Alibaba Cloud
Virtual Private Cloud

SDK Reference

Issue: 20191128

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequent**

ial, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

Style	Description	Example
	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 **Danger:** Resetting will result in the loss of user configuration data.
	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 **Notice:** If the weight is set to 0, the server no longer receives new requests.
	A note indicates supplemental instructions, best practices, tips, and other content.	 **Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type.
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands.	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *`Instance_ID`*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`

Style	Description	Example
{} or {a|b}	**This format is used for a required value, where only one item can be selected.**	`switch {active|stand}`

Contents

1 SDK development guide

VPC supports Java, Python, Go, .NET, and PHP SDK development.

The following table provides the download links and development guides for the supported SDKs. For more information, see *Alibaba Cloud SDK platform*.

SDK	GitHub download address	Development guide
Java SDK	*VPC Java SDK*	*Get started*
Python SDK	*VPC Python SDK*	*Get started*
Go SDK	*VPC Go SDK*	*Get started*
.NET SDK	*VPC .NET SDK*	*Get started*
PHP SDK	*VPC PHP SDK*	*Get started*

2 Python SDK examples

2.1 Install Alibaba Cloud SDK for Python

This topic describes how to build a Python development environment on a local PC so you can run VPC Python SDK examples. Alibaba Cloud SDK for Python supports Python 2.7. To run VPC Python SDK examples, you first need to install the core library of Alibaba Cloud SDK for Python and then install VPC Python SDK. Python SDK supports Windows, Linux, and Mac operating systems, and all VPC Python SDK examples can run in a Windows system.

Procedure

Procedure

1. **Install Python.**
   a) *Download Python 2.7*.

   Select and download the correct installation package from the download list. The package format should be `python-XYZ.msi` **and XYZ represents the version number.**

   b) **Double-click the package to enter the Python installation wizard and use the default configurations directly.**

   c) **Set environment variables.**

   Add the python installation path and the directories of the pip program in the `Path` **row. Separate the directories with** ;

   ```
   C:\Python27;C:\Python27\Scripts
   ```

   d) **In the** `cmd` **command line, run the** `python` **command. If the following codes are displayed, you have entered the Python interactive environment and Python has been installed.**

   ```
   Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:30:26) [MSC v.
   1500 64 bit (AM
   D64)] on win32
   Type "help", "copyright", "credits" or "license" for more
   information.
   >>>
   ```

2. **Run the following command to install the core library of Alibaba Cloud SDK for Python.**

```
pip install aliyun-python-sdk-core
```

For more information about Python and PIP, see *Python document* **and** *PIP document*.

3. **Run the following command to install VPC SDK.**

```
pip install aliyun-python-sdk-vpc
```

4. **Run the following command to use** `--user` **to install SDK.**

```
pip install --user aliyun-python-sdk-cms
```

2.2 Preparations

Before you run example VPC Python SDK files, you must complete the required configurations and ensure the following:

1. **There is at least 14.88 USD in your account.**
2. **You have obtained an AccessKey. For more information, see** *#unique_12*.
3. **The** *VPC Python Example library* **is downloaded.**
4. **The** *setup.py* **directory is accessed and the following command is run to initiate the environment.**

```
python setup.py install --user
```

2.3 Virtual Private Cloud (VPC)

2.3.1 Create and delete a VPC/VSwitch

This topic describes how to use the Alibaba Cloud Python SDK to create and delete a VPC and a VSwitch.

Context

Context

In this tutorial, we will create a VPC in the China (Zhangjiakou) region and create a VSwitch under this VPC. Then we will delete the VPC and VSwitch.

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **file.**
2. **Open the** *consts.py* **file in your text editor and configure the** ACS_CLIENT **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\vpc* **folder.**
4. **Open the** *vpc_quick_start.py* **file in your text editor and configure your required parameters. Then, save the configurations and exit the editor.**
5. **Enter the directory of** *vpc_quick_start.py* **and run the following command to run the example that creates and deletes the VPC and VSwitch.**

```
#encoding=utf-8
import sys
import json

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateVpcRequest
from aliyunsdkvpc.request.v20160428 import CreateVSwitchRequest
from aliyunsdkvpc.request.v20160428 import DeleteVSwitchRequest
from aliyunsdkvpc.request.v20160428 import DeleteVpcRequest
from aliyunsdkvpc.request.v20160428 import DescribeVSwitchAttri
butesRequest
from aliyunsdkvpc.request.v20160428 import DescribeVpcAttribute
Request
from sdk_lib.exception import ExceptionHandler
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *
from sdk_lib.common_util import CommonUtil

class VpcQuickStart(object):
    def __init__(self, client):
        self.client = client

    def create_vpc(self):
        """
        create_vpc: Create a VPC
        API reference link: https://www.alibabacloud.com/help/doc-
detail/35737.htm
        """
        try:
            request = CreateVpcRequest.CreateVpcRequest()
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the VPC is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_vpc_status,
                                        AVAILABLE, response_json['
VpcId']):
                return response_json
        except ServerException as e:
```

```
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_vpc(self, params):
        """
        delete_vpc: Delete a VPC
        API reference link: https://www.alibabacloud.com/help/doc-
detail/35738.htm
        """
        try:
            request = DeleteVpcRequest.DeleteVpcRequest()
            # The ID of the VPC to be deleted
            request.set_VpcId(params['vpc_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_vpc_attribute(self, vpc_id):
        """
        describe_vpc_attribute: Query the VPC created in the
specified region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/94565.htm
        """
        try:
            request = DescribeVpcAttributeRequest.DescribeVp
cAttributeRequest()
            # The ID of the VPC to be queried
            request.set_VpcId(vpc_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_vpc_status(self, vpc_id):
        """
        describe_vpc_status: Query the status of the VPC created in
the specified region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/94565.htm
        """
        response = self.describe_vpc_attribute(vpc_id)
        return response["Status"]

    def create_vswitch(self, params):
        """
        create_vswitch: Create a VSwitch instance
        API reference link: https://www.alibabacloud.com/help/doc-
detail/35745.htm
        """
        try:
            request = CreateVSwitchRequest.CreateVSwitchRequest()
            # The ID of the zone to which the VSwitch belongs. To
query the region ID, call the DescribeZones action.
            request.set_ZoneId(params['zone_id'])
            # The ID of the VPC to which the VSwitch belongs.
```

```
            request.set_VpcId(params['vpc_id'])
            # The CIDR block of the VSwitch
            request.set_CidrBlock(params['cidr_block'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the VSwitch is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_vswitch_status
,
                                        AVAILABLE, response_json['
VSwitchId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_vswitch_attribute(self, vswitch_id):
        """
        describe_vswitch_attribute: Query the status of the VSwitch
created in the specified region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/94567.htm
        """
        try:
            request = DescribeVSwitchAttributesRequest.DescribeVS
witchAttributesRequest()
            request.set_VSwitchId(vswitch_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_vswitch_status(self, vswitch_id):
        """
        describe_vswitch_status: Query the status of the VSwitch
created in the specified region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/94567.htm
        """
        response = self.describe_vswitch_attribute(vswitch_id)
        return response["Status"]

    def delete_vswitch(self, params):
        """
        delete_vswitch: Delete a VSwitch instance
       API reference link: https://www.alibabacloud.com/help/doc-
detail/35746.htm
        """
        try:
            request = DeleteVSwitchRequest.DeleteVSwitchRequest()
            # The ID of the VSwitch to be deleted
            request.set_VSwitchId(params['vswitch_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the VSwitch has been deleted
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_vswitch_status
,
```

```
                                            '', params['vswitch_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)


def main():
    client = ACS_CLIENT

    vpc_quick_start = VpcQuickStart(client)

    params = {}
    params['zone_id'] = "cn-zhangjiakou-b"
    params['cidr_block'] = "172.16.0.0/16"

    # Create a vpc
    vpc_json = vpc_quick_start.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    # Create a vswitch
    params['vpc_id'] = vpc_json['VpcId']
    vswitch_json = vpc_quick_start.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)

    # Delete the vswitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vpc_quick_start.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", vswitch_json)

    # Delete the vpc
    vpc_json = vpc_quick_start.delete_vpc(params)
    CommonUtil.log("delete_vpc", vpc_json)


if __name__ == "__main__":
    sys.exit(main())
```

```
python vpc_quick_start.py
```

The system displays the following output:

```
----------------------------create_vpc----------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-8vbf9ud7xrcn9xxxxxxxx",
  "VRouterId": "vrt-8vb1qjnxcm03nxxxxxxxx",
  "VpcId": "vpc-8vb67v4ozd8wfxxxxxxxx",
  "RequestId": "5052F988-75CC-46AD-A1A6-0E9E445BD0D5"
}
----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-8vbqn2at0kljjxxxxxxxx",
  "RequestId": "0BA1ABF7-21CF-4460-9A86-0BB783886E58"
}
----------------------------delete_vswitch----------------------------
{
  "RequestId": "D691F04B-A6EE-49A7-A434-4A45DD3AA0B8"
}
----------------------------delete_vpc----------------------------
{
```

```
  "RequestId": "4570F816-AB8D-45EA-8913-6AE787C1632C"
}
```

2.3.2 Create a custom route table

This topic describes how to use the Alibaba Cloud Python SDK to create a custom route table.

Context

Context

To create a custom route table, the procedure described in this tutorial is as follows:

1. **Create a VPC in the China (Zhangjiakou) region.**
2. **Create a VSwitch under the VPC.**
3. **Create a custom route table named as sdk_route_table.**
4. **Query the created VSwitch.**
5. **Associate the route table with the VSwitch in the same VPC.**
6. **Dissociate the route table from the VSwitch in the same VPC.**
7. **Delete the custom route table.**
8. **Delete the VSwitch.**
9. **Delete the VPC.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\vpc* **folder.**
4. **Open the** *vpc_quick_start.py* **file in your text editor and configure your required parameters. Then, save the configurations and exit the editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateRouteEntryRequest
```

```
from aliyunsdkvpc.request.v20160428 import DeleteRouteEntryRequest
from aliyunsdkvpc.request.v20160428 import AssociateEipAddressR
equest
from aliyunsdkvpc.request.v20160428 import UnassociateEipAddres
sRequest
from aliyunsdkvpc.request.v20160428 import DescribeRouteTablesR
equest
from sdk_lib.common_util import CommonUtil
from sdk_lib.sdk_vpc import Vpc
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.exception import ExceptionHandler
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *

class RouteTable(object):
    def __init__(self, client):
        self.client = client


    def create_route_table(self, params):
        """
        create_route_table:
        API reference link: https://www.alibabacloud.com/help/doc-
detail/87586.htm
        """
        try:
            request = CreateRouteEntryRequest.CreateRouteEntryRequ
est()
            request.set_action_name("CreateRouteTable")
            # The ID of the VPC to which the custom route table
belongs.
            request.add_query_param("VpcId", params['vpc_id'])
            # The name of the route table.
            request.add_query_param("RouteTableName", params['
route_table_name'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            route_table_id = response_json['RouteTableId']
            # Check if the route table is in the available state.
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_route_table
_status,
                                        route_table_id, route_tabl
e_id):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def associate_route_table(self, params):
        """
        associate_route_table: Associate the custom route table with
 the VSwitch in the same VPC
        API reference link: https://www.alibabacloud.com/help/doc-
detail/87599.htm
        """
        try:
            request = AssociateEipAddressRequest.AssociateE
ipAddressRequest()
            request.set_action_name("AssociateRouteTable")
            # The ID of the route table
```

```
            request.add_query_param("RouteTableId", params['
route_table_id'])
            # The ID of the VSwitch
            request.add_query_param("VSwitchId", params['vswitch_id
'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            time.sleep(DEFAULT_TIME * 5)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def unassociate_route_table(self, params):
        """
        unassociate_route_table: Dissociate the route table from the
 VSwitch
        API reference link: https://www.alibabacloud.com/help/doc-
detail/87628.htm
        """
        try:
            request = UnassociateEipAddressRequest.Unassociat
eEipAddressRequest()
            request.set_action_name("UnassociateRouteTable")
            # The ID of the route table
            request.add_query_param("RouteTableId", params['
route_table_id'])
            # The ID of the VSwitch
            request.add_query_param("VSwitchId", params['vswitch_id
'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            time.sleep(DEFAULT_TIME * 5)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_route_table(self, params):
        """
        delete_route_table: Delete the custom route table
        API reference link: https://www.alibabacloud.com/help/doc-
detail/87601.htm
        """
        try:
            request = DeleteRouteEntryRequest.DeleteRouteEntryRequ
est()
            request.set_action_name("DeleteRouteTable")
            # The ID of the route table
            request.add_query_param("RouteTableId", params['
route_table_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the route table has been deleted
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_route_table
_status,
                                        '', params['route_table_id
']):
                return response_json
        except ServerException as e:
```

```
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_route_table(self, route_table_id):
        """
        describe_route_table: Query the route table
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36014.htm
        """
        try:
            request = DescribeRouteTablesRequest.DescribeRo
uteTablesRequest()
            # The ID of the route table
            request.set_RouteTableId(route_table_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_route_table_status(self, route_table_id):
        """
        describe_route_table_status: Query the status of the route
table
        """
        response = self.describe_route_table(route_table_id)
        if len(response["RouteTables"]["RouteTable"]) == 0:
            return ''
        return response["RouteTables"]["RouteTable"][0]["RouteTable
Id"]

def main():
    client = ACS_CLIENT

    vpc = Vpc(client)
    vswitch = VSwitch(client)
    route_table = RouteTable(client)

    params = {}
    params['cidr_block'] = "172.16.0.0/16"
    params['zone_id'] = "cn-zhangjiakou-b"
    params['route_table_name'] = "sdk_route_table"

    #Create a vpc
    vpc_json = vpc.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    #Create a vswitch
    params['vpc_id'] = vpc_json['VpcId']
    vswitch_json = vswitch.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)

    #Create a route table
    route_table_json = route_table.create_route_table(params)
    CommonUtil.log("create_route_table", route_table_json)

    #Query the vswitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vswitch.describe_vswitch_attribute(params['
vswitch_id'])
    CommonUtil.log("describe_vswitch_attribute", vswitch_json)
```

```
    #Associate the route table with the vswitch
    params['route_table_id'] = route_table_json['RouteTableId']
    associate_json = route_table.associate_route_table(params)
    CommonUtil.log("associate_route_table", associate_json)

    #Dissociate the route table from the vswitch
    unassociate_json = route_table.unassociate_route_table(params)
    CommonUtil.log("unassociate_route_table", unassociate_json)

    #Delete the route table
    delete_route_table_json = route_table.delete_route_table(params)
    CommonUtil.log("delete_route_table", delete_route_table_json)

    #Delete the vswitch
    delete_vswitch_json = vswitch.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", delete_vswitch_json)

    #Delete the vpc
    delete_vpc_json = vpc.delete_vpc(params)
    CommonUtil.log("delete_vpc", delete_vpc_json)

if __name__ == "__main__":
    sys.exit(main())
```

5. **Enter the *vpc_route_table.py* directory and run the following command to create the custom route entry.**

```
python vpc_route_table.py
```

The system displays the following output:

```
----------------------------create_vpc----------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-8vb65a5hqy8pcxxxxxxxx",
  "VRouterId": "vrt-8vbbbiftzizc3xxxxxxxx",
  "VpcId": "vpc-8vbebihln001gxxxxxxxx",
  "RequestId": "862F279B-4A27-4300-87A1-047FB9961AF2"
}

----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-8vb30klhn2is5xxxxxxxx",
  "RequestId": "1DA17173-CB61-4DCE-9C29-AABFDF3001A6"
}

----------------------------create_route_table
----------------------------
{
  "RouteTableId": "vtb-8vbc4iwpo13apxxxxxxxx",
  "RequestId": "01E66E67-7801-4705-A02A-853BA7EEA89F"
}

----------------------------describe_vswitch_attribute
----------------------------

{
  "Status": "Available",
  "NetworkAclId": "",
  "VpcId": "vpc-8vbebihln001gxxxxxxxx",
```

```
  "Description": "",
  "RouteTable": {
    "RouteTableId": "vtb-8vb65a5hqy8pcxxxxxxxx",
    "RouteTableType": "System"
  },
  "CidrBlock": "172.16.0.0/16",
  "CreationTime": "2019-04-12T03:08:43Z",
  "CloudResources": {
    "CloudResourceSetType": []
  },
  "ZoneId": "cn-zhangjiakou-b",
  "ResourceGroupId": "rg-acfmxazbxxxxxxxx",
  "VSwitchId": "vsw-8vb30klhn2is5xxxxxxxx",
  "RequestId": "C5A20BA3-E998-498D-8900-35AE5FDFFB77",
  "Ipv6CidrBlock": "",
  "VSwitchName": "",
  "AvailableIpAddressCount": 252,
  "IsDefault": false
}

----------------------------associate_route_table
----------------------------
{
  "RequestId": "5FC0143B-D34B-47DC-8D49-AFD222EA5876"
}

----------------------------unassociate_route_table
----------------------------
{
  "RequestId": "F0194718-6E4C-496C-9DA8-1B88DF1D6FAD"
}

----------------------------delete_route_table
----------------------------
{
  "RequestId": "B5C068A6-137C-4337-8E3A-9E30E1726703"
}

----------------------------delete_vswitch----------------------------
{
  "RequestId": "26DEDBF8-2F0D-4A13-8CB3-23A84C947704"
}

----------------------------delete_vpc----------------------------
{
  "RequestId": "E1B2641F-5911-40E4-9F36-CC0B2EDD1747"
}
```

2.3.3 Create a custom route entry

This topic describes how to use the Alibaba Cloud Python SDK to create a custom route entry on a VRouter or VBR.

Prerequisites

Prerequisites

Before you can create a custom route entry, you need to ensure that:

- **A VPC and a VSwitch are created in the China (Zhangjiakou) region.**

- **The next hop for the custom route entry is created in the China (Zhangjiakou) region. The next hop can be an ECS instance, an HAVIP, a VPN Gateway, a NAT Gateway, an ENI, or a router interface.**

Context

Context

The detailed steps used in this tutorial to create a custom route entry are as follows:

1. **Create a custom route table named sdk_route_table in the China (Zhangjiakou) region.**
2. **Query the VSwitches under the VPC vpc-8vb7ztbjqomi9xxxxxxxx.**
3. **Associate the custom route table with the VSwitch vsw-8vbfqpcijj0d1xxxxxxxx.**
4. **Add a custom route entry to the custom route table. The destination CIDR block of the entry is 168.168.0.0/16, the type of the next hop is ECS instance, and the ID of the next hop is i-8vbgsnt7046axxxxxxxx.**
5. **Delete the route entry.**
6. **Disassociate the custom route table from the VSwitch.**
7. **Delete the custom route table.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\vpc* **folder.**
4. **Open the** *vpc_route_entry.py* **file in your editor and configure your required parameters. Then, save the configurations and exit the editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateRouteEntryRequest
from aliyunsdkvpc.request.v20160428 import DeleteRouteEntryRequest
from aliyunsdkvpc.request.v20160428 import DescribeRouteTablesR
equest
from sdk_lib.exception import ExceptionHandler
```

```
from sdk_lib.check_status import CheckStatus
from sdk_lib.common_util import CommonUtil
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.sdk_route_table import RouteTable
from sdk_lib.consts import *

class RouteEntry(object):
    def __init__(self, client):
        self.client = client

    def create_route_entry(self, params):
        """
        create_route_entry: Create a route entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36012.htm
        """
        try:
            request = CreateRouteEntryRequest.CreateRouteEntryRequ
est()
            # The ID of the route table
            request.set_RouteTableId(params['route_table_id'])
            # The destination CIDR block of the custom route entry
            request.set_DestinationCidrBlock(params['destinatio
n_cidr_block'])
            # The type of the next hop
            request.set_NextHopType(params['nexthop_type'])
            # The ID of the instance used as the next hop
            request.set_NextHopId(params['nexthop_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the route entry is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_route_entry
_status,
                                        AVAILABLE, params['
route_table_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_route_entry(self, params):
        """
        delete_route_entry: Delete the route entry
         API reference link: https://www.alibabacloud.com/help/doc-
detail/36013.htm
        """
        try:
            request = DeleteRouteEntryRequest.DeleteRouteEntryRequ
est()
            # The ID of the route table to which the route entry
belongs
            request.set_RouteTableId(params['route_table_id'])
            # The destination CIDR block of the route entry
            request.set_DestinationCidrBlock(params['destinatio
n_cidr_block'])
            # The ID of the instance used as the next hop
            request.set_NextHopId(params['nexthop_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            time.sleep(DEFAULT_TIME)
            return response_json
```

```
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_route_entry_vrouter(self, params):
        """
        describe_route_entry_vrouter: Query the route entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36014.htm
        """
        try:
            request = DescribeRouteTablesRequest.DescribeRo
uteTablesRequest()
            # The ID of the VRouter or VBR to which the route table
belongs
            request.set_VRouterId(params['vrouter_id'])
            # The ID of the route table
            request.set_RouteTableId(params['route_table_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_route_entry(self, route_table_id):
        """
        describe_route_entry: Query the route entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36014.htm
        """
        try:
            request = DescribeRouteTablesRequest.DescribeRo
uteTablesRequest()
            request.set_RouteTableId(route_table_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_route_entry_status(self, route_table_id):
        """
        describe_route_entry_status: Query the status of the route
entry
        """
        response = self.describe_route_entry(route_table_id)
        return response["RouteTables"]["RouteTable"][0]["RouteEntrys
"]["RouteEntry"][0]["Status"]

def main():
    client = ACS_CLIENT
    vswitch = VSwitch(client)
    route_table = RouteTable(client)
    route_entry = RouteEntry(client)

    params = {}
    params['route_table_name'] = "sdk_route_table"
    params['destination_cidr_block'] = "0.0.0.0/0"
    params['nexthop_id'] = "i-bp1e82xlhob2xxxxxxxx"
```

```
    params['nexthop_type'] = "Instance"

    params['vpc_id'] = "vpc-bp15opprpg0rgxxxxxxxx"
    params['vswitch_id'] = "vsw-bp1e67w26n2sjxxxxxxxx"

    #Create a route table
    route_table_json = route_table.create_route_table(params)
    CommonUtil.log("create_route_table", route_table_json)

    #Query the vswitch
    vswitch_json = vswitch.describe_vswitch_attribute(params)
    CommonUtil.log("describe_vswitch_attribute", vswitch_json)

    #Associate the route table with the vswitch
    params['route_table_id'] = route_table_json['RouteTableId']
    associate_json = route_table.associate_route_table(params)
    CommonUtil.log("associate_route_table", associate_json)

    #Create a route entry
    create_route_entry_json = route_entry.create_route_entry(params)
    CommonUtil.log("create_route_entry", create_route_entry_json)

    #Delete the route entry
    delete_route_entry_json = route_entry.delete_route_entry(params)
    CommonUtil.log("delete_route_entry", delete_route_entry_json)

    #Disassociate the route table from the vswitch
    unassociate_json = route_table.unassociate_route_table(params)
    CommonUtil.log("unassociate_route_table", unassociate_json)

    #Delete the route table
    delete_route_table_json = route_table.delete_route_table(params)
    CommonUtil.log("delete_route_table", delete_route_table_json)


if __name__ == "__main__":
    sys.exit(main())
```

5. **Access the *vpc_route_entry.py* directory and run the following command to create a custom route entry.**

```
python vpc_route_entry.py
```

The system displays the following output:

```
----------------------------create_route_table
----------------------------
{
  "RouteTableId": "vtb-8vbn7px9zxwr2xxxxxxxx",
  "RequestId": "8B351EE1-614F-44E4-93AF-1CADA4BF02E8"
}

----------------------------describe_vswitch_attribute
----------------------------

{
  "Status": "",
  "NetworkAclId": "",
  "VpcId": "",
  "Description": "",
  "Ipv6CidrBlock": "",
```

```
  "CreationTime": "",
  "CloudResources": {
    "CloudResourceSetType": []
  },
  "ZoneId": "",
  "ResourceGroupId": "",
  "VSwitchId": "",
  "RequestId": "5E199415-BBA3-443D-B1EC-06341FE267F4",
  "VSwitchName": "",
  "CidrBlock": ""
}

----------------------------associate_route_table
----------------------------
{
  "RequestId": "5F33E444-5CCD-4677-91AB-3E234A9A64E4"
}

----------------------------create_route_entry
----------------------------
{
  "RequestId": "D6035ECA-DD81-4FAB-B084-55BE60FB18ED"
}

----------------------------delete_route_entry
----------------------------
{
  "RequestId": "54108FD7-8609-4111-919D-B2983466F480"
}

----------------------------unassociate_route_table
----------------------------
{
  "RequestId": "0F36A76A-1E54-41DC-852E-1D970FDE8F3F"
}

----------------------------delete_route_table
----------------------------
{
  "RequestId": "F3151A59-4F90-4531-AFDC-B7B7CF70A8C1"
}
```

2.4 Elastic IP (EIP)

2.4.1 Attach an EIP to an ECS instance

This topic describes how to use the Alibaba Cloud Python SDK to attach an EIP to an ECS instance.

Prerequisites

Prerequisites

Before you attach an EIP to an ECS instance, you need to ensure that:

- **The ECS instance is of the VPC network type.**
- **The ECS instance and the EIP are locate in the same region.**

- The ECS instance is in the running or stopped status.
- The ECS instance is not attached to a public IP address or another EIP.

Context

Context

The detailed steps used in this tutorial to attach an EIP to an ECS instance are as follows:

1. Create an EIP in the China (Hangzhou) region.
2. Attach the EIP to the ECS instance.
3. Query EIPs attached to the ECS instance.
4. Modify the peak bandwidth and name of the EIP.
5. Query the modified EIP.
6. Disassociate an EIP from an ECS instance.
7. Release the EIP.

Procedure

Procedure

1. In the downloaded SDK directory, open the *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* folder.
2. Open the *consts.py* file in your text editor and configure the `ACS_CLIENT` parameter used for user authentication.
3. In the downloaded SDK directory, open the *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\eip* folder.
4. Open the *eip_quick_start.py* file in your editor and configure your required parameters. Then, save the configurations and exit the editor.

```
#encoding=utf-8
import sys
import json
from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import AllocateEipAddressRequest
from aliyunsdkvpc.request.v20160428 import AssociateEipAddressR
equest
from aliyunsdkvpc.request.v20160428 import DescribeEipAddresses
Request
from aliyunsdkvpc.request.v20160428 import UnassociateEipAddres
sRequest
from aliyunsdkvpc.request.v20160428 import ModifyEipAddressAttr
ibuteRequest
from aliyunsdkvpc.request.v20160428 import ReleaseEipAddressRequest
from sdk_lib.exception import ExceptionHandler
from sdk_lib.check_status import CheckStatus
```

```
from sdk_lib.consts import *
from sdk_lib.common_util import CommonUtil

"""
Create an EIP->Attach the EIP to an ECS instance->Query EIPs->Modify
 the configurations and name of the EIP->Query the EIP->Disassociate
 the EIP->Release the EIP
"""
class Eip(object):
    def __init__(self, client):
        self.client = client

    def allocate_eip_address(self, params):
        """
        allocate_eip_address: Apply for an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36016.htm
        """
        try:
            request = AllocateEipAddressRequest.AllocateEipAddressRe
quest()
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, response_json["
AllocationId"]):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def associate_eip_address(self, params):
        """
        associate_eip_address: Attach the EIP to a cloud product
instance in the same region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36017.htm
        """
        try:
            request = AssociateEipAddressRequest.AssociateE
ipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the cloud product instance to attach
            request.set_InstanceType(params['instance_type'])
            # The ID of the instance to attach
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        InUse, params['allocation_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_address(self, allocation_id):
```

```
        """
        describe_eip_status: Query created EIPs in a region.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        try:
            request = DescribeEipAddressesRequest.DescribeEi
pAddressesRequest()
            # The ID of the EIP
            request.set_AllocationId(allocation_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_status(self, allocation_id):
        """
        describe_eip_status: Query the status of created EIPs in a
region.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        # The ID of the EIP
        response = self.describe_eip_address(allocation_id)
        return response["EipAddresses"]["EipAddress"][0]["Status"]


    def unassociate_eip_address(self, params):
        """
        unassociate_eip_address: Disassociate an EIP from a cloud
resource.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36021.htm
        """
        try:
            request = UnassociateEipAddressRequest.Unassociat
eEipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the resource to disassociate
            request.set_InstanceType(params['instance_type'])
            # The ID of the cloud product instance to disassociate
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, params['
allocation_id']):
                return response_json
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def modify_eip_address(self, params):
        """
        modify_eip_address: Modify the name, description, and peak
bandwidth of an EIP
```

```
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36019.htm
        """
        try:
            request = ModifyEipAddressAttributeRequest.ModifyEipA
ddressAttributeRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The peak bandwidth in Mbps of the EIP
            request.set_Bandwidth(params['bandwidth'])
            # The name of the EIP
            request.set_Name(params['name'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def release_eip_address(self, params):
        """
        release_eip_address: Release an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36020.htm
        """
        try:
            request = ReleaseEipAddressRequest.ReleaseEipAddressReq
uest()
            # The ID of the EIP to release
            request.set_AllocationId(params['allocation_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

def main():

    client = ACS_CLIENT
    eip = Eip(client)

    params = {}

    # Create an EIP
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)

    # Attach the EIP to an ECS instance
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['instance_id'] = ECS_INSTANCE_ID
    params['instance_type'] = 'EcsInstance'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address", eip_response_json)

    # Query EIPs
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Modify the configurations and name of an EIP
    params['bandwidth'] = BANDWIDTH_50
```

```
    params['name'] = EIP_NEW_NAME
    eip_response_json = eip.modify_eip_address(params)
    CommonUtil.log("modify_eip_address", eip_response_json)

    # Query the EIP
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Disassociate the EIP
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address", eip_response_json)

    # Release the EIP
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

if __name__ == '__main__':
    sys.exit(main())
```

5. **Access the *eip_quick_start.py* directory and run the following command to attach the EIP to the ECS instance.**

```
python eip_quick_start.py
```

The system displays the following output:

```
--------------------------allocate_eip_address
--------------------------
{
  "EipAddress": "47.xx.xx.23",
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "RequestId": "C438312E-F7A4-4A04-901F-D22FE23EDB4D",
  "AllocationId": "eip-bp1wybucvhhx5xxxxxxxx"
}
--------------------------associate_eip_address
--------------------------
{
  "RequestId": "6EC6605E-3D2B-4EE8-BD13-F1964CD1EAB1"
}
--------------------------describe_eip_address
--------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "47.xx.xx.23",
        "AllocationId": "eip-bp1wybucvhhx5xxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "InUse",
        "BandwidthPackageId": "",
        "InstanceId": "i-bp1e82xlhob2xxxxxxxx",
        "InstanceRegionId": "cn-hangzhou",
        "RegionId": "cn-hangzhou",
        "AvailableRegions": {
```

```
          "AvailableRegion": [
            "cn-hangzhou"
          ]
        },
        "ResourceGroupId": "rg-acfm4odxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "EcsInstance",
        "AllocationTime": "2019-04-17T11:57:43Z",
        "Name": "",
        "OperationLocks": {
          "LockReason": []
        },
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "5",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "8715A878-A808-4CC4-AAD5-E414FDAB5B0E"
}
---------------------------modify_eip_address
---------------------------
{
  "RequestId": "2108AE1C-94FB-475D-BFEE-EC88598BF6A6"
}
---------------------------describe_eip_address
---------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "47.xx.xx.23",
        "AllocationId": "eip-bp1wybucvhhx5xxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "InUse",
        "BandwidthPackageId": "",
        "InstanceId": "i-bp1e82xlhob2xxxxxxxx",
        "InstanceRegionId": "cn-hangzhou",
        "RegionId": "cn-hangzhou",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-hangzhou"
          ]
        },
        "ResourceGroupId": "rg-acfm4odxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "EcsInstance",
        "AllocationTime": "2019-04-17T11:57:43Z",
        "Name": "EIP_NEW_NAME",
        "OperationLocks": {
          "LockReason": []
        },
        "Mode": "NAT",
```

```
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "50",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "6694D35B-B5DD-4506-8AB1-2D16477646DE"
}
----------------------------unassociate_eip_address
----------------------------
{
  "RequestId": "EDE86CF6-EE68-4922-B919-85A4F11BF668"
}
----------------------------release_eip_address
----------------------------
{
  "RequestId": "53FEE062-B595-4D64-AB47-834015D32888"
}
```

2.4.2 Add an EIP to an Internet Shared Bandwidth

This topic describes how to use the Alibaba Cloud Python SDK to add an EIP to an Internet Shared Bandwidth.

Context

Context

The detailed steps used in this tutorial to add an EIP to an Internet Shared Bandwidth are as follows:

1. **Create an EIP in the China (Hangzhou) region.**
2. **Create an Internet Shared Bandwidth in the China (Hangzhou) region.**
3. **Add the EIP to the Internet Shared Bandwidth.**
4. **Query the Internet Shared Bandwidth.**
5. **Remove the EIP from the Internet Shared Bandwidth.**
6. **Delete the Internet Shared Bandwidth.**
7. **Release the EIP.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for authentication.**

3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\eip* **folder.**
4. **Open the** *eip_add_cbwp.py* **file in your editor and configure your required parameters. Then, save the configurations and exit the editor.**

```
#encoding=utf-8
import sys
from aliyunsdkvpc.request.v20160428 import CreateCommonBandwidt
hPackageRequest
from aliyunsdkvpc.request.v20160428 import AddCommonBandwidthPa
ckageIpRequest
from aliyunsdkvpc.request.v20160428 import DescribeCommonBandwi
dthPackagesRequest
from aliyunsdkvpc.request.v20160428 import RemoveCommonBandwidt
hPackageIpRequest
from aliyunsdkvpc.request.v20160428 import DeleteCommonBandwidt
hPackageRequest
from sdk_lib.common_util import CommonUtil
from sdk_lib.sdk_eip import *

"""
Create an EIP->Create an Internet Shared Bandwidth->The ID of the
Internet Shared Bandwidth->Add the EIP
to the Internet Shared Bandwidth->Query the Internet Shared
Bandwidth->Remove the EIP from the
Internet Shared Bandwidth->Delete the Internet Shared Bandwidth->
Release the EIP
"""
class CommonBandwidthPackage(object):
    def __init__(self, client):
        self.client = client

    def create_common_bandwidth_package(self, params):
        """
        create_common_bandwidth_package: Create an Internet Shared
Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/55930.htm
        """
        try:
            request = CreateCommonBandwidthPackageRequest.CreateComm
onBandwidthPackageRequest()
            # The peak bandwidth in Mbps of the Internet Shared
Bandwidth
            request.set_Bandwidth(params['bandwidth'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_cbwp_status,
                                        AVAILABLE, response_json["
BandwidthPackageId"]):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def add_common_bandwidth_packageIp(self, params):
        """
```

```
        add_common_bandwidth_packageIp: Add the EIP to the Internet
Shared Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/55989.htm
        """
        try:
            request = AddCommonBandwidthPackageIpRequest.AddCommonB
andwidthPackageIpRequest()
            # The ID of the EIP instance
            request.set_IpInstanceId(params['ip_instance_id'])
            # The ID of the Internet Shared Bandwidth
            request.set_BandwidthPackageId(params['bandwidth_
package_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_cbwp(self, cbwp_id):
        """
        describe_cbwp: Query the Internet Shared Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/55997.htm
        """
        try:
            request = DescribeCommonBandwidthPackagesRequest.
DescribeCommonBandwidthPackagesRequest()
            # The ID of the Internet Shared Bandwidth
            request.set_BandwidthPackageId(cbwp_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_cbwp_status(self, cbwp_id):
        """
        describe_cbwp: Query the status of the Internet Shared
Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/55997.htm
        """
        # The ID of the Internet Shared Bandwidth
        response = self.describe_cbwp(cbwp_id)
        return response["CommonBandwidthPackages"]["CommonBand
widthPackage"][0]["Status"]


    def remove_common_bandwidth_packageIp(self, params):
        """
        remove_common_bandwidth_packageIp: Remove the EIP from the
Internet Shared Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/55995.htm
        """
        try:
            request = RemoveCommonBandwidthPackageIpRequest.
RemoveCommonBandwidthPackageIpRequest()
            # The ID of the EIP
```

```
            request.set_IpInstanceId(params['ip_instance_id'])
            # The ID of the Internet Shared Bandwidth
            request.set_BandwidthPackageId(params['bandwidth_
package_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)


    def delete_common_bandwidth_package(self, params):
        """
        delete_common_bandwidth_package: Delete the Internet Shared
Bandwidth
        API reference link: https://www.alibabacloud.com/help/doc-
detail/56000.htm
        """
        try:
            request = DeleteCommonBandwidthPackageRequest.DeleteComm
onBandwidthPackageRequest()
            # The ID of the Internet Shared Bandwidth
            request.set_BandwidthPackageId(params['bandwidth_
package_id'])
            # Indicate whether to forcibly delete the Internet
Shared Bandwidth
            request.set_Force(params['force'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

def main():

    client = ACS_CLIENT
    eip = Eip(client)
    cbwp = CommonBandwidthPackage(client)

    params = {}

    # Create an EIP
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)
    params['allocation_id'] = eip_response_json["AllocationId"]

    # Create an Internet Shared Bandwidth
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['ip_instance_id'] = eip_response_json["AllocationId"]
    params['bandwidth'] = BANDWIDTH_10
    cbwp_repsonse_json = cbwp.create_common_bandwidth_package(params
)
    CommonUtil.log("create_common_bandwidth_package", cbwp_repso
nse_json)

    # Add the EIP to the Internet Shared Bandwidth
    params['bandwidth_package_id'] = cbwp_repsonse_json['BandwidthP
ackageId']
    cbwp_repsonse_json = cbwp.add_common_bandwidth_packageIp(params)
```

```
    CommonUtil.log("add_common_bandwidth_packageIp", cbwp_repso
nse_json)

    # Query the Internet Shared Bandwidth
    cbwp_repsonse_json = cbwp.describe_cbwp(params['bandwidth_
package_id'])
    CommonUtil.log("add_common_bandwidth_packageIp", cbwp_repso
nse_json)

    # Remove the EIP from the Internet Shared Bandwidth
    cbwp_repsonse_json = cbwp.remove_common_bandwidth_packageIp(
params)
    CommonUtil.log("remove_common_bandwidth_packageIp", cbwp_repso
nse_json)

    # Delete the Internet Shared Bandwidth
    params['force'] = True
    cbwp_repsonse_json = cbwp.delete_common_bandwidth_package(params
)
    CommonUtil.log("delete_common_bandwidth_package", cbwp_repso
nse_json)

    # Release the EIP
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

if __name__ == '__main__':
    sys.exit(main())
```

5. **Access the *eip_add_cbwp.py* directory and run the following command to add the EIP to the Internet Shared Bandwidth.**

```
python eip_add_cbwp.py
```

The system displays the following output:

```
----------------------------allocate_eip_address
----------------------------
{
  "EipAddress": "118.xx.xx.198",
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "RequestId": "A830A607-B7C4-49FE-A6EE-7237D64CDE2D",
  "AllocationId": "eip-bp1mdyvr22qvgxxxxxxxx"
}
----------------------------create_common_bandwidth_package
----------------------
-----
{
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "BandwidthPackageId": "cbwp-bp12k058pjjiexxxxxxxx",
  "RequestId": "93127320-DD79-4F83-A3B9-DC99D0597B0C"
}
----------------------------add_common_bandwidth_packageIp
-----------------------
----
{
  "RequestId": "7F314AFE-B398-4348-AF61-B7D27B731286"
}
----------------------------add_common_bandwidth_packageIp
-----------------------
----
```

```
{
  "TotalCount": 1,
  "CommonBandwidthPackages": {
    "CommonBandwidthPackage": [
      {
        "Status": "Available",
        "PublicIpAddresses": {
          "PublicIpAddresse": [
            {
              "IpAddress": "118.xx.xx.198",
              "AllocationId": "eip-bp1mdyvr22qvgxxxxxxxx"
            }
          ]
        },
        "BusinessStatus": "Normal",
        "RegionId": "cn-hangzhou",
        "BandwidthPackageId": "cbwp-bp12k058pjjiexxxxxxxx",
        "Name": "",
        "ISP": "BGP",
        "CreationTime": "2019-04-18T01:46:17Z",
        "ResourceGroupId": "rg-acfm4odxxxxxxxx",
        "Bandwidth": "10",
        "InstanceChargeType": "PostPaid",
        "HasReservationData": false,
        "InternetChargeType": "PayByBandwidth",
        "ExpiredTime": "",
        "Ratio": 100,
        "Description": ""
      }
    ]
  },
  "PageNumber": 1,
  "RequestId": "015DD0FA-742B-4431-92EA-E3F03FDEB8CD",
  "PageSize": 10
}
----------------------------remove_common_bandwidth_packageIp
--------------------
-------
{
  "RequestId": "A49C9126-B703-4D34-B552-A7FE283FB5DD"
}
---------------------------delete_common_bandwidth_package
----------------------
-----
{
  "RequestId": "E423F648-C169-4B63-A2CF-5E6C8E441DE1"
}
----------------------------release_eip_address
--------------------------
{
  "RequestId": "7E0D34AE-58C3-468A-B021-378F8938AE6B"
}
```


2.4.3 Modify the peak bandwidth of an EIP

This topic describes how to use the Alibaba Cloud Python SDK to modify the peak bandwidth of an EIP.

Context

Context

The example used in the procedure of this topic includes the following steps:

1. **Create an EIP in the China (Hangzhou) region.**
2. **Modify the peak bandwidth of the EIP to 50 Mbit/s.**
3. **Query the modified EIP.**
4. **Modify the peak bandwidth of the EIP to 10 Mbit/s.**
5. **Query the modified EIP.**
6. **Release the EIP.**

Procedure

Procedure

1. **In the SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and set the** `ACS_CLIENT` **parameter for user authentication.**
3. **In the SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\eip* **folder.**
4. **Open the** *eip_modify_attribute.py* **file in your text editor, set the parameters as needed, and then save the settings and exit the editor.**

```
#encoding=utf-8
import sys
import json
from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import AllocateEipAddressRequest
from aliyunsdkvpc.request.v20160428 import AssociateEipAddressR
equest
from aliyunsdkvpc.request.v20160428 import DescribeEipAddresses
Request
from aliyunsdkvpc.request.v20160428 import UnassociateEipAddres
sRequest
from aliyunsdkvpc.request.v20160428 import ModifyEipAddressAttr
ibuteRequest
from aliyunsdkvpc.request.v20160428 import ReleaseEipAddressRequest
from sdk_lib.exception import ExceptionHandler
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *
from sdk_lib.common_util import CommonUtil

"""
Create an EIP -> Modify the peak bandwidth of the EIP to 50 Mbit/s -
> Query the EIP -> Modify the peak bandwidth of the EIP to 10 Mbit/s
 -> Query the EIP -> Release the EIP
"""
class Eip(object):
    def __init__(self, client):
        self.client = client
```

```
    def allocate_eip_address(self, params):
        """
        allocate_eip_address: Create an EIP.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36016.htm
        """
        try:
            request = AllocateEipAddressRequest.AllocateEipAddressRe
quest()
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, response_json["
AllocationId"]):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def associate_eip_address(self, params):
        """
        associate_eip_address: Associate the EIP with an instance in
 the same region.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36017.htm
        """
        try:
            request = AssociateEipAddressRequest.AssociateE
ipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the instance with which the EIP is
associated
            request.set_InstanceType(params['instance_type'])
            # The ID of the instance with which the EIP is
associated
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        InUse, params['allocation_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_address(self, allocation_id):
        """
        describe_eip_status: Query the EIP created in the specified
region.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        try:
            request = DescribeEipAddressesRequest.DescribeEi
pAddressesRequest()
            # The ID of the EIP
```

```
            request.set_AllocationId(allocation_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_status(self, allocation_id):
        """
        describe_eip_status: Query the status of the EIP created in
the specified region.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        # The ID of the EIP
        response = self.describe_eip_address(allocation_id)
        return response["EipAddresses"]["EipAddress"][0]["Status"]


    def unassociate_eip_address(self, params):
        """
        unassociate_eip_address: Disassociate the EIP from the
instance.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36021.htm
        """
        try:
            request = UnassociateEipAddressRequest.Unassociat
eEipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the instance from which the EIP is
disassociated
            request.set_InstanceType(params['instance_type'])
            # The ID of the instance from which the EIP is
disassociated
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, params['
allocation_id']):
                return response_json
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def modify_eip_address(self, params):
        """
        modify_eip_address: Modify the name, description, and peak
bandwidth of the specified EIP.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36019.htm
        """
        try:
            request = ModifyEipAddressAttributeRequest.ModifyEipA
ddressAttributeRequest()
            # The ID of the EIP
```

```
            request.set_AllocationId(params['allocation_id'])
            # The peak bandwidth (Mbit/s) of the EIP
            request.set_Bandwidth(params['bandwidth'])
            # The name of the EIP
            request.set_Name(params['name'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def release_eip_address(self, params):
        """
        release_eip_address: Release the specified EIP.
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36020.htm
        """
        try:
            request = ReleaseEipAddressRequest.ReleaseEipAddressReq
uest()
            # The ID of the EIP to be released
            request.set_AllocationId(params['allocation_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

def main():

    client = ACS_CLIENT
    eip = Eip(client)

    params = {}

    # Create an EIP.
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)

    params['allocation_id'] = eip_response_json["AllocationId"]

    # Modify the peak bandwidth of the EIP to 50 Mbit/s.
    params['name'] = EIP_NEW_NAME
    params['bandwidth'] = BANDWIDTH_50
    eip_response_json = eip.modify_eip_address(params)
    CommonUtil.log("modify_eip_address", eip_response_json)

    # Query the EIP.
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Modify the peak bandwidth of the EIP to 10 Mbit/s.
    params['bandwidth'] = BANDWIDTH_10
    eip_response_json = eip.modify_eip_address(params)
    CommonUtil.log("modify_eip_address", eip_response_json)

    # Query the EIP.
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
```

```
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Release the EIP.
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

if __name__ == '__main__':
    sys.exit(main())
```

5. **Enter the directory where the *eip_modify_attribute.py* file is located, and then run the following command to modify the peak bandwidth of the EIP.**

```
python eip_modify_attribute.py
```

The system output is displayed as follows:

```
----------------------------allocate_eip_address
----------------------------
{
  "EipAddress": "47.xx.xx.225",
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "RequestId": "9318DD7A-F065-4EA6-9EA0-20A9C46EDADC",
  "AllocationId": "eip-bp15bzjk5djcsxxxxxxxx"
}
----------------------------modify_eip_address
----------------------------
{
  "RequestId": "C39D55A1-6B47-489B-8614-FDB9736EDE73"
}
----------------------------describe_eip_address
----------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "47.xx.xx.225",
        "AllocationId": "eip-bp15bzjk5djcsxxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "Available",
        "BandwidthPackageId": "",
        "InstanceId": "",
        "InstanceRegionId": "",
        "RegionId": "cn-hangzhou",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-hangzhou"
          ]
        },
        "ResourceGroupId": "rg-acfm4odxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "",
        "AllocationTime": "2019-04-18T04:01:28Z",
        "Name": "EIP_NEW_NAME",
        "OperationLocks": {
          "LockReason": []
```

```
        },
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "50",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "51ECAB45-2518-4A46-89DC-8ADEE1AFDBE9"
}
--------------------------modify_eip_address
--------------------------
{
  "RequestId": "ACAB5724-D05C-46A4-8C2B-6064AEEC792B"
}
--------------------------describe_eip_address
--------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "47.xx.xx.225",
        "AllocationId": "eip-bp15bzjk5djcsxxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "Available",
        "BandwidthPackageId": "",
        "InstanceId": "",
        "InstanceRegionId": "",
        "RegionId": "cn-hangzhou",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-hangzhou"
          ]
        },
        "ResourceGroupId": "rg-acfm4odxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "",
        "AllocationTime": "2019-04-18T04:01:28Z",
        "Name": "EIP_NEW_NAME",
        "OperationLocks": {
          "LockReason": []
        },
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "10",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "6F653A80-AB28-4842-84A8-CD444EB81A29"
```

```
}
----------------------------release_eip_address
----------------------------
{
  "RequestId": "C407633A-5658-482F-AB5E-069028C3B06C"
}
```

2.4.4 Attach an EIP to an SLB instance or an ENI

This topic describes how to use the Alibaba Cloud Python SDK to attach an EIP to an SLB instance or an ENI.

Prerequisites

Prerequisites

Before you attach an EIP to an SLB instance or an ENI, you must ensure that a VPC and a VSwitch are created in the China (Hangzhou) region and an ENI is created under the VPC and the VSwitch.

Context

Context

The detailed steps used in this tutorial to attach an EIP to an SLB instance/ENI are as follows:

1. **Create an SLB instance in the China (Hangzhou) region.**
2. **Create an EIP in the China (Hangzhou) region.**
3. **Attach the EIP to the SLB instance.**
4. **Detach the EIP from the SLB instance.**
5. **Attach the detached EIP to the ENI.**
6. **Detach the EIP from the ENI.**
7. **Delete the SLB instance.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\eip* **folder.**

4. **Open the *eip_associate_slb.py* file in your editor and configure your required parameters. Then, save the configurations and exit the editor.**

```
#encoding=utf-8
import sys
import json
from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import AllocateEipAddressRequest
from aliyunsdkvpc.request.v20160428 import AssociateEipAddressR
equest
from aliyunsdkvpc.request.v20160428 import DescribeEipAddresses
Request
from aliyunsdkvpc.request.v20160428 import UnassociateEipAddres
sRequest
from aliyunsdkvpc.request.v20160428 import ModifyEipAddressAttr
ibuteRequest
from aliyunsdkvpc.request.v20160428 import ReleaseEipAddressRequest
from sdk_lib.exception import ExceptionHandler
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *
from sdk_lib.common_util import CommonUtil
from sdk_lib.sdk_load_balancer import LoadBalancer

"""
Create a VPC->Create a VSwitch->Call an ECS API action to generate
an ENI (these steps are not included
in the following codes)
Create an SLB instance->Create an EIP->Attach the EIP to the SLB
instance->Detach the EIP from the SLB instance
->Attach the EIP to the ENI->Detach the EIP from the ENI->Delete the
 SLB instance
"""
class Eip(object):
    def __init__(self, client):
        self.client = client

    def allocate_eip_address(self, params):
        """
        allocate_eip_address: Create an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36016.htm
        """
        try:
            request = AllocateEipAddressRequest.AllocateEipAddressRe
quest()
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, response_json["
AllocationId"]):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def associate_eip_address(self, params):
        """
```

```
        associate_eip_address: Attach the EIP to an cloud product
instance in the same region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36017.htm
        """
        try:
            request = AssociateEipAddressRequest.AssociateE
ipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the cloud product instance to attach
            request.set_InstanceType(params['instance_type'])
            # The ID of the instance to attach
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        InUse, params['allocation_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_address(self, allocation_id):
        """
        describe_eip_status: Query one or more EIPs in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        try:
            request = DescribeEipAddressesRequest.DescribeEi
pAddressesRequest()
            # The ID of the EIP
            request.set_AllocationId(allocation_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_eip_status(self, allocation_id):
        """
        describe_eip_status: Query the status of an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36018.htm
        """
        # The ID of the EIP
        response = self.describe_eip_address(allocation_id)
        return response["EipAddresses"]["EipAddress"][0]["Status"]


    def unassociate_eip_address(self, params):
        """
        unassociate_eip_address: Detach the EIP from the cloud
resource
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36021.htm
        """
```

```
        try:
            request = UnassociateEipAddressRequest.Unassociat
eEipAddressRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The type of the resource to detach
            request.set_InstanceType(params['instance_type'])
            # The ID of the cloud product instance to detach
            request.set_InstanceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_eip_status,
                                        AVAILABLE, params['
allocation_id']):
                return response_json
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def modify_eip_address(self, params):
        """
        modify_eip_address: Modify the name, description and peak
bandwidth of an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36019.htm
        """
        try:
            request = ModifyEipAddressAttributeRequest.ModifyEipA
ddressAttributeRequest()
            # The ID of the EIP
            request.set_AllocationId(params['allocation_id'])
            # The peak bandwidth in Mbps of the EIP
            request.set_Bandwidth(params['bandwidth'])
            # The name of the EIP
            request.set_Name(params['name'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def release_eip_address(self, params):
        """
        release_eip_address: Release an EIP
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36020.htm
        """
        try:
            request = ReleaseEipAddressRequest.ReleaseEipAddressReq
uest()
            # The ID of the EIP to release
            request.set_AllocationId(params['allocation_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
```

```
            ExceptionHandler.client_exception(e)
def main():

    client = ACS_CLIENT
    eip = Eip(client)
    load_balancer = LoadBalancer(client)
    #Create a VPC, create a VSwitch, and call an ECS API action to
generate an ENI
    params = {}
    params['vpc_id'] = VPC_ID
    params['vswitch_id'] = VSWITCH_ID

    #Create an SLB instance
    load_balancer_repsonse_json = load_balancer.create_load_balancer
_private(params)
    CommonUtil.log("create_load_balancer", load_balancer_repson
se_json)

    #Create an EIP
    params['load_balancer_id'] = load_balancer_repsonse_json['
LoadBalancerId']
    params['instance_id'] = load_balancer_repsonse_json['LoadBalanc
erId']
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address slb", eip_response_json)

    # Attach the EIP to the SLB instance
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['instance_type'] = 'SlbInstance'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address eip", eip_response_json)

    # Detach the EIP from the SLB instance
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address slb", eip_response_json)

    # Attach the EIP to the ENI
    params['instance_id'] = ENI_ID
    params['instance_type'] = 'NetworkInterface'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address eni", eip_response_json)

    # Detach the EIP from the ENI
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address eni", eip_response_json)

    # Delete the SLB instance
    load_balancer_repsonse_json = load_balancer.delete_load_balancer
(params)
    CommonUtil.log("delete_load_balancer", load_balancer_repson
se_json)

if __name__ == '__main__':
```

```
    sys.exit(main())
```

5. **Access the *eip_associate_slb.py* directory and run the following command to attach the EIP to the SLB instance or the ENI.**

```
Python eip_associate_slb.py
```

The system displays the following output:

```
----------------------------create_load_balancer
----------------------------
{
  "VpcId": "vpc-bp15opprpg0rgxxxxxxxx",
  "AddressIPVersion": "ipv4",
  "LoadBalancerName": "auto_named_slb",
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "VSwitchId": "vsw-bp1e67w26n2sjxxxxxxxx",
  "RequestId": "D3651A96-008C-4B35-A36E-54C2902535C5",
  "Address": "172.xx.xx.146",
  "NetworkType": "vpc",
  "LoadBalancerId": "lb-bp15u6kumammdxxxxxxxx"
}
----------------------------allocate_eip_address slb
----------------------------
{
  "EipAddress": "47.xx.xx.76",
  "ResourceGroupId": "rg-acfm4odxxxxxxxx",
  "RequestId": "15FD58CD-B186-4E2C-B4B3-74F712168832",
  "AllocationId": "eip-bp1ofhmmep6rkxxxxxxxx"
}
----------------------------associate_eip_address eip
----------------------------
{
  "RequestId": "5EDABF0B-A067-474E-9556-0A3AA870960A"
}
----------------------------unassociate_eip_address slb
----------------------------
-
{
  "RequestId": "89556510-D726-490A-9092-9BEA0644CC43"
}
----------------------------associate_eip_address eni
----------------------------
{
  "RequestId": "FAE87FDD-232A-4859-803B-F9B57508AEDC"
}
----------------------------unassociate_eip_address eni
----------------------------
-
{
  "RequestId": "7DF556E8-12BE-481A-B83D-B3D9E8836534"
}
----------------------------delete_load_balancer
----------------------------
{
  "RequestId": "2B70C01A-A440-40A5-A98B-83A687537CCE"
```

```
}
```

2.5 NAT Gateway

2.5.1 Create a NAT Gateway

This topic describes how to use the Alibaba Cloud Python SDK to create a NAT Gateway.

Context

Context

The detailed steps used in this tutorial to create a NAT Gateway are as follows:

1. **Create a VPC in the China (Shanghai) region.**
2. **Create a VSwitch under the VPC.**
3. **Create a NAT Gateway under the VPC.**
4. **Query the NAT Gateway.**
5. **Delete the NAT Gateway.**
6. **Delete the VSwitch.**
7. **Delete the VPC.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\natgw* **folder.**
4. **Open the** *natgw_quick_start.py* **file in your text editor and configure your required parameters. Then, save the configurations and exit the text editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DeleteNatGatewayRequest
```

```
from aliyunsdkvpc.request.v20160428 import DescribeNatGatewaysR
equest
from sdk_lib.sdk_vpc import Vpc
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.exception import ExceptionHandler
from sdk_lib.consts import *

client = ACS_CLIENT


class NatGateway(object):
    def __init__(self, client):
        self.client = client

    def create_nat_gateway(self, params):
        """
        create_nat_gateway: Create a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36048.htm
        """
        try:
            request = CreateNatGatewayRequest.CreateNatGatewayRequ
est()
            request.set_VpcId(params['vpc_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_nat_gateway
_status,
                                        AVAILABLE, response_json['
NatGatewayId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway(self, nat_gateway_id):
        """
        describe_nat_gateway: Query NAT Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        try:
            request = DescribeNatGatewaysRequest.DescribeNa
tGatewaysRequest()
            request.set_NatGatewayId(nat_gateway_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_nat_gateway(self, params):
        """
        delete_nat_gateway: Delete a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36051.htm
```

```
        """
        try:
            request = DeleteNatGatewayRequest.DeleteNatGatewayRequ
est()
            request.set_NatGatewayId(params['nat_gateway_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_nat_gateway
_status,
                                        '', params['nat_gateway_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway_status(self, nat_gateway_id):
        """
        describe_nat_gateway_status: Query the status of NAT
Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        response = self.describe_nat_gateway(nat_gateway_id)
        if len(response["NatGateways"]["NatGateway"]) == 0:
            return ''
        return response["NatGateways"]["NatGateway"][0]['Status']


def main():
    vpc = Vpc(client)
    vswitch = VSwitch(client)
    nat_gateway = NatGateway(client)

    params = {}

    # Create a VPC
    vpc_json = vpc.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    # Create a VSwitch
    params['vpc_id'] = vpc_json['VpcId']
    params['zone_id'] = "cn-shanghai-b"
    params['cidr_block'] = "172.16.1.0/24"
    vswitch_json = vswitch.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)

    # Create a NAT Gateway
    nat_gateway_json = nat_gateway.create_nat_gateway(params)
    CommonUtil.log("create_nat_gateway", nat_gateway_json)

    # Query the NAT Gateway
    params['nat_gateway_id'] = nat_gateway_json['NatGatewayId']
    nat_gateway_json = nat_gateway.describe_nat_gateway(params['
nat_gateway_id'])
    CommonUtil.log("describe_nat_gateway", nat_gateway_json)

    # Delete the NAT Gateway
    nat_gateway_json = nat_gateway.delete_nat_gateway(params)
    CommonUtil.log("delete_nat_gateway", nat_gateway_json)
```

```
    # Delete the VSwitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vswitch.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", vswitch_json)

    # Delete the VPC
    vpc_json = vpc.delete_vpc(params)
    CommonUtil.log("delete_vpc", vpc_json)


if __name__ == "__main__":
    sys.exit(main())
```

5. **Access the *natgw_quick_start.py* directory and run the following command to create a NAT Gateway.**

```
python natgw_quick_start.py
```

The system displays the following output:

```
----------------------------create_vpc----------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-uf6wzp25d8lkbxxxxxxxx",
  "VRouterId": "vrt-uf6di7voecmyqxxxxxxxx",
  "VpcId": "vpc-uf63cqupghmk1xxxxxxxx",
  "RequestId": "97D36E19-F789-424F-A473-660D63EF8CF9"
}

----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-uf6fovepnk4yexxxxxxxx",
  "RequestId": "18DA0E81-34A6-4877-9771-E2C4EEEBADD1"
}

----------------------------create_nat_gateway
----------------------------
{
  "NatGatewayId": "ngw-uf6mfrcmzktstxxxxxxxx",
  "BandwidthPackageIds": {
    "BandwidthPackageId": []
  },
  "ForwardTableIds": {
    "ForwardTableId": [
      "ftb-uf6411str8n9sxxxxxxxx"
    ]
  },
  "RequestId": "B1F791C8-73B1-46C5-8A20-726A615BC627",
  "SnatTableIds": {
    "SnatTableId": [
      "stb-uf6t4eijvq3aexxxxxxxx"
    ]
  }
}

----------------------------describe_nat_gateway
----------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
```

```
  "RequestId": "1F9303B1-4024-4A92-B67E-FB6BE1DC76D1",
  "PageSize": 10,
  "NatGateways": {
    "NatGateway": [
      {
        "Status": "Available",
        "BandwidthPackageIds": {
          "BandwidthPackageId": []
        },
        "VpcId": "vpc-uf63cqupghmk1xxxxxxxx",
        "Description": "",
        "ForwardTableIds": {
          "ForwardTableId": [
            "ftb-uf6411str8n9sxxxxxxxx"
          ]
        },
        "IpLists": {
          "IpList": []
        },
        "BusinessStatus": "Normal",
        "RegionId": "cn-shanghai",
        "CreationTime": "2019-04-24T09:09:12Z",
        "NatGatewayId": "ngw-uf6mfrcmzktstxxxxxxxx",
        "SnatTableIds": {
          "SnatTableId": [
            "stb-uf6t4eijvq3aexxxxxxxx"
          ]
        },
        "AutoPay": false,
        "InstanceChargeType": "PostPaid",
        "ExpiredTime": "",
        "Spec": "Small",
        "Name": ""
      }
    ]
  }
}

----------------------------delete_nat_gateway
----------------------------
{
  "RequestId": "A0B71FE4-4756-4D91-899E-1DFA52D8615E"
}

----------------------------delete_vswitch----------------------------
{
  "RequestId": "F224307E-3DE4-4415-AE19-DDCF24695462"
}

----------------------------delete_vpc----------------------------
{
  "RequestId": "1BFFCBC3-7F83-436C-96E9-CA4A620072DA"
}
```


2.5.2 Associate and disassociate an EIP

This topic describes how to use the Alibaba Cloud Python SDK to associate and disassociate an EIP.

Context

Context

The detailed steps used in this tutorial to associate and disassociate an EIP are as follows:

1. **Create a VPC in the China (Shanghai) region.**
2. **Create a VSwitch under the VPC.**
3. **Create a NAT Gateway under the VPC.**
4. **Create an EIP in the China (Shanghai) region.**
5. **Associate the EIP to the NAT Gateway.**
6. **Query the EIP.**
7. **Create an Internet Shared Bandwidth instance in the China (Shanghai) region.**
8. **Add the EIP to the Internet Shared Bandwidth instance.**
9. **Query the created NAT Gateway.**
10. **Disassociate the EIP from the NAT Gateway.**
11. **Remove the EIP from the Internet Shared Bandwidth instance.**
12. **Delete the Internet Shared Bandwidth instance.**
13. **Delete the NAT Gateway.**
14. **Release the EIP.**
15. **Delete the VSwitch.**
16. **Delete the VPC.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\natgw* **folder.**
4. **Open the** *natgw_associate_eip.py* **file in your text editor and configure your required parameters. Then, save the configurations and exit the text editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
```

```
from aliyunsdkvpc.request.v20160428 import CreateNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DeleteNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DescribeNatGatewaysR
equest
from sdk_lib.sdk_vpc import Vpc
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.sdk_eip import Eip
from sdk_lib.sdk_cbwp import CommonBandwidthPackage
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.exception import ExceptionHandler
from sdk_lib.consts import *

client = ACS_CLIENT


class NatGateway(object):
    def __init__(self, client):
        self.client = client

    def create_nat_gateway(self, params):
        """
        create_nat_gateway: Create a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36048.htm
        """
        try:
            request = CreateNatGatewayRequest.CreateNatGatewayRequ
est()
            request.set_VpcId(params['vpc_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_nat_gateway
_status,
                                        AVAILABLE, response_json['
NatGatewayId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway(self, nat_gateway_id):
        """
        describe_nat_gateway: Query NAT Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        try:
            request = DescribeNatGatewaysRequest.DescribeNa
tGatewaysRequest()
            request.set_NatGatewayId(nat_gateway_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_nat_gateway(self, params):
```

```
        """
        delete_nat_gateway: Delete a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36051.htm
        """
        try:
            request = DeleteNatGatewayRequest.DeleteNatGatewayRequ
est()
            request.set_NatGatewayId(params['nat_gateway_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_nat_gateway
_status,
                                        '', params['nat_gateway_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway_status(self, nat_gateway_id):
        """
        describe_nat_gateway_status: Query the status of NAT
Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        response = self.describe_nat_gateway(nat_gateway_id)
        if len(response["NatGateways"]["NatGateway"]) == 0:
            return ''
        return response["NatGateways"]["NatGateway"][0]['Status']


def main():
    vpc = Vpc(client)
    vswitch = VSwitch(client)
    eip = Eip(client)
    cbwp = CommonBandwidthPackage(client)
    nat_gateway = NatGateway(client)

    params = {}

    # Create a VPC
    vpc_json = vpc.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    # Create a VSwitch
    params['vpc_id'] = vpc_json['VpcId']
    params['zone_id'] = "cn-shanghai-d"
    params['cidr_block'] = "172.16.1.0/24"
    vswitch_json = vswitch.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)

    # Create a NAT Gateway
    nat_gateway_json = nat_gateway.create_nat_gateway(params)
    CommonUtil.log("create_nat_gateway", nat_gateway_json)

    # Create an EIP
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)
```

```
    params['allocation_id'] = eip_response_json["AllocationId"]

    # Associate the EIP to the NAT Gateway
    params['instance_id'] = nat_gateway_json['NatGatewayId']
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['instance_type'] = 'Nat'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address eip", eip_response_json)

    # Query the EIP
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Create an Internet Shared Bandwidth
    params['bandwidth'] = BANDWIDTH_10
    cbwp_repsonse_json = cbwp.create_common_bandwidth_package(params
)
    CommonUtil.log("create_common_bandwidth_package", cbwp_repso
nse_json)

    # Add the EIP to the Internet Shared Bandwidth
    params['ip_instance_id'] = params['allocation_id']
    params['bandwidth_package_id'] = cbwp_repsonse_json['BandwidthP
ackageId']
    cbwp_repsonse_json = cbwp.add_common_bandwidth_packageIp(params)
    CommonUtil.log("add_common_bandwidth_packageIp", cbwp_repso
nse_json)

    # Query the NAT Gateway
    params['nat_gateway_id'] = nat_gateway_json['NatGatewayId']
    nat_gateway_json = nat_gateway.describe_nat_gateway(params['
nat_gateway_id'])
    CommonUtil.log("describe_nat_gateway", nat_gateway_json)

    # Disassociate the EIP
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address nat", eip_response_json)

    # Remove the EIP from the Internet Shared Bandwidth
    cbwp_repsonse_json = cbwp.remove_common_bandwidth_packageIp(
params)
    CommonUtil.log("remove_common_bandwidth_packageIp", cbwp_repso
nse_json)

    # Delete the Internet Shared Bandwidth
    params['force'] = True
    cbwp_repsonse_json = cbwp.delete_common_bandwidth_package(params
)
    CommonUtil.log("delete_common_bandwidth_package", cbwp_repso
nse_json)

    # Delete the NAT Gateway
    nat_gateway_json = nat_gateway.delete_nat_gateway(params)
    CommonUtil.log("delete_nat_gateway", nat_gateway_json)

    # Release the EIP
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

    # Delete the VSwitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vswitch.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", vswitch_json)
```

```
    # Delete the VPC
    vpc_json = vpc.delete_vpc(params)
    CommonUtil.log("delete_vpc", vpc_json)


if __name__ == "__main__":
    sys.exit(main())
```

5. **Access the *natgw_associate_eip.py* directory and run the following command to associate and disassociate an EIP.**

```
python natgw_associate_eip.py
```

The system displays the following output:

```
----------------------------create_vpc----------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-uf6agemvkcmd8xxxxxxxx",
  "VRouterId": "vrt-uf6r7lqtsv65dxxxxxxxx",
  "VpcId": "vpc-uf6mqfqx8vjmoxxxxxxxx",
  "RequestId": "ADF806C6-FCD6-4E46-B8E3-72C2BE895344"
}

----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-uf6rm6add6w89xxxxxxxx",
  "RequestId": "897FFCC1-E6BA-484E-A245-C5DAEBBA269C"
}

----------------------------create_nat_gateway
----------------------------
{
  "NatGatewayId": "ngw-uf681h38pbvlyxxxxxxxx",
  "BandwidthPackageIds": {
    "BandwidthPackageId": []
  },
  "ForwardTableIds": {
    "ForwardTableId": [
      "ftb-uf6jd0vbyao2dxxxxxxxx"
    ]
  },
  "RequestId": "7C7CD3CB-041A-4B80-80B4-8BF8D5EF0D26",
  "SnatTableIds": {
    "SnatTableId": [
      "stb-uf6uj997htg3uxxxxxxxx"
    ]
  }
}

----------------------------allocate_eip_address
----------------------------
{
  "EipAddress": "106.xx.xx.129",
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RequestId": "DB795B99-1CEA-4FC1-9CE3-9DE2B977BF02",
  "AllocationId": "eip-uf62tf8y4uyacxxxxxxxx"
}
```

```
--------------------------associate_eip_address eip
--------------------------
{
  "RequestId": "443D7060-B716-4193-A44D-23FE762004F8"
}

--------------------------describe_eip_address
--------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "106.xx.xx.129",
        "AllocationId": "eip-uf62tf8y4uyacxxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "InUse",
        "BandwidthPackageId": "",
        "InstanceId": "ngw-uf681h38pbvlyxxxxxxxx",
        "InstanceRegionId": "cn-shanghai",
        "RegionId": "cn-shanghai",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-shanghai"
          ]
        },
        "ResourceGroupId": "rg-acfmxazxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "Nat",
        "AllocationTime": "2019-04-24T10:03:08Z",
        "Name": "",
        "OperationLocks": {
          "LockReason": []
        },
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "5",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "F0AEE605-14AD-4ADD-980C-4B508CE7EE4B"
}

--------------------------create_common_bandwidth_package
----------------------
-----
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "BandwidthPackageId": "cbwp-uf6dmfvq0gzzgxxxxxxxx",
  "RequestId": "D5E02777-2A72-42EC-8308-5D4B6E56D900"
}

--------------------------add_common_bandwidth_packageIp
-----------------------
```

```
----
{
  "RequestId": "3B4CD99C-6E59-4DA2-9256-EACF3F699412"
}

---------------------------describe_nat_gateway
---------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "RequestId": "A07498A0-4D60-4E0F-A7DE-3A832B174D59",
  "PageSize": 10,
  "NatGateways": {
    "NatGateway": [
      {
        "Status": "Available",
        "BandwidthPackageIds": {
          "BandwidthPackageId": []
        },
        "VpcId": "vpc-uf6mqfqx8vjmoxxxxxxxx",
        "Description": "",
        "ForwardTableIds": {
          "ForwardTableId": [
            "ftb-uf6jd0vbyao2dxxxxxxxx"
          ]
        },
        "IpLists": {
          "IpList": [
            {
              "UsingStatus": "Idle",
              "IpAddress": "106.xx.xx.129",
              "AllocationId": "eip-uf62tf8y4uyacxxxxxxxx"
            }
          ]
        },
        "BusinessStatus": "Normal",
        "RegionId": "cn-shanghai",
        "CreationTime": "2019-04-24T10:03:05Z",
        "NatGatewayId": "ngw-uf681h38pbvlyxxxxxxxx",
        "SnatTableIds": {
          "SnatTableId": [
            "stb-uf6uj997htg3uxxxxxxxx"
          ]
        },
        "AutoPay": false,
        "InstanceChargeType": "PostPaid",
        "ExpiredTime": "",
        "Spec": "Small",
        "Name": ""
      }
    ]
  }
}

---------------------------unassociate_eip_address nat
---------------------------
-
{
  "RequestId": "92CB670E-239D-4659-B91F-E0565D5C0F2D"
}

---------------------------remove_common_bandwidth_packageIp
--------------------
-------
```

```
{
  "RequestId": "A58E9647-6761-4CA3-8786-3CD4E7D2A7AB"
}

----------------------------delete_common_bandwidth_package
----------------------
-----
{
  "RequestId": "4AA428BC-B72F-4567-94B8-AC2398EF7529"
}

----------------------------delete_nat_gateway
---------------------------
{
  "RequestId": "EC6C5D04-AF7D-4560-A30E-80EC141D174D"
}

----------------------------release_eip_address
---------------------------
{
  "RequestId": "9B1380B3-EE97-49BD-88FE-DBF356304208"
}

----------------------------delete_vswitch----------------------------
{
  "RequestId": "A9A1D63E-5709-4B98-90BF-9069AA264230"
}

----------------------------delete_vpc----------------------------
{
  "RequestId": "3B687C37-5315-4E0B-BE13-103BB287A80D"
}
```

2.5.3 Create a DNAT entry

This topic describes how to use the Alibaba Cloud Python SDK to create a DNAT entry.

Context

Context

The detailed steps used in this tutorial to create a DNAT entry are as follows:

1. **Create a VPC in the China (Shanghai) region.**
2. **Create a VSwitch under the VPC.**
3. **Create a NAT Gateway under the VPC.**
4. **Create an EIP in the China (Shanghai) region.**
5. **Attach the EIP to the NAT Gateway.**
6. **Create a DNAT entry.**
7. **Query the EIP attached to the NAT Gateway.**
8. **Query the NAT Gateway.**
9. **Delete the DNAT entry.**

10. **Detach the EIP from the NAT Gateway.**
11. **Delete the NAT Gateway.**
12. **Release the EIP.**
13. **Delete the VSwitch.**
14. **Delete the VPC.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\natgw* **folder.**
4. **Open the** *natgw_dnat.py* **file in your text editor and configure your required parameters. Then, save the configurations and exit the text editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DeleteNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DescribeNatGatewaysR
equest
from aliyunsdkvpc.request.v20160428 import CreateForwardEntryRequest
from aliyunsdkvpc.request.v20160428 import DescribeForwardTable
EntriesRequest
from aliyunsdkvpc.request.v20160428 import DeleteForwardEntryRequest
from sdk_lib.sdk_vpc import Vpc
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.sdk_eip import Eip
from sdk_lib.sdk_cbwp import CommonBandwidthPackage
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.exception import ExceptionHandler
from sdk_lib.consts import *

client = ACS_CLIENT


class NatGateway(object):
    def __init__(self, client):
        self.client = client

    def create_nat_gateway(self, params):
        """
```

```
        create_nat_gateway: Create a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36048.htm
        """
        try:
            request = CreateNatGatewayRequest.CreateNatGatewayRequ
est()
            request.set_VpcId(params['vpc_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_nat_gateway
_status,
                                        AVAILABLE, response_json['
NatGatewayId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway(self, nat_gateway_id):
        """
        describe_nat_gateway: Query NAT Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        try:
            request = DescribeNatGatewaysRequest.DescribeNa
tGatewaysRequest()
            request.set_NatGatewayId(nat_gateway_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_nat_gateway(self, params):
        """
        delete_nat_gateway: Delete a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36051.htm
        """
        try:
            request = DeleteNatGatewayRequest.DeleteNatGatewayRequ
est()
            request.set_NatGatewayId(params['nat_gateway_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_nat_gateway
_status,
                                        '', params['nat_gateway_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
```

```
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway_status(self, nat_gateway_id):
        """
        describe_nat_gateway_status: Query the status of NAT
Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        response = self.describe_nat_gateway(nat_gateway_id)
        if len(response["NatGateways"]["NatGateway"]) == 0:
            return ''
        return response["NatGateways"]["NatGateway"][0]['Status']

    def create_forward_entry(self, params):
        """
        create_forward_entry: Create a forward entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36058.htm
        """
        try:
            request = CreateForwardEntryRequest.CreateForwardEntryRe
quest()
            request.set_ForwardTableId(params['forward_table_id'])
            request.set_ExternalIp(params['external_ip'])
            request.set_IpProtocol(params['ip_protocol'])
            request.set_ExternalPort(params['external_port'])
            request.set_InternalIp(params['internal_ip'])
            request.set_InternalPort(params['internal_port'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the forward entry is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_forward_status
,
                                        AVAILABLE, params['
forward_table_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_forward(self, forward_table_id):
        """
        describe_forward: Query DNAT entries in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36053.htm
        """
        try:
            request = DescribeForwardTableEntriesRequest.DescribeFo
rwardTableEntriesRequest()
            request.set_ForwardTableId(forward_table_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_forward_status(self, forward_table_id):
        """
```

```
        describe_forward_status: Query the status of DNAT entries in
 a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36053.htm
        """
        response = self.describe_forward(forward_table_id)
        if len(response["ForwardTableEntries"]["ForwardTableEntry"])
 == 0:
            return ''
        return response["ForwardTableEntries"]["ForwardTableEntry"][
0]['Status']

    def delete_forward_entry(self, params):
        """
        delete_forward_entry: Delete a forward entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36050.htm
        """
        try:
            request = DeleteForwardEntryRequest.DeleteForwardEntryRe
quest()
            request.set_ForwardTableId(params['forward_table_id'])
            request.set_ForwardEntryId(params['forward_entry_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the forward entry is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_forward_status
,
                                        '', params['forward_table_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)


def main():
    vpc = Vpc(client)
    vswitch = VSwitch(client)
    eip = Eip(client)
    cbwp = CommonBandwidthPackage(client)
    nat_gateway = NatGateway(client)

    params = {}

    # Create a VPC
    vpc_json = vpc.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    # Create a VSwitch
    params['vpc_id'] = vpc_json['VpcId']
    params['zone_id'] = "cn-shanghai-b"
    params['cidr_block'] = "172.16.1.0/24"
    vswitch_json = vswitch.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)
    params['vswitch_id'] = vswitch_json['VSwitchId']

    # Create a NAT Gateway
    nat_gateway_json = nat_gateway.create_nat_gateway(params)
    CommonUtil.log("create_nat_gateway", nat_gateway_json)
```

```
    # Create an EIP
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['external_ip'] = eip_response_json['EipAddress']

    # Attach the EIP to the NAT Gateway
    params['instance_id'] = nat_gateway_json['NatGatewayId']
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['instance_type'] = 'Nat'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address eip", eip_response_json)

    # Create a DNAT entry
    params['forward_table_id'] = nat_gateway_json['ForwardTableIds']
['ForwardTableId'][0]
    params['ip_protocol'] = 'tcp'
    params['external_port'] = '8080'
    params['internal_port'] = '80'
    params['internal_ip'] = '172.16.1.0'
    forward_entry_json = nat_gateway.create_forward_entry(params)
    CommonUtil.log("create_forward_entry", forward_entry_json)

    # Query the EIP
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Query the NAT Gateway
    params['nat_gateway_id'] = nat_gateway_json['NatGatewayId']
    nat_gateway_json = nat_gateway.describe_nat_gateway(params['
nat_gateway_id'])
    CommonUtil.log("describe_nat_gateway", nat_gateway_json)

    # Delete the DNAT entry
    params['forward_entry_id'] = forward_entry_json['ForwardEntryId
']
    forward_entry_json = nat_gateway.delete_forward_entry(params)
    CommonUtil.log("delete_forward_entry", forward_entry_json)

    # Detach the EIP
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address nat", eip_response_json)

    # Delete the NAT Gateway
    nat_gateway_json = nat_gateway.delete_nat_gateway(params)
    CommonUtil.log("delete_nat_gateway", nat_gateway_json)

    # Release the EIP
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

    # Delete the VSwitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vswitch.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", vswitch_json)

    # Delete the VPC
    vpc_json = vpc.delete_vpc(params)
    CommonUtil.log("delete_vpc", vpc_json)


if __name__ == "__main__":
```

```
    sys.exit(main())
```

5. **Access the *natgw_dnat.py* directory and run the following command to create a DNAT entry.**

```
python natgw_dnat.py
```

The system displays the following output:

```
----------------------------create_vpc----------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-uf63rln6gbb50xxxxxxxx",
  "VRouterId": "vrt-uf6p1hfo0ho8gxxxxxxxx",
  "VpcId": "vpc-uf6c3r8yca7dhxxxxxxxx",
  "RequestId": "1F97FC59-77DF-4D76-BE62-0A13EB4E614C"
}

----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-uf6liy66d9ssuxxxxxxxx",
  "RequestId": "88CCCFED-1448-49D2-8550-71952981A47A"
}

----------------------------create_nat_gateway
----------------------------
{
  "NatGatewayId": "ngw-uf6aolgwhssvsxxxxxxxx",
  "BandwidthPackageIds": {
    "BandwidthPackageId": []
  },
  "ForwardTableIds": {
    "ForwardTableId": [
      "ftb-uf6unjiun4i12xxxxxxxx"
    ]
  },
  "RequestId": "62A58351-D608-43A4-849E-1E177E917BEA",
  "SnatTableIds": {
    "SnatTableId": [
      "stb-uf65utljwcdkpxxxxxxxx"
    ]
  }
}

----------------------------allocate_eip_address
----------------------------
{
  "EipAddress": "101.xx.xx.110",
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RequestId": "0565295E-2F49-4511-93BC-747A2D19A6BD",
  "AllocationId": "eip-uf683xrl32ge8xxxxxxxx"
}

----------------------------associate_eip_address eip
----------------------------
{
  "RequestId": "8759FCE8-F8C2-4372-91D5-7A25D43FD78C"
}

----------------------------create_forward_entry
----------------------------
```

```
{
  "ForwardEntryId": "fwd-uf6ng3wt8sfwmxxxxxxxx",
  "RequestId": "CC81BCF6-2F64-40CF-85B0-676A83AC3902"
}

---------------------------describe_eip_address
---------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "101.xx.xx.110",
        "AllocationId": "eip-uf683xrl32ge8xxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "InUse",
        "BandwidthPackageId": "",
        "InstanceId": "ngw-uf6aolgwhssvsxxxxxxxx",
        "InstanceRegionId": "cn-shanghai",
        "RegionId": "cn-shanghai",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-shanghai"
          ]
        },
        "ResourceGroupId": "rg-acfmxazxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "Nat",
        "AllocationTime": "2019-04-24T10:56:53Z",
        "Name": "",
        "OperationLocks": {
          "LockReason": []
        },
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "5",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "CD2B3613-2A99-4687-9C23-A8E9F1F03048"
}

---------------------------describe_nat_gateway
---------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "RequestId": "D7519663-8D3B-4CC5-894F-A6798C89688D",
  "PageSize": 10,
  "NatGateways": {
    "NatGateway": [
      {
        "Status": "Available",
        "BandwidthPackageIds": {
```

```
          "BandwidthPackageId": []
        },
        "VpcId": "vpc-uf6c3r8yca7dhxxxxxxxx",
        "Description": "",
        "ForwardTableIds": {
          "ForwardTableId": [
            "ftb-uf6unjiun4i12xxxxxxxx"
          ]
        },
        "IpLists": {
          "IpList": [
            {
              "UsingStatus": "UsedByForwardTable",
              "IpAddress": "101.xx.xx.110",
              "AllocationId": "eip-uf683xrl32ge8xxxxxxxx"
            }
          ]
        },
        "BusinessStatus": "Normal",
        "RegionId": "cn-shanghai",
        "CreationTime": "2019-04-24T10:56:50Z",
        "NatGatewayId": "ngw-uf6aolgwhssvsxxxxxxxx",
        "SnatTableIds": {
          "SnatTableId": [
            "stb-uf65utljwcdkpxxxxxxxx"
          ]
        },
        "AutoPay": false,
        "InstanceChargeType": "PostPaid",
        "ExpiredTime": "",
        "Spec": "Small",
        "Name": ""
      }
    ]
  }
}

----------------------------delete_forward_entry
----------------------------
{
  "RequestId": "32C76D08-5738-4B07-A638-ACE5F5F5220E"
}

---------------------------unassociate_eip_address nat
--------------------------
-
{
  "RequestId": "AE686920-2CD1-4850-AADC-C249484D4B1A"
}

----------------------------delete_nat_gateway
----------------------------
{
  "RequestId": "FEBB1E7A-BA5B-4445-B2AB-5B828C17BBE6"
}

----------------------------release_eip_address
----------------------------
{
  "RequestId": "812D5E78-5113-4B92-892D-0B293BAD66F6"
}

----------------------------delete_vswitch----------------------------
{
```

```
  "RequestId": "8E13EEE4-21B5-4280-B46B-5C168736DC3A"
}

----------------------------delete_vpc----------------------------
{
  "RequestId": "DCBA91E7-F355-4EB6-83E3-27F2E68A8435"
}
```

2.5.4 Create an SNAT entry

This topic describes how to use the Alibaba Cloud Python SDK to create an SNAT entry.

Context

Context

The detailed steps used in this tutorial to create an SNAT entry are as follows:

1. **Create a VPC in the China (Shanghai) region.**
2. **Create a VSwitch under the VPC.**
3. **Create a NAT Gateway under the VPC.**
4. **Create an EIP in the China (Shanghai) region.**
5. **Attach the EIP to the NAT Gateway.**
6. **Create an SNAT entry.**
7. **Query the EIP attached to the NAT Gateway.**
8. **Query the NAT Gateway.**
9. **Delete the SNAT entry.**
10. **Detach the EIP from the NAT Gateway.**
11. **Delete the NAT Gateway.**
12. **Release the EIP.**
13. **Delete the VSwitch.**
14. **Delete the VPC.**

Procedure

Procedure

1. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\sdk_lib* **folder.**
2. **Open the** *consts.py* **file in your text editor and configure the** `ACS_CLIENT` **parameter used for user authentication.**
3. **In the downloaded SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\natgw* **folder.**

4. **Open the *natgw_snat.py* file in your text editor and configure your required parameters. Then, save the configurations and exit the text editor.**

```
#encoding=utf-8
import sys
import json
import time

from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DeleteNatGatewayRequest
from aliyunsdkvpc.request.v20160428 import DescribeNatGatewaysR
equest
from aliyunsdkvpc.request.v20160428 import CreateSnatEntryRequest
from aliyunsdkvpc.request.v20160428 import DescribeSnatTableEnt
riesRequest
from aliyunsdkvpc.request.v20160428 import DeleteSnatEntryRequest
from sdk_lib.sdk_vpc import Vpc
from sdk_lib.sdk_vswitch import VSwitch
from sdk_lib.sdk_eip import Eip
from sdk_lib.sdk_cbwp import CommonBandwidthPackage
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.exception import ExceptionHandler
from sdk_lib.consts import *

client = ACS_CLIENT


class NatGateway(object):
    def __init__(self, client):
        self.client = client

    def create_nat_gateway(self, params):
        """
        create_nat_gateway: Create a NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36048.htm
        """
        try:
            request = CreateNatGatewayRequest.CreateNatGatewayRequ
est()
            request.set_VpcId(params['vpc_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_nat_gateway
_status,
                                        AVAILABLE, response_json['
NatGatewayId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway(self, nat_gateway_id):
        """
        describe_nat_gateway: Query NAT Gateways in a region
```

```
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        try:
            request = DescribeNatGatewaysRequest.DescribeNa
tGatewaysRequest()
            request.set_NatGatewayId(nat_gateway_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_nat_gateway(self, params):
        """
        delete_nat_gateway: Delete the NAT Gateway
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36051.htm
        """
        try:
            request = DeleteNatGatewayRequest.DeleteNatGatewayRequ
est()
            request.set_NatGatewayId(params['nat_gateway_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the NAT Gateway is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_nat_gateway
_status,
                                        '', params['nat_gateway_id
']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_nat_gateway_status(self, nat_gateway_id):
        """
        describe_nat_gateway_status: Query the status of NAT
Gateways in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/36054.htm
        """
        response = self.describe_nat_gateway(nat_gateway_id)
        if len(response["NatGateways"]["NatGateway"]) == 0:
            return ''
        return response["NatGateways"]["NatGateway"][0]['Status']

    def create_snat_entry(self, params):
        """
        describe_snat: Create an SNAT entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/42672.htm
        """
        try:
            request = CreateSnatEntryRequest.CreateSnatEntryRequest
()
            request.set_SnatTableId(params['snat_table_id'])
            request.set_SourceVSwitchId(params['vswitch_id'])
            request.set_SnatIp(params['snat_ip'])
```

```
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the SNAT entry is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_snat_status,
                                        AVAILABLE, params['
snat_table_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_snat(self, snat_table_id):
        """
        describe_snat: Query SNAT entries in a region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/42677.htm
        """
        try:
            request = DescribeSnatTableEntriesRequest.DescribeSn
atTableEntriesRequest()
            request.set_SnatTableId(snat_table_id)
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_snat_status(self, snat_table_id):
        """
        describe_snat_status: Query the status of SNAT entries in a
region
        API reference link: https://www.alibabacloud.com/help/doc-
detail/42677.htm
        """
        response = self.describe_snat(snat_table_id)
        if len(response["SnatTableEntries"]["SnatTableEntry"]) == 0:
            return ''
        return response["SnatTableEntries"]["SnatTableEntry"][0]['
Status']

    def delete_snat_entry(self, params):
        """
        delete_snat_entry: Delete an SNAT entry
        API reference link: https://www.alibabacloud.com/help/doc-
detail/42678.htm
        """
        try:
            request = DeleteSnatEntryRequest.DeleteSnatEntryRequest
()
            request.set_SnatTableId(params['snat_table_id'])
            request.set_SnatEntryId(params['snat_entry_id'])
            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the SNAT entry is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_snat_status,
                                        '', params['snat_table_id
']):
```

```
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)


def main():
    vpc = Vpc(client)
    vswitch = VSwitch(client)
    eip = Eip(client)
    cbwp = CommonBandwidthPackage(client)
    nat_gateway = NatGateway(client)

    params = {}

    # Create a VPC
    vpc_json = vpc.create_vpc()
    CommonUtil.log("create_vpc", vpc_json)

    # Create a VSwitch
    params['vpc_id'] = vpc_json['VpcId']
    params['zone_id'] = "cn-shanghai-b"
    params['cidr_block'] = "172.16.1.0/24"
    vswitch_json = vswitch.create_vswitch(params)
    CommonUtil.log("create_vswitch", vswitch_json)
    params['vswitch_id'] = vswitch_json['VSwitchId']

    # Create a NAT Gateway
    nat_gateway_json = nat_gateway.create_nat_gateway(params)
    CommonUtil.log("create_nat_gateway", nat_gateway_json)

    # Create an EIP
    eip_response_json = eip.allocate_eip_address(params)
    CommonUtil.log("allocate_eip_address", eip_response_json)
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['snat_ip'] = eip_response_json['EipAddress']

    # Attach the EIP to the NAT Gateway
    params['instance_id'] = nat_gateway_json['NatGatewayId']
    params['allocation_id'] = eip_response_json["AllocationId"]
    params['instance_type'] = 'Nat'
    eip_response_json = eip.associate_eip_address(params)
    CommonUtil.log("associate_eip_address eip", eip_response_json)

    # Create an SNAT entry
    params['snat_table_id'] = nat_gateway_json['SnatTableIds']['
SnatTableId'][0]
    snat_entry_json = nat_gateway.create_snat_entry(params)
    CommonUtil.log("create_snat_entry", snat_entry_json)

    # Query the EIP
    eip_response_json = eip.describe_eip_address(params['allocation
_id'])
    CommonUtil.log("describe_eip_address", eip_response_json)

    # Query the NAT Gateway
    params['nat_gateway_id'] = nat_gateway_json['NatGatewayId']
    nat_gateway_json = nat_gateway.describe_nat_gateway(params['
nat_gateway_id'])
    CommonUtil.log("describe_nat_gateway", nat_gateway_json)

    # Delete the SNAT entry
    params['snat_entry_id'] = snat_entry_json['SnatEntryId']
```

```
    snat_entry_json = nat_gateway.delete_snat_entry(params)
    CommonUtil.log("delete_snat_entry", snat_entry_json)

    # Detach the EIP
    eip_response_json = eip.unassociate_eip_address(params)
    CommonUtil.log("unassociate_eip_address nat", eip_response_json)

    # Delete the NAT Gateway
    nat_gateway_json = nat_gateway.delete_nat_gateway(params)
    CommonUtil.log("delete_nat_gateway", nat_gateway_json)

    # Release the EIP
    eip_response_json = eip.release_eip_address(params)
    CommonUtil.log("release_eip_address", eip_response_json)

    # Delete the VSwitch
    params['vswitch_id'] = vswitch_json['VSwitchId']
    vswitch_json = vswitch.delete_vswitch(params)
    CommonUtil.log("delete_vswitch", vswitch_json)

    # Delete the VPC
    vpc_json = vpc.delete_vpc(params)
    CommonUtil.log("delete_vpc", vpc_json)


if __name__ == "__main__":
    sys.exit(main())
```

5. **Access the *natgw_snat.py* directory and run the following command to create an SNAT entry.**

```
python natgw_snat.py
```

The system displays the following output:

```
----------------------------create_vpc--------------------------
{
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RouteTableId": "vtb-uf6a8ccj9ne58xxxxxxxx",
  "VRouterId": "vrt-uf6qqqaf1o1ptxxxxxxxx",
  "VpcId": "vpc-uf6hxer3h07wgxxxxxxxx",
  "RequestId": "8F483A7B-8A38-47ED-85BD-1E83C075AEA4"
}

----------------------------create_vswitch----------------------------
{
  "VSwitchId": "vsw-uf6lbov9tyetqxxxxxxxx",
  "RequestId": "2EE2E11B-EF60-4C88-BE2A-F45517290B31"
}

----------------------------create_nat_gateway
---------------------------
{
  "NatGatewayId": "ngw-uf6l3c3rswubuxxxxxxxx",
  "BandwidthPackageIds": {
    "BandwidthPackageId": []
  },
  "ForwardTableIds": {
    "ForwardTableId": [
      "ftb-uf6086r1hyecbxxxxxxxx"
    ]
```

```
  },
  "RequestId": "9037D769-24C8-46AD-83F3-4C0538FA5970",
  "SnatTableIds": {
    "SnatTableId": [
      "stb-uf6ppo11rsecmxxxxxxxx"
    ]
  }
}

---------------------------allocate_eip_address
---------------------------
{
  "EipAddress": "101.xx.xx.110",
  "ResourceGroupId": "rg-acfmxazxxxxxxxx",
  "RequestId": "0DE621B4-6BDE-4E17-A294-8F71FBB9F710",
  "AllocationId": "eip-uf6d311cpmr0nxxxxxxxx"
}

---------------------------associate_eip_address eip
---------------------------
{
  "RequestId": "C95B2EDC-F081-4784-B60B-2600F60E684D"
}

---------------------------create_snat_entry
---------------------------
{
  "SnatEntryId": "snat-uf6ppbwshdu40xxxxxxxx",
  "RequestId": "BB9F8FD2-3CB5-4F84-8006-FE64BF3BEA06"
}

---------------------------describe_eip_address
---------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "PageSize": 10,
  "EipAddresses": {
    "EipAddress": [
      {
        "ISP": "BGP",
        "ExpiredTime": "",
        "InternetChargeType": "PayByBandwidth",
        "IpAddress": "101.xx.xx.110",
        "AllocationId": "eip-uf6d311cpmr0nxxxxxxxx",
        "PrivateIpAddress": "",
        "Status": "InUse",
        "BandwidthPackageId": "",
        "InstanceId": "ngw-uf6l3c3rswubuxxxxxxxx",
        "InstanceRegionId": "cn-shanghai",
        "RegionId": "cn-shanghai",
        "AvailableRegions": {
          "AvailableRegion": [
            "cn-shanghai"
          ]
        },
        "ResourceGroupId": "rg-acfmxazxxxxxxxx",
        "HasReservationData": false,
        "InstanceType": "Nat",
        "AllocationTime": "2019-04-24T11:20:09Z",
        "Name": "",
        "OperationLocks": {
          "LockReason": []
        },
```

```
        "Mode": "NAT",
        "BandwidthPackageType": "",
        "BandwidthPackageBandwidth": "",
        "Bandwidth": "5",
        "HDMonitorStatus": "OFF",
        "ChargeType": "PostPaid",
        "SecondLimited": false,
        "Descritpion": ""
      }
    ]
  },
  "RequestId": "19052237-6E84-4258-89B9-05772C33C0DC"
}

---------------------------describe_nat_gateway
--------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "RequestId": "26CAA3FE-B400-4522-9582-2DAAF69129AE",
  "PageSize": 10,
  "NatGateways": {
    "NatGateway": [
      {
        "Status": "Available",
        "BandwidthPackageIds": {
          "BandwidthPackageId": []
        },
        "VpcId": "vpc-uf6hxer3h07wgxxxxxxxx",
        "Description": "",
        "ForwardTableIds": {
          "ForwardTableId": [
            "ftb-uf6086r1hyecbxxxxxxxx"
          ]
        },
        "IpLists": {
          "IpList": [
            {
              "UsingStatus": "UsedBySnatTable",
              "IpAddress": "101.xx.xx.110",
              "AllocationId": "eip-uf6d311cpmr0nxxxxxxxx"
            }
          ]
        },
        "BusinessStatus": "Normal",
        "RegionId": "cn-shanghai",
        "CreationTime": "2019-04-24T11:20:06Z",
        "NatGatewayId": "ngw-uf6l3c3rswubuxxxxxxxx",
        "SnatTableIds": {
          "SnatTableId": [
            "stb-uf6ppo11rsecmxxxxxxxx"
          ]
        },
        "AutoPay": false,
        "InstanceChargeType": "PostPaid",
        "ExpiredTime": "",
        "Spec": "Small",
        "Name": ""
      }
    ]
  }
}
```

```
----------------------------delete_snat_entry
----------------------------
{
  "RequestId": "CDA82881-ACF0-4DD1-887B-A764F56F180D"
}

----------------------------unassociate_eip_address nat
--------------------------
-
{
  "RequestId": "140C46FF-0DB0-47F9-B0F4-3459DF117EAF"
}

----------------------------delete_nat_gateway
----------------------------
{
  "RequestId": "8709747C-6786-443C-8AEA-51647AA49769"
}

----------------------------release_eip_address
----------------------------
{
  "RequestId": "0BC21C23-0FB3-4594-8B14-6552DF788C93"
}

----------------------------delete_vswitch--------------------------
{
  "RequestId": "16E840EC-E058-40C1-A5BF-8CDF672EA139"
}

----------------------------delete_vpc----------------------------
{
  "RequestId": "B5F18126-FF2A-4005-9973-3984034DF0F4"
}
```

2.6 Router interface

2.6.1 Interconnect two VPCs under the same account in the same region

This topic describes how to use Alibaba Cloud SDK for Python to interconnect two VPCs under the same account in the same region.

Prerequisites

Prerequisites

Context

Context

Before you interconnect two VPCs, you must create a router interface for each VPC. When you create router interfaces, note the following limits:

- **Only one pair of interconnected router interfaces can be created between two VRouters.**

- A maximum of five router interfaces can be created for each VRouter.
- If a router interface with an overdue payment is under your account, you cannot create router interfaces.
- Route entries in the same route table cannot have the same destination CIDR blocks.

The example used in this topic includes the following steps:

1. Create an initiator router interface.
2. Create an acceptor router interface.
3. Modify the initiator router interface.
4. Modify the acceptor router interface.
5. Query the initiator router interface.
6. Query the acceptor router interface.
7. Connect the initiator router interface to the acceptor router interface.
8. Create a route entry whose next hop is the initiator router interface.
9. Create a route entry whose next hop is the acceptor router interface.
10. Delete the route entry whose next hop is the acceptor router interface.
11. Delete the route entry whose next hop is the initiator router interface.
12. Deactivate the initiator router interface.
13. Deactivate the acceptor router interface.
14. Delete the initiator router interface.
15. Delete the acceptor router interface.

Procedure

Procedure

1. In the SDK directory, open the *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\ec* folder.
2. Open the *ec_same_account.py* file in your text editor, set the parameters as needed, and then save the settings and exit the editor.

   The sample code is as follows:

```
#encoding=utf-8
import sys
import json
from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateRouterInterfac
eRequest
```

```
from aliyunsdkvpc.request.v20160428 import DeleteRouterInterfac
eRequest
from aliyunsdkvpc.request.v20160428 import DescribeRouterInterf
acesRequest
from aliyunsdkvpc.request.v20160428 import ConnectRouterInterfa
ceRequest
from aliyunsdkvpc.request.v20160428 import DeactivateRouterInte
rfaceRequest
from aliyunsdkvpc.request.v20160428 import ModifyRouterInterfac
eAttributeRequest
from sdk_lib.sdk_route_entry import RouteEntry
from sdk_lib.exception import ExceptionHandler
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *

client = AcsClient(
            '<your-access-key-id>',     # your zone ID
            '<your-access-key-secret>', # your AccessKey ID
            '<your-region-id>')         # your AccessKey Secret

class RouterInterface(object):
    def __init__(self, client):
        self.client = client

    def create_router_interface(self, params):
        """
        create_router_interface: Create a VRouter interface.
        """
        try:
            request = CreateRouterInterfaceRequest.CreateRout
erInterfaceRequest()
            # The specification of the router interface
            request.set_Spec(params['spec'])
             # The role of the router interface
            request.set_Role(params['role'])
            # The ID of the router associated with the router
interface
            request.set_RouterId(params['router_id'])
            # The type of the router associated with the router
interface
            request.set_RouterType(params['router_type'])
            # The region ID of the connection receiver
            request.set_OppositeRegionId(params['opposite_region_id
'])
            # The type of the router associated with the peer router
 interface
            request.set_OppositeRouterType(params['opposite_r
outer_type'])

            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the router interface is available.
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_ri_status,
                                        'Idle', response_json['
RouterInterfaceId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)
```

```
    def connect_router_interface(self, params):
        """
        connect_router_interface: The initiator router interface
initiates a connection to the acceptor.
        """
        try:
            request = ConnectRouterInterfaceRequest.ConnectRou
terInterfaceRequest()
            # The ID of the initiator router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT * 5,
DEFAULT_TIME * 5,
                                        self.describe_ri_status,
                                        'Active', params['router_int
erface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def deactivate_router_interface(self, params):
        """
        deactivate_router_interface: Deactivate the router interface
.
        """
        try:
            request = DeactivateRouterInterfaceRequest.Deactivate
RouterInterfaceRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the router interface is available.
            if CheckStatus.check_status(TIME_DEFAULT_OUT * 5,
DEFAULT_TIME * 5,
                                        self.describe_ri_status,
                                        'Inactive', params['
router_interface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def modify_router_interface_attribute(self, params):
        """
        modify_router_interface_attribute: Modify the configuration
of the router interface.
        """
        try:
            request = ModifyRouterInterfaceAttributeRequest.
ModifyRouterInterfaceAttributeRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            # The ID of the peer router interface
            request.set_OppositeInterfaceId(params['opposite_i
nterface_id'])
            # The ID of the peer router
```

```
            request.set_OppositeRouterId(params['opposite_router_id
'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the router interface is available.
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_ri_status,
                                        'Idle', params['router_int
erface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)



    def describe_router_interface(self, instance_id):
        """
        describe_router_interface: Query the router interfaces in
the specified region.
        """
        try:
            request = DescribeRouterInterfacesRequest.DescribeRo
uterInterfacesRequest()
            # The query conditions
            request.add_query_param('Filter.1.Key', "RouterInte
rfaceId")
            # The ID of the queried instance
            request.add_query_param('Filter.1.Value.1', instance_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_ri_status(self, instance_id):
        """
        describe_ri_status: Query the status of the router interface
 in the specified region.
        """
        response = self.describe_router_interface(instance_id)
        if len(response['RouterInterfaceSet']['RouterInterfaceType
']) == 0:
            return ''
        return response['RouterInterfaceSet']['RouterInterfaceType']
[0]['Status']

    def delete_router_interface(self, params):
        """
        delete_router_interface: Delete the router interface.
        """
        try:
            request = DeleteRouterInterfaceRequest.DeleteRout
erInterfaceRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check whether the router interface is available.
```

```
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_ri_status,
                                        '', params['instance_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

def main():
    client = ACS_CLIENT
    router_interface = RouterInterface(client)
    route_entry = RouteEntry(client)

    params = {}
    params['spec'] = "Large.2"
    params['role'] = "InitiatingSide"
    params['router_id'] = ROUTER_ID
    params['router_type'] = "VRouter"
    params['opposite_region_id'] = "cn-hangzhou"
    params['opposite_router_type'] = "VRouter"

    # Create an initiator router interface.
    router_interface_json = router_interface.create_router_interface
(params)
    CommonUtil.log("create_router_interface", router_interface_json)

    # Create an acceptor router interface.
    params['spec'] = "Negative"
    params['role'] = "AcceptingSide"
    params['router_id'] = ROUTER_ID2
    router_interface_json2 = router_interface.create_router_interf
ace(params)
    CommonUtil.log("create_router_interface", router_interface_json2
)

    # Modify the initiator router interface.
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
    params['opposite_interface_id'] = router_interface_json2['
RouterInterfaceId']
    params['opposite_router_id'] = ROUTER_ID2
    modify_ri_json = router_interface.modify_router_interface_attrib
ute(params)
    CommonUtil.log("modify_router_interface_attribute", modify_ri_
json)

    # Modify the acceptor router interface.
    params['router_interface_id'] = router_interface_json2['
RouterInterfaceId']
    params['opposite_interface_id'] = router_interface_json['
RouterInterfaceId']
    params['opposite_router_id'] = ROUTER_ID
    modify_ri_json2 = router_interface.modify_router_interf
ace_attribute(params)
    CommonUtil.log("modify_router_interface_attribute", modify_ri_
json2)

    # Query the initiator router interface.
    describe_ri_json = router_interface.describe_router_interface(
router_interface_json['RouterInterfaceId'])
    CommonUtil.log("describe_router_interface", describe_ri_json)
```

```
    # Query the acceptor router interface.
    describe_ri_json2 = router_interface.describe_router_interface(
router_interface_json2['RouterInterfaceId'])
    CommonUtil.log("describe_router_interface", describe_ri_json2)

    # Initiate a connection.
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
    connect_ri_json = router_interface.connect_router_interface(
params)
    CommonUtil.log("connect_router_interface", connect_ri_json)

    # Create a route entry whose next hop is the initiator router
interface.
    params['route_table_id'] = TABLE_ID
    params['destination_cidr_block'] = "0.0.0.0/0"
    params['nexthop_type'] = 'RouterInterface'
    params['nexthop_id'] = router_interface_json['RouterInterfaceId
']
    route_entry_json = route_entry.create_route_entry(params)
    CommonUtil.log("create_route_entry", route_entry_json)

    # Create a route entry whose next hop is the acceptor router
interface.
    params['route_table_id'] = TABLE_ID2
    params['destination_cidr_block'] = "0.0.0.0/0"
    params['nexthop_type'] = 'RouterInterface'
    params['nexthop_id'] = router_interface_json2['RouterInterfaceId
']
    route_entry_json2 = route_entry.create_route_entry(params)
    CommonUtil.log("create_route_entry", route_entry_json2)

    # Delete the route entry whose next hop is the acceptor router
interface.
    route_entry_json = route_entry.delete_route_entry(params)
    CommonUtil.log("delete_route_entry", route_entry_json)

    # Delete the route entry whose next hop is the initiator router
interface.
    params['route_table_id'] = TABLE_ID
    params['nexthop_id'] = router_interface_json['RouterInterfaceId
']
    route_entry_json = route_entry.delete_route_entry(params)
    CommonUtil.log("delete_route_entry", route_entry_json)

    # Deactivate the initiator router interface.
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
    deactivate_ri_json = router_interface.deactivate_router_in
terface(params)
    CommonUtil.log("deactivate_router_interface", deactivate_ri_json
)

    # Deactivate the acceptor router interface.
    params['router_interface_id'] = router_interface_json2['
RouterInterfaceId']
    deactivate_ri_json2 = router_interface.deactivate_router_in
terface(params)
    CommonUtil.log("deactivate_router_interface", deactivate
_ri_json2)

    # Delete the initiator router interface.
    params['instance_id'] = router_interface_json['RouterInterfaceId
']
```

```
    router_interface_json = router_interface.delete_router_interface
(params)
    CommonUtil.log("delete_router_interface", router_interface_json)

    # Delete the acceptor router interface.
    params['instance_id'] = router_interface_json2['RouterInte
rfaceId']
    router_interface_json2 = router_interface.delete_router_interf
ace(params)
    CommonUtil.log("delete_router_interface", router_interface_json2
)


if __name__ == '__main__':
    sys.exit(main())
```

3. **Enter the directory where the *ec_same_account.py* file is stored, and then run the following command:**

```
python ec_same_account.py
```

Result

Result

The system output is displayed as follows:

```
----------------------------create_router_interface
----------------------------
{
  "RequestId": "F5493DC1-53B0-4916-874F-87773A54525F",
  "RouterInterfaceId": "ri-bp1ujwb6xsw16xxxxxxxx"
}
----------------------------create_router_interface
----------------------------
{
  "RequestId": "E3F5BE99-B2C5-4751-8173-0F590300EE72",
  "RouterInterfaceId": "ri-bp1vze2rusg2cxxxxxxxx"
}
----------------------------modify_router_interface_attribute
----------------------------
{
  "RequestId": "8D691507-F31A-41E5-9C72-457EFF1F7727"
}
----------------------------modify_router_interface_attribute
----------------------------
{
  "RequestId": "2E664208-8F37-4F19-B719-00B4F1AF03B2"
}
----------------------------describe_router_interface
----------------------------
{
  "TotalCount": 1,
  "RouterInterfaceSet": {
    "RouterInterfaceType": [
      {
        "BusinessStatus": "Normal",
        "CreationTime": "2019-04-30T06:09:16Z",
        "Role": "InitiatingSide",
        "OppositeRouterId": "vrt-bp141no9pds2bxxxxxxxx",
```

```
        "Spec": "Large.2",
        "Status": "Idle",
        "EndTime": "2999-09-08T16:00:00Z",
        "OppositeInterfaceSpec": "Negative",
        "RouterInterfaceId": "ri-bp1ujwb6xsw16xxxxxxxx",
        "RouterType": "VRouter",
        "OppositeBandwidth": 0,
        "OppositeVpcInstanceId": "vpc-bp1v31by9jix2xxxxxxxx",
        "HasReservationData": false,
        "OppositeInterfaceBusinessStatus": "Normal",
        "OppositeRouterType": "VRouter",
        "OppositeRegionId": "cn-hangzhou",
        "VpcInstanceId": "vpc-bp15opprpg0rgxxxxxxxx",
        "RouterId": "vrt-bp1ltkytn6lgmxxxxxxxx",
        "CrossBorder": false,
        "OppositeInterfaceOwnerId": "",
        "Bandwidth": 2048,
        "OppositeInterfaceId": "ri-bp1vze2rusg2cxxxxxxxx",
        "ChargeType": "AfterPay"
      }
    ]
  },
  "PageNumber": 1,
  "RequestId": "89EF0631-0A36-41AD-A586-AF4FFDA6E68B",
  "PageSize": 10
}
--------------------------describe_router_interface
--------------------------
{
  "TotalCount": 1,
  "RouterInterfaceSet": {
    "RouterInterfaceType": [
      {
        "Status": "Idle",
        "OppositeRegionId": "cn-hangzhou",
        "BusinessStatus": "Normal",
        "OppositeRouterId": "vrt-bp1ltkytn6lgmxxxxxxxx",
        "VpcInstanceId": "vpc-bp1v31by9jix2xxxxxxxx",
        "RouterInterfaceId": "ri-bp1vze2rusg2cxxxxxxxx",
        "CreationTime": "2019-04-30T06:09:18Z",
        "RouterType": "VRouter",
        "OppositeInterfaceOwnerId": "",
        "RouterId": "vrt-bp141no9pds2bxxxxxxxx",
        "Bandwidth": 0,
        "OppositeInterfaceId": "ri-bp1ujwb6xsw16xxxxxxxx",
        "EndTime": "2999-09-08T16:00:00Z",
        "ChargeType": "AfterPay",
        "OppositeVpcInstanceId": "vpc-bp15opprpg0rgxxxxxxxx",
        "HasReservationData": false,
        "CrossBorder": false,
        "OppositeInterfaceBusinessStatus": "Normal",
        "Spec": "Negative",
        "OppositeRouterType": "VRouter",
        "Role": "AcceptingSide"
      }
    ]
  },
  "PageNumber": 1,
  "RequestId": "578448D7-9DCF-4703-8337-EF88DDF2C325",
  "PageSize": 10
}
--------------------------connect_router_interface
--------------------------
{
```

```
  "RequestId": "A5DBB86B-F6F5-4A53-899E-8CF4FEF510F2"
}
----------------------------create_route_entry
----------------------------
{
  "RequestId": "70D896FE-986B-48EF-9734-17D6BDC8327A"
}
----------------------------create_route_entry
----------------------------
{
  "RequestId": "A2233E25-4D6B-4713-A96F-E7CA745973CA"
}
----------------------------delete_route_entry
----------------------------
{
  "RequestId": "464C62A4-EE65-4414-AF0A-4984AE6B8696"
}
----------------------------delete_route_entry
----------------------------
{
  "RequestId": "0C11A332-969B-47CA-A683-5BFFECA28B3D"
}
----------------------------deactivate_router_interface
----------------------------
{
  "RequestId": "018305AD-FB9E-450A-91E8-3830634F5AC2"
}
----------------------------deactivate_router_interface
----------------------------
{
  "RequestId": "89B03203-9224-4CA3-8679-E0A49029A2D2"
}
----------------------------delete_router_interface
----------------------------
{
  "RequestId": "FB0424CE-D0C7-438B-A3FA-BCF24EE9CC8A"
}
----------------------------delete_router_interface
----------------------------
{
  "RequestId": "8A0A0BB6-A69D-461B-A117-14AD6670DECA"
}
```

2.6.2 Interconnect a VBR and a VPC in the same region and under the same account

This topic describes how to use the Alibaba Cloud SDK for Python to interconnect a VBR and a VPC that are located in the same region and are under the same account.

Prerequisites

Prerequisites

Context

Context

Note the following when you create router interfaces:

- **Only one pair of interconnected router interfaces can be created between two routers.**
- **A maximum of five router interfaces can be created for each VRouter.**
- **If a router interface with an overdue payment is under your account, you cannot create router interfaces.**
- **Route entries in the same route table cannot have the same destination CIDR blocks.**
- **The VBR must operate as the connection initiator and be in the Activated state.**

The example used in this topic includes the following steps:

1. **Create a VBR.**
2. **Create the initiator router interface.**
3. **Create the acceptor router interface.**
4. **Modify the initiator router interface.**
5. **Modify the acceptor router interface.**
6. **Query the initiator router interface.**
7. **Query the acceptor router interface.**
8. **Query the ID of the route table in the VBR.**
9. **Connect the initiator router interface to the acceptor router interface.**
10. **Create a route entry whose next hop is the initiator router interface.**
11. **Create a route entry whose next hop is the acceptor router interface.**
12. **Delete the route entry whose next hop is the acceptor router interface.**
13. **Delete the route entry whose next hop is the initiator router interface.**
14. **Deactivate the initiator router interface.**
15. **Deactivate the acceptor router interface.**
16. **Delete the initiator router interface.**
17. **Delete the acceptor router interface.**
18. **Delete the VBR.**

Procedure

Procedure

1. **In the SDK directory, open the** *$aliyun-openapi-python-sdk-examples\sdk_examples\examples\ec* **folder.**

2. **Open the** *ec_vbr_vpc.py* **file in your text editor, set the parameters as needed, and then save the settings and exit the editor.**

   The sample code is as follows:

```
#encoding=utf-8
import sys
import json
from aliyunsdkcore.acs_exception.exceptions import ServerException,
ClientException
from aliyunsdkvpc.request.v20160428 import CreateRouterInterfac
eRequest
from aliyunsdkvpc.request.v20160428 import DeleteRouterInterfac
eRequest
from aliyunsdkvpc.request.v20160428 import DescribeRouterInterf
acesRequest
from aliyunsdkvpc.request.v20160428 import ConnectRouterInterfa
ceRequest
from aliyunsdkvpc.request.v20160428 import DeactivateRouterInte
rfaceRequest
from aliyunsdkvpc.request.v20160428 import ModifyRouterInterfac
eAttributeRequest
from aliyunsdkvpc.request.v20160428 import CreateVirtualBorderR
outerRequest
from aliyunsdkvpc.request.v20160428 import DescribeVirtualBorde
rRoutersRequest
from aliyunsdkvpc.request.v20160428 import DescribeRouteTablesR
equest
from aliyunsdkvpc.request.v20160428 import DeleteVirtualBorderR
outerRequest
from sdk_lib.sdk_route_entry import RouteEntry
from sdk_lib.exception import ExceptionHandler
from sdk_lib.common_util import CommonUtil
from sdk_lib.check_status import CheckStatus
from sdk_lib.consts import *

client = AcsClient(
            '<your-access-key-id>',     # your zone ID
            '<your-access-key-secret>', # your AccessKey ID
            '<your-region-id>')         # your AccessKey Secret

class RouterInterface(object):
    def __init__(self, client):
        self.client = client

    def create_router_interface(self, params):
        """
        create_router_interface: Create a router interface
        """
        try:
            request = CreateRouterInterfaceRequest.CreateRout
erInterfaceRequest()
            # The specification of the router interface
            request.set_Spec(params['spec'])
            # The role of the router interface
            request.set_Role(params['role'])
            # The ID of the router associated with the router
interface
            request.set_RouterId(params['router_id'])
            # The type of the router associated with the router
interface
            request.set_RouterType(params['router_type'])
```

```
            # The ID of the region to which the connection acceptor
belongs
            request.set_OppositeRegionId(params['opposite_region_id
'])
            # The type of the router associated with the peer router
 interface
            request.set_OppositeRouterType(params['opposite_r
outer_type'])
            # The ID of the access point to which the peer end
belongs
            request.set_OppositeAccessPointId(params['opposite_a
ccess_point_id'])
            # The ID of the access point to which the VBR belongs
            #request.set_AccessPointId(params['access_point_id'])

            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the router interface is in the available
state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_ri_status,
                                        'Idle', response_json['
RouterInterfaceId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def create_router_interface_vbr(self, params):
        """
        create_router_interface: Create a router interface
        """
        try:
            request = CreateRouterInterfaceRequest.CreateRout
erInterfaceRequest()
            # The specification of the router interface
            request.set_Spec(params['spec'])
            # The role of the router interface
            request.set_Role(params['role'])
            # The ID of the router associated with the router
interface
            request.set_RouterId(params['router_id'])
            # The type of the router associated with the router
interface
            request.set_RouterType(params['router_type'])
            # The ID of the region to which the connection acceptor
belongs
            request.set_OppositeRegionId(params['opposite_region_id
'])
            # The type of the router associated with the peer router
 interface
            request.set_OppositeRouterType(params['opposite_r
outer_type'])
            # The ID of the access point to which the VBR belongs
            request.set_AccessPointId(params['access_point_id'])

            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the router interface is in the available
state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
```

```
                                            self.describe_ri_status,
                                            'Idle', response_json['
RouterInterfaceId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def connect_router_interface(self, params):
        """
        connect_router_interface: The initiator router interface
initiates a connection to the acceptor router interface
        """
        try:
            request = ConnectRouterInterfaceRequest.ConnectRou
terInterfaceRequest()
            # The ID of the initiator router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT * 5,
DEFAULT_TIME * 5,
                                        self.describe_ri_status,
                                        'Active', params['router_int
erface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def deactivate_router_interface(self, params):
        """
        deactivate_router_interface: Deactivate a router interface
        """
        try:
            request = DeactivateRouterInterfaceRequest.Deactivate
RouterInterfaceRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the router interface is in the available
state
            if CheckStatus.check_status(TIME_DEFAULT_OUT * 5,
DEFAULT_TIME * 5,
                                        self.describe_ri_status,
                                        'Inactive', params['
router_interface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def modify_router_interface_attribute(self, params):
        """
        modify_router_interface_attribute: Modify the configurations
 of a router interface
        """
        try:
```

```
            request = ModifyRouterInterfaceAttributeRequest.
ModifyRouterInterfaceAttributeRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['router_int
erface_id'])
            # The ID of the peer router interface
            request.set_OppositeInterfaceId(params['opposite_i
nterface_id'])
            # The ID of the peer router
            request.set_OppositeRouterId(params['opposite_router_id
'])
            # The type of the peer router
            request.set_OppositeRouterType(params['opposite_r
outer_type'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the router interface is in the available
state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_ri_status,
                                        'Idle', params['router_int
erface_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)



    def describe_router_interface(self, instance_id):
        """
        describe_router_interface: Query router interfaces in a
region
        """
        try:
            request = DescribeRouterInterfacesRequest.DescribeRo
uterInterfacesRequest()
            # The queried filter type
            request.add_query_param('Filter.1.Key', "RouterInte
rfaceId")
            # The ID of the queried instance
            request.add_query_param('Filter.1.Value.1', instance_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_ri_status(self, instance_id):
        """
        describe_ri_status: Query the status of router interfaces in
 a region
        """
        response = self.describe_router_interface(instance_id)
        if len(response['RouterInterfaceSet']['RouterInterfaceType
']) == 0:
            return ''
        return response['RouterInterfaceSet']['RouterInterfaceType']
[0]['Status']
```

```
    def delete_router_interface(self, params):
        """
        delete_router_interface: Delete a router interface
        """
        try:
            request = DeleteRouterInterfaceRequest.DeleteRout
erInterfaceRequest()
            # The ID of the router interface
            request.set_RouterInterfaceId(params['instance_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the router interface is in the available
state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME * 5,
                                        self.describe_ri_status,
                                        '', params['instance_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def create_virtual_border_router(self, params):
        """
        create_virtual_border_router: Create a VBR
        """
        try:
            request = CreateVirtualBorderRouterRequest.CreateVirt
ualBorderRouterRequest()
            # The ID of the physical connection
            request.set_PhysicalConnectionId(params['physical_c
onnection_id'])
            # The Alibaba Cloud-side IP address of the VBR
            request.set_LocalGatewayIp(params['local_gateway_ip'])
            # The peer IP address of the physical connection-side
interface of the VBR
            request.set_PeerGatewayIp(params['peer_gateway_ip'])
            # The subnet mask for the Alibaba Cloud-side and
customer-side IP addresses of the VBR
            request.set_PeeringSubnetMask(params['peering_su
bnet_mask'])
            # The VLAN ID of the VBR
            request.set_VlanId(params['vlan_id'])

            response = client.do_action_with_exception(request)
            response_json = json.loads(response)
            # Check if the VBR is in the available state
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_vbr_status,
                                        'active', response_json['
VbrId']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def delete_virtual_border_router(self, params):
        """
        delete_virtual_border_router: Delete the VBR
        """
        try:
```

```
            request = DeleteVirtualBorderRouterRequest.DeleteVirt
ualBorderRouterRequest()
            request.set_VbrId(params['vbr_id'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            if CheckStatus.check_status(TIME_DEFAULT_OUT, DEFAULT_TI
ME,
                                        self.describe_vbr_status,
                                        '', params['vbr_id']):
                return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_virtual_border_router(self, instance_id):
        """
        describe_virtual_border_router: Query VBRs
        """
        try:
            request = DescribeVirtualBorderRoutersRequest.DescribeVi
rtualBorderRoutersRequest()
            # The queried filter type
            request.add_query_param('Filter.1.Key', "VbrId")
            # The ID of the queried instance
            request.add_query_param('Filter.1.Value.1', instance_id)
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
            ExceptionHandler.client_exception(e)

    def describe_vbr_status(self, instance_id):
        """
        describe_virtual_border_router: Query the status of VBRs
        """
        response = self.describe_virtual_border_router(instance_id)
        if len(response['VirtualBorderRouterSet']['VirtualBor
derRouterType']) == 0:
            return ''
        return response['VirtualBorderRouterSet']['VirtualBor
derRouterType'][0]['Status']

    def describe_route_table(self, params):
        """
        describe_route_table: Query route tables
        """
        try:
            request = DescribeRouteTablesRequest.DescribeRo
uteTablesRequest()
            # The ID of the VRouter or VBR to which the route table
belongs
            request.set_RouterId(params['router_id'])
            # The type of the router to which the route table
belongs
            request.set_RouterType(params['router_type'])
            response = self.client.do_action_with_exception(request)
            response_json = json.loads(response)
            return response_json
        except ServerException as e:
            ExceptionHandler.server_exception(e)
        except ClientException as e:
```

```
            ExceptionHandler.client_exception(e)


def main():
    router_interface = RouterInterface(client)
    route_entry = RouteEntry(client)

    params = {}
    params['spec'] = "Large.2"
    params['role'] = "InitiatingSide"
    params['router_type'] = "VBR"
    params['opposite_region_id'] = "cn-hangzhou"
    params['opposite_router_type'] = "VRouter"
    params['access_point_id'] = "ap-cn-hangzhou-xx-x"

    # Create a VBR
    params['physical_connection_id'] = PC_ID
    params['local_gateway_ip'] = "116.xx.xx.254"
    params['peer_gateway_ip'] = "116.xx.xx.254"
    params['peering_subnet_mask'] = "255.255.255.252"
    params['vlan_id'] = 30
    vbr_json = router_interface.create_virtual_border_router(params)
    CommonUtil.log("create_virtual_border_router", vbr_json)

    # Create an initiator router interface
    params['router_id'] = vbr_json['VbrId']
    router_interface_json = router_interface.create_router_interf
ace_vbr(params)
    CommonUtil.log("create_router_interface", router_interface_json)

    # Create an acceptor router interface
    params['router_type'] = "VRouter"
    params['spec'] = "Negative"
    params['role'] = "AcceptingSide"
    params['router_id'] = ROUTER_ID2
    params['opposite_router_type'] = 'VBR'
    params['opposite_access_point_id'] = params['access_point_id']
    router_interface_json2 = router_interface.create_router_interf
ace(params)
    CommonUtil.log("create_router_interface", router_interface_json2
)

    # Modify the initiator router interface
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
    params['opposite_interface_id'] = router_interface_json2['
RouterInterfaceId']
    params['opposite_router_id'] = ROUTER_ID2
    params['opposite_router_type'] = "VRouter"
    modify_ri_json = router_interface.modify_router_interface_attrib
ute(params)
    CommonUtil.log("modify_router_interface_attribute", modify_ri_
json)

    # Modify the acceptor router interface
    params['router_interface_id'] = router_interface_json2['
RouterInterfaceId']
    params['opposite_interface_id'] = router_interface_json['
RouterInterfaceId']
    params['opposite_router_id'] = vbr_json['VbrId']
    params['opposite_router_type'] = "VBR"
    modify_ri_json2 = router_interface.modify_router_interf
ace_attribute(params)
```

```
    CommonUtil.log("modify_router_interface_attribute", modify_ri_
json2)

    # Query the initiator router interface
    describe_ri_json = router_interface.describe_router_interface(
router_interface_json['RouterInterfaceId'])
    CommonUtil.log("describe_router_interface", describe_ri_json)

    # Query the acceptor router interface
    describe_ri_json2 = router_interface.describe_router_interface(
router_interface_json2['RouterInterfaceId'])
    CommonUtil.log("describe_router_interface", describe_ri_json2)

    # Query the ID of the route table of the VBR
    params['router_id'] = vbr_json['VbrId']
    params['router_type'] = 'VBR'
    route_table_json = router_interface.describe_route_table(params)
    CommonUtil.log("describe_route_table", route_table_json)

    # Initiate a connection
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
    connect_ri_json = router_interface.connect_router_interface(
params)
    CommonUtil.log("connect_router_interface", connect_ri_json)

    # Create a route entry whose next hop is the initiator router
interface
    params['route_table_id'] = route_table_json["RouteTables"]["
RouteTable"][0]["RouteTableId"]
    params['destination_cidr_block'] = "0.0.0.0/0"
    params['nexthop_type'] = 'RouterInterface'
    params['nexthop_id'] = router_interface_json['RouterInterfaceId
']
    route_entry_json = route_entry.create_route_entry(params)
    CommonUtil.log("create_route_entry", route_entry_json)

    # Create a route entry whose next hop is the acceptor router
interface
    params['route_table_id'] = TABLE_ID2
    params['destination_cidr_block'] = "0.0.0.0/0"
    params['nexthop_type'] = 'RouterInterface'
    params['nexthop_id'] = router_interface_json2['RouterInterfaceId
']
    route_entry_json2 = route_entry.create_route_entry(params)
    CommonUtil.log("create_route_entry", route_entry_json2)

    # Delete the route entry whose next hop is the acceptor router
interface
    route_entry_json = route_entry.delete_route_entry(params)
    CommonUtil.log("delete_route_entry", route_entry_json)

    # Delete the route entry whose next hop is the initiator router
interface
    params['route_table_id'] = route_table_json["RouteTables"]["
RouteTable"][0]["RouteTableId"]
    params['nexthop_id'] = router_interface_json['RouterInterfaceId
']
    route_entry_json = route_entry.delete_route_entry(params)
    CommonUtil.log("delete_route_entry", route_entry_json)

    # Deactivate the initiator router interface
    params['router_interface_id'] = router_interface_json['
RouterInterfaceId']
```

```
    deactivate_ri_json = router_interface.deactivate_router_in
terface(params)
    CommonUtil.log("deactivate_router_interface", deactivate_ri_json
)

    # Deactivate the acceptor router interface
    params['router_interface_id'] = router_interface_json2['
RouterInterfaceId']
    deactivate_ri_json2 = router_interface.deactivate_router_in
terface(params)
    CommonUtil.log("deactivate_router_interface", deactivate
_ri_json2)

    # Delete the initiator router interface
    params['instance_id'] = router_interface_json['RouterInterfaceId
']
    router_interface_json = router_interface.delete_router_interface
(params)
    CommonUtil.log("delete_router_interface", router_interface_json)

    # Delete the acceptor router interface
    params['instance_id'] = router_interface_json2['RouterInte
rfaceId']
    router_interface_json2 = router_interface.delete_router_interf
ace(params)
    CommonUtil.log("delete_router_interface", router_interface_json2
)

    # Delete the VBR
    params['vbr_id'] = vbr_json['VbrId']
    vbr_json = router_interface.delete_virtual_border_router(params)
    CommonUtil.log("delete_virtual_border_router", vbr_json)

if __name__ == '__main__':
    sys.exit(main())
```

3. **Enter the directory where the *ec_vbr_vpc.py* file is stored, and then run the following command to interconnect a VBR and a VPC that are located in the same region and are under the same account.**

```
python ec_vbr_vpc.py
```

Result

Result

The system output is displayed as follows:

```
----------------------------create_virtual_border_router
----------------------------
{
  "VbrId": "vbr-bp1vudncgk9jtxxxxxxxx",
  "RequestId": "0C4FABDB-FF18-4E70-9E43-6DC03197F0EA"
}
----------------------------create_router_interface
----------------------------
{
  "RequestId": "11F14FA2-ECFA-4B27-A75D-7C6D5FECACDF",
  "RouterInterfaceId": "ri-bp1vabte8nbdexxxxxxxx"
```

```
}
--------------------------create_router_interface
--------------------------
{
  "RequestId": "72FAB86D-470B-40E4-8476-F9223A911486",
  "RouterInterfaceId": "ri-bp1mxkc61ly0nxxxxxxxx"
}
--------------------------modify_router_interface_attribute
--------------------------
{
  "RequestId": "93D5FFF7-6C05-41B1-92F0-AC8F1809BC98"
}
--------------------------modify_router_interface_attribute
--------------------------
{
  "RequestId": "1C461452-42BD-4649-B318-2214222B4FCA"
}
--------------------------describe_router_interface
--------------------------
{
  "TotalCount": 1,
  "RouterInterfaceSet": {
    "RouterInterfaceType": [
      {
        "BusinessStatus": "Normal",
        "CreationTime": "2019-04-30T02:54:22Z",
        "AccessPointId": "ap-cn-hangzhou-xx-x",
        "Role": "InitiatingSide",
        "OppositeRouterId": "vrt-bp141no9pds2bxxxxxxxx",
        "Spec": "Large.2",
        "Status": "Idle",
        "EndTime": "2999-09-08T16:00:00Z",
        "OppositeInterfaceSpec": "Negative",
        "RouterInterfaceId": "ri-bp1vabte8nbdexxxxxxxx",
        "RouterType": "VBR",
        "OppositeBandwidth": 0,
        "OppositeVpcInstanceId": "vpc-bp1v31by9jix2xxxxxxxx",
        "HasReservationData": false,
        "OppositeInterfaceBusinessStatus": "Normal",
        "OppositeRouterType": "VRouter",
        "OppositeRegionId": "cn-hangzhou",
        "RouterId": "vbr-bp1vudncgk9jtxxxxxxxx",
        "CrossBorder": false,
        "OppositeInterfaceOwnerId": "",
        "Bandwidth": 2048,
        "OppositeInterfaceId": "ri-bp1mxkc61ly0nxxxxxxxx",
        "ChargeType": "AfterPay"
      }
    ]
  },
  "PageNumber": 1,
  "RequestId": "3553DB38-A4A0-4453-976C-542E96B1B5A9",
  "PageSize": 10
}
--------------------------describe_router_interface
--------------------------
{
  "TotalCount": 1,
  "RouterInterfaceSet": {
    "RouterInterfaceType": [
      {
        "Status": "Idle",
        "OppositeRegionId": "cn-hangzhou",
        "BusinessStatus": "Normal",
```

```
        "OppositeRouterId": "vbr-bp1vudncgk9jtxxxxxxxx",
        "VpcInstanceId": "vpc-bp1v31by9jix2xxxxxxxx",
        "RouterInterfaceId": "ri-bp1mxkc61ly0nxxxxxxxx",
        "CreationTime": "2019-04-30T02:54:24Z",
        "RouterType": "VRouter",
        "OppositeInterfaceOwnerId": "",
        "RouterId": "vrt-bp141no9pds2bxxxxxxxx",
        "Bandwidth": 0,
        "OppositeInterfaceId": "ri-bp1vabte8nbdexxxxxxxx",
        "EndTime": "2999-09-08T16:00:00Z",
        "ChargeType": "AfterPay",
        "OppositeAccessPointId": "ap-cn-hangzhou-xx-x",
        "HasReservationData": false,
        "CrossBorder": false,
        "OppositeInterfaceBusinessStatus": "Normal",
        "Spec": "Negative",
        "OppositeRouterType": "VBR",
        "Role": "AcceptingSide"
      }
    ]
  },
  "PageNumber": 1,
  "RequestId": "217D8D94-C508-4285-8101-7DE3B53A88A5",
  "PageSize": 10
}
--------------------------describe_route_table
--------------------------
{
  "TotalCount": 1,
  "PageNumber": 1,
  "RequestId": "CA6BBE52-DF5E-496A-93CF-9857AF22D2AC",
  "PageSize": 10,
  "RouteTables": {
    "RouteTable": [
      {
        "RouteTableId": "vtb-bp1s126yz0swpxxxxxxxx",
        "RouteEntrys": {
          "RouteEntry": []
        },
        "CreationTime": "2019-04-30T02:54:19Z",
        "VSwitchIds": {
          "VSwitchId": []
        },
        "ResourceGroupId": "",
        "VRouterId": "vbr-bp1vudncgk9jtxxxxxxxx",
        "RouteTableType": "System"
      }
    ]
  }
}
--------------------------connect_router_interface
--------------------------
{
  "RequestId": "93476D4E-6C08-44B8-83E4-5759E1A08F29"
}
--------------------------create_route_entry
--------------------------
{
  "RequestId": "66D4F46B-5E0F-4565-8740-845EC26EBE00"
}
--------------------------create_route_entry
--------------------------
{
  "RequestId": "E1F99FEF-2499-40A7-84ED-6F9D440D4FF8"
```

```
}
----------------------------delete_route_entry
----------------------------
{
  "RequestId": "8C983886-F058-4234-A6D7-ECDEE2C2D945"
}
----------------------------delete_route_entry
----------------------------
{
  "RequestId": "DDEC56B9-CDD4-4D0A-B460-040B85B97FE9"
}
----------------------------deactivate_router_interface
----------------------------
{
  "RequestId": "04069B7C-6A42-43F2-A086-50F802940045"
}
----------------------------deactivate_router_interface
----------------------------
{
  "RequestId": "B2EBF829-E1C0-43E5-9BAD-DFFCBFA301F7"
}
----------------------------delete_router_interface
----------------------------
{
  "RequestId": "20EEC4A6-E468-4F3C-A743-89193F7504E1"
}
----------------------------delete_router_interface
----------------------------
{
  "RequestId": "59DCE168-B9CC-43A3-A54F-0D8C194BABE0"
}
----------------------------delete_virtual_border_router
----------------------------
{
  "RequestId": "E1D71EEC-FE9F-4834-8780-19EBBD91A638"
}
```