Alibaba Cloud
物#网平台

アカウントとログイン

Document Version20191220

目次

1 サーバー SDK

1.1 プライマリアカウントを使用したコンソールへのログイン

プライマリアカウントは、このアカウントのすべてのリソースに対するフル操作権限を持ち、アカウント情報の変更も可能です。

プライマリアカウントを使用した IoT Platformコンソールへのログイン

プライマリアカウントでコンソールにログインすると、**IoT Platform** のフル操作権限を持ちます。

1. *Alibaba Cloud 公式 Web サイト* を開きます。
2. **[**コンソール**]** をクリックします。
3. アカウント名とパスワードを入力してコンソールにログインします。

   注：
   ユーザー名やパスワードを忘れた場合は、ログインページで **[**ユーザー名をお忘れですか？**]** または **[**パスワードをお忘れですか？**]** をクリックすると、ユーザー名やパスワードを再取得できます。
4. コンソールで **[**プロダクト**]** をクリックすると、**Alibaba Cloud** が提供するすべてのプロダクトとサービスが表示されます。
5. 「**IoT Platform**」と検索し、結果に表示された **[IoT Platform]** をクリックして **[IoT Platform** コンソール**]** に入ります。

   注：
   IoT Platform サービスを有効にしていない場合は、**[IoT Platform** コンソール**]** のトップページでこのサービスを有効にするように指示されます。 **[**今すぐ有効化**]** をクリックすると、すぐに有効化されます。

[IoT Platform コンソール**]** に入ると、プロダクト、機器、およびルールの管理ができます。

プライマリアカウントを使用したアクセス制御の作成

プライマリアカウントにはフル権限が与えられているため、プライマリアカウントの漏洩によりセキュリティ上の重大なリスクが発生する可能性があります。 したがって、**Alibaba Cloud** のリソースへのアクセスを他人に許可する際に、プライマリアカウントのユーザー名とパスワードを開示しないでください。 代わりに、 **RAM (Resource Access Management)** を使用してサ

ブアカウントを作成し、サブアカウントに必要なアクセス権限を割り当てる必要があります。 プライマリアカウントのユーザーと管理者以外のすべてのユーザーは、サブアカウントを使用してリソースにアクセスします。 **RAM** ユーザーを使用した **IoT Platform** へのアクセスについて詳しくは、「*RAM* ユーザーの使用」と「権限のカスタマイズ」をご参照ください。

1.2 RAM (Resource Access Management)

この章では、**IoT Platform** のアクセス制御について説明します。

1.2.1 RAM と STS

RAM (Resource Access Management) と **STS (Security Token Service)** は、**Alibaba Cloud** が提供するアクセス制御システムです。 **RAM** と **STS** について詳しくは、「*RAM* のヘルプ」をご参照ください。

RAMは、アカウントのアクセス権限を制御する際に使用します。 **RAM**を使用すると、**RAM** ユーザーの作成と管理が行えます。 さまざまなアクセス権限を与えることで、**RAM** ユーザーがアクセスできるリソースを制御できます。

STS はセキュリティトークン管理システムです。 これは、**RAM** ユーザーに付与された短期間のアクセス権限を管理するために使用します。 **STS** を使用すると、ユーザーに一時的なアクセス権限を与えることができます。

背景

RAM と **STS** を使用すると、プライマリアカウントの **AccessKey** を開示することなく、安全にユーザーへアクセス権を付与することができます。 プライマリアカウントの **AccessKey** が漏洩すると、リソースは大きなセキュリティリスクにさらされます。 **AccessKey** を入手すれば、そのユーザーのリソースに対するあらゆる操作が実行でき、個人情報を盗むこともできます。

RAM は、長期間のアクセス権限を制御するためのメカニズムです。 **RAM** ユーザーを作成すると、各 **RAM** ユーザーごとに異なる権限を与えることができます。 もし **RAM** ユーザーの **AccessKey** が漏洩しても、プライマリアカウントの **AccessKey** が漏洩した場合ほどのリスクはありません。 **RAM** ユーザーの **AccessKey** が漏洩しても、そこから漏れる可能性のある情報は限られています。 **RAM** ユーザーは長期間有効です。

ユーザーに長期間のアクセス許可を与えることができる **RAM** とは異なり、**STS** ではユーザーに一時的なアクセス権を与えることができます。 **STS API** を呼び出すことで、一時的な **AccessKey** とトークンを取得できます。 一時的な **AccessKey** と トークンを **RAM** ユーザーに割り当てることで、特定のリソースにアクセスすることができるようになります。 **STS** で取得で

きる権限は厳しく制限されており、有効期限も設定されています。 したがって、たとえ情報が予期せず流出したとしても、システムが重大なセキュリティ侵害を負うということはありません。

RAM および **STS** の使用方法について詳細は、「用例」をご参照ください。

基本概念

RAM および **STS** を使用する前に、次の基本概念について簡単に理解しておくことを推奨します。

- **RAM** ユーザー**: RAM** コンソールを使用して作成したユーザー。 **RAM** ユーザーの作成中または作成後に、**AccessKey** を生成することができます。 **RAM** ユーザーを作成したら、パスワードを設定し、権限を付与する必要があります。 これが完了すると、**RAM** ユーザーは許可された操作を実行できるようになります。 **RAM** ユーザーは、特定の操作権限を持つユーザーと考えることができます。
- ロール**:** 権限のグループを表す仮想的なエンティティ。 ロール自体には、ログインパスワードや **AccessKey** はありません。 **RAM** ユーザーにロールを割り当てることができます。 ロールが割り当てられるた **RAM** ユーザーは、そのロールに付与された権限を持ちます。
- ポリシー**:** ポリシーはアクセス権を定義します。 たとえば、ポリシーは、 **RAM** ユーザーが特定のリソースを読み書きするための権限を定義します。
- リソース**: RAM** ユーザーがアクセス可能なクラウドリソースを指します。たとえば、全ての **Table Store** インスタンス、ある **1** つの **Table Store** インスタンス、または **1** つの **Table Store** インスタンス内のある **1** つのテーブルなどです。

RAM ユーザーとロールとの関係は、個人とその立場との関係に似ています。 たとえば、ある人物の役割 (ロール) は、職場では従業員であり、自宅では父親であるということがいえるでしょう。 同じ **1** 人の人物でも、状況 (シナリオ) が変われば役割 (ロール) も変わります。 ある人物が特定のロールを引き受けると、その人はそのロールに伴う権限を持ちます。 ロール自体は操作上のエンティティではありません。 ユーザーがロールを引き受けることによって初めて、ロールは完全な操作エンティティとなります。 複数のユーザーが共同で **1** つのロールを引き受けることができます。

例

アカウントの **AccessKey** が漏洩した場合にセキュリティリスクにさらされるのを防ぐため、アカウント管理者は **2** つの **RAM** ユーザーを作成します。 **2** つの **RAM** ユーザーは **A** と **B** とします。各々について、**AccessKey** が生成されます。 **A** には読み取り権限があり、**B** には書き込み権限があります。 管理者は、**RAM** コンソールからいつでも **RAM** ユーザーの権限を取り消すことができます。

さらに、**IoT Platform** の **API** に対する一時的なアクセス権を関係者に与える必要があるとします。 このとき、**A** の **AccessKey** を開示してはいけません。 代わりに、管理者はロール (**C** とします) を作成します。 そして、そのロールに **IoT Platform** の **API** へのアクセス権を与えます。 この段階では、直接 **C** を使用することはできないということにご注意ください。**C** の **AccessKey** は存在せず、**C** は **IoT Platform API** へのアクセス権を持つ仮想エンティティに過ぎないからです。

管理者は、**STS API** の**AssumeRole** を呼び出して、**IoT Platform API** へのアクセスに使用できる一時的なセキュリティ認証情報を取得する必要があります。 **AssumeRole** を呼び出す際、**RoleArn** の値は **C** の **ARN (Alibaba Cloud resource name)** となります。**AssumeRole** の呼び出しが成功すると、**STS** はセキュリティ認証情報として 一時的な**AccessKeyId**、**AccessKeySecret**、**SecurityToken** を返します。 これらの認証情報の有効期間は、**AssumeRole** を呼び出す際に指定できます。 アカウント管理者は、**IoT Platform** の **API** にアクセスする必要があるユーザーにこれらの認証情報を提供することができます。 この認証情報による **API** へのアクセス権は一時的なものです。

RAM と STS の使用方法が複雑な理由

RAM と **STS** の概念および使用方法は複雑です。 サービスの利便性と引き換えに、アカウントのセキュリティと柔軟なアクセス制御を保証しています。

操作を実行するエンティティと、権限グループを表す仮想エンティティと分離して保持するために、**RAM** ユーザーとロールは分離されています。 あるユーザーが複数のアクセス権限 (読み取り権限と書き込み権限など) を必要としていますが、実際には一度に **1** つのアクセス権限しか必要としない場合は、**2** つのロールを作成することができます。 作成した **2** つのロールに、それぞれ読み取り権限と書き込み権限を与えます。 次に、**RAM** ユーザーを **1** つ作成し、そのユーザーに両方のロールを割り当てます。 その **RAM** ユーザーが読み取り権限を必要とする場合は、読み取り権限を持つロールを利用し、 書き込み権限を必要とする場合は、書き込み権限を持つロールを利用することになります。 これによって、各操作で必要とする以上の権限が与えられることを防ぎ、リスクを軽減することができます。 さらに、ロールを他のアカウントや **RAM** ユーザーに割り当て、ロールに含まれる権限を付与することもできます。 これにより、ロールのアクセス権限は使用しやすくなります。

STS を使用すると、より柔軟なアクセス制御が可能になります。 たとえば、認証情報の有効期間を設定することができます。 ただし、長期的な認証情報が必要な場合は、**RAM** のみを使用して **RAM** ユーザーの管理を行ってください。

これに続く文書では、**RAM** と **STS** を使用するためのガイドラインとその使用例を示します。**RAM** と **STS** が提供する **API** について詳しくは、「*API リファレンス - RAM*」、「*API リファレンス - STS*」をご参照ください。

1.2.2 権限のカスタマイズ

権限は、対象のリソースに対する特定の操作を、システムが許可したり拒否したりする条件を定義します。

権限は、権限付与ポリシーで定義されています。 権限をカスタマイズすると、カスタム権限付与ポリシーを使用して特定の権限を定義できます。 **RAM (Resource Access Management)** コンソールの**[**ポリシー**]** ページで **[**権限付与ポリシーを作成**]** をクリックし、権限付与ポリシーのカスタマイズを行います。 権限付与ポリシーをカスタマイズする場合は、空白のテンプレートを選択します。

権限付与ポリシーは **JSON** 文字列で、以下のパラメーターを必要とします。

- `Action`**:** 権限を付与するアクションを示します。 **IoT** アクションは "*iot:*" で始まります。アクションについての詳細と用例は、「アクションの定義」 をご参照ください。
- `Effect` **:** 権限タイプを示します。"`Allow`（許可）" または "`Deny`（拒否）" のいずれかです。
- `Resource` **: IoT Platform** では、リソースに対する権限付与をサポートしていないため、アスタリスク **(**`*`**)** を入力してください。
- `Condition`**:** 認証条件を示します。 詳細については、「条件の定義」をご参照ください。

アクションの定義

`Action` は、**API (application programming interface)** 操作の名前です。 権限付与ポリシーを作成するときは、各アクションの頭に "*iot:*" を付け、複数のアクションはコンマ **(** `,` **)** で区切ります。 ワイルドカード文字としてアスタリスク **(** `*` **)** を使用することもできます。 **IoT Platform** で使用される **API** 名の定義について詳しくは、「*API アクセス権限*」をご参照ください。

アクション定義の例を次に示します。

- 単一の **API** 操作を定義します。

```
"Action": "iot:CreateProduct"
```

- 複数の **API** 操作を定義します。

```
"Action": [
"iot:UpdateProduct",
"iot:QueryProduct"
```

```
]
```

- すべての読み取り専用 **API** 操作を定義します。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "iot:Query*",
        "iot:List*",
        "iot:Get*",
        "iot:BatchGet*",
        "iot:Check*"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDatabases",
        "rds:DescribeAccounts",
        "rds:DescribeDBInstanceNetInfo"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "ram:ListRoles",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "mns:ListTopic",
        "mns:GetTopicRef"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ots:ListInstance",
        "ots:GetInstance",
        "ots:ListTable",
        "ots:DescribeTable"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "fc:ListServices",
        "fc:GetService",
        "fc:GetFunction",
        "fc:ListFunctions"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
```

```
        "log:ListShards",
        "log:ListLogStores",
        "log:ListProject"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "cms:QueryMetricList"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

・ すべての読み書き **API** 操作を定義します。

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": "iot:*",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "rds:DescribeDBInstances",
        "rds:DescribeDatabases",
        "rds:DescribeAccounts",
        "rds:DescribeDBInstanceNetInfo",
        "rds:ModifySecurityIps"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "ram:ListRoles",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "mns:ListTopic",
        "mns:GetTopicRef"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ots:ListInstance",
        "ots:ListTable",
        "ots:DescribeTable",
        "ots:GetInstance"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
```

```
        "fc:ListServices",
        "fc:GetService",
        "fc:GetFunction",
        "fc:ListFunctions"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "log:ListShards",
        "log:ListLogStores",
        "log:ListProject"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "ram:PassRole",
      "Resource": "*",
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "acs:Service": "iot.aliyuncs.com"
        }
      }
    },
    {
      "Action": [
        "cms:QueryMetricList"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

条件の定義

RAM 権限付与ポリシーでは、現在、アクセス **IP** アドレス制限、**HTTPS (Hypertext Transfer Protocol Secure)** ベースのアクセス許可、**MFA (**マルチファクター認証**)** ベースのアクセス許可、アクセス時間制限など複数の認証条件をサポートしています。 **IoT Platform** 上のすべての **API** 操作は、上記の認証条件をサポートします。

アクセス元 **IP** アドレスによるアクセス制御

このアクセス制御は、**IoT Platform** にアクセスできる発信元 **IP** アドレスを制限し、**CIDR (Classless Inter-Domain Routing)** ブロックによるフィルタリングをサポートします。 典型的なシナリオは次の通りです。

- 単一の **IP** アドレスまたは **CIDR** ブロックにアクセス制御ルールを適用します。 たとえば、次のコードは、**IP** アドレス **10.101.168.111** および **10.101.169.111/24** からのアクセス要求のみが許可されていることを示します。

```
{
"Statement": [
```

```
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.168.111",
"10.101.169.111/24"
]
}
}
}
],
"Version": "1"
}
```

- アクセス制御ルールを複数の **IP** アドレスに適用します。 たとえば、次のコードは、**IP** アドレス**10.101.168.111** および **10.101.169.111** からのアクセス要求のみが許可されていることを示します。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IPaddress ":{
"acs:SourceIp": [
"10.101.168.111",
"10.101.169.111"
]
}
}
}
],
"Version": "1"
}
```

HTTPS ベースのアクセス制御

このアクセス制御では、**HTTPS** ベースのアクセスを有効または無効にすることができます。

たとえば、次のコードは、**HTTPS** ベースのアクセスのみが許可されていることを示しています。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"Bool": {
"acs:SecureTransport": "true"
}
```

```
}
}
],
"Version": "1"
}
```

MFA ベースのアクセス制御

このアクセス制御では、**MFA** ベースのアクセスを有効または無効にすることができます。

たとえば、次のコードは、**MFA** ベースのアクセスのみが許可されていることを示しています。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"Bool": {
"acs:MFAPresent ": "true"
}
}
}
],
"Version": "1"
}
```

アクセス時間制限

このアクセス制御では、リクエストのアクセス時間を制限することができます。 指定された時刻より前のアクセス要求が許可または拒否されます。

たとえば、次のコードは、北京時間 **(UTC+8)** で **2019** 年 **1** 月 **1** 日 **00:00:00** より前のアクセス要求のみが許可されていることを示します。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"DateLessThan": {
"acs:CurrentTime": "2019-01-01T00:00:00+08:00"
}
}
}
],
"Version": "1"
}
```

典型的なシナリオ

これらのアクション、リソース、および条件の定義に基づいた、典型的なシナリオにおける権限付与ポリシーを以下に示します。

次の例は、アクセスを許可する権限付与ポリシーです。

シナリオ: CIDR ブロック 10.101.168.111/24 にIoT Platform のアクセス権限を割り当て、北京時間 (UTC+8) で 2019 年 1 月 1 日 00:00:00 より前の HTTPS ベースのアクセスだけを許可します。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IPaddress ":{
"acs:SourceIp": [
"10.101.168.111/24"
]
},
"DateLessThan": {
"acs:CurrentTime": "2019-01-01T00:00:00+08:00"
},
"Bool": {
"acs:SecureTransport": "true"
}
}
}
],
"Version": "1"
}
```

次の例は、特定の条件のアクセスを拒否する権限付与ポリシーです。

シナリオ: IP アドレス 10.101.169.111 からの読み取り要求を拒否します。

```
{
"Statement": [
{
"Effect": "Deny",
"Action": [
"iot:Query*",
"iot:List*",
"iot:Get*",
"iot:BatchGet*"
],
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.169.111"
]
}
}
}
],
"Version": "1"
```

```
}
```

権限付与ポリシーを作成したら、**RAM** コンソール内の **[ユーザー管理]** ページで権限付与ポリシーを **RAM** ユーザー に適用します。 許可された **RAM** ユーザーは、このポリシーで定義されている操作を実行できます。 **RAM** ユーザーの作成とアクセス権付与の詳細については、「*RAM* ユーザーの使用」をご参照ください。

1.2.3 API アクセス権限

次の表の各操作は、**RAM** ユーザーの権限付与ポリシーを作成する際に指定する `Action` の値を示しています。

RAM ユーザーの権限付与ポリシーの作成について詳細は、「権限のカスタマイズ」をご参照ください。

操作	RAM アクション	リソース	説明
CreateProduct	**iot:CreateProduct**	*	プロダクトを作成します。
UpdateProduct	**iot:UpdateProduct**	*	プロダクトの情報を更新します。
QueryProduct	**iot:QueryProduct**	*	プロダクトの詳細情報を取得します。
QueryProductList	**iot:QueryProductList**	*	全てのプロダクトを取得します。
DeleteProduct	**iot:DeleteProduct**	*	プロダクトを削除します。
CreateProductTags	**iot:CreateProductTags**	*	プロダクトのタグを作成します。
UpdateProductTags	**iot:UpdateProductTags**	*	プロダクトのタグを更新します。
DeleteProductTags	**iot:DeleteProductTags**	*	プロダクトのタグを削除します。
ListProductTags	**iot:ListProductTags**	*	プロダクトのタグを取得します。
ListProductByTags	**iot:ListProductByTags**	*	タグを指定してプロダクトを取得します。
RegisterDevice	**iot:RegisterDevice**	*	デバイスを登録します。
QueryDevice	**iot:QueryDevice**	*	指定されたプロダクトに紐付くデバイスを全て取得します。
DeleteDevice	**iot:DeleteDevice**	*	デバイスを削除します。

操作	RAM アクション	リソース	説明
QueryPageByApplyId	**iot:QueryPageByApplyId**	*	同時に登録されたデバイスの情報を取得します。
BatchGetDeviceState	**iot:BatchGetDeviceState**	*	一度に複数のデバイスのステータスを取得します。
BatchRegisterDeviceWithApplyId	**iot:BatchRegisterDeviceWithApplyId**	*	指定された申請 ID を使用して、複数のデバイスを同時に登録します。
BatchRegisterDevice	**iot:BatchRegisterDevice**	*	同時に複数のデバイスを登録します (デバイス名は指定しません)。
QueryBatchRegisterDeviceStatus	**iot:QueryBatchRegisterDeviceStatus**	*	複数デバイスの登録について、進行状況と結果を取得します。
BatchCheckDeviceNames	**iot:BatchCheckDeviceNames**	*	デバイス名を一括で指定します。
QueryDeviceStatistics	**iot:QueryDeviceStatistics**	*	デバイスの統計情報を照会します。
QueryDeviceEventData	**iot:QueryDeviceEventData**	*	デバイスのイベント履歴を照会します。
QueryDeviceServiceData	**iot:QueryDeviceServiceData**	*	デバイスのサービス履歴を照会します。
SetDeviceProperty	**iot:SetDeviceProperty**	*	指定したデバイスのプロパティを設定します。
SetDevicesProperty	**iot:SetDevicesProperty**	*	複数のデバイスのプロパティを設定します。
InvokeThingService	**iot:InvokeThingService**	*	デバイス上のサービスを起動します。
InvokeThingsService	**iot:InvokeThingsService**	*	複数のデバイス上のサービスを起動します。
QueryDevicePropertyStatus	**iot:QueryDevicePropertyStatus**	*	デバイスのプロパティのスナップショットを照会します。
QueryDeviceDetail	**iot:QueryDeviceDetail**	*	デバイスの詳細情報を取得します。
DisableThing	**iot:DisableThing**	*	デバイスを無効にします。

操作	RAM アクション	リソース	説明
EnableThing	**iot:EnableThing**	*	無効化されたデバイスを有効にします。
GetThingTopo	**iot:GetThingTopo**	*	デバイスのトポロジ関係を照会します。
RemoveThingTopo	**iot:RemoveThingTopo**	*	デバイスのトポロジ関係を削除します。
NotifyAddThingTopo	**iot:NotifyAddThingTopo**	*	指定されたサブデバイスのトポロジ関係を追加するようゲートウェイ機器に通知します。
QueryDevicePropertyData	**iot:QueryDevicePropertyData**	*	デバイスプロパティの履歴を照会します。
QueryDevicePropertiesData	**iot:QueryDevicePropertiesData**	*	複数のデバイスプロパティの履歴を照会します。
GetGatewayBySubDevice	**iot:GetGatewayBySubDevice**	*	サブデバイス情報を使用してゲートウェイ機器の情報を照会します。
SaveDeviceProp	**iot:SaveDeviceProp**	*	デバイスのタグを作成します。
QueryDeviceProp	**iot:QueryDeviceProp**	*	デバイスのタグをすべて取得します。
DeleteDeviceProp	**iot:DeleteDeviceProp**	*	デバイスのタグを削除します。
QueryDeviceByTags	**iot:QueryDeviceByTags**	*	タグを指定してデバイスを検索します。
CreateDeviceGroup	**iot:CreateDeviceGroup**	*	デバイスグループを作成します。
UpdateDeviceGroup	**iot:UpdateDeviceGroup**	*	デバイスグループの情報を更新します。
DeleteDeviceGroup	**iot:DeleteDeviceGroup**	*	デバイスグループを削除します。
BatchAddDeviceGroupRelations	**iot:BatchAddDeviceGroupRelations**	*	デバイスをグループに追加します。
BatchDeleteDeviceGroupRelations	**iot:BatchDeleteDeviceGroupRelations**	*	デバイスをグループから削除します。

操作	RAM アクション	リソース	説明
QueryDeviceGroupInfo	**iot:QueryDeviceGroupInfo**	*	グループの詳細情報を照会します。
QueryDeviceGroupList	**iot:QueryDeviceGroupList**	*	デバイスグループを全て取得します。
SetDeviceGroupTags	**iot:SetDeviceGroupTags**	*	グループのタグを作成、更新、削除します。
QueryDeviceGroupTagList	**iot:QueryDeviceGroupTagList**	*	グループのタグを全て取得します。
QueryDeviceGroupByDevice	**iot:QueryDeviceGroupByDevice**	*	指定したデバイスが含まれているグループを照会します。
QueryDeviceListByDeviceGroup	**iot:QueryDeviceListByDeviceGroup**	*	デバイスグループ内のデバイスを取得します。
QuerySuperDeviceGroup	**iot:QuerySuperDeviceGroup**	*	デバイスグループの親グループを取得します。
QueryDeviceGroupByTags	**iot:QueryDeviceGroupByTags**	*	タグを指定してデバイスグループを照会します。
StartRule	**iot:StartRule**	*	ルールを有効化します。
StopRule	**iot:StopRule**	*	ルールを無効化します。
ListRule	**iot:ListRule**	*	全てのルールを照会します。
GetRule	**iot:GetRule**	*	ルールの詳細を取得します。
CreateRule	**iot:CreateRule**	*	ルールを作成します。
UpdateRule	**iot:UpdateRule**	*	ルールの情報を更新します。
DeleteRule	**iot:DeleteRule**	*	ルールを削除します。
CreateRuleAction	**iot:CreateRuleAction**	*	ルールのデータ転送メソッドを作成します。
UpdateRuleAction	**iot:UpdateRuleAction**	*	データ転送メソッドを更新します。
DeleteRuleAction	**iot:DeleteRuleAction**	*	データ転送メソッドを削除します。
GetRuleAction	**iot:GetRuleAction**	*	データ転送メソッドの詳細情報を照会します。
ListRuleActions	**iot:ListRuleActions**	*	ルールに含まれるデータ転送メソッドを全て取得します。

操作	RAM アクション	リソース	説明
Pub	**iot:Pub**	*	メッセージをパブリッシュします。
PubBroadcast	**iot:PubBroadcast**	*	ブロードキャストトピックをサブスクライブしているデバイスにメッセージをパブリッシュします。
RRpc	**iot:RRpc**	*	デバイスにメッセージを送信し、デバイスからのレスポンスを取得します。
CreateProductTopic	**iot:CreateProductTopic**	*	プロダクトのトピックカテゴリを作成します。
DeleteProductTopic	**iot:DeleteProductTopic**	*	トピックカテゴリを削除します。
QueryProductTopic	**iot:QueryProductTopic**	*	プロダクトのトピックカテゴリをすべて取得します。
UpdateProductTopic	**iot:UpdateProductTopic**	*	トピックカテゴリを更新します。
CreateTopicRouteTable	**iot:CreateTopicRouteTable**	*	トピック間のメッセージルーティング関係を作成します。
DeleteTopicRouteTable	**iot:DeleteTopicRouteTable**	*	トピック間のメッセージルーティング関係を削除します。
QueryTopicReverseRouteTable	**iot:QueryTopicReverseRouteTable**	*	対象のトピックの発信元となるトピックを照会します。
QueryTopicRouteTable	**iot:QueryTopicRouteTable**	*	発信元のトピックから対象となるトピックを照会します。
GetDeviceShadow	**iot:GetDeviceShadow**	*	デバイスのシャドウ情報を照会します。
UpdateDeviceShadow	**iot:UpdateDeviceShadow**	*	デバイスのシャドウ情報を更新します。

1.2.4 RAM ユーザーの使用

RAM ユーザー (サブアカウント) は **IoT Platform** のコンソールにログインして **IoT** リソースを管理することができます。また、対応する **AccessKeyId** と **AccessKeySecret** を使って **IoT API (application programming interface)** を使用できます。

まず RAM ユーザーを作成し、次に、権限付与ポリシーを使ってこの RAM ユーザーに IoT Platform へのアクセス権限を割り当てる必要があります。 権限付与ポリシーのカスタマイズについて詳細は、権限のカスタマイズをご参照ください。

RAM ユーザーの作成

すでに RAM ユーザーを作成している場合は、この手順をスキップしてください。

1. 「*RAM コンソール*」にログインします。
2. 左側のメニューで、[ユーザー] をクリックします。
3. [ユーザーの作成] をクリックします。
4. ユーザー情報を入力し、 [このユーザーの AccessKey を自動生成] にチェックを入れ、[OK] をクリックします。

   > 注：
   > [OK] をクリックすると、AccessKey を保存するように求められます。 AccessKey をダウンロードできるのは この時だけです。 すぐにこの AccessKey を保存して、安全に保管する必要があります。 この AccessKey は、RAM ユーザーが API 操作を呼び出すときに必要になります。

5. 初期ログインパスワードを設定します。
   a. [ユーザー管理] ページで、作成した RAM ユーザーの [管理] をクリックし、[ユーザーの詳細] ページを表示します。
   b. [コンソールへのログインの有効化] をクリックします。
   c. この RAM ユーザーの初期パスワードを設定し、[次回ログイン時にパスワードのリセットが必要] にチェックを入れ、[OK] をクリックします。
6. マルチファクター認証 (MFA) を有効にします。 (任意)

   [ユーザーの詳細] ページで、[VMFAデバイスの有効化] をクリックします。

RAM ユーザーを作成すると、RAM ユーザーは、RAM ユーザー用ログインリンクから Alibaba Cloud の公式 Web サイトと IoT Platform コンソールにログインできます。 RAM ユーザー用ログインリンクを取得するには、[RAM コンソール] の [RAM の概要] ページにアクセスします。

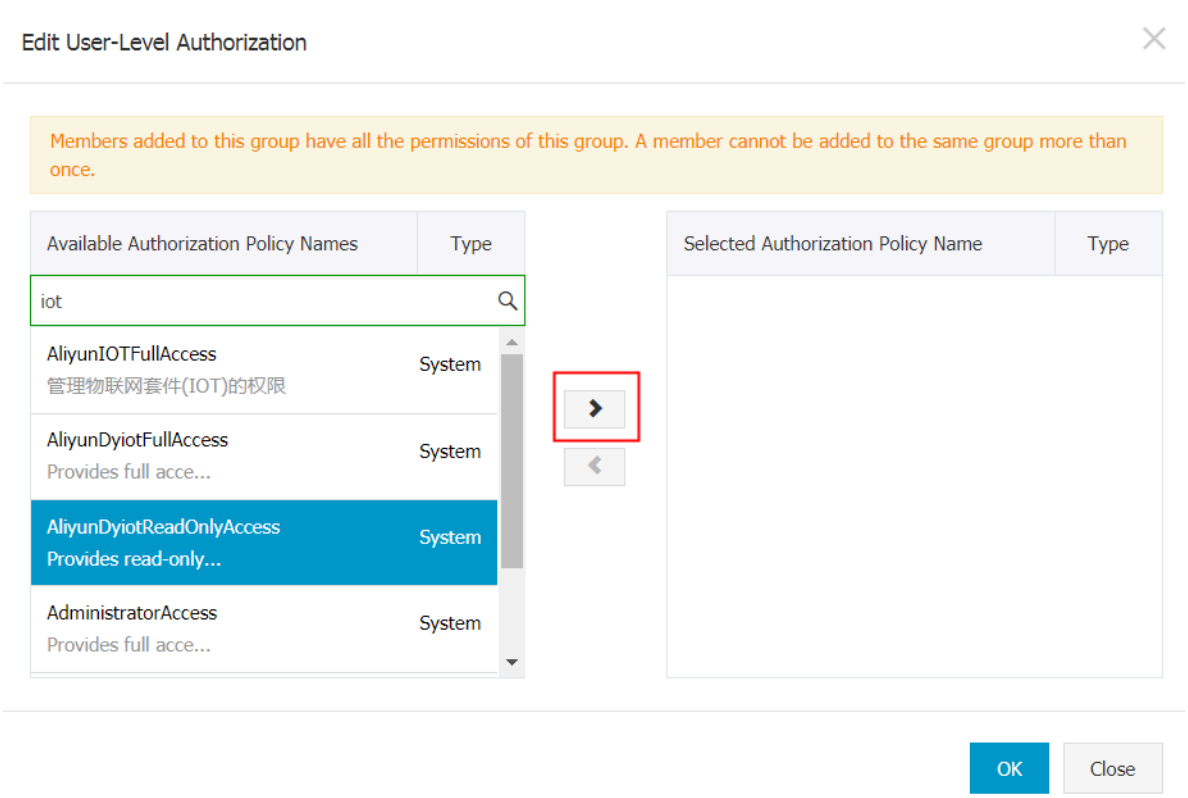
ただし、アクセス権が付与されていないと、RAM ユーザーは Alibaba Cloud リソースにアクセスすることができません。 したがって、この RAM ユーザーに IoT Platform にアクセスする権限を割り当てる必要があります。

RAM ユーザーに IoT Platform へのアクセスを許可する

[RAM コンソール] 内の [ユーザー管理] ページから RAM ユーザーに権限を割り当てるか、[グループ管理] ページから同じ権限をグループに割り当てます。 RAM ユーザーにアクセス権限を割り当てるには、次の手順を実行します。

1. プライマリアカウントを使用して [*RAMコンソール*] にログインします。
2. 画面左側のメニューで、[ユーザー] をクリックします。
3. 権限を付与したい RAM ユーザーの横にある [許可] をクリックします。
4. [許可] ダイアログボックスで、適用したい権限付与ポリシーを選択します。 ページの中央にある右矢印をクリックして、選択した権限付与ポリシーを [選択済みの権限付与ポリシー名] に移動し、[OK] をクリックします。

> 注：
> RAM ユーザーにカスタム権限を割り当てるには、まず権限付与ポリシーを作成する必要があります。 権限付与ポリシーのカスタマイズについて詳細は、権限のカスタマイズをご参照ください。



権限を付与された RAM ユーザーは、権限付与ポリシーに定義されたリソースにアクセスし、指定された操作を実行することができます。

RAM ユーザーを使用したコンソールへのログイン

プライマリアカウントのユーザーは、公式 **Web** サイトからコンソールにログインできます。**RAM** ユーザーは、**[RAM** ユーザーログイン**]** ページからコンソールにログインする必要があります。

1. **[RAM** ユーザーログイン**]** ページにログインするためのリンクを取得します。

   プライマリアカウントで **[RAM** コンソール**]** に ログインし、 **[RAM** の概要**]** ページの **[RAM** ユーザー用ログインリンク**]** を探し、ログインリンクを **RAM** ユーザーに送信します。
2. **RAM** ユーザーは **[RAM** ユーザーログイン**]** ページにアクセスし、**RAM** ユーザー名とパスワードを使ってコンソールにログインします。

   > 注 **:**
   > ログイン名の形式は次のとおりです**: RAM** ユーザー名@社名のエイリアス **(username@company-alias** など)。 **RAM** ユーザーは、初回ログイン時にログインパスワードを変更する必要があります。
3. ページの右上隅にある **[**コンソール**]** をクリックして、**[**ホーム**]** ページに移動します。
4. **[**プロダクト**]** をクリックし、**[IoT Platform]** を選択して **[IoT Platform** コンソール**]** に移動します。

このコンソールで、**RAM** ユーザーは許可された操作を実行できます。

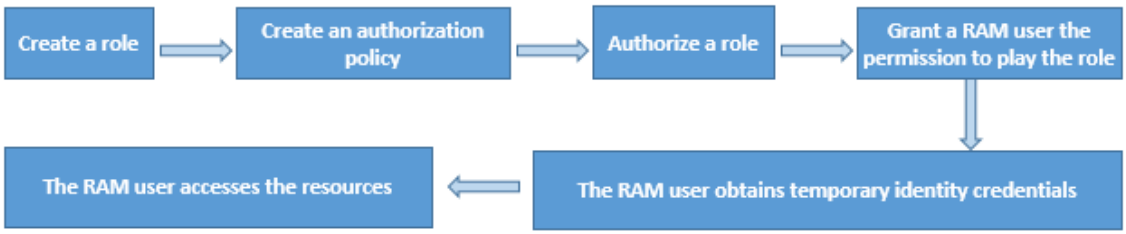
1.2.5 STS 上級ガイド

STS (Security Token Service) では、**RAM (Resource Access Management)** よりも厳格な権限管理を行うことができます。 **STS** を使用してリソースアクセス制御を実装するには、複雑な権限付与プロセスを伴います。**STS** を使うと、**RAM** ユーザーに、リソースへの一時的なアクセス権を与えることができます。

RAM ユーザーならびに、**RAM** ユーザーに付与された権限は長期的に有効です。 **RAM** ユーザーを手動で削除するか、**RAM** ユーザーのアクセス権限を無効にする必要があります。 もし **RAM** ユーザーのアカウント情報が漏洩した場合、そのユーザー情報や関連するアクセス権の削除に時間がかかってしまうと、お客様の **Alibaba Cloud** リソースや重要な情報が侵害されかね

ません。 そのため、主要な権限や長期的に有効である必要のない権限を管理する際には **STS** の使用を推奨します。

図 1-1 : RAM **ユーザーに一時的な権限を付与するプロセス**



ステップ 1: ロールの作成

ロールは、一連の権限を持った仮想ユーザーを表す仮想エンティティです。

1. 『*RAM コンソール*』にログインします。
2. [ロール] の[ロールの作成] を選択して、 ロールを作成します。
3. [ユーザーロール] を選択します。
4. [デフォルトのアカウント情報を使用] をオンにし、[次へ] をクリックします。
5. [ロール名] と [説明] を入力し、[作成] をクリックします。
6. [閉じる] または [許可] をクリックします。

   このロールに適用する権限付与ポリシーを既に作成している場合は、[許可] をクリックしてこのユーザーに権限を付与します。

   まだ権限付与ポリシーを作成していない場合は、[閉じる] をクリックします。 [ポリシー] をクリックすると、このロールに適用する権限付与ポリシーを作成できます。

ステップ 2: 権限付与ポリシーの作成

権限付与ポリシーは、ロールに付与するリソースアクセス権限を定義します。

1. 『*RAM コンソール*』で [ポリシー] の [権限付与ポリシーの作成] をクリックします。
2. 空白のテンプレートを選択します。

3. 権限付与ポリシー名とポリシーの内容を設定し、[権限付与ポリシーの作成] をクリックします。

   ポリシーの内容の書き方について詳しく知りたい場合は、[権限付与ポリシーの形式] をクリックします。

   権限付与ポリシーの例 : IoT リソースに対する読み取り権限

```
{
"Version": "1",
"Statement": [
{
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action":[
"mns:ListTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action":[
"log:ListShards",
"log:ListLogStores",
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
```

```
{
"Effect": "Allow",
"Action": [
"iot:Query*",
"iot:List*",
"iot:Get*",
"iot:BatchGet*"
],
"Resource": "*"
}
]
}
```

権限付与ポリシーの例:**IoT** リソースに対する読み取りと書き込みの権限

```
{
"Version": "1",
"Statement": [
{
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action":[
"mns:ListTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action":[
"log:ListShards",
"log:ListLogStores",
```

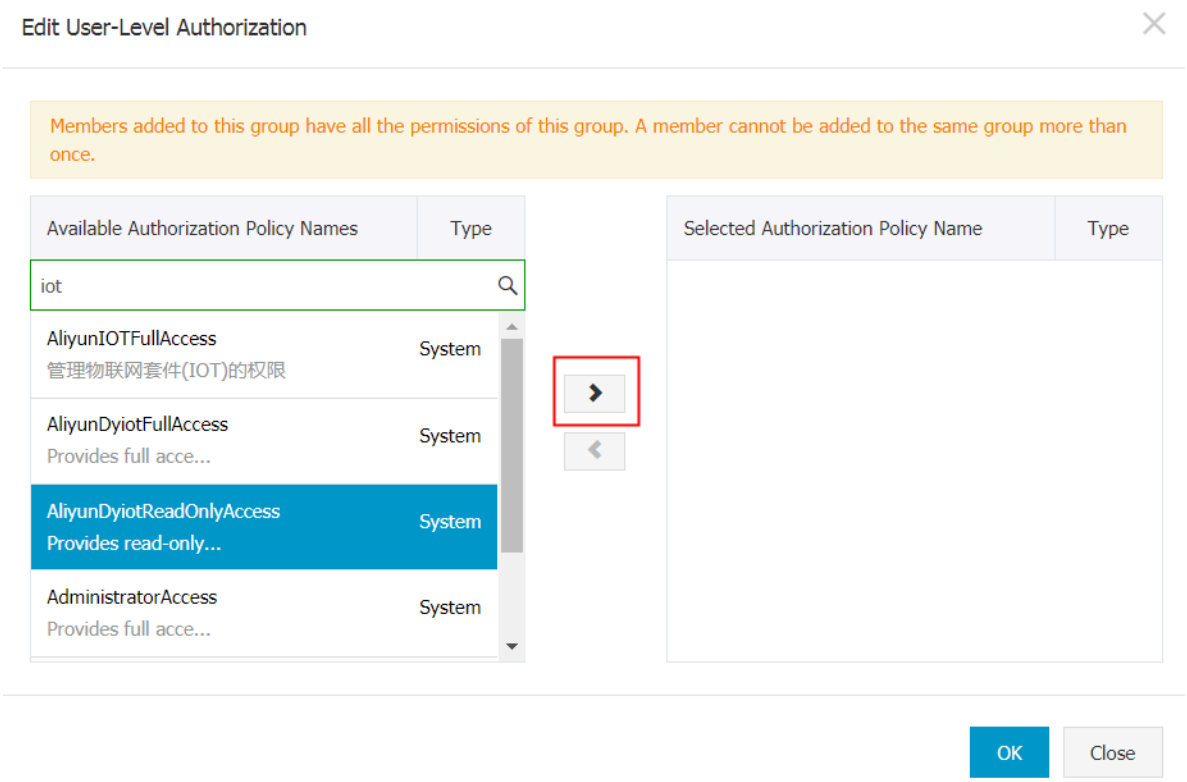
```
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*"
}
]
}
```

権限付与ポリシーが作成できたら、そのポリシーに定義されている権限をロールに付与することができます。

ステップ 3: ロールに権限を付与

ロールは、権限を付与されて初めてリソースへのアクセス権を持ちます。

1. 『*RAM コンソール*』内で、[ロール] をクリックします。
2. 権限を付与したいロールを選択し、[許可] をクリックします。
3. 表示されたダイアログボックスで、指定したロールに付与したいカスタム権限付与ポリシーを選択します。中央の右矢印をクリックして選択した権限付与ポリシーを [選択された権限付与ポリシー名] リストに移動させ、**[OK]** をクリックします。



権限付与が完了すると、ロールは選択した権限付与ポリシーに定義されたアクセス権を持つようになります。 [管理] をクリックして [ロールの詳細] ページを開くと、そのロールの基本情報や付与されているアクセス権限を表示できます。

次に、RAM ユーザーにロールの権限を付与する必要があります。

ステップ 4: RAM ユーザーにロールの権限を付与

権限付与の完了後、ロールは権限付与ポリシーで定義されたアクセス権を持ちます。 しかし、ロールは仮想ユーザーにすぎません。 許可された操作を実行するには、RAM ユーザーにそのロールを適用する必要があります。 もしすべての RAM ユーザーにそのロールが適用された場合、セキュリティ上のリスクが発生します。 そのロールを適用してアクセス権を与えるのは、特定の RAM ユーザーのみにすべきです。

RAM ユーザーにこのロールを適用してアクセス権を与えるには、"Resource" パラメーターにこのロールの ID が設定されているカスタム権限付与ポリシーを作成する必要があります。 その後、その権限付与ポリシーを RAM ユーザーに適用して権限を付与することができます。

1. 『*RAM コンソール*』内で、[ポリシー] > [権限付与ポリシーの作成] の順にクリックします。
2. 空白のテンプレートを選択します。
3. 権限付与ポリシー名とポリシーの内容を入力し、[権限付与ポリシーの作成] をクリックします。

    注：

   ポリシーの内容で、"リソース" パラメーターの値をロールの ARN (Alibaba Cloud resource name) にします。 [ロール] ページで指定したいロールを見つけます。[管理] をクリックして [ロールの詳細] ページを開くと、ロールの ARN を確認できます 。

   権限付与ポリシーの例

   ```
   {
   "Version": "1",
   "Statement": [
   {
   "Effect": "Allow",
   "Action": "iot:QueryProduct",
   "Resource": "Role Arn"
   }
   ]
   }
   ```

4. 権限付与ポリシーが作成できたら、[RAM コンソール] のトップページを開きます。
5. 左側のメニューで [ユーザー] をクリックし、[RAM ユーザー管理] ページを開きます。
6. 権限を付与したい RAM ユーザーを選択し、[許可] をクリックします。

7. 表示されたダイアログボックスで、先ほど作成した権限付与ポリシーを選択します。中央の右矢印をクリックしてその権限付与ポリシーを [選択された権限付与ポリシー名] リストに移動させ、[OK] をクリックします。

権限付与が完了すると、RAM ユーザーはこのロールへのアクセス権を持つことになります。 そして、STS を使ってリソースにアクセスするための一時的な認証情報を取得します。

ステップ 5: RAM **ユーザーの一時的な認証情報の取得**

許可された RAM ユーザーは、STS API の呼び出しや STS SDK の使用によって、ロールを適用した一時的な認証情報を取得することができます。 一時的な認証情報には、AccessKeyId、AccessKeySecret、SecurityToken があります。 STS API および STS SDK についての詳細は、「*API リファレンス (STS)*」、「*SDK リファレンス(STS)*」をご参照ください。

STS の API や SDK を使って一時的な認証情報を取得する際には、次のパラメーターを指定する必要があります。

- **RoleArn:** RAM ユーザーに適用するロールの ARN。
- **RoleSessionName:** 一時的な認証情報の名前。 これはカスタムパラメーターです。
- **Policy:** 権限付与ポリシー。 このパラメーターは、ロールの権限に制約を追加します。 このパラメーターを使って、トークンの権限を制限することができます。 このパラメーターを指定しないと、指定したロールの全ての権限を持つトークンが生成されます。
- **DurationSeconds:** 一時的認証情報の有効期間。 このパラメーターは、秒で表されます。 既定値は 3,600 で、900 から 3,600 の間で指定できます。
- **id** と **secret:** RAM ユーザーの AccessKeyId と AccessKeySecret。

一時的な認証情報を取得する例

API の例: RAM ユーザーは、ロールを適用した認証情報を取得するため、STS API の `AssumeRole` を呼び出します。

```
https://sts.aliyuncs.com?Action=AssumeRole
&RoleArn=acs:ram::1234567890123456:role/iotstsrole
&RoleSessionName=iotreadonlyrole
&DurationSeconds=3600
&Policy=<url_encoded_policy>
```

```
&<Common request parameters>
```

SDK の例**: RAM** ユーザーは、**STS** 用の **Python CLI** インターフェイスを介して一時的な認証情報を取得します。

```
$python ./sts.py AssumeRole RoleArn=acs:ram::1234567890123456:role/
iotstsrole RoleSessionName=iotreadonlyrole Policy='{"Version":"1","
Statement":[{"Effect":"Allow","Action":"iot:*","Resource":"*"}]}'
DurationSeconds=3600 --id=id --secret=secret
```

リクエストが受信されると、ロールを適用した一時的な認証情報が返されます。 認証情報には、**AccessKeyId**、**AccessKeySecret**、**SecurityToken** があります。

ステップ 6: RAM ユーザーによるリソースへのアクセス

一時的な認証情報を取得すると、**RAM** ユーザーは特定のロールを適用した認証情報を **SDK** のリクエストに含めて送ることができます。

Java SDK の例**: RAM** ユーザーは、一時的な認証情報に含まれる **AccessKeyId**、**AccessKeySecret**、**SecurityToken** の各パラメーターをリクエストに含めて渡し、**IAcsClient** オブジェクトを作成します。

```
IClientProfile profile = DefaultProfile.getProfile("cn-hangzhou",
AccessKeyId,AccessSecret);
RpcAcsRequest request.putQueryParameter("SecurityToken", Token);
IAcsClient client = new DefaultAcsClient(profile);
AcsResponse response = client.getAcsResponse(request);
```