访问控制

STS API文档

阿里云
aliyun.com

STS API文档

STS介绍

阿里云STS (Security Token Service) 是为阿里云账号（或RAM用户）提供短期访问权限管理的云服务。通过STS，您可以为联盟用户（您的本地账号系统所管理的用户）颁发一个自定义时效和访问权限的访问凭证。联盟用户可以使用STS短期访问凭证直接调用阿里云服务API，或登录阿里云管理控制台操作被授权访问的资源。

访问点

STS的默认访问点地址是: https://sts.aliyuncs.com ，用户必须使用https接入访问点。

术语表

术语	中文	说明
Federated identity	联盟用户身份	联盟用户的身份认证由客户自己管理
Policy	访问策略	用来描述授权策略的一种描述语言
Grantor	授权者	授权令牌的颁发者(云账号或RAM用户)
Name	被授权者	授权令牌的使用者(即联盟用户)

调用方式

请求结构

服务地址

STS服务的API接入地址为

```
https://sts.aliyuncs.com
```

通信协议

为了保证通信的安全性，STS服务仅支持使用HTTPS安全通道发送请求。

HTTP请求方法

支持HTTP GET/POST方法发送请求，这种方式下请求参数需要包含在请求的URL中。(GET请求最大不得超过4KB, POST请求最大不得超过10MB)

请求参数

每个请求都需要指定要执行的操作，即Action参数（例如AddUser），以及每个操作接口都需要包含的公共请求参数和指定操作接口所特有的请求参数。

字符编码

请求及返回结果都使用UTF-8字符集进行编码。

公共请求参数

Format

- 名称: Format
- 类型: String
- 必须: 否
- 说明: 返回值的类型，支持JSON与XML，默认为XML。

Version

- 名称: Version
- 类型: String
- 必须: 是
- 说明: API版本号，为日期形式：YYYY-MM-DD，本版本对应为2015-04-01。

AccessKeyId

- 名称: AccessKeyId
- 类型: String
- 必须: 是
- 说明: 阿里云颁发给用户的访问服务所用的密钥ID。

Signature

- 名称: Signature
- 类型: String
- 必须: 是
- 说明: 签名结果串，关于签名的计算方法，请参见签名机制。

SignatureMethod

- 名称: SignatureMethod
- 类型: String
- 必须: 是
- 说明: 签名方式，目前仅支持HMAC-SHA1。

SignatureVersion

- 名称: SignatureVersion
- 类型: String
- 必须: 是
- 说明: 签名算法版本，目前版本是1.0

SignatureNonce

- 名称: SignatureNonce
- 类型: String
- 必须: 是
- 说明: 唯一随机数，用于防止网络重放攻击。用户在不同请求间要使用不同的随机数值。

Timestamp

- 名称: Timestamp
- 类型: String

- 必须: 是
- 说明: 请求的时间戳。日期格式按照ISO8601标准表示，并需要使用UTC时间。格式为： YYYY-MM-DDThh:mm:ssZ 例如，2013-01-10T12:00:00Z（为北京时间2013年1月10日20点0分0秒）

请求示例

```
https://sts.aliyuncs.com/
?Format=xml
&Version=2015-04-01
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&Timestamp=2012-06-01T12:00:00Z
...
```

公共返回参数

用户发送的每次接口调用请求，无论成功与否，系统都会返回一个唯一识别码RequestId给用户。

示例

XML示例

```
<?xml version="1.0" encoding="utf-8"?>
<!-结果的根结点-->
<接口名称+Response>
    <!-返回请求标签-->
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!-返回结果数据-->
</接口名称+Response>
```

JSON示例

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216"
    /* 返回结果数据 */
}
```

返回结果处理

调用API服务后返回数据采用统一格式，返回的HTTP状态码为2xx，代表调用成功；返回4xx或5xx的HTTP状态码代表调用失败。调用成功返回的数据格式主要有XML和JSON两种，外部系统可以在请求时传入参数来制定返回的数据格式，默认为XML格式。本文档中的返回示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

成功结果

XML示例

XML返回结果包括请求是否成功信息和具体的业务数据。示例如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!-结果的根结点-->
<接口名称+Response>
   <!-返回请求标签-->
   <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
   <!-返回结果数据-->
</接口名称+Response>
```

JSON示例

```
{
   "RequestId": "4C467B38-3910-447D-87BC-AC049166F216",
   /* 返回结果数据 */
}
```

错误结果

调用接口出错后，将不会返回结果数据。调用方可根据附表<错误代码表>来定位错误原因。

当调用出错时，HTTP请求返回一个4xx或5xx的HTTP状态码。返回的消息体中是具体的错误代码及错误信息。另外还包含一个全局唯一的请求ID：RequestId和一个您该次请求访问的站点ID：HostId。在调用方找不到错误原因时，可以联系阿里云客服，并提供该HostId和RequestId，以便我们尽快帮您解决问题。

XML示例

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <RequestId>8906582E-6722-409A-A6C4-0E7863B733A5</RequestId>
  <HostId>sts.aliyuncs.com</HostId>
```

```
  <Code>InvalidParameter</Code>
  <Message>The specified parameter "Action or Version" is not valid.</Message>
</Error>
```

JSON示例

```
{
  "RequestId": "7463B73D-35CC-4D19-A010-6B8D65D242EF",
  "HostId": "sts.aliyuncs.com",
  "Code": "InvalidParameter",
  "Message": "The specified parameter \"Action or Version\" is not valid."
}
```

签名机制

STS服务会对每个访问的请求进行身份验证，所以无论使用HTTP还是HTTPS协议提交请求，都需要在请求中包含签名（Signature）信息。STS通过使用**Access Key ID**和**Access Key Secret**进行对称加密的方法来验证请求的发送者身份。**Access Key ID**和**Access Key Secret**由阿里云官方颁发给访问者（可以通过阿里云官方网站申请和管理），其中**Access Key ID**用于标识访问者的身份；**Access Key Secret**是用于加密签名字符串和服务器端验证签名字符串的密钥，必须严格保密，只有阿里云和用户知道。

签名步骤

使用请求参数构造规范化的请求字符串（Canonicalized Query String）

a) 按照参数名称的字典顺序对请求中所有的请求参数（包括文档中描述的"公共请求参数"和给定了的请求接口的自定义参数，但不能包括"公共请求参数"中提到Signature参数本身）进行排序。

注：当使用GET方法提交请求时，这些参数就是请求URI中的参数部分（即URI中"?"之后由"&"连接的部分）。

b) 对每个请求参数的名称和值进行编码。名称和值要使用UTF-8字符集进行URL编码，URL编码的编码规则是：

- 对于字符 A-Z、a-z、0-9以及字符"-"、"_"、"."、"~"不编码；
- 对于其他字符编码成"%XY"的格式，其中XY是字符对应ASCII码的16进制表示。比如英文的双引号（"）对应的编码就是%22
- 需要说明的是英文空格（ ）要被编码是%20，而不是加号（+）。

注：一般支持URL编码的库（比如Java中的java.net.URLEncoder）都是按照"application/x-www-form-urlencoded"的MIME类型的规则进行编码的。实现时可以直接使用这类方式进行编码，把编

码后的字符串中加号（+）替换成%20、星号（*）替换成%2A、%7E替换回波浪号（~），即可得到上述规则描述的编码字符串。

c) 对编码后的参数名称和值使用英文等号（=）进行连接。

d) 再把英文等号连接得到的字符串按参数名称的字典顺序依次使用&符号连接，即得到规范化请求字符串。

使用上一步构造的规范化字符串按照下面的规则构造用于计算签名的字符串：

```
StringToSign=
HTTPMethod + "&" +
percentEncode("/") + "&" +
percentEncode(CanonicalizedQueryString)
```

其中HTTPMethod是提交请求用的HTTP方法，比GET。 percentEncode("/")是按照1.b中描述的URL编码规则对字符"/"进行编码得到的值，即"%2F"。

percentEncode(CanonicalizedQueryString)是对第1步中构造的规范化请求字符串按1.b中描述的URL编码规则编码后得到的字符串。

按照RFC2104的定义，使用上面的用于签名的字符串计算签名HMAC值。注意：计算签名时使用的Key就是用户持有的Access Key Secret并加上一个"&"字符(ASCII:38)，使用的哈希算法是SHA1。

按照Base64编码规则把上面的HMAC值编码成字符串，即得到签名值（Signature）。

5. 将得到的签名值作为Signature参数添加到请求参数中，即完成对请求签名的过程。 注意：得到的签名值在作为最后的请求参数值提交给STS服务器的时候，要和其他参数一样，按照RFC3986的规则进行URL编码）。

示例

以AssumeRole为例，签名前的请求URL为：

```
https://sts.aliyuncs.com/?SignatureVersion=1.0&Format=JSON&Timestamp=2015-09-
01T05%3A57%3A34Z&RoleArn=acs%3Aram%3A%3A1234567890123%3Arole%2Ffirstrole&RoleSessionName=clien
t&AccessKeyId=testid&SignatureMethod=HMAC-SHA1&Version=2015-04-
01&Action=AssumeRole&SignatureNonce=571f8fb8-506e-11e5-8e12-b8e8563dc8d2
```

对应的StringToSign是：

```
GET&%2F&AccessKeyId%3Dtestid%26Action%3DAssumeRole%26Format%3DJSON%26RoleArn%3Dacs%253Aram
%253A%253A1234567890123%253Arole%252Ffirstrole%26RoleSessionName%3Dclient%26SignatureMethod%3DH
MAC-SHA1%26SignatureNonce%3D571f8fb8-506e-11e5-8e12-
b8e8563dc8d2%26SignatureVersion%3D1.0%26Timestamp%3D2015-09-
```

```
01T05%253A57%253A34Z%26Version%3D2015-04-01
```

假如使用的Access Key Id是"testid"，Access Key Secret是"testsecret"，用于计算HMAC的Key就是"testsecret&"，则计算得到的签名值是：

```
gNI7b0AyKZHxDgjBGPDgJ1Ce3L4=
```

签名后的请求URL为（注意增加了Signature参数）：

```
https://sts.aliyuncs.com/?SignatureVersion=1.0&Format=JSON&Timestamp=2015-09-01T05%3A57%3A34Z&RoleArn=acs%3Aram%3A%3A1234567890123%3Arole%2Ffirstrole&RoleSessionName=client&AccessKeyId=testid&SignatureMethod=HMAC-SHA1&Version=2015-04-01&Signature=gNI7b0AyKZHxDgjBGPDgJ1Ce3L4%3D&Action=AssumeRole&SignatureNonce=571f8fb8-506e-11e5-8e12-b8e8563dc8d2
```

操作接口

扮演角色(AssumeRole)

接口描述

通过该接口，获取一个扮演该角色的临时身份。

请求参数

Action

- 类型：String
- 必须：是
- 描述：系统规定参数，取值：AssumeRole

RoleArn

- 类型：String
- 必须：是
- 描述：指定角色的资源描述符；每个角色都有一个唯一的资源描述符(Arn)，您可以在RAM的角色管

理中查看一个角色的Arn

RoleSessionName

- 类型：String
- 必须：是
- 描述：此参数用来区分不同的Token，以标明谁在使用此Token，便于审计;
- 格式：^[a-zA-Z0-9\.@\-_]+$ 2-50个字符

Policy

- 名称：Policy
- 类型：String
- 必须：否
- 描述：长度限制为1024字节；您可以通过此参数限制生成的Token的权限，不指定则返回的Token将拥有指定角色的所有权限；

DurationSeconds

- 名称：DurationSeconds
- 类型：Integer
- 必须：否
- 描述：指定的过期时间，单位为秒。过期时间范围：900 ~ 3600，默认值为3600。

返回参数

Credentials

- 类型：Credentials
- 描述：访问凭证

AssumedRoleUser

- 类型：AssumedRoleUser
- 描述：角色扮演临时身份

操作示例

HTTP Request

```
https://sts.aliyuncs.com?Action=AssumeRole
&RoleArn=acs:ram::1234567890123456:role/adminrole
```

```
&RoleSessionName=alice
&DurationSeconds=3600
&Policy=<url_encoded_policy>
&<公共请求参数>
```

HTTP Response

XML格式

```
<AssumeRoleResponse>
  <RequestId>6894B13B-6D71-4EF5-88FA-F32781734A7F</RequestId>
  <AssumedRoleUser>
    <arn>acs:sts::1234567890123456:assumed-role/AdminRole/alice</arn>
    <AssumedRoleUserId>344584339364951186:alice<AssumedRoleUserId>
  </AssumedRoleUser>
  <Credentials>
    <AccessKeyId>STS.L4aBSCSJVMuKg5U1vFDw</AccessKeyId>
    <AccessKeySecret>wyLTSmsyPGP1ohvvw8xYgB29dlGI8KMiH2pKCNZ9</AccessKeySecret>

<SecurityToken>CAESrAIIARKAAShQquMnLIlbvEcIxO6wCoqJufs8sWwieUxu45hS9AvKNEte8KRUWiJWJ6Y+YHAPgNwi7yfRecMFydL2uPOgBI7LDio0RkbYLmJfIxHM2nGBPdml7kYEOXmJp2aDhbvvwVYIyt/8iES/R6N208wQh0Pk2bu+/9dvalp6wOHF4gkFGhhTVFMuTDRhQlNDU0pWTXVLZzVVMXZGRHciBTQzMjc0KgVhbGljZTCpnJjwySk6BlJzYU1ENUJuCgExGmkKBUFsbG93Eh8KDEFjdGlvbkVxdWFscxIGQWN0aW9uGgcKBW9zczoqEj8KDlJlc291cmNlRXF1YWxzEghSZXNvdXJjZRojCiFhY3M6b3NzOio6NDMyNzQ6c2FtcGxlYm94L2FsaWNlLyo=</SecurityToken>
    <Expiration>2015-04-09T11:52:19Z</Expiration>
  </Credentials>
</AssumeRoleResponse>
```

JSON格式

```
{
  "Credentials": {
    "AccessKeyId": "STS.L4aBSCSJVMuKg5U1vFDw",
    "AccessKeySecret": "wyLTSmsyPGP1ohvvw8xYgB29dlGI8KMiH2pKCNZ9",
    "Expiration": "2015-04-09T11:52:19Z",
    "SecurityToken":
"CAESrAIIARKAAShQquMnLIlbvEcIxO6wCoqJufs8sWwieUxu45hS9AvKNEte8KRUWiJWJ6Y+YHAPgNwi7yfRecMFydL2
uPOgBI7LDio0RkbYLmJfIxHM2nGBPdml7kYEOXmJp2aDhbvvwVYIyt/8iES/R6N208wQh0Pk2bu+/9dvalp6wOHF4gkF
GhhTVFMuTDRhQlNDU0pWTXVLZzVVMXZGRHciBTQzMjc0KgVhbGljZTCpnJjwySk6BlJzYU1ENUJuCgExGmkKBUFsb
G93Eh8KDEFjdGlvbkVxdWFscxIGQWN0aW9uGgcKBW9zczoqEj8KDlJlc291cmNlRXF1YWxzEghSZXNvdXJjZRojCiFhY3
M6b3NzOio6NDMyNzQ6c2FtcGxlYm94L2FsaWNlLyo="
  },
  "AssumedRoleUser": {
    "arn": "acs:sts::1234567890123456:assumed-role/AdminRole/alice",
    "AssumedRoleUserId":"344584339364951186:alice"
    },
  "RequestId": "6894B13B-6D71-4EF5-88FA-F32781734A7F"
}
```

数据类型

Credentials

描述

访问凭证

节点名称

Credentials

子节点

AccessKeyId

- 类型：String
- 描述：访问密钥标识

AccessKeySecret

- 类型：String
- 描述：访问密钥

SecurityToken

- 类型：String
- 描述：安全令牌

Expiration

- 类型：String
- 描述：失效时间

AssumedRoleUser

描述

通过扮演角色接口获取的临时身份

节点名称

AssumedRoleUser

子节点

Arn

- 类型：String
- 描述：该角色临时身份的资源描述符

子节点

AssumedRoleUserId

- 类型：String
- 描述：该角色临时身份的用户ID

附录

错误代码表

HTTP Status 400

InvalidParameter

InvalidParameter.RoleArn

- HTTP Status: 400
- ErrorMessage: The parameter RoleArn is wrongly formed.

InvalidParameter.RoleSessionName

- HTTP Status: 400
- ErrorMessage: The parameter RoleSessionName is wrongly formed.

InvalidParameter.DurationSeconds

- HTTP Status: 400
- ErrorMessage: The Min/Max value of DurationSeconds is 15min/1hr.

InvalidParameter.PolicyGrammar

- HTTP Status: 400
- ErrorMessage: The parameter Policy has not passed grammar check.

InvalidParameter.PolicySize

- HTTP Status: 400
- ErrorMessage: The size of Policy must be smaller than 1024 bytes.

HTTP Status 403

NoPermission

- HTTP Status: 403
- ErrorMessage: You are not authorized to do this action. You should be authorized by RAM.

HTTP Status 500

InternalError

- HTTP Status: 500
- ErrorMessage: STS Server Internal Error happened.