
            "certId": "cert-xsdfwerdty67",
            "certName": "TestCertFail",
            "certCommonName": "httpstestfail.baidu.com",
            "certStartTime": "2014-06-01T23:00:10Z",
            "certStopTime": "2015-06-01T23:00:10Z",
            "certCreateTime": "2014-06-01T23:00:10Z",
            "certUpdateTime": "2014-06-01T23:00:10Z"
        }
    ]
}
```

3.4 获取证书信息(无证书公钥私钥)

方法	API	说明
GET	/v1/certificate/{certId}	获取证书ID为{certId}的应用信息

返回参数

返回值为一个certificate对象

certificate参数列表

参数名称	参数类型	说明
certId	String	证书ID
certName	String	证书名称
certCommonName	String	证书通用名称
certStartTime	DateTime	证书生效时间
certStopTime	DateTime	证书到期时间
certCreateTime	DateTime	证书创建时间
certUpdateTime	DateTime	证书更新时间
certType	Integer	证书类型

可能异常

异常code	说明
AccessDeniedException	无权限访问
ResourceNotFoundException	证书不存在

请求示例

```
GET /v1/certificate/cert-5atue8m3sxsv HTTP/1.1
HOST: certificate.bj.bce-internal.baidu.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
x-bce-date: 2014-06-01T23:00:10Z
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4

{
    "certId": "cert-5atue8m3sxsv",
    "certName": "TestCert",
    "certCommonName": "httpstest.baidu.com",
    "certStartTime": "2014-06-01T23:00:10Z",
    "certStopTime": "2015-06-01T23:00:10Z",
    "certCreateTime": "2014-06-01T23:00:10Z",
    "certUpdateTime": "2014-06-01T23:00:10Z",
    "certType": 1
}
```

3.5 删除证书

方法	API	说明
DELETE	/v1/certificate/{certId}	删除证书

可能异常

异常code	说明
OperationNotAllowedException	证书使用中
AccessDeniedException	无权限访问
ResourceNotFoundException	证书不存在

请求示例

```
DELETE /v1/certificate/cert-5atue8m3sxsv HTTP/1.1
HOST: certificate.bj.bce-internal.baidu.com
Authorization: {authorization}
Content-Type: application/json; charset=utf-8
x-bce-date: 2014-06-01T23:00:10Z
```

返回示例

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4
```

3.6 替换证书（id不变）

方法	API	说明
PUT	/v1/certificate/{certId}?certData	替换过期且不在使用中的证书

请求参数

参数名称	参数类型	是否必须	说明
certName	String	必须	证书的名称。长度限制为1-65个字符，以字母开头，只允许包含字母、数字、’-‘、’/’、’.’、’’，*Java正则表达式*`^[a-zA-Z]a-zA-Z0-9\-/\.]{2,64}$`
certServerData	String	必须	服务器证书的数据内容 (Base64编码)
certPrivateData	String	必须	证书的私钥数据内容 (Base64编码)
certLinkData	String	可选	证书链数据内容 (Base64编码)
certType	Integer	可选	证书类型，非必填，默认为1

可能异常

异常code	说明
OperationNotAllowedException(409)	证书使用中或者证书
AccessDeniedException（403）	证书非本用户或子用户无该证书运维权限
ResourceNotFoundException（404）	无证书
CertExceedLimit (409)	超过用户最大证书数
UnmatchedPairParameterInvalidException (400)	证书有效时间不包含当前时间
PrivateKeyParameterInvalid (400)	私钥解析异常
CertificateParameterInvalid (400)	证书解析异常
CertChainParameterInvalid (400)	证书链解析异常
UnmatchedPairParameterInvalid (400)	公钥私钥不匹配

请求示例

```
    PUT /v1/certificate/cert-5atue8m3sxsv?certData HTTP/1.1
    HOST: certificate.baidubce.com
    Authorization: {authorization}
    Content-Type: application/json; charset=utf-8
    x-bce-date: 2014-06-01T23:00:10Z

    {
        "certName": "TestCert",
        "certServerData": "-----BEGIN CERTIFICATE-----
\ngBs4mWchJjzl0IM3B+TrAD...\n-----END CERTIFICATE-----",
        "certPrivateData": "-----BEGIN RSA PRIVATE KEY-----
\n6JCfAxrrh7AoCg0jhqjgN/by0U2jwG/xFe...\n-----END RSA PRIVATE KEY-----",

    }
```

返回示例

```
    HTTP/1.1 200 OK
    Content-Type: application/json; charset=utf-8
    x-bce-request-id: 9ebc57ed-1ff5-480f-b5b1-6847ff54f2b4
```

6 附表

6.1 错误状态码

CODE	MESSAGE	HTTP Status Code	说明
OperationNotAllowed	Resource status conflict.	409 CONFLICT	资源状态冲突，不能执行请求操作
ResourceNotFound	Resource not found.	404 NOT_FOUND	请求资源不存在
ParametersNotChanged	Parameters not changed.	403 FORBIDDEN	请求参数未发生变化
ResourceNameDuplicated	Resource name duplicated.	409 CONFLICT	资源名称重复
ParametersInvalid	Parameters invalid.	400 BAD REQUEST	请求参数不合法
AccessDenied	Access denied.	403 FORBIDDEN	无权限访问

6.2 时间格式

日期与时间的表示有多种方式。为统一起见，除非是约定俗成或者有相应规范的，凡需要日期时间表示的地方一律采用UTC时间，遵循ISO 8601，并做以下约束：

1. 表示日期一律采用`YYYY-MM-DD`方式，例如`2014-06-01`表示2014年6月1日
2. 表示时间一律采用`hh:mm:ss`方式，并在最后加一个大写字母Z表示UTC时间。例如`23:00:10Z`表示UTC时间23点0分10秒。
3. 凡涉及日期和时间合并表示时，在两者中间加大写字母T，例如`2014-06-01T23:00:10Z`表示UTC时间2014年6月1日23点0分10秒。

6.3 RequestId

所有请求都应该唯一地对应一个ID，用于标识该请求。requestId可用于问题定位、性能分析等等多个场景。所有的日志都应该带有requestId以便后续分析。

JAVA-SDK

概述

本文档主要介绍证书管理模块Java SDK的安装和使用。

安装SDK工具包

运行环境

Java SDK工具包可在jdk1.8以上的环境下运行。

方式一：使用Maven安装

在Maven的pom.xml文件中添加bce-java-sdk的依赖：

```
<dependency>
<groupId>com.baidubce</groupId>
<artifactId>bce-java-sdk</artifactId>
<version>{version}</version>
</dependency>
```

其中，`{version}`为版本号，可以在SDK下载页面找到。

方式二：直接使用JAR包安装

1. 在官方网站下载Java SDK压缩工具包。
2. 将下载的`bce-java-sdk-version.zip`解压后，复制到工程文件夹中。
3. 在Eclipse右键“工程 -> Properties -> Java Build Path -> Add JARs”。
4. 添加SDK工具包`lib/bce-java-sdk-{version}.jar`和第三方依赖工具包`third-party/*.jar`。 其中，`version`为版本号。

SDK目录结构

```
com.baidubce
├── auth                                    //BCE签名相关类
├── http                                    //BCE的Http通信相关类
├── internal                                //SDK内部类
├── model                                   //BCE公用model类
├── services
│       └── cert                            //证书管理服务相关类
│           ├── model                       //证书管理服务内部model，如Request
或Response
│           └── CertClient.class            //证书管理服务客户端入口类
├── util                                    //BCE公用工具类
├── BceClientConfiguration.class            //对BCE的HttpClient的配置
├── BceClientException.class                //BCE客户端的异常类
├── BceServiceException.class               //与BCE服务端交互后的异常类
├── ErrorCode.class                         //BCE通用的错误码
└── Region.class                            //BCE提供服务的区域
```

创建CertClient（必做）

用户可以参考如下代码新建一个CertClient，需要传入aksk和endpoint（证书服务的url）

```
String endpoint = "https://certificate.baidubce.com";
// 服务url
String accessKey = "your-access-key-id";
// 用户ak
String secretKey = "your-secret-access-key";
// 用户sk
CertClient certClient = CertClient.createCertClient(accessKey, secretKey,
endpoint);
```

方法列表

1.创建证书

请求参数

参数名称	参数类型	是否必须	说明
certName	String	必须	证书的名称。长度限制为1-65个字符，以字母开头，只允许包含字母、数字、'-'、'/'、'.'、''，*Java正则表达式* ^[a-zA-Z]a-zA-Z0-9\-/\.]{2,64}$
certServerData	String	必须	服务器证书的数据内容 (Base64编码)
certPrivateData	String	可须	证书的私钥数据内容 (Base64编码)，证书类型为1时必填
certLinkData	String	可选	证书链数据内容 (Base64编码)
certType	Integer	可选	证书类型，1表示服务端证书，2表示客户端证书，默认为1

返回值:CertCreateResponse

属性名	属性类型	说明
certId	String	证书id
certName	String	证书名

参考代码

```
// 准备参数
String certName = "Your-certificate-name";
String certServerData = "Your-certificate-server-data";
String certPrivateData = "Your-certificate-private-data";
String certLinkData = "Your-certificate ";

// 构造请求
CertCreateRequest request = new CertCreateRequest();
request.setCertName(certName);
request.setCertPrivateData(certPrivateData);
request.setCertServerData(certServerData);
request.setCertLinkData(certLinkData);

// 发出请求，获取结果
CertCreateResponse createResponse = certClient.createCert(request);
```

可能异常

异常code	说明
CertExceedLimit (409)	超过用户最大证书数
UnmatchedPairParameterInvalidException (400)	证书有效时间不包含当前时间
PrivateKeyParameterInvalid (400)	私钥解析异常
CertificateParameterInvalid (400)	证书解析异常
CertChainParameterInvalid (400)	证书链解析异常
UnmatchedPairParameterInvalid (400)	公钥私钥不匹配

2.获取证书列表

请求参数：无

返回值:CertCreateResponse

属性名	属性类型	说明
certs	List<CertificateMeta>	证书信息的列表

CertificateMeta

属性名	属性类型	说明
certId	String	证书ID
certName	String	证书名称
certCommonName	String	证书通用名称
certStartTime	DateTime	证书生效时间
certStopTime	DateTime	证书到期时间
certCreateTime	DateTime	证书创建时间
certUpdateTime	DateTime	证书更新时间

参考代码

```
// 发出请求，获取结果
CertListResponse listResponse = certClient.listUserCerts();
```

3.获取单个证书的信息(不包含证书公钥私钥信息)

请求参数

参数名称	参数类型	是否必须	说明
certId	String	必须	证书id

返回值:CertificateMeta

属性名	属性类型	说明
certId	String	证书ID
certName	String	证书名称
certCommonName	String	证书通用名称
certStartTime	DateTime	证书生效时间
certStopTime	DateTime	证书到期时间
certCreateTime	DateTime	证书创建时间
certUpdateTime	DateTime	证书更新时间

参考代码

```
// 准备参数
String certId = "Your-cert-id";

// 发出请求，获取结果
CertificateMeta certificateMeta = certClient.getCertInfo(certId);
```

4.重命名证书

请求参数

参数名称	参数类型	是否必须	说明
certName	String	必须	证书的名称。长度限制为1-65个字符，以字母开头，只允许包含字母、数字、’-‘、’/’、’.’、’’，*Java正则表达式* ^[a-zA-Z]a-zA-Z0-9\-/\.]{2,64}$
certId	String	必须	证书id

返回值:CertInServiceListResponse

属性名	属性类型	说明
certId	String	证书id
certName	String	证书名

参考代码

```
// 准备参数
String certId = "Your-cert-id";
String newName = "Cert-new-name";

// 发出请求
certClient.updateCertName(certId, newName);
```

可能异常

异常code	说明
AccessDeniedException（403）	无权限访问
ResourceNotFoundException（404）	证书不存在

5.删除证书

请求参数

参数名称	参数类型	是否必须	说明
certId	String	必须	证书id

返回值：无

参考代码

```
// 准备参数
String certId = "Your-cert-id";

// 发出请求
certClient.delete(certId);
```

可能异常

异常code	说明
OperationNotAllowedException（409）	证书使用中
AccessDeniedException（403）	无权限访问
ResourceNotFoundException（404）	证书不存在

6.替换证书（证书id不变）

请求参数

参数名称	参数类型	是否必须	说明
certId	String	必须	证书id
certName	String	必须	证书的名称。长度限制为1-65个字符，以字母开头，只允许包含字母、数字、’-‘、’/’、’.’、’’，*Java正则表达式* ^[a-zA-Z]a-zA-Z0-9\-/\.]{2,64}$
certServerData	String	必须	服务器证书的数据内容 (Base64编码)
certPrivateData	String	可须	证书的私钥数据内容 (Base64编码)，证书类型为1时必填
certLinkData	String	可选	证书链数据内容 (Base64编码)
certType	Integer	可选	证书类型，1表示服务端证书，2表示客户端证书，默认为1

返回值：无

参考代码

```
// 准备参数
String certId = "Your-certificate-id";
String certName = "Your-certificate-name";
String certServerData = "Your-certificate-server-data";
String certPrivateData = "Your-certificate-private-data";
String certLinkData = "Your-certificate ";

// 构造请求
CertCreateRequest request = new CertCreateRequest();
request.setCertName(certName);
request.setCertPrivateData(certPrivateData);
request.setCertServerData(certServerData);
request.setCertLinkData(certLinkData);

// 发出请求
certClient.replaceCertData(certId, request);
```

可能异常

异常code	说明
OperationNotAllowedException（409）	证书使用中
AccessDeniedException（403）	无权限访问
ResourceNotFoundException（404）	证书不存在
UnmatchedPairParameterInvalidException (400)	证书有效时间不包含当前时间
PrivateKeyParameterInvalid (400)	私钥解析异常
CertificateParameterInvalid (400)	证书解析异常
CertChainParameterInvalid (400)	证书链解析异常
UnmatchedPairParameterInvalid (400)	公钥私钥不匹配

版本说明

v1.0.0

首次发布。

常见问题

证书上传失败如何处理?

1.提示【证书解析异常, 请修改后重试】

私钥填写错误导致系统无法解析时会导致此错误。请检查私钥是否填写错误；检查证书和私钥是否为PEM格式，如果证书不是PEM格式，需要进行转换，具体转换方法，请查看格式转换。检查是否漏填前缀和后缀，证书私钥以`-----BEGIN RSA PRIVATE KEY-----`开头，以`-----END RSA PRIVATE KEY-----`结尾；或者以`-----BEGIN PRIVATE KEY-----`开头，以`-----END PRIVATE KEY-----`结尾；

2.提示【公钥私钥不匹配, 请修改后重试】

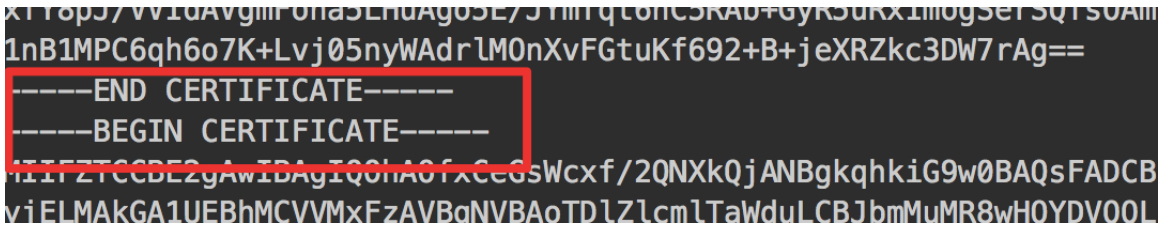
私钥和证书中的公钥不匹配时会导致此错误。检查私钥是否与证书匹配，具体检查方式，请查看格式转换。

3.提示【证书解析异常, 请修改后重试】

证书数据填写错误导致系统无法解析时会导致此错误。请检查证书是否填写错误；检查证书和私钥是否为PEM格式，如果证书不是PEM格式，需要进行转换，具体转换方法，请查看格式转换。检查是否漏填前缀和后缀，证书以`-----BEGIN CERTIFICATE-----`开头，以`-----END CERTIFICATE-----`结尾。

4.提示【证书链解析异常, 请修改后重试】

证书链填写不完整时会导致此错误；如果CA提供的证书中包含了证书链（有些CA会提供bundle形式的打包证书，打包证书中从第二个CERTIFICATE开始就是证书链），可直接将包含证书链的证书文件直接上传。如果证书文件中包含多份证书，需要人为合并后上传。拼接规则为服务器证书放第一份，中间证书放第二份，中间不要有空行。一般情况下，机构在颁发证书的时候会有对应说明，请注意查阅规则说明。



5.提示【证书不在有效期内】

证书过期时会导致此错误；过期证书暂不支持上传。

6.提示【请求创建的证书数量超过限制】

上传的证书数量超过默认配额200时会引发此错误；解决方法是删除已经废弃或者过期的证书，或者联系客服增加配额。

术语表

A

ABC Robot

百度智能云ABC Robot是百度面向机器人合作伙伴推出的集语音、视觉、NLP、知识库、运动控制等AI技术为一体的机器人开放平台。依托百度全球领先的AI技术，搭载ABC Robot平台的机器人可以实现看、听、说、动多模态的人机交互，胜任复杂场景下的业务咨询、业务办理、人机协作等需求。

Access Key ID / Secret Access Key

用户开通BOS服务后，系统会自动分配一对 Access Key ID / Secret Access Key，该密钥对将在用户向BOS发起请求时用做签名验证。Access Key ID用于标示用户，Secret Access Key 是用户用于加密签名字符串和 BOS 服务用来验证签名字符串的密钥。

ACL

访问控制列表（Access Control List）作为应用在子网上的防火墙组件帮助用户实现子网级别的安全访问控制。

ADX

Ad Exchange，广告交易平台

Application Programming Interface 应用程序编程接口

API是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

Auto Scaling 百度智能云弹性伸缩

百度智能云弹性伸缩(AS)是自动化扩缩容用户云资源的管理服务，当您业务所需的云资源用量经常性变化时，弹性伸缩会是您使用云资源的理想方式。

A记录

A (Address) 记录是用来指定主机名（或域名）对应的IP地址记录。用户可以将该域名下的网站服务器指向到自己的网页服务器(web server)上。同时也可以设置域名的子域名。通俗来说A记录就是服务器的IP，域名绑定A记录就是告诉DNS，当输入域名的时候给你引导向设置在DNS的A记录所对应的服务器。

安全策略(SecurityPolicy)

安全策略是用于配置直播流（Stream）安全机制的一组策略，包括在直播过程中使用的推流认证、播放认证、内容加密(DRM方案)、防盗链（Referer/IP 黑白名单）等。

B

Baidu Cloud Compute 百度智能云服务器

云服务器BCC是一种简单高效，处理能力可弹性伸缩的计算服务。无需购买硬件，即可迅速创建或释放任意多台云服务器。提升了运维效率，降低了IT成本。为用户快速构建稳定可靠的应用，降低了网络规模计算的难度，便于用户专注于核心业务的创新。

Baidu Cloud Monitor 百度智能云监控

通过监控采集、存储、展示、计算、汇聚报警、统计的一体化解决方案，为百度智能云用户提供全面、可靠、及时的监控服务。

Baidu IntelliEdge 智能边缘

百度边缘计算产品，包含了智能边缘本地运行包和智能边缘云端管理套件。

Baidu Load Balance 百度负载均衡

负载均衡BLB是对多台云服务器服务进行流量分发的负载均衡服务，BLB能够均衡应用程序的流量，将前端并发访问转发给后台多台云服务器，实现业务水平扩展。BLB还能保证故障自动切换，及时的消除服务的单点故障，提升服务的可用性。

Baidu Object Storage 百度对象存储

百度对象存储BOS提供稳定、安全、高效、高扩展性以及高性价比的存储服务，支持5TB以内任意数据多地域跨集群的存储，以达到资源统一利用，降低使用难度，提高工作效率。

BCC安全组

安全组即对一组BCC实例所约定的安全访问规则的集合。BCC实例可以选择默认安全组或者自定义安全组。默认安全组允许所有入站和出站访问全部通过。自定义安全组定义自己想要的入站与出站规则。

BCC镜像

镜像是BCC实例运行环境的模板，包括操作系统和已安装的软件及配置信息。您可以使用镜像创建BCC实例和更换BCC实例的系统盘。百度智能云提供了多种公共镜像供您在创建BCC实例时选择。您还可以基于某一时刻的BCC系统盘创建您专属的自定义镜像，该镜像只能您自己看到。

BCC快照

快照是您指定的数据集合的一个可用拷贝，该拷贝包括相应数据在您指定的时间点（拷贝开始的时间点）的镜像。快照的作用主要是在线数据备份与恢复，当设备发生应用故障或者文件损坏时可以快速恢复数据，将数据恢复到某个时间点的状态。快照的另一个作用是为您提供了另外一个数据访问通道，当原数据在处理线应用时，用户可以访问快照数据，还可以利用快照进行测试等工作。

BCC实例

BCC实例是一个虚拟的计算环境，包含CPU、内存等最基础的计算组件，是云服务器呈献给每个用户的实际操作实体。BCC实例是云服务器最为核心的概念，支持IP绑定，镜像和快照等功能，诸如CDS磁盘、SCS简单缓存服务只有挂载在BCC实例后才可使用。

BCM-Agent

BCM-Agent是BCM服务提供的监控采集客户端，用户可以选择下载并安装到自己的BCC（百度智能云服务器）中。

BINLOG

BINLOG是数据库执行写操作产生的SQL日志，在主从架构环境下可以作为从库的同步数据的输入，在数据恢复时可以根据BINLOG进行点对点的恢复。

Border Gateway Protocol 边界网关协议

是运行于 TCP 上的一种自治系统的路由协议。

BOS FS

BOS FS 是BOS提供的一个免费工具，可以将远程BOS服务挂载到本地文件系统中，实现类似于网络文件

系统的功能。目的是便于用户使用系统命令或者标准的 POSIX 接口访问 BOS 上的文件，就像使用本地磁盘一样方便。

BOS 权限控制

BOS提供用户签名验证、访问控制列表ACL和对象限时访问相结合的权限控制方式，为用户提供安全可靠的数据保护。其中用户签名验证用AK/SK非对称加密的方法对URL进行签名来实现用户身份验证；ACL根据签名识别用户身份后，提出请求Bucket的访问权限信息，并根据相应的权限信息对请求做出响应；对象限时访问让用户可以提供自定义时间内有效的URL用于下载等应用场景。

BRN（Baidu Resource Name）

是百度智能云资源的唯一标识，您可以通过该标识调用或者识别百度智能云资源，比如API调用、DuerOS技能调用和IAM策略识别等。

BSC 百度流式计算

百度流式计算（BSC）提供低延迟、高吞吐、高准确性的流式数据实时处理能力，完全兼容Apache Spark基础上进行深度优化；基于百度自研BigSQL，通过SQL语句实现复杂业务逻辑数据处理，简单易用；提供全面、实时的监控指标。

BTS 百度表格服务

百度表格服务BTS（Baidu Table Service）是构建在百度自研的分布式表格存储Table上的Nosql数据存储服务，提供海量结构化、半结构化数据的存储和实时访问。

Bucket

Bucket可以理解为命名空间。Bucket相当于单机系统中的一块硬盘，而Object相当于存放在硬盘上的文件。用户最多可创建100个Bucket，但每个Bucket中存放的Object数量和大小不受限制。 Bucket的名称在一个Region内具有唯一性，且不可修改。

百度数据工厂Pingo

百度数据工厂Pingo是百度智能云上提供的集成的批量和流式数据处理系统，它在弹性计算资源管理和改进的数据访问管理层之上，运行优化的Spark计算引擎，提供SQL分析和DataFrame API，支持低延时的批量和流式数据加工和处理，对外提供REST Service任务执行接口。

百度数据科学平台 JARVIS

Jarvis是一款可视化的数据挖分析科学环境，集成了百度开源的PaddlePaddle深度学习框架，也支持其他优秀的框架如Tensorflow。帮助开发者管理复杂的项目环境，同时提供代码和可视化的交互模式，帮助用户便捷的完成数据挖分析。

报警

报警泛指用户利用监控项、报警策略、报警动作来实现事件通报功能。

报警策略

报警策略是指一种规则，用于表达监控项在满足什么样的条件下处于“异常”状态。

报警策略状态

报警策略状态是指报警监控项所属状态，分为“正常”、“异常”和“数据不足”三种状态。

- 当报警策略中设置的报警规则不成立，不满足报警条件时，属于“正常”状态；
- 当报警策略中设置的报警规则成立，满足报警条件时，属于“异常”状态；
- 当报警策略所关联的监控项在所判断的时间范围内无任何数据时，属于“数据不足”状态。

报警动作

报警动作是指用户针对报警策略的某一状态执行相应的触发动作。目前仅支持“发送报警邮件”和“发送报警短信”两种动作，后续会支持“自动扩容”、“BCC开关机”等动作。

报警屏蔽

报警屏蔽是指不触发报警动作，包括邮件和短信，但仍然会对报警策略的状态进行判断和展示。

备份存储

用于持久化保存数据库数据或日志等备份的底层存储资源。

备节点

主实例中用来保证服务可靠性的节点，备节点通过异步复制与主节点保持同步，备节点对用户不可见。

本地磁盘

专属实例中常用的磁盘介质，可以为专属实例提供高IO的读写性能。在客户专属的物理服务器上以RAID5实现物理磁盘的高可用，当有一块物理磁盘发生故障时，可以保证磁盘阵列和客户专属实例的正常使用。在专属实例中可以支持对本地磁盘进行离线扩容的操作。

边缘节点

云计算中心之外的任意地点，根据不同的场景需要和延迟容忍程度，选择包括但不限于：本地设备侧、本地局域网络节点（网关、控制器、控制系统服务器）、广域网络节点（基站、CDN）。

标签（tag）

一个标签是一个key-value对，用于提供额外的信息，如"设备号=95D8-7913"、"型号=ABC123"、"出厂编号=1234567890"等。

播放认证(Play)

为了避免播放地址长期暴露带来风险，LSS支持播放地址加密策略。开启播放认证后，用户可设置播放地址的有效期，并且需要加入播放token才可以进行播放。

C

CC (Challenge Collapsar)攻击

HTTP Flood，是针对Web服务在 OSI 协议第七层协议发起的攻击，攻击者极力模仿正常用户的网页请求行为，发起方便、过滤困难，极其容易造成目标服务器资源耗尽无法提供服务。

CLI

Command Line Interface 命令行界面，用户在命令行界面根据命令提示符键入命令，执行需要的操作。

Cloud Disk Service 云磁盘服务

云磁盘服务CDS是支持随机读写的弹性存储系统，为云服务器提供原始的块级存储设备，为云磁盘提供后端的存储支持。 从云磁盘的需求来看，衡量CDS的指标主要有可靠性和可用性、吞吐、IOPS以及异常恢复时间等。CDS云磁盘服务比本地磁盘的可靠性、可用性更高，有独立于云服务器的生命周期，支持快速扩容、在线备份和回滚。

CNAME记录

即别名记录。这种记录允许您将多个名字映射到同一台计算机。 通常用于同时提供WWW和MAIL服务的计算机。例如，有一台计算机名为“host.mydomain.com”（A记录）。 它同时提供WWW和MAIL服务，为了

便于用户访问服务。可以为该计算机设置两个别名（CNAME）：WWW和MAIL。 这两个别名的全称就是“www.mydomain.com”和以“mail.mydomain.com”。实际上它们都指向“host.mydomain.com”。

CNAME域名

在百度智能云CDN接入加速域名后，系统给对应域名分配一个"CNAME域名"，用将加速域名在域名服务提供商处完成CNAME配置指向这个CNAME域名。配置生效后，域名解析的工作就正式转向百度智能云，该域名所有的请求都将转向百度智能云CDN的节点。

CockroachDB

HTAP 数据库 CockroachDB（HTAP Database for CockroachDB）是百度智能云基于全球最新的 NewSQL 开源数据库 CockroachDB 打造的一款同时支持联机事务处理（OLTP）和联机分析处理（OLAP）两种业务的融合型分布式数据库产品。

Column

即属性列，每行都可按需定义若干属性列，每个属性对应记录该行的一部分数据信息。

Content Delivery Network 内容分发网络

将源站内容分发至全国所有的节点，缩短用户查看内容的延迟，提高用户访问网站的响应速度与网站的可用性，解决网络带宽小、用户访问量大、网点分布不均等问题。

Cookie

有时也用其复数形式Cookies，指某些网站为了辨别用户身份、进行session跟踪而储存在用户本地终端上的数据（通常经过加密）。

CPA

Cost Per Action，每次行动成本，一般指每注册成本

CPC

Cost Per Click，单次点击费用

CPM

Cost Per Mile，千次展示费用

CPT

Cost Per Time，单位时间费用

CTR

Click Through Rate，点击率

CU（Compute Unit）

是流计算所提供计算资源的基本单位，1CU 包含1核CPU和4G内存。 计费的标准是CU 单价使用量使用时长。

CVCA节点

节点是会话流中的基本单位。节点分为根节点、判断节点和动作节点。

CVCA实体

用户话术中包含的关键信息或限定条件，可以理解为用户需要提供的筛选条件。

CVCA意图

意图表示用户的目的。

CVCA中控

智能客服的调度中心，负责多轮会话、知识库，以及对接的第三方服务等模块的开关设置和调度策略。

CVR

Click Value Rate，广告点击转化率，CVR=(转化次数/点击次数)*100%

策略（Policy）

策略是用户权限的一个集合，描述了用户对资源的操作权限，每个策略关联了一个策略语法（ACL）。可以给用户和用户组关联相关的策略，然后通过策略来控制用户具体的访问权限。

策略语法（ACL）

策略语法描述了用户对哪些资源的哪些操作具有权限，因此它是一个服务，资源和操作的集合。

触发器（Trigger）

用户通过触发器定义和管理事件的生成方式。

存储网关（storage gateway）

存储网关是云存储服务的一种，提供了存储协议转换功能。存储网关自身不提供存储能力，其数据是持久化保存在其他云存储服务（对象存储BOS）中的，存储网关主要是起到中转和协议转换作用。

D

DCC安全组

安全组即对一组专属实例所约定的安全访问规则的集合。专属实例可以选择默认安全组或者自定义安全组。默认安全组允许所有入站和出站访问全部通过。自定义安全组定义自己想要的入站与出站规则。

DCC镜像

镜像是专属实例运行环境的模板，包括操作系统和已安装的软件及配置信息。用户可以使用镜像创建专属实例和更换专属实例的系统盘。百度智能云提供了多种公共镜像供用户在创建专属实例时选择。用户还可以基于某一时刻的专属实例系统盘创建自定义镜像，该镜像只有用户自己能看到。

DCC快照

快照是用户指定的数据集合的一个可用拷贝，该拷贝包括相应数据在用户指定的时间点（拷贝开始的时间点）的镜像。当前专属实例仅支持创建系统盘快照，不支持对本地磁盘创建快照。当用户有备份本地磁盘数据需求时，建议重要数据可以存入数据库中，或者购买CDS或BOS等专业的存储产品来进行存储。

DDoS

分布式拒绝服务DDoS攻击指借助于客户/服务器技术，将多个计算机联合起来作为攻击平台，对一个或多个目标发动DDoS攻击，从而成倍地提高拒绝服务攻击的威力。通常，攻击者使用一个偷窃帐号将DDoS主控程序安装在一个计算机上，在一个设定的时间主控程序将与大量代理程序通讯，代理程序已经被安装在Internet上的许多计算机上。代理程序收到指令时就发动攻击。利用客户/服务器技术，主控程序能在几秒钟内激活成百上千次代理程序的运行。

DMP

Data Management Platform，数据管理平台

DNS

Domain Name System，域名系统，因特网上作为域名和IP地址相互映射的一个分布式数据库，能够使用户更方便的访问互联网，而不用去记住能够被机器直接读取的IP数串。通过主机名得到该主机名对应的IP地址的过程叫做域名解析（或主机名解析）。

DOGO实例

实例的概念类似于资源，同一个项目下每个实例有唯一一个实例ID，每一辆车对应一个实例ID，形成实例与车辆的绑定。

DOGO项目

一个账号下可以创建多个项目，同一个项目的车模型一致，当用户创建项目后自动为用户创建一个项目的车辆模型。

DomainKeys Identified Mail 域名密钥识别邮件标准

DKIM是一种电子邮件验证标准。由互联网工程任务组（IETF）开发而成，针对的目标是互联网最严重的威胁之一的电子邮件欺诈。一般来说，发送方会在电子邮件的标头插入DKIM-Signature及电子签名资讯，而接收方则透过DNS查询得到公钥后进行验证。

DSP

Demand Side Platform，需求方平台

DuMap

位置服务 DuMap基于百度地图海量的位置数据以及先进的定位能力，为用户提供精度高、覆盖广、稳定性强的位置服务。

代理实例

代理实例可将数据库请求按读写类型，自动分摊到其对应的主实例或只读实例，实现读写自动分离。每个RDS主实例仅能创建一个代理实例，创建代理实例前须先创建好主实例，主实例释放时，其对应的代理实例会自动释放。

代理实例专用帐号

主实例的已创建帐号无法在代理实例中同步，因此如果需要建立连接，还需要创建一个代理实例专用帐号。代理实例专用帐号的创建、修改等操作，将自动同步到主实例，创建同名帐号。

地域

BCC是一个多区域部署的服务，地域（Region）指的是BCC实例所在的物理位置。百度智能云目前开放多区域支持，您可以根据您的客户群体分布的不同选择不同地域的云服务器服务，请参考区域选择说明。地域内的BCC实例内网间是可以互通的，不同地域之间的BCC实例内网不互通。因为获取物理资源的成本不一，所以每个区域的价格都略有差异。

读写分离

主实例处理数据库写请求，而只读实例处理数据库读请求。

度量（metric）

数据指标的类别，如发动机的温度、发动机转速、模拟量等。

短信配额

短信配额是一天之内用户能发送短信数量的最大限额。

对等连接（Peer connection）

对等连接为用户提供了VPC级别的网络互联服务，使用户实现在不同虚拟网络之间的流量互通，实现同区域/跨区域，同用户/不同用户之间稳定高速的虚拟网络互联。

E

endpoint

IoT Hub 的服务实例，代表一个完整的IoT Hub 服务。

F

File Transfer Protocol 文件传输协议

一种文件传输协议，属于网络协议组的应用层，使得主机间可以共享文件。

FusionDB

百度智能云FusionDB是一种在线分布式云数据库，提供了简单、快速、高性能、高可靠的大规模并行处理（MPP）数据仓库服务。百度智能云FusionDB基于开源Greenplum Database项目开发，提供了完善的监控、安全管理等功能，同时可以轻松实现在线扩展。百度智能云FusionDB具有良好的兼容性，可以轻松使用PostgreSQL相关生态工具，对海量数据进行分析处理，在使用方便、高效、可靠的同时，让您更专注于业务处理。

防盗链

我们支持用户在LSS控制台设置防盗链Referer/IP黑白名单功能。设置Referer黑名单时，禁止黑名单上的URL源站访问直播播放地址；设置Referer白名单时，只允许白名单上的URL源站访问直播播放地址。设置IP黑名单时，禁止黑名单上的IP用户访问直播播放地址；设置IP白名单时，只允许白名单上的IP用户访问播放地址。

分块上传

MultiPartUpload支持将一个大文件切割为一系列特定大小的小数据块，将这些小数据块分别上传到服务端，全部上传完后再在服务端将这些小数据块合并成为一个Object。

分组（group）

可以按标签（tag）对数据点进行分组。

服务地址

负载均衡对外提供服务的访问地址。BLB实例默认提供内网IP作为服务地址，您还可以通过开启公网访问获得公网IP来对公网流量进行负载均衡。

负载均衡

一种解决大量并发访问问题的机制，将访问请求或数据流量均匀地分担到多台节点设备上，并分别进行处理，使接收到请求的服务器独立地回应用户的请求。

负载均衡实例

由百度负载均衡集群提供的虚拟服务。可以提供基于多种监听器（TCP、UDP、HTTP、HTTPS）的负载均衡服务。

G

高可靠

通常用来描述一个系统经过专门的设计，从而减少停工时间而保证服务的高度可用性。RDS具有故障自动

单点切换、数据库自动备份等功能，来保证RDS实例高可用和数据安全。

公网IP

一种公网接入方式，即计算机得到的在互联网上的IP，连接公网的计算机可通过该地址随意互相访问。公网IP在全世界是唯一的。

H

Hadoop

Hadoop是由Apache基金会所开发的一个分布式系统基础架构，使得用户在不了解分布式底层细节的情况下，借助集群对大数据进行高速运算和存储。

Hadoop Streaming

是Hadoop提供的一个编程工具，它允许用户使用任何可执行文件或者脚本文件作为Mapper和Reducer。

Hbase

是一个开源的、与Google Big table类似的非关系型分布式数据库。您可以使用Hbase 随机查询、更新数据，并保持低延迟。

Hive

是基于Hadoop的一种开源数据仓库。它使您能够避免使用较低级别的计算机语言（如java）去编写MapReduce复杂的程序。想了解Hive的更多信息，请查看http://hive.apache.org/。

Hive SQL

是Hive（基于Hadoop构建的数据仓库分析系统）提供的SQL查询方式，用来分析存储在Hadoop分布式文件系统中的数据，并将结构化的数据映射为一张数据库表，提供完整的SQL查询功能。另外可以把SQL语句转换为MapReduce任务进行，使用自身的SQL语句查询所要分析的内容。Hive SQL有利于不熟悉的Hadoop的用户对数据进行查询、分析和汇总。

HTTP referer

HTTP Referer是header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器籍此可以获得一些信息用于处理。

Hypertext Transfer Protocol 超文本传输协议

超文本传输协议HTTP是一种详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议，是ISO7层中的第7层协议。

函数（Function）

函数是用户编写的，由事件触发，执行特定功能的一段代码。

后端服务器

负责接收前端并发访问的一组百度智能云服务器，前端的并发访问会根据BLB实例定义的规则转发到后端服务器进行处理。

环境变量（Environment Variables）

是指在函数计算中用来指定函数运行环境的一些参数。

缓存

是临时的数据交换区域，具有快速的存储速率，能够减小系统负荷，提高数据传输速率。

缓存命中率

用户程序运行时，从缓存内调取的数据量与调用的数据总量的比。

会话

指用户与系统进行的一次交互，包括从发起到退出的整个过程。

会话保持

一种能识别用户与服务器交互过程连续性的机制，在限定的会话生命周期内，负载均衡的同时，能将同一用户相关连的访问请求分配到同一台服务器上。

I

IAM

Identity and Access Management 身份识别与访问管理。IAM是一套建立和维护数字身份，并提供有效、安全的IT资源访问的业务流程和管理手段。它实现了组织信息资产统一的身份认证、授权和身份数据集中管理与审计。

IAM 权限（permission）

权限是允许（allow）或拒绝（deny）一个用户对某个资源执行某种操作。

Instance

BTS产品的基本单位，一个Instance在概念上相当于一个独立的数据库。用户可创建多个Instance。Instance在Region内是唯一的。用户可在Instance中创建表格。

Internet Service Provider 互联网服务提供商

ISP是向广大用户综合提供互联网接入业务、信息业务、和增值业务的电信运营商。在邮件营销领域，ISP主要指电子邮箱服务商。RFC 6650给电子邮箱服务商的定义是：为终端用户提供邮件发送、接收、存储服务的公司或组织。

J

集群

是一组相互独立的、通过高速网络互联的计算单元。对用户，集群是一个独立的服务器。集群内部能够灵活调度，具有高可靠性、高可用性和可缩放性等。

监控项

监控项是指通过采集、推送等手段获取到的某类数据，比如CPU使用率、磁盘占用率等，BCM会对其进行存储，可用于趋势图和信息的展示，也可用于配置报警策略。

监控项（Metric）

监控项（Metric）又称指标，通常是衡量系统某一方面的状态或状况的衡量标准，如系统、服务或程序的性能。监控项随着时间的推移会产生一系列监控数据点。BCE中对云服务默认提供了一系列的监控项，如BCC、RDS等，并有辅助的设施自动将这些监控项数据点发布到BCM中。用户也可以自定义一系列监控项，并采用BCM的API或bcm-tool工具将它们的数据点发布到BCM中，然后按照一定的筛选条件来检索这些数据点的数据或统计数据。

监听器

监听器用于将用户请求转发至后端服务器，再把后端服务器的响应返回给用户。监听器配置有监听协议和监听端口，并按照转发规则将请求转发给后端服务器的端口。监听器还提供了后端服务器的健康检查功

能。

健康检查

检查后端服务器实例的运行状况的服务，对用户指定的路径及服务器端口等项进行检查，当后端服务器实例出现问题时，停止向其分发请求，并且做后续的检查，直至该实例恢复正常。

鉴权（Authorization）

认证和鉴权是两个分离的概念，但是这两个操作又是紧密相关的。认证是获得用户的身份，而鉴权是判断用户是否有操作的权限。因此，鉴权是在认证之后的判断。IAM使用策略的方式来完成权限的判断，用户的策略和RBAC中的角色类似。它定义了资源和操作的集合。服务端在需要鉴权的场景中，传递鉴权上下文（操作者，操作的资源和操作）给IAM，IAM根据策略的配置来进行相应的鉴权操作，最终返回用户是否有权限的结果。

交换分区

在物理内存没有剩余空间时，将磁盘的一些空间当作虚拟内存来使用。

角色（Role）

IAM用户是一种实体身份，有确定的身份凭证，它通常与某个确定的人或应用程序一一对应。IAM role是一种虚拟身份，它关联一组权限，没有确定的身份凭证，必须关联到某个实体身份上才能使用。

节点

伸缩组中每个BCC都被称为一个节点，伸缩组的扩容或者缩容即是通过增加和减少节点数量进行。1个伸缩组中包含N个节点，用户可以为伸缩组配置最大和最小的节点数范围。

聚合函数（aggregator）

可以对一段时间的数据点做聚合，如每10分钟的和值、平均值、最大值、最小值等。

K

克隆实例

按指定的RDS实例创建一个和原实例数据及配置（备份设置、参数设置等）均一致的新实例。

跨区域专线

依托百度超高性能的基础网络资源，实现同一用户/多用户的跨区域内网高速、稳定连接。

快照

是指云服务器或云磁盘在某一特定时间点的完整内容及结构的副本。用户可以随时按快照点进行数据恢复，从而实现包括增量备份等各种形式的备份服务。

扩容配置

伸缩组在扩容时将会按照扩容配置创建新的节点，扩容配置由用户定义并可更改。不同于伸缩组配置，扩容配置仅在新建节点时发生作用，当您更新扩容配置时，已经存在的节点不会受到影响。1个伸缩组对应1个扩容配置。

L

LSS模板

模板是用于配置音视频直播流（Stream）的具体操作功能参数集合。通过模板不仅可以配置音视频编码标准、码率、输出流类型等基本参数，也可以在新建域名时配置同步录制、水印和缩略图。

冷备份

在数据库停机或维护状态下的备份。

临时安全凭证（Security Token）

安全令牌是用户临时访问的密钥，有时效性控制。用户申请了临时的身份一般包括临时的AKSK和Security Token，在使用临时身份进行请求的时候，需要同时使用临时AKSK签名，并在header中传递Security Token，此时IAM认证的结果就是用户的一个临时的身份信息。

临时数据盘

一种基于SSD RAID5高速固态磁盘阵列构建的块级存储设备，具备高吞吐与高速读写能力。通过RAID5冗余阵列实现数据的可靠性。暂不支持基于临时数据盘创建快照。通常用于存储缓存数据或有高IOPS需求的应用场景，区别于CDS云磁盘，不建议在临时数据盘中存储高可靠性和持久性需求的数据。

令牌认证（Token）

IAM进行认证的一种方式是Token认证，此方式是兼容Openstack产品的一种认证。用户提供了用户和密码之后会颁发给用户一个Token Id，此Id对应的用户信息保存在IAM中，用户拿到Token Id之后在请求中添加Token Id 的 header 即可以认证通过，Token 具有时效性。Token方式是对内服务使用的一种方式，不建议对外暴露。

流式计算作业

是流式计算的业务单元，一个作业描述了一个流计算完整的数据处理业务逻辑。

路由表

路由表是指路由器上管理路由条目的列表。

逻辑备份

逻辑备份又称为“导出”，是将数据从数据库中导出并保存在MySQL能够解析的文件中，该文件的格式或者是SQL，或者是以某种符号分割的文本。逻辑备份文件只能用来对数据库进行逻辑恢复，即数据导入，而不能按数据库原来的存储特征进行物理恢复。

M

Mail Delivery Agent 邮件投递代理

MDA的主要功能就是将MTA接收的信件依照信件的流向放置到本机账户下的邮件文件中，或者再经由MTA将信件送到下个MTA。如果信件的流向是到本机，它的功能除了将由MTA传来的邮件放置到每个用户的收件箱，还可以进行邮件过滤（filtering）等。

Mail Exchange 邮件路由

MX记录指向一个邮件服务器。用户可以在MX记录中将该域名指向自己的邮件服务器，然后即可自行操控所有的邮箱设置。例如，当Internet上的某用户要发一封信给user@domain时，该用户的邮件系统通过DNS查找该域名的MX记录，如果MX记录存在，用户计算机就将邮件发送到MX记录所指定的邮件服务器上。

Mail Transfer Agent 邮件传送代理

MTA负责把邮件由一个服务器传到另一个服务器或邮件投递代理。

Mail User Agent 邮件用户代理

MUA是电子邮件客户端程。它接收邮件主机的电子邮件，并提供用户浏览与编写邮件的功能。常见的MUA有Foxmail,Outlook等。

MapReduce

是Hadoop处理数据的一种编程模型，通常用于大规模数据集（>1T）的并行运算。编程思想是将数据的处理方式分为“Map(映射)”和“Reduce(规约)”两种。Map函数首先对大规模数据集进行操作，分发给一个Slave主节点管理下的master分节点共同完成。然后通过Reduce函数整合各个节点的中间结果，从而得到最终计算结果。

MapReduce Jar

是Hadoop执行数据处理的一种作业输入类型。Hadoop由JAVA程序进行编写，因此通常默认的情况下，将作业打包成Jar格式进行处理。

Master

Master是BMR集群中的一种角色。主要是负责管理分布式数据和分解任务的执行，每一个集群中只有一台机器扮演master角色。

MCT 队列（Pipeline）

通过队列，用户可以更灵活地管理转码任务。当用户创建一个任务时，用户必须为该任务指定所属的队列。

MCT 模板（Preset）

模板是系统预设的对于一个视频资源在做转码计算时所需定义的集合。用户可以更简便的将一个模板应用于一个和多个视频的转码任务，以使这些任务输出相同规格的目标视频资源。

MCT 任务（Job）

任务是音视频转码中最基本的执行单元，每个任务将一个原始的音视频资源转码成目标规格的音视频资源。因此，任务和转码的目标是一一对应的，也就是说如果用户需要将一个原始视频规格转换成三种目标规格，比如从AVI格式转码成FLV/MP4/HLS格式，那么用户将会需要创建三个任务。

MCT 缩略图

从原视频中截取一帧或多帧画面，并根据用户指定的大小和伸缩策略生成图片，满足视频预览、特殊帧提取等需求。

MCT 音视频转码

音视频转码MCT（Multimedia Cloud Transcoder），结合百度智能云平台，为开发者和企业用户提供包括音视频存储、转码、CDN加速、多端安全播放器（Web/Android/iOS）、DRM（Digital Rights Management）数字版权管理等服务。

Memcache

是一款开源的高性能分布式缓存系统。其官方定义如下：Free & open source, high-performance, distributed memory object caching system.

Modbus通讯协议

Modbus协议是应用于电子控制器上的一种通用语言。通过此协议，控制器相互之间、控制器经由网络（例如以太网）和其它设备之间可以通信。通过Modbus，不同厂商生产的控制设备可以连成工业网络，进行集中监控。

Modbus协议模式

"物接入按照Modbus协议支持3种模式，TCP, RTU, ASCII。目前物解析支持TCP和RTU两种模式。

- TCP模式：Modbus主站和从站之间通过TCP协议通信。

- RTU模式：Modbus主站和从站之间通过串口连接，如RS232, RS485。数据包里面数据是二进制格式，没有经过任何编码，传输效率最高。
- ASCII模式：Modbus主站和从站之间通过串口连接，如RS232, RS485。数据包里面数据是将二进制数据进行ASCII编码，每个byte编码成2个字符，如byte 0x4f会编码成'4f' 2个字符进行传输。传输效率相对RTU偏低，好处是可以在传输线路上监控可读的数据。"

MongoDB

百度智能云MongoDB产品，100%兼容业界MongoDB协议，为云上的金融、游戏、电商等客户提供了高可靠、高弹性、免运维的云上文档数据库服务。

MQTT

MQTT是基于二进制消息的发布/订阅（Publish/Subscribe）模式的协议，最早由IBM提出的，如今已经成为OASIS规范，更符合M2M大规模沟通。

MIME 多功能Internet 邮件扩充

MIME是一种多用途网际邮件扩充协议。它扩展了基本的面向文本的Internet邮件系统，在不改动现有邮件协议的情况下，实现了用标准的文本格式邮件传输非文本（二进制）数据的功能。服务器将邮件中多媒体数据的MIME类型告诉浏览器，从而让浏览器使用相应的插件读取文件。

媒资

VOD 主要管理的视频对象称为媒资。

密钥管理服务KMS

密钥管理服务（Key Management Service，简称：KMS）是百度智能云提供的一款密钥管理服务，您可以通过该项服务便捷、安全、可靠的在云上管理密钥类信息。用户可以从繁杂的密钥设备管理和安全机制实现中解脱出来，只需要专注于业务上层的加解密功能场景。

名字空间（Scope）

名字空间（Scope）是指监控项的可见范围。不同Scope的监控项彼此完全隔离，从而不会发生冲突。通常将不同的应用程序的监控项放在不同的Scope中。在任何使用监控项的地方，都必须增加Scope的指定。

模板ID

即templateID，在调用API或SDK进行短信下发时，templateID是必需的请求参数。也就是说，用户在用此模板进行短信发送时，必须要用到相应的模板ID这个参数。

N

NAT网关

NAT（Network Address Translation）网关为私有网络提供访问Internet服务，支持多台云服务器共享公网IP资源访问Internet。NAT网关可以绑定EIP实例及共享带宽包，为云服务器实现从内网IP到公网IP的多对一或多对多的地址转换服务。

O

Object

在BOS中，用户操作的基本数据单元是Object。每个Object包含Key、Meta和Data。其中，Key是Object的名字；Meta是用户对该Object的描述，由一系列Name-Value对组成；Data是Object的数据。

Object Relational Mapping 对象关系映射

通过使用描述对象与数据库之间的映射关系，将程序中的对象自动持久化到数据库中。

operation

对topic的操作权限。目前基于MQTT协议，IoT Hub 支持创建发布PUBLISH和订阅SUBSCRIBE两种权限。

P

paddlepaddle

百度深度学习框架，支持机器视觉、自然语言处理、推荐系统等先进算法。Paddle(Parallel Distributed Deep Learning，并行分布式深度学习)， 具有易用性，灵活性，高效性，扩展性等特点，详见官网。

permission

为每一个policy设置一组权限permission，其中包括主题topic，和对该主题的操作权限operation。

Pig

是在Hadoop顶部运行的一种开源分析软件包。由类似 SQL 的语言 Pig Latin 操作，允许用户构建、汇总和查询数据。 与 SQL 操作类似，Pig Latin 也支持一流的 map/reduce 函数及复杂的由用户定义的可扩展数据类型。该功能能够处理复杂的非结构化的数据源，如文本文档和日志文件。另外Pig 支持通过 Java 中编写的用户定义函数使用用户扩展名。在BMR中支持pig 0.11版本。

policy

为身份principal设置对应的策略policy，一个principal对应一个policy。

Polling

Polling是一种CPU决策提供周边设备服务的方式，又称“程控输出入”（Programmed I/O）。具体是指由CPU定时发出询问，依序询问每一个周边设备是否需要其服务，有即给予服务，服务结束后再问下一个周边，接着不断周而复始。

Portal

系统的应用界面

principal

principal是一个抽象概念，表示设备(thing)的身份。每个thing可以绑定一个principal，每个principal拥有一个policy权限。

Q

QPS
Query Per Second，每秒广告请求数

签名认证（Signature）

IAM认证的另外一种方式是签名认证，这种方式是使用AKSK来进行API签名的方式。具体的流程是用户使用SK和API的接口信息生成一个签名串，此签名串中含有AK的信息。百度智能云收到请求之后相应的进行签名串的校验，校验的方式就是重放用户签名的过程，然后判断签名串是否有效。可以理解是一种对称加密的方式。默认接入百度智能云的认证需要使用签名认证方式。

全量备份

在某一时间点对所有数据所做的备份。

全维度分析

视频内容分析VCA 为用户预设了丰富的系统模板，满足用户在语音、文字、人脸、物体、场景识别等多个维度进行视频分析，对于希望使用全维度分析能力的用户来说，是最佳的选择。

R

Redis

是一款遵守BSD协议的，支持高级的键值对操作的，开源缓存与存储系统。其官方定义如下：

Redis is an open source, BSD licensed, advanced key-value cache and store.

Referer白名单

Referer白名单即准入列表，基于HTTP header referer字段的防盗链方法，目的是防止用户存储在BOS上的数据被其他人盗链。用户可以通过BOS控制台设置Referer字段的白名单。设置白名单后，只有Referer字段在白名单内的用户才可以访问Bucket中存储的数据，不在白名单内的请求会被拒绝。但如果用户的Referer为空，默认可以访问，不受白名单限制。

Region

Region代表着一个独立的地域，是百度智能云中的重要概念。百度智能云中的服务除了极少数如账号服务全局有效之外，绝大部分服务都是区域间隔离的。每个区域的服务独立部署互不影响。服务间共享数据需要通过显式拷贝完成。在API中引用区域必须使用其ID。

Relational Database Service 关系型数据库服务

RDS是专业的、高性能、高可靠的云数据库服务。用户可以使用WEB界面轻松的配置、操作数据库实例。RDS还为用户提供可靠的数据备份和恢复、完备的安全管理、完善的监控、轻松扩展等功能支持。相对于用户自建数据库，RDS具有更经济、更专业、更高效、更可靠、简单易用等特点，使您能更专注于核心业务。

Representational State Transfer 表述性状态转移

REST是Roy Fielding博士在2000年他的博士论文中提出来的一种软件架构风格。 目前在三种主流的Web服务实现方案中，因为REST模式与复杂的SOAP和XML-RPC相比更加简洁，越来越多的web服务开始采用REST风格设计和实现。例如，Amazon.com提供接近REST风格的Web服务进行图书查找；雅虎提供的Web服务也是REST风格的。 需要注意的是，REST是设计风格而不是标准。REST通常基于使用HTTP、URI、和XML以及HTML这些现有的广泛流行的协议和标准。

ROI

Return On Investment，投资回报率

Row

表格的行，用于记录用户输入的数据信息，每行包含一个主键及若干属性

Rowkey

主键，是表格中的一个特殊列，它的值可以唯一标识表格的每一行。

RTB

Real-Time Bidding，实时竞价

热备份

在数据库正常工作状态下的备份。

S

SDK

Software Development Kits 软件开发工具包。软件开发工具包SDK一般都是一些被百度智能云软件工程师用于为特定的软件包、软件实例、软件框架、硬件平台、操作系统、文档包等建立应用软件的开发工具的集合。

Sender Policy Framework 发送方策略框架

SPF是一种以IP地址认证电子邮件发件人身份的技术，是非常高效的垃圾邮件解决方案。接收邮件方会首先检查域名的SPF记录，来确定发件人的IP地址是否被包含在SPF记录里面，如果在，就认为是一封正确的邮件，否则会认为是一封伪造的邮件进行退回。

Simple Cache Service 简单缓存服务

简单缓存服务SCS (Simple Cache Service) 是高性能、高可用的分布式内存缓存服务。

Simple Email Service 简单邮件服务

SES为用户提供便捷可靠的邮件批量发送服务。它对邮件内容的垃圾过滤和病毒检测、多种统计信息查询、多种发送方认证、多种访问认证和授权、多种邮件防伪识别标准、多种邮件发送控制策略、邮件发送模拟测试提供支持。

Simple Mail Transfer Protocol 简单邮件传输协议

SMTP是电子邮件从客户机传输到服务器或从某一个服务器传输到另一个服务器使用的传输协议，属于TCP/IP协议族。它帮助每台计算机在发送或中转信件时找到下一个目的地。它是用于在互联网上发送电子邮件的主要协议，几乎所有的互联网电子邮件都是由基于SMTP的客户端和服务器程序发送和接收的。

Simple Message Service 简单消息服务

是构建在可靠云基础设施之上，便捷高效、稳定可靠的短消息下行服务，帮助用户完成短消息的及时下发，而用户只需按使用量付费。

Spark

是一个开源数据分析的集群计算框架，以HDFS（Hadoop Distributed File System）为基础进行搭建。Spark提供了内存集群计算，允许用户程序从集群内存中加载数据，使得内存可以最大化的利用，缓存中间结果。因此，对于某些应用程序，相比于Hadoop MapReduce，处理作业的速度将提升100倍以上。从而Spark不仅提升了集群资源的使用率，还增加作业的时效性。

SSH

SSH是较可靠，专为远程登录会话和其他网络服务提供安全性的协议，支持用户可以通过ssh登入master节点对集群进行操作，并通过配置ssh代理访问各服务监控页面。

SSP

Supply Side Platform，供给方平台

STS（security token service）

是百度智能云的授信用户提供临时访问凭证的一种云服务。通过STS，可以为调用方颁发一个自定义时效和访问权限的临时访问凭证。

伸缩组

一组有弹性伸缩进行自动缩放的BCC被称为伸缩组，您对弹性伸缩的使用通常是从创建伸缩组开始的。

伸缩组配置

伸缩组配置用于定义伸缩组的基本属性，包括所在区域、网络、可伸缩的节点数范围等。伸缩组配置始终

约束伸缩组中所有的节点，因此对伸缩组配置的更改会对所有已经存在和未来将会加入的节点产生影响。1个伸缩组对应1个伸缩组配置。

审核维度

视频内容审核服务提取审核对象中的多维度物料以进行审核，包括图像、音频、语音、图像文字。

时间戳（timestamp）

数据产生的时间点。

时间序列

“1个metric +1个field + n个tag(n>=1)”定义了一个时间序列。如 metric为wind，field为speed，tag为“型号=ABC123”、“出厂编号=1234567890"的数据点都属于同一个时间序列。

实例备份

对RDS实例进行的备份，即在热备节点上产生一份数据快照并上传BOS。

手动备份

RDS支持用户手动对数据库实例进行备份。

数据点（data point）

“1个metric + 1个field(可选) + 1个timestamp + n个tag(n>=1)”唯一定义了一个数据点。当写入的metric、field、timestamp、n个tag都相同时，后写入的value会覆盖先写入的value。

数据库（database）

一个用户可以有多个数据库，一个数据库可以写入多个“度量”的“数据点”。

数据库存储

用于持久化保存数据库数据和日志的底层存储资源。

数据库迁移

随着业务的变化，数据库也需要随着业务从一个环境迁移到另一个环境。如从本地IDC迁移到云上，或者从一个云迁移到另一个云。

数据库实例

数据库实例是运行在百度智能云中的数据库环境，数据库实例中可以包含多个用户创建的数据库。

数据库引擎

数据库引擎是用于存储、处理和保护数据的核心服务。利用数据库引擎可控制访问权限并快速处理事务，从而满足大多数需要处理大量数据的应用程序的要求。RDS目前支持MySQL、SQL Server、Postgresql数据库。

数据库帐号

在使用数据库前，需要在RDS实例中创建帐号。数据库帐号由小写字母、数字、下划线组成，字母开头、字母或数字结尾，最长16个字符。

数据同步

两个数据源之间数据的实时同步，数据同步适用于本地灾备或异地灾备等场景。

数值（value）

度量对应的数值，如56°C、1000r/s等（实际中不带单位）。如果有多个field，每个field都有相应的value。不同的field支持不同的数据类型写入。对于同一个field，如果写入了某个数据类型的value之后，相同的field不允许写入其他数据类型。

私网IP

私网IP是专门给一些局域网内用的。也就是说在网络上是不唯一的，公网上是不能通这个私有IP来找到对应的设备的。

T

Table

表格，在Instance中创建的数据单位。一个Instance可以包含多张表格，每张表格包含行和列。表格的名字在Instance内唯一，不同Instance内可创建同名表。

Tez

是一个运行在YARN之上支持DAG作业的计算框架。具体请参考： http://tez.apache.org/

thing

表示IoT Hub 设备，用户可以在每个endpoint中创建一个或多个thing。

topic

每一个policy都需要指定一个主题项目（topic），在进行使用IoT Hub 服务之前，需要先为我们即将开展的订阅发布信息创建一个主题名称，该主题应用于MQTT客户端。topic规则允许字符串可以带一个通配符"#"，例如"temperature/#"就是匹配前缀是temperature的所有topic；单独的"#"表示匹配所有topic。

Transmission Control Protocol 传输控制协议

传输控制协议TCP是一种面向连接（连接导向）的、可靠的、基于IP的第4层传输层协议，由IETF的RFC 793说明（specified）。TCP在IP报文的协议号是6。

TSDB

Time Series Database，时序数据库，用于保存时间序列（按时间顺序变化）的海量数据。

弹性伸缩规则

弹性伸缩规则用于规定伸缩组将在何时进行扩容或者缩容，以及扩缩容的节点数量。

天工物联卡

百度智能云天工和联通合作推出的物联网卡，物联网卡专用13位号段，支持定向发到天工物接入。

图片服务

对媒资封面图（cover）进行缩放、裁剪、格式转换、旋转、添加水印等实时处理并将处理结果通过CDN实时返回给用户的服务称为图片服务。

推流认证(Push)

为确保推流频道不被非法占用（即防止非法推流），音视频直播LSS支持采用带token认证的可设置有效期的推流地址。支持用户使用推流认证功能，以便保护音视频直播发布安全。开启推流认证后，直播推流地址可设置有效期，有效期过后该地址将失效，再次推流需重新获取推流地址。关闭推流认证，直播推流地址固定不变。

V

VCA 模板

模板是 VCA 对每个视频做分析计算时所需定义的集合。一个模板能够应用于一个和多个视频分析任务，确保视频分析结果的维度相同。

Virtual Network Computer 虚拟网络计算机

VNC远程控制能力强大，高效实用。百度VNC远程登录软件支持RDP远程桌面、SSH客户端、Windows、Linux操作系统的远程登录。

VPC

私有网络 VPC(Virtual private Cloud) 是一个用户能够自定义的虚拟网络，能够帮助用户构建属于自己的网络环境。通过指定IP地址范围和子网等配置，即可快速创建一个VPC，不同的 VPC 之间完全隔离，用户可以在VPC内创建和管理BCC实例。

W

WAF

Web Application Firewall Web应用防火墙是通过执行一系列针对HTTP/HTTPS的安全策略来专门为Web应用提供保护的一款产品。

Web攻击

针对Web应用发起的攻击，包括但不限于以下攻击类型：SQL注入、XSS跨站、Webshell上传、命令注入、非法HTTP协议请求、非授权文件访问等。

Web漏洞

Web漏洞通常是指网站程序上的漏洞，可能是由于代码编写者在编写代码时考虑不周全等原因而造成的漏洞，常见的WEB漏洞有SQL注入、CSRF、XSS漏洞、上传漏洞、任意文件读取漏洞、命令执行、文件包含等。

外网专线

单独建设一条专用的物理线路，将用户的数据中心接入百度智能云，实现用户独占一条物理线路，该线路不与其他用户共享。

网关设备

网关设备是连接企业内部设备（比如PC、PLC、SCADA、DCS）与企业外网云服务的桥梁，它接受设备轮询配置管理服务下发的配置、执行设备轮询配置的读取策略、并在收到设备返回的数据后，将数据上传回云端。

网关子设备

指某网关设备所在网络内与之连接的的工业设备，工业设备采集的数据将经该网关上传至云端。

维度（Dimension）

维度（Dimension）是用来描述监控项特征的多个键值对（Key/Value，简称K/V对）。维度不同，则意味着监控项是不同的。例如，对于虚拟机的监控项CPUUsagePercent来说，一个典型的维度就是机器ID（InstanceId），同时，描述这个监控项的维度还有虚拟机的镜像ID（ImageID）。

文档存储

DOC存储源文档文件、转码后的文件，以备在线浏览使用，称为文档存储。

文档分发

将文档内容分发至全国所有的节点，缩短用户查看内容的延迟，提高用户访问网站的响应速度与网站的可用性，解决网络带宽小、用户访问量大、网点分布不均等问题称为文档分发。

文档转码

DOC将源文档文件转换成适合PC、WAP、APP等多终端环境在线浏览的HTML5文件的计算服务称为文档转码。

文件存储（elastic file system, EFS）

文件存储是云存储的一种，为云主机、Docker容器等提供标准文件访问接口（NFS、CIFS）的云存储服务，具备无限容量、高性能、多共享、高可用等特性。

文件系统（file system）

文件系统是操作系统用来组织和管理存储于物理介质上数据的一种手段。

物理备份

将实际组成数据库的原始文件从一处拷贝到另一处的备份过程。

物联网卡

运营商基于物联网公共服务网络，面向物联网用户提供的移动通信接入业务。

物模型

物模型由一个或多个属性构成，您可以用他来表示一类设备。有了物模型的定义，可以更方便的基于模型的接口来处理物影子数据，以及反控。也可以通过直接配置模型的属性实现上传的数据转发至TSDB。不需要写复杂的SQL语句。

物影子

物影子反映物理世界中的一个物，是物在云端的『影子』或『数字双胞胎』。运行时，物将监控值上报给物影子，物影子会用一个json文档给这个物做一个状态暂存。天工其他产品或您的应用程序可以直接通过MQTT或HTTP访问。同时，物影子也提供反控功能。

X

下行短信

通信服务提供商发送给目标用户的短信，一般情况下目标用户免费接收。在SMS中，下行短信是指SMS发出的短信。

消息通知

通过选择消息通知方式，可以使用户随时了解任务的状态。不论处理成功还是失败，在处理任务结束时均会触发消息通知。

信封加密

Envelope Encryption，为敏感内容进行“一次一密钥”的加解密技术方案。随机生成的密钥加密后和内容一起保存，解密时先还原密钥，再对内容进行解密。

虚拟专用网络（VPN）

通过VPN服务，将百度智能云与客户的多个数据中心快速、灵活搭建VPN隧道，实现混合云。百度智能云VPN网关，基于主备模式的高可靠架构实现，支持VPN健康性自动检测、故障自动恢复等功能。

Y

音视频存储

VOD 存储音视频转码前的源文件与转码后的目标文件以备点播服务调用，称为音视频存储。

音视频分发

将音视频源站内容分发至全国所有的节点，缩短用户查看内容的延迟，提高用户访问网站的响应速度与网站的可用性，解决网络带宽小、用户访问量大、网点分布不均等问题称为音视频分发。

音视频媒资

开发者上传一个媒体文件后，VOD 对媒体文件进行转码、生成缩略图、获取元信息等处理后形成的资源集合称为音视频媒资。

音视频转码

VOD 将源音视频文件转换成各种终端设备所需的文件格式以满足手机、平板、智能电视和 PC 等多终端播放需求的计算服务称为转码。

硬件密码设备

Hardware Security Machine，符合国家主管部门要求的硬件密码管理设备。

用户（User）

用户是账户系统下的概念，账户和用户是1对n的关系。用户是资源的操作者，但是不是资源的拥有者。百度智能云系统中存在全局唯一的user_id来标示一个用户，并且user_name在一个账户下是唯一的。

用户密钥（Credential）

用户密钥是和用户关联的安全信息，这里主要指用户的AKSK信息。用户在第一次创建的时候，系统会默认给用户创建一组AKSK，用于签名认证。针对临时授权的场景，还会有一种Security Token也是用户密钥的一种。

用户主密钥

Customer Master Key，用户在百度智能云密钥管理服务中创建的主密钥。

邮件验证

邮件验证（Email Authentication）的目的是确认一封邮件的有效身份，它是反垃圾邮件、反伪造、反欺骗中的一个重要步骤。由于SMTP协议中没有涉及发件人确认，所以邮件验证非常必要。邮件认证有SPF、DKIM、ADSP、DMARC、VBR、iprev等方式。

邮箱地址认证

邮箱地址认证是一种通过向待认证邮箱发送激活邮件来确保邮箱的有效性的方式。用户响应链接认证请求，验证认证信息的合法性，完成认证。

邮箱域名认证

邮箱域名认证是一种通过令牌来认证邮件域有效性的方式。IAM生成base64编码的令牌后，用户通过将令牌加入邮件域的DNS记录来认证整个域的有效性。

域（field）

在指定度量下数据的子类别。即一个metric支持多个field，如metric为wind，该metric可以有两个field：direction和speed。

域名

Domain Name，由一串用点分隔的名字组成的Internet上某一台计算机或计算机组的名称，用于在数据传输

时标识计算机的电子方位（有时也指地理位置，地理上的域名，指代有行政自主权的一个地方区域）。域名是一个IP地址上有“面具”。一个域名的目的是便于记忆和沟通的一组服务器的地址（网站，电子邮件，FTP等）。

域名解析

域名解析是把域名指向网站空间IP，让人们通过注册的域名可以方便地访问到网站一种服务。IP地址是网络上标识站点的数字地址，为了方便记忆，采用域名来代替IP地址标识站点地址。域名解析就是域名到IP地址的转换过程。域名的解析工作由DNS服务器完成。

域名注册

域名注册是Internet中用于解决地址对应问题的一种方法。域名注册遵循先申请先注册原则，管理机构对申请人提出的域名是否违反了第三方的权利不进行任何实质审查。每个域名都是独一无二且不可重复。

Z

增量备份

自从任意类型的上一次备份后，对所有修改部分所做的备份。

站点监控

站点监控是指从外部对用户的站点进行可用性监控，目前支持HTTP、HTTPS、PING和TCP监控。

帐户（Account）

在百度智能云console上注册时创建，account对应了account id，此id百度智能云唯一。帐户是百度智能云资源的拥有者，是最小的计费主体。在创建account的时候，系统同时会默认创建root用户（超级管理员），之后使用帐户的用户名密码登陆实际的操作身份是root用户。创建帐户时同时会为其关联一个相同id的domain（openstack概念），因此在百度智能云系统中root_user_id = account_id = domain_id。

只读实例

为扩容RDS主实例读请求能力，支持用户只读请求、与主实例具有同步关系的RDS实例。只读实例从属于主实例，二者是主从关系。创建只读实例前必须先创建好主实例，主实例释放时，其从属的只读实例会自动释放。

直播拉流

拉流是指通过LSS 云端到用户指定的源站拉取直播流的过程。LSS 支持整站拉流，单个直播间拉流两种类型。

直播推流

直播推流，是指将直播场景实况信息实时编码压缩后，推送到 LSS 云端的过程。

置信度

视频内容审核服务会在审核结果中输出置信度(Confidence Level)。若置信度越高，说明审核结果越负面；同理，若置信度越低，说明审核结果越正面。VCR 中置信度的取值范围为0~100浮点数，保留2位小数。

主节点

主实例中用来给用户提供读写请求服务的节点，主实例故障时会自动切换到备节点，从而保证RDS的高可用。

主实例

支持用户读写请求的具有主备架构的RDS实例。

专属服务器

即用户专属的物理服务器资源，自用户确认购买行为后，服务器即标记为此用户的专属服务器。专属服务器默认部署了虚拟化程序，用户需要将专属服务器划分为专属实例后，才可以真正开始使用。

专属实例

即用户在专属服务器实例上使用虚拟化系统创建的虚拟机。用户可以在专属服务器资源允许的情况下自由创建专属实例，一个专属实例相当于一台可独立使用的BCC实例。

资源（resource）

资源是云服务呈现给用户与之交互的对象实体的一种抽象，如BCC实例，BOS存储桶。

资源管控操作

指云资源生命周期管理及运维管理操作，比如BCC实例的创建、停止、重启，BOS的bucket的创建、修改、删除等。

资源使用操作

是指使用资源的核心功能，比如BOS bucket的数据上传/下载。

子网（Subnet）

子网是 VPC 内的用户可定义的IP地址范围，根据业务需求，通过CIDR(无类域间路由)可以指定不同的地址空间和IP段。未来用户可以将子网作为一个单位，用来定义Internet访问权限、路由规则和安全策略。

自定义监控

自定义监控是指用户可以将自己产生的数据推送到BCM中并对其进行监控。

自定义人脸库

公众人物，是指在一定范围内拥有特殊社会地位，具有重要影响，为人们所广泛知晓和关注，且能因此从社会中得到巨大利益，并与社会公共利益密切相关的人物。其社会知名度、社会地位、是否因此得到巨大的利益，和较长时间内影响社会公共利益等四个共众人物要件，缺一不可，共同体现了公众人物的特性。

自定义维度分析

百度智能云为需要单维度或几个维度的分析视频的用户，提供了可定制化的转码模板，以帮助他们满足复杂业务条件下的个性化需求。

自动备份

RDS对备份集每天自动备份一次，RDS支持通过设置自定义备份周期全面满足用户不同的备份需求，从而减轻用户自运维的负担。

32960协议

GB/T 32960协议规定了电动汽车远程服务与管理系统中协议结构、通信连接、数据包结构与定义、数据单元格式与定义。该协议适用于电动汽车远程服务与管理系统中平台间的通信，车载终端至平台的传输宜参照执行。

808协议

JT T808协议规定了道路运输车辆卫星定位系统北斗兼容车载终端与监管/监控平台之间的通讯协议与数据格式，包括协议基础、通信连接、消息处理、协议分类与说明及数据格式。

区域选择说明

区域

区域(Region)是指分布在全球范围内的各个物理节点，您可以根据客户群体分布的地理位置，选择在不同区域部署BCC。百度智能云目前提供服务区域为： **华北区（北京）** 、**华南区（广州）**、**华东区（苏州）**、**金融华东（上海）**、**金融华中（武汉）**、**香港**。

国内华北区、华南区、华东区、金融华中、金融华东数据中心提供多线BGP骨干网线路，网络能力覆盖全国各省市，实现稳定高速访问。

请关注以下详情，它们可能与您所选择的服务有关：

1. 处在同一区域的云服务产品之间通过内网互通。
2. 当前处在不同区域的云服务产品之间内网不能互通。

产品服务	区域说明	举例
云服务器BCC	- 不同区域之间的云服务器BCC内网默认不互通。 - 不同区域之间的云服务器BCC与BLB、EIP、CDS等云服务内网默认不互通。	同在华北区的BCC、CDS内网互通；但华北区的BCC与华南区的CDS内网不互通。
容器引擎CCE	- 不同区域之间的容器集群内网默认不互通。 - 不同区域之间的容器集群与BLB内网默认不互通。	同在华北区的容器集群、BLB内网互通；但华北区的集群与华南区的BLB内网不互通。
负载均衡BLB	不支持跨区域部署	BLB绑定BCC时，只能选择绑定本区域的云服务器BCC。
关系型数据库RDS	- 不同区域之间的RDS内网默认不互通。 - 不同区域之间的云服务器BCC与RDS内网不互通。	华北区的BCC和华南区的RDS内网不互通。
简单缓存服务SCS	不同区域之间的BCC与SCS内网不互通。	华北区的BCC和华南区的SCS内网不互通。
对象存储BOS	- 同一区域之间的云服务器BCC和对象存储BOS内网互通。 - 不同区域之间的BCC与BOS内网不互通。	同在华北区的BCC和BOS内网互通，但华北区的BCC和华南区的BOS内网不互通。

3. 购买百度智能云服务时建议您选择和您目标用户所在区域最为接近的数据中心，可以降低访问时延，提升用户访问速度。
4. 华南区产品服务等级协议SLA保障与华北区保持一致，后续如有差异，会在官网通知。
5. 有关每个服务所支持的区域信息请参阅下面的表格，其中全局表示该服务不需要进行区域选择。：
    - 计算相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
云服务器 BCC								
专属服务器 DCC								
物理服务器 BBC								
容器引擎 CCE								
弹性伸缩 AutoScaling								
应用引擎专业版 BAEPRO								
函数计算 CFC								

- 网络相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
弹性公网IP EIP								
负载均衡 BLB								
智能云解析								
私有网络 VPC								
专线 ET								

- 存储和CDN相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
对象存储 BOS								
云磁盘 CDS								
文件存储 CFS								
内容分发网络 CDN								
存储网关BSG								

- 数据库相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
云数据库 RDS								
云数据库 SCS								
云数据库 TableStorage								
云数据库 DRDS								
数据传输服务 DTS								
云数据库 DocDB for MongoDB								
云数据库FusionDB								
云数据库 HTAP for CockroachDB								

- 安全和管理相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
DDoS基础防护								
应用防火墙服务 WAF								
业务安全风控 AFD								
云监控 BCM								
云审计 BCT								
SSL证书服务								
主机安全客户端								
安全检测服务 SRD								
流量审计 IDS								
密钥管理服务 KMS								

- 数据分析相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
百度 MapReduce BMR								
百度数据工厂 Pingo								
百度数据科学平台								
百度机器学习 BML								
百度深度学习								
百度数据仓库 Palo								
百度 Elasticsearch								
百度日志服务 BLS								
百度消息服务								

- 网站服务相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
站点服务								
域名服务 BCD								
云虚拟主机 BCH								
智能流量管理 ITM								
移动解析 HTTPDNS								

- 智能多媒体服务相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
音视频转码 MCT								
音视频直播 LSS								
音视频点播 VOD								
文档服务 DOC								
视频内容审核 VCR								
视频内容分析 VCA								

- 物联网服务相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
物接入 IoT Hub								
物解析 IoT Parser								
时序数据库 TSDB								
规则引擎 Rule Engine								
物可视 IoT Visualization								
智能边缘 BIE								
度家 DuHome								
度行智能车辆云 DuGo								
位置服务 DuMap								
物流地图 LPS								

- 人工智能服务相关产品区域信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
AI开发平台 Infinite								
语音技术								
人脸识别								
文字识别								
自然语言处理								
内容审核								
图像识别								
图像搜索								
智能外呼								
人体分析								
机器人平台 ABC Robot								
智能呼叫中心								
AR增强现实								
Apollo训练平台								
EasyDL定制化训练与服务平台								
理解与交互技术UNIT								

- 数字营销云服务相关产品信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
百度信息流推广API								
搜索Referer API								
智能推荐 BRS								
百度舆情API								

- 区块链服务相关产品信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
百度区块链引擎 BBE								

- 应用服务相关产品信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
百度慧推 SPP								
简单邮件服务 SES								
云呼叫中心 CCC								
百度效率云 Devops								
简单消息服务 SMS								
应用性能管理服务 APM								
问卷调研服务								
人工测试								
移动App测试								

- 云市场服务相关产品信息。

产品名	全局	华北-北京	华北-保定	华南-广州	华东-苏州	金融华东-上海	金融华中-武汉	香港
云市场								

常见问题：

1. **如何选择服务区域呢?**

   购买百度智能云服务时建议您选择和您目标用户所在区域最为接近的数据中心，可以降低访问时延，提升用户访问速度。

2. **华南区是否可以备案，与备案系统是否已经打通?**

   可以，备案与区域选择没有直接联系，与业务形态有关。建站类的服务无论部署在哪个区域，都需要备案。

3. **华南区是什么线路?**

   多线BGP骨干网线路，保证全国用户高速访问。

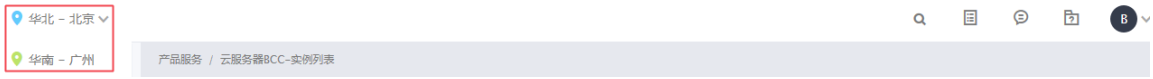
4. **华南区的计费方式及金额是否和北京区相同?**

   同样产品，计费方式是相同的。后续如果受到运营商影响而产生的计费差异，会在官网通知，敬请关

注。

5. **控制台区域如何切换?**

   官网控制台页面导航栏支持选择区域，用户可根据需求自行切换。例如，云服务器BCC支持华北－华南两个区域的切换。



6. **华北云服务器BCC是否可以访问华南的BOS存储文件?**

   可以访问。用户可以跨区域访问Bos bucket文件，会产生额外费用，详细参见BOS计费规则。

可用区

可用区(Availability Zone)是指在同一个区域(Region)下，电力、网络等基础设施相互隔离的一个或多个数据中心。一个区域包含一个或多个可用区，当一个可用区出现故障后，不会影响其他可用区的使用，保护您的应用程序或数据库不受单一位置故障影响。

除此之外，可用区能提供比单个数据中心更强的可用性、容错能力以及可扩展性。请关注以下详情，可能与您选择的可用区有关：

- 同一区域下的不同可用区之间的云服务通过内网可以访问。
- 不同区域下的可用区数目可能不相同。
- 可用区是针对用户层面定义的标识符，不同用户在同一区域下选择相同的可用区，可能对应不同的物理数据中心。

常见问题：

1. **为什么要分可用区?**

   可用区是指在同一区域下，电力和网络互相独立的区域，故障会被隔离在一个可用区内。如果您的应用程序需要更高的高可用性，建议您将云服务创建在不同的可用区内。

2. **如何选择可用区呢?**

   登录百度智能云控制台，选择您所在的区域，购买云服务时选择不同可用区。不同的区域可能拥有不同的可用区数目。

3. **在同一个区域内，可用区A的CDS可以挂载到另一可用区B的BCC实例上吗?**

   不可以，一块云磁盘CDS只能挂载到同一可用区的一台BCC实例上，不可跨可用区挂载。

SDK入门指南

Java-SDK入门指南

前提条件

本文旨在帮助您快速获取百度智能云Java SDK并开始调用。

前提条件

开发前请确保下述前提条件已准备就绪：

- Java SDK工具包可在jdk1.7，jdk1.8环境下运行。
- **AK/SK**：SDK 认证时必须传入 AK/SK 参数，在安全认证页面 获取 Access Key和Secret Key。

  用户可以通过两种方式与百度智能云进行交互，包括认证方式和匿名方式。对于认证方式，需要通过使

用Access Key Id / Secret Access Key加密的方法来验证某个请求的发送者身份。Access Key Id（AK）用于标示用户，Secret Access Key（SK）是用户用于加密认证字符串和百度智能云用来验证认证字符串的密钥，其中SK必须保密，只有用户和百度智能云知道。

安装SDK包

- 方式一：使用Maven的安装

  在Maven的pom.xml中的文件中添加BCE-Java的SDK的依赖：

```
 <dependency>
    <groupId>com.baidubce</groupId>
    <artifactId>bce-java-sdk</artifactId>
    <version>{version}</version>
</dependency>
```

  其中，{version}为版本号，可以在SDK页面下载找到。

- 方式二：直接使用JAR包安装

  在百度智能云官网下载Java SDK压缩工具包。将的bce-java-sdk-version.zip解压后，复制到工程文件夹在Eclipse右键“工程 - >属性 - > Java构建路径 - >添加JAR 添加SDK包工具lib/bce-java-sdk-version.jar 状语从句：第三方依赖工具包third-party/*.ja 其中，version为版本号。

SDK目录结构

```
com.baidubce
        ├── auth                                        //BCE签名相关类
        ├── http                                        //BCE的Http通信相关类
        ├── internal                                    //SDK内部类
        ├── model                                       //BCE公用model类
        ├── services
        │       └── iotdm                               //以物管理服务举例
        │           ├── model                          //物管理内部model，如
Request或Response
        │           └── IotDmV3Client.class              //以物管理服务举例，物管
理客户端入口类
        ├── util                                        //BCE公用工具类
        ├── BceClientConfiguration.class                //对BCE的HttpClient的配置
        ├── BceClientException.class                    //BCE客户端的异常类
        ├── BceServiceException.class                   //与BCE服务端交互后的异常
类
        ├── ErrorCode.class                             //BCE通用的错误码
        └── Region.class                                //BCE提供服务的区域
```

使用Java SDK

调用Java SDK的3个主要步骤：

- 创建并初始化xxxClient，百度智能云服务分别配置了详细的Java SDK 帮助指南，详细请参考产品指南。
- 创建API请求并设置参数。
- 发起请求并处理应答或异常。

PHP-SDK入门指南

前提条件

本文旨在帮助您快速获取百度智能云PHP SDK并开始调用。

前提条件

开发前请确保下述前提条件已准备就绪：

- PHP SDK包要求运行环境至少为PHP 5.3.2 版本。
- **AK/SK**：SDK 认证时必须传入 AK/SK 参数，在安全认证页面 获取 Access Key和Secret Key。

  用户可以通过两种方式与百度智能云进行交互，包括认证方式和匿名方式。对于认证方式，需要通过使用Access Key Id / Secret Access Key加密的方法来验证某个请求的发送者身份。Access Key Id（AK）用于标示用户，Secret Access Key（SK）是用户用于加密认证字符串和百度智能云用来验证认证字符串的密钥，其中SK必须保密，只有用户和百度智能云知道。

安装SDK包

1. 在百度智能云官网下载PHP SDK压缩工具包。
2. 解压ZIP包：

```
BaiduBce.phar            //PHP SDK
BosClientSample.php         //示例
SampleConf.php           //参考配置文件，具体内容见下文
```

> **说明：** 用户在配置好AK/SK和Host之后，可以使用`phpunit --debug BosClientSample.php`来运行Sample。

3. 在脚本文件中添加以下代码，即可以使用SDK包：

```
include ‘BaiduBce.phar’;
require ‘YourConf.php’;
```

关于配置文件的引用请参考下各产品PHP SDK帮助手册中的配置client。

SDK目录结构

```
BaiduBce.phar
├──src
│   └── BaiduBce
│       ├── Auth                //BCE签名相关
│       ├── Exception           //BCE客户端的异常
│       ├── Http                //BCE的Http通信相关
│       ├── Log                 //BCE日志
│       ├── Services
│       │   └── Bos                   //以对象存储BOS为例，BOS主目录必须保留
│       │       ├── BosClient.php     //BOS操作类，所有操作可以通过BosClient类可以完
成
│       │       ├── BosOptions.php    //BOS自定义配置
│       │       └── CannedAcl.php     //CannedAcl模块
│       └── Util                //BCE公用工具
└──vendor                       //第三方库
```

使用PHP SDK

调用PHP SDK的3个主要步骤：

- 创建并初始化xxxClient，百度智能云服务分别配置了详细的PHP SDK 帮助指南，详细请参考产品指南。

- 创建API请求并设置参数。
- 发起请求并处理应答或异常。

Python-SDK帮助指南

前提条件

本文旨在帮助您快速获取百度智能云Python SDK并开始调用。

前提条件

开发前请确保下述前提条件已准备就绪：

- Python SDK工具包支持在Python 2.7环境下运行。
- 验证**AK/SK**：SDK 认证时必须传入 AK/SK 参数，在安全认证页面 获取 Access Key和Secret Key。

  用户可以通过两种方式与百度智能云进行交互，包括认证方式和匿名方式。对于认证方式，需要通过使用Access Key Id / Secret Access Key加密的方法来验证某个请求的发送者身份。Access Key Id（AK）用于标示用户，Secret Access Key（SK）是用户用于加密认证字符串和百度智能云用来验证认证字符串的密钥，其中SK必须保密，只有用户和百度智能云知道。

安装SDK包

在百度智能云官网下载Python SDK压缩工具包。执行下列命令，安装SDK包：

```
python setup.py install
```

SDK目录结构

```
baidubce
     ├── auth                            //公共权限目录
   ├── http                             //Http请求模块
     ├── services                        //服务公共目录
     │     └── bos                      //以对象存储BOS为例，BOS目录
     │           ├──bos_client.py       //BOS客户端入口类
     │           ├──bos_handle.py       //HTTP reponse处理函数
     │           ├──canned_acl.py       //权限控制需要的常量
     │           └──storage_class.py    //存储类型定义模块
     ├── bce_base_client.py              //BCE客户端入口类的基类
     ├── bce_client_configuration.py     //针对BOS特有的HttpClient的配置类
     ├── bce_response.py                 //BCE客户端的请求类
     ├── exception.py                    //BCE客户端的异常类
     ├── protocol.py                     //网络协议定义
     ├── region.py                       //区域处理模块
     ├── retry_policy.py                 //BCE服务公共配置类
     └── utils.py                        //BCE公用工具类
```

使用Python SDK

调用Python SDK的3个主要步骤：

- 创建并初始化xxxClient，百度智能云服务分别配置了详细的Python SDK 帮助指南，详细请参考产品指南。
- 创建API请求并设置参数。
- 发起请求并处理应答或异常。

API SDK

Java-SDK

物接入IoT Hub

Interface	Method	API
Endpoint	iotHubClient.createEndpoint();	POST /v1/endpoint
Endpoint	iotHubClient.listEndpoints();	GET /v1/endpoint
Endpoint	iotHubClient.queryEndpoint();	GET /v1/endpoint/{endpointName}
Endpoint	iotHubClient.deleteEndpoint();	DELETE /v1/endpoint/{endpointName}
Thing	iotHubClient.createThing();	POST /v1/endpoint/{endpointName}/thing
Thing	iotHubClient.listThings();	GET /v1/endpoint/{endpointName}/thing
Thing	iotHubClient.listPrincipals();	GET /v1/endpoint/{endpointName}/thing/{thingN
Thing	iotHubClient.deleteThing();	DELETE /v1/endpoint/{endpointName}/thing/{th
Principal	iotHubClient.createPrincipal();	POST /v1/endpoint/{endpointName}/principal
Principal	iotHubClient.listPrincipals();	GET /v1/endpoint/{endpointName}/principal?thi {thingName}
Principal	iotHubClient.listPrincipals();	GET /v1/endpoint/{endpointName}/principal/{pr
Principal	iotHubClient.attachThingToPrincipal();	POST /v1/action/attach-thing-principal
Principal	iotHubClient.removeThingToPrincipal();	POST /v1/action/remove-thing-principal
Principal	iotHubClient.regeneratePassword();	POST /v1/endpoint/{endpointName}/principal/{p
Principal	iotHubClient.deletePrincipal();	DELETE /v1/endpoint/{endpointName}/principal
Policy	iotHubClient.createPolicy();	POST /v1/endpoint/{endpointName}/policy
Policy	iotHubClient.listPolicy();	GET /v1/endpoint/{endpointName}/policy?princ {principalName}
Policy	iotHubClient.queryPolicy();	GET /v1/endpoint/{endpointName}/policy/{polic
Policy	iotHubClient.attachPrincipalToPolicy();	POST /v1/action/attach-principal-policy
Policy	iotHubClient.removePrincipalToPolicy();	POST /v1/action/remove-thing-principal-policy
Policy	iotHubClient.deletePolicy();	DELETE /v1/endpoint/{endpointName}/policy/{p
Permission	iotHubClient.updatePermission();	PUT /v1/endpoint/{endpointName}/permission/{
Permission	iotHubClient.listPermission();	GET /v1/endpoint/{endpointName}/permission? {policyName}
Permission	iotHubClient.queryPermission()	GET /v1/endpoint/{endpointName}/permission/{
Certificate	iotHubClient.createPrincipalWithCert();	POST /v1/endpoint/:endpointName/principal?w
Certificate	iotHubClient.renewCert();	POST /v1/endpoint/:endpointName/principal/:principa certificate

内容分发网络CDN

Class	Method	API
CdnClient	createDomain(CreateDomainRequest request)	PUT /v2/domain/{domain}
CdnClient	deleteDomain(DeleteDomainRequest request)	DELETE /v2/domain/{domain}
CdnClient	deleteDomain(String domain)	DELETE /v2/domain/{domain}
CdnClient	describeIp(DescribeIpRequest request)	GET /v2/utils
CdnClient	describeIp(String ip)	GET /v2/utils
CdnClient	disableDomain(DisableDomainRequest request)	POST /v2/domain/{domain}?disable
CdnClient	disableDomain(String domain)	POST /v2/domain/{domain}?disable
CdnClient	enableDomain(EnableDomainRequest request)	POST /v2/domain/{domain}?enable
CdnClient	enableDomain(String domain)	POST /v2/domain/{domain}?enable
CdnClient	getCacheQuota()	GET /v2/cache/quota
CdnClient	getCacheQuota(GetCacheQuotaRequest request)	GET /v2/cache/quota
CdnClient	getDomainCacheTTL(GetDomainCacheTTLRequest request)	GET /v2/domain/{domain}/config?cacheTTL
CdnClient	getDomainCacheTTL(String domain)	GET /v2/domain/{domain}/config?cacheTTL
CdnClient	getDomainConfig(GetDomainConfigRequest request)	GET /v2/domain/{domain}/config
CdnClient	getDomainConfig(String domain)	GET /v2/domain/{domain}/config
CdnClient	getDomainLog(GetDomainLogRequest request)	GET /v2/log/{domain}/log
CdnClient	getPrefetchStatus(GetPrefetchStatusRequest request)	GET /v2/cache/prefetch
CdnClient	getPurgeStatus(GetPurgeStatusRequest request)	GET /v2/cache/purge
CdnClient	getStatAvgSpeed(GetStatAvgSpeedRequest request)	GET /v2/stat/avgspeed
CdnClient	getStatFlow(GetStatFlowRequest request)	GET /v2/stat/flow
CdnClient	getStatHitRate(GetStatHitRateRequest request)	GET /v2/stat/flow
CdnClient	getStatHttpCode(GetStatHttpCodeRequest request)	GET /v2/stat/httpcode
CdnClient	getStatPv(GetStatPvRequest request)	GET /v2/stat/pv

Class	Method	API
CdnClient	getStatSrcFlow(GetStatSrcFlowRequest request)	GET /v2/stat/srcflow
CdnClient	getStatTopReferer(GetStatTopRefererRequest request)	GET /v2/stat/topn/referer
CdnClient	getStatTopUrl(GetStatTopUrlRequest request)	GET /v2/stat/topn/url
CdnClient	getStatUv(GetStatUvRequest request)	GET /v2/stat/uv
CdnClient	listDomains()	GET /v2/domain
CdnClient	listDomains(ListDomainsRequest request)	GET /v2/domain
CdnClient	prefetch(PrefetchRequest request)	POST /v2/cache/prefetch
CdnClient	prefetch(String url)	POST /v2/cache/prefetch
CdnClient	purge(PurgeRequest request)	POST /v2/cache/purge
CdnClient	purge(String url)	POST /v2/cache/purge
CdnClient	purgeDirectory(String directory)	POST /v2/cache/purge
CdnClient	setDomainCacheFullUrl(SetDomainCacheFullUrlRequest request)	/v2/domain/{domain}/config?cacheFullUrl
CdnClient	setDomainCacheFullUrl(String domain, boolean setting)	/v2/domain/{domain}/config?cacheFullUrl
CdnClient	setDomainCacheTTL(SetDomainCacheTTLRequest request)	PUT /v2/domain/{domain}/config?cacheTTL
CdnClient	setDomainIpACL(SetDomainIpACLRequest request)	PUT /v2/domain/{domain}/config?ipACL
CdnClient	setDomainLimitRate(SetDomainLimitRateRequest request)	PUT /v2/domain/{domain}/config?limitRate
CdnClient	setDomainLimitRate(String domain, int limitRate)	PUT /v2/domain/{domain}/config?limitRate
CdnClient	setDomainOrigin(SetDomainOriginRequest request)	PUT /v2/domain/{domain}/config?origin
CdnClient	setDomainOrigin(String domain, String peer)	PUT /v2/domain/{domain}/config?origin
CdnClient	setDomainRefererACL(SetDomainRefererACLRequest request)	PUT /v2/domain/{domain}/config?refererACL

Class	Method	API
CdnClient	setHttpsConfig(SetHttpsConfigRequest request)	PUT /v2/domain/{domain}/config?https
CdnClient	setHttpsConfig(String domain, HttpsConfig https)	PUT /v2/domain/{domain}/config?https
CdnClient	setRequestAuth(SetRequestAuthRequest request)	PUT /v2/domain/{domain}/config?requestAuth
CdnClient	setRequestAuth(String domain, RequestAuth requestAuth)	PUT /v2/domain/{domain}/config?requestAuth

功能发布记录

发布时间	功能概述
2019-08	证书管理提供API接口及Java SDK，方便开发人员管理、创建以及查看SSL证书
2019-04	签名认证增加V2版本。V2认证字符串是百度智能云最新的API认证协议方式。基于V2协议，服务可提供更高的请求响应性能，用户也可更大程度保证自己的密钥安全
2019-03	增加生成签名认证错误排查步骤以及常见错误case，降低鉴权认证过程中的签名错误
2018-11	新增单元管理功能。企业组织创建者可以通过创建单元对组织成员进行分组管理，每个单元下可以包含多个账户，创建者可以给每个单元添加不同的策略，单元下的账户自动继承该策略