

An abstract background featuring a collection of 3D rectangular blocks in white, grey, and red, arranged in a staggered, architectural pattern.

FUJITSU Cloud Service K5 IaaS API User Guide

Version 1.18.2
FUJITSU LIMITED

All Rights Reserved, Copyright FUJITSU LIMITED 2015-2018

Preface

Purpose of this Manual

This document explains how to use the FUJITSU Cloud Server K5 IaaS REST API by providing sample scripts. It consists of two parts:

- Preparing the environment to use the API which includes shell setup as well as managing projects, users, groups and roles.
- API usage detailing how to create a basic implementation as well as how to delete resources from an implementation.

Audience for this Manual

This manual is intended for those involved in the planning and developing of applications or services using K5 IaaS.

This manual is written on the assumption that the reader possesses the following knowledge:

- Basic knowledge of virtualization technology (hypervisors, virtual servers, virtual storage, virtual networks)
- Basic knowledge of OpenStack
- Basic knowledge of your OS
- Basic knowledge of the Internet and Intranet
- Basic security knowledge
- Basic knowledge of system operation, including backups, monitoring, and redundancy

Target region of this document

The target regions of this document are East Japan Region 1, East Japan Region 2, West Japan Region 1, West Japan Region 2, UK Region 1, Finland Region 1, Germany Region 1, Spain Region 1, US Region 1.

Organisation of Manuals

Refer to the related manuals listed below according to your purposes and methods of use.

Manual	Purpose and usage
FUJITSU Cloud Service K5 IaaS API User Guide (this document)	This document provides instructions on how to use the K5 REST APIs, including how to build an API execution environment and how to use a sample scripts.
FUJITSU Cloud Service K5 IaaS Features Handbook (hereinafter "Features Handbook")	This document explains the functions provided by this service in detail.
FUJITSU Cloud Service K5 IaaS API Reference (hereinafter "API Reference")	This document provides detailed information about how to use REST API.
FUJITSU Cloud Service K5 IaaS Heat Template Specifications	This document explains the format of the Heat Orchestration Template (HOT) used for orchestration.
FUJITSU Cloud Service K5 IaaS Service Portal User Guide	This document explains how to use the functions provided by this service via Service Portal (Web GUI).

Manual	Purpose and usage
FUJITSU Cloud Service K5 Portal User Guide (hereinafter "K5 Portal User Guide")	This document explains how to use the functions, including registration and user information management, provided by the K5 Portal.

Abbreviations used in this Manual

In this manual, product names are abbreviated as follows.

Official name	Abbreviation	
FUJITSU Cloud Service K5 IaaS	K5 IaaS	
Microsoft® Windows Server® 2012 SE R2	Windows 2012 R2	Windows
Microsoft® Windows Server® 2008 SE R2	Windows 2008 R2	
Microsoft® Windows Server® 2008 EE R2		
Red Hat® Enterprise Linux® 6.5(for Intel64)	RHEL6.5	Linux
Community Enterprise Operating System 6.5	CentOS6.5	CentOS
Red Hat Update Infrastructure	RHUI	
Windows Server Update Services	WSUS	

Trademarks

- Microsoft, Windows, Windows Server and other Microsoft product names and model names are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- Java is a registered trademark of Oracle Corporation and its subsidiaries or affiliates in the United States and/or other countries.
- Xeon is a trademark of Intel Corporation in the United States and/or other countries.
- Linux® is a registered trademark of Linus Torvalds in the United States and/or other countries.
- Red Hat and Red Hat Enterprise Linux are trademarks of Red Hat, Inc. registered in the United States and/or other countries.
- Ubuntu is a registered trademark of Canonical Ltd.
- The OpenStack Word Mark is either a registered trademark/service mark or trademark/service mark of the OpenStack Foundation, in the United States and other countries and is used with the OpenStack Foundation's permission.
- Other company names and product names mentioned in this manual are trademarks or registered trademarks of their respective companies.

In this manual, the registered trademark symbols (™ or ®) next to system names or product names have been omitted.

Export Administration Regulations

When exporting or giving this document to a third party, be sure to familiarize yourself with the regulations related to export administration valid in your country of residence and the United States, and follow the necessary procedures.

Note....

- The content of this manual may change without prior notice.
- The reproduction of this manual without permission is prohibited.
- Fujitsu do not assume responsibility for any violation of patent rights or any other rights of a third party that may occur due to the use of the data in this manual.

Revision History

Version	Update date	Section	Changes
1.1	January 18, 2016	All chapters	Corrected typographical errors
		All chapters	Corrected drawings
		Creating the setup shells on page 5	Added explanations
1.2	February 18, 2016	Creating the setup shells on page 5	Added explanations
		Assigning and referencing a role on page 18	Corrected typographical errors
		All chapters	Added jumping to index function
1.3	April 1, 2016	Creating the setup shells on page 5	Added explanations
		All chapters	Delete expressions of user management
1.4	October 6, 2016	Cautionary notes regarding the use of APIs on page 10	Added "Cautionary notes regarding the use of APIs"
		Environment on page 5	Added explanations
		Creating the setup shells on page 5	Added explanations
		Creating a group on page 13	Added explanations
		Adding a user to a group on page 14	Added explanations
		Deleting a group on page 15	Added explanations
		All chapters	Corrected typographical errors
1.5	December 27, 2016	Preparing to create a virtual server (listing virtual server types) on page 78	Explanation correction
		All chapters	Corrected typographical errors

Version	Update date	Section	Changes
		Creating the setup shells on page 5	Corrected typographical errors
		Creating a security group rule on page 35	Corrected typographical errors
1.6	February 27, 2016	System structure diagram on page 21	Explanation correction
		Creating certificates for SSL-VPN	Added explanations
		Registering SSL-VPN certificates	Added explanations
		Creating a key container for SSL-VPN	Added explanations
		Allocating a global IP address for SSL-VPN	Added explanations
		Creating a VPN service	Added explanations
		Creating an SSL-VPN connection	Added explanations
		Configuring a connection to a VPN client	Added explanations
		Deleting an SSL-VPN connection	Added explanations
		Deleting a VPN service	Added explanations
		Deleting a global IP address for SSL-VPN	Added explanations
		Deleting a container for SSL-VPN keys	Added explanations
		Deleting certificates for SSL-VPN	Added explanations
		Chapter 4: Appendix	Revised the structure
1.7	April 25, 2017	System structure diagram on page 21	Explanation correction
		SSL-VPN connection (V2 service/K5 client certificate) on page 51	Added explanations
		SSL-VPN connection (V2 service/self-signed certificate) on page 58	Added explanations
		Deleting an SSL-VPN connection (V2 service) on page 101	Added explanations
		SSL-VPN connection (V1 service)	Deleted explanations
		Deleting an SSL-VPN connection (V1 service)	Deleted explanations

Version	Update date	Section	Changes
1.8	June 6, 2017	Creating an SSL-VPN connection (V2 service/K5 client certificate) on page 52	Corrected typographical errors
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Corrected typographical errors
		Registering SSL-VPN certificates (V2 service/self-signed certificate) on page 62	Corrected typographical errors
		Creating an SSL-VPN connection (V2 service/self-signed certificate) on page 70	Corrected typographical errors
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Corrected typographical errors
		Deleting an SSL-VPN connection (V2 service) on page 101	Corrected typographical errors
		Deleting an SSL-VPN connection (V2 service) on page 101	Corrected typographical errors
		Deleting a container for SSL-VPN keys (V2 service) on page 102	Corrected typographical errors
		Deleting certificates for SSL-VPN (V2 service) on page 102	Corrected typographical errors
1.9	Jun 19, 2017	Creating the setup shells on page 5	Explanation correction
		Retrieving the authentication token on page 9	Explanation correction
		System structure diagram on page 21	Corrected typographical errors
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Explanation correction
		Converting certificate formats on page 112	Explanation correction
1.10	July 26, 2017	Introduction on page 1	Added explanations
		Creating a security group rule on page 35	Explanation correction
		Examples of security group rules on page 37	Corrected typographical errors
		Creating a firewall rule (IP address specification) on page 39	Explanation correction

Version	Update date	Section	Changes
		Creating a firewall rule (port number specification) on page 41	Explanation correction
		Creating a firewall rule (deny rule) on page 44	Explanation correction
		Creating a firewall policy on page 46	Explanation correction
		Creating a firewall on page 48	Explanation correction
		Creating a subnet on page 26	Explanation correction
		Creating an SSL-VPN connection (V2 service/self-signed certificate) on page 70	Explanation correction
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Explanation correction
		Creating an SSL-VPN connection (V2 service/K5 client certificate) on page 52	Explanation correction
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Explanation correction
		Creating a port on page 79	Explanation correction
		Converting certificate formats on page 112	Added explanations
1.11	Sept 1, 2017	All chapters	Added "\" (backslash) to indicate the positions of line feeds in commands
		Creating a key pair on page 75	Explanation correction
		Creating the setup shells on page 5	Explanation correction
1.12	Oct 19, 2017	Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Deleted explanations
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Deleted explanations
1.13	Dec 4 , 2017	Preface on page ii	Corrected typographical errors
		Creating a subnet on page 26	Corrected explanation
		Creating a VPN service (V2 service/K5 client certificate) on page 51	Corrected explanation
		Creating a VPN service (V2 service/self-signed certificate) on page 68	Corrected explanation

Version	Update date	Section	Changes
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Added explanations
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Added explanations
		Creating a group on page 13	Changed the locations of line feeds in commands
		Creating a VPN service (V2 service/K5 client certificate) on page 51	
		Creating a VPN service (V2 service/self-signed certificate) on page 68	
		Creating a virtual server (Windows and port specification) on page 84	
		Retrieving a global IP address and assigning it to a virtual server on page 91	
		Creating a key container for SSL-VPN (V2 service/self-signed certificate) on page 66	Corrected typographical errors
1.14	Jan 11 , 2018	Environment on page 5	Corrected explanation
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Added explanations
		Configuring the API access environment (Windows) on page 113	Added explanations
1.15	January 26, 2018	Retrieving the authentication token on page 9	Corrected output results
		Creating an SSL-VPN connection (V2 service/K5 client certificate) on page 52	Added explanations
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Corrected explanation
		Creating an SSL-VPN connection (V2 service/self-signed certificate) on page 70	Added explanations
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Corrected explanation
1.16	February 22, 2018	Retrieving the authentication token on page 9	Corrected typographical errors
		Changing the information of a virtual router (attaching to an external network) on page 31	Added explanations
		SSL-VPN connection (V2 service/K5 client certificate) on page 51	Added explanations
		Configuring a connection to a VPN client (V2 service/K5 client certificate) on page 56	Added explanations
		SSL-VPN connection (V2 service/self-signed certificate) on page 58	Added explanations
		Registering SSL-VPN certificates (V2 service/self-signed certificate) on page 62	Corrected explanation

Version	Update date	Section	Changes
		Creating a key container for SSL-VPN (V2 service/self-signed certificate) on page 66	Corrected explanation
		Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Corrected explanation
1.17	March 22, 2018	Creating a subnet on page 26	Added explanations
		Creating a security group rule on page 35	Corrected typographical errors
		High-security configuration for SSL-VPN connection (V2 service) on page 94	Added explanations
1.17.1	March 30, 2018	High-security configuration for SSL-VPN connection (V2 service) on page 94	Added explanations
1.18	April 19, 2018	Creating a security group rule on page 35	Corrected explanation
		Creating a firewall on page 39	Corrected explanation
		Simple configuration for SSL-VPN connection (V2 service) on page 51	Corrected chapter name
		Creating an SSL-VPN connection (V2 service/K5 client certificate) on page 52	Corrected explanation
		Creating an SSL-VPN connection (V2 service/self-signed certificate) on page 70	Corrected explanation
1.18.1	June 20, 2018	Configuring a connection to a VPN client (V2 service/self-signed certificate) on page 73	Corrected hyper link
1.18.2	June 29, 2018	Preface on page ii	Target region names added
		Creating the setup shells on page 5	corrected a typo

Contents

Part 1: Introduction.....	1
1.1 Commands used in this manual.....	2
Part 2: Preparing the environment.....	4
2.1 Building the API operating environment.....	5
2.1.1 Environment.....	5
2.1.2 Creating the setup shells.....	5
2.1.3 Retrieving the authentication token.....	9
2.1.4 Cautionary notes regarding the use of APIs.....	10
2.2 Managing projects.....	11
2.2.1 Creating a project.....	11
2.3 Managing groups.....	13
2.3.1 Creating a group.....	13
2.3.2 Adding a user to a group.....	14
2.3.3 Deleting a group.....	15
2.4 Managing users and groups.....	17
2.4.1 Listing the existing roles.....	17
2.4.2 Assigning and referencing a role.....	18
Part 3: Creating resources.....	20
3.1 Assumed system configuration.....	21
3.1.1 System structure diagram.....	21
3.2 Building a network environment.....	24
3.2.1 Creating a network.....	24
3.2.2 Create a subnet.....	26
3.2.2.1 Creating a subnet.....	26
3.2.2.2 Adding routing.....	28
3.2.3 Creating a virtual router.....	30
3.2.4 Changing the information of a virtual router (attaching to an external network).....	31
3.2.5 Changing the information of a virtual router (attaching to a subnet).....	32
3.3 Security group settings.....	34
3.3.1 Creating a security group.....	34
3.3.2 Creating a security group rule.....	35
3.3.3 Examples of security group rules.....	37
3.4 Creating a firewall.....	39
3.4.1 Creating a firewall rule (IP address specification).....	39
3.4.2 Creating a firewall rule (port number specification).....	41
3.4.3 Creating a firewall rule (ICMP permission).....	42
3.4.4 Creating a firewall rule (deny rule).....	44
3.4.5 Creating a firewall policy.....	46
3.4.6 Creating a firewall.....	48

3.5 Simple configuration for SSL-VPN connection (V2 service).....	51
3.5.1 SSL-VPN connection (V2 service/K5 client certificate).....	51
3.5.1.1 Creating a VPN service (V2 service/K5 client certificate).....	51
3.5.1.2 Creating an SSL-VPN connection (V2 service/K5 client certificate).....	52
3.5.1.3 Configuring a connection to a VPN client (V2 service/K5 client certificate).....	56
3.5.2 SSL-VPN connection (V2 service/self-signed certificate).....	58
3.5.2.1 Creating certificates for SSL-VPN (V2 service/self-signed certificate).....	59
3.5.2.2 Registering SSL-VPN certificates (V2 service/self-signed certificate).....	62
3.5.2.3 Creating a key container for SSL-VPN (V2 service/self-signed certificate).....	66
3.5.2.4 Creating a VPN service (V2 service/self-signed certificate).....	68
3.5.2.5 Creating an SSL-VPN connection (V2 service/self-signed certificate).....	70
3.5.2.6 Configuring a connection to a VPN client (V2 service/self-signed certificate)....	73
3.6 Create a virtual server.....	75
3.6.1 Creating a key pair.....	75
3.6.2 Preparing to create a virtual server (listing virtual server images).....	76
3.6.3 Preparing to create a virtual server (listing virtual server types).....	78
3.6.4 Creating a port.....	79
3.6.5 Creating a virtual server (CentOS and port specification).....	80
3.6.6 Creating a virtual server (CentOS and DHCP retrieval).....	82
3.6.7 Creating a virtual server (Windows and port specification).....	84
3.6.8 Creating a virtual server (Windows and DHCP retrieval).....	87
3.6.9 Retrieving a virtual server port.....	90
3.6.10 Retrieving a global IP address and assigning it to a virtual server.....	91
3.7 Confirming connection.....	93
3.7.1 Logging in to a virtual server (CentOS).....	93
3.7.2 Logging in to a virtual server (Windows).....	93
3.8 High-security configuration for SSL-VPN connection (V2 service).....	94
3.8.1 System structure diagram.....	94
3.8.2 Building a network environment.....	94
3.8.2.1 Building a management network.....	95
3.8.2.2 Building a business service network.....	96
3.8.2.2.1 Attaching ports.....	97
3.8.2.2.2 Configuring interface files (Linux OS only).....	98
Part 4: Deleting resources.....	99
4.1 Deleting the basic system structure.....	100
4.1.1 The order of deletion.....	100
4.2 Deleting an SSL-VPN connection (V2 service).....	101
4.2.1 Deleting an SSL-VPN connection (V2 service).....	101
4.2.2 Deleting a VPN service (V2 service).....	101
4.2.3 Deleting a container for SSL-VPN keys (V2 service).....	102
4.2.4 Deleting certificates for SSL-VPN (V2 service).....	102
4.3 Deleting a firewall.....	104
4.3.1 Deleting a firewall.....	104
4.3.2 Deleting a firewall policy.....	104
4.3.3 Deleting a firewall rule.....	104
4.4 Deleting a virtual server.....	106
4.4.1 Deleting a global IP address.....	106

4.4.2 Deleting a virtual server.....	106
4.4.3 Deleting a port.....	106
4.4.4 Deleting a key pair.....	107
4.5 Deleting a security group.....	108
4.5.1 Deleting a security group rule.....	108
4.5.2 Deleting a security group.....	108
4.6 Deleting a network environment.....	109
4.6.1 Detaching a virtual router from a subnet.....	109
4.6.2 Deleting a virtual router.....	109
4.6.3 Deleting a subnet.....	109
4.6.4 Deleting a network.....	110
Appendix A: Appendix.....	111
A.1 Specification format for payloads during certificate registration.....	111
A.2 Converting certificate formats.....	112
A.3 Configuring the API access environment (Windows).....	113

Part 1: Introduction

Topics:

- [Commands used in this manual](#)

1.1 Commands used in this manual

This section explains an overview of the commands and the command options indicated in this manual.

Handling of the commands indicated in this manual

When writing this manual, the operation of multiple OSS commands when using the K5 APIs has been confirmed, but this does not mean that the OSS commands in this manual themselves are supported.

When using options other than those indicated in the OSS command explanations below, please be aware that you do so at your own risk.

The command examples given in this manual assume that "bash" is being used for execution.

cURL command

An OSS command for performing data communication that is compatible with various protocols. In this manual it is used for executing K5 IaaS APIs using HTTPS communication

Command options used in this manual

Options	Explanation
-s	Specify this to prevent display of the progress of communication or command syntax errors.
-S	Specify this to display command syntax errors when the "-s" option is specified.
-i	Specify this to display the HTTP headers of responses.
-H	Specify this to use HTTP headers in requests.
-X	Specify the request method.
-d	Specify the data to send as the request body.
--cert	Specify the client certificate to use for client authentication.
--key	Specify the private key corresponding to the client certificate used for client authentication.

jq command

An OSS command used to operate JSON format data from the command line. In this manual it is used to add line feeds to the JSON data contained in response bodies and display the data clearly.

No command options are used in this manual.

Openssl command

An OSS command for encryption related operations, such as the creation and format conversion of the client certificates used in encryption. In this manual it is used to convert the format of the K5 client certificate used in client authentication.

Command options used in this manual

Options	Explanation
x509	Specify this to operate X.509 format client certificates. Formats such as PEM and DER are included in the X.509 format.
pkcs12	Specify this to operate PKCS#12 format client certificates.
rsa	Specify this to operate private keys.
-in	Specify this to indicate the client certificate that is the target of operation
-out	Specify the name of the output file when performing format conversion, etc.
-clcerts	Specify this when converting a PKCS#12 format client certificate to a PEM format client certificate.
-nokeys	Specify this when extracting a private key from a PKCS#12 client certificate.
-nodes	Specify this when not performing encryption using a passphrase when extracting a private key from a PKCS#12 format client certificate.
-modulus	Specify this to display the information necessary to confirm the consistency of a client certificate and a private key.
-noout	Specify this when not outputting a file during operation of a client certificate or private key.

Part 2: Preparing the environment

Topics:

- [Building the API operating environment](#)
- [Managing projects](#)
- [Managing groups](#)
- [Managing users and groups](#)

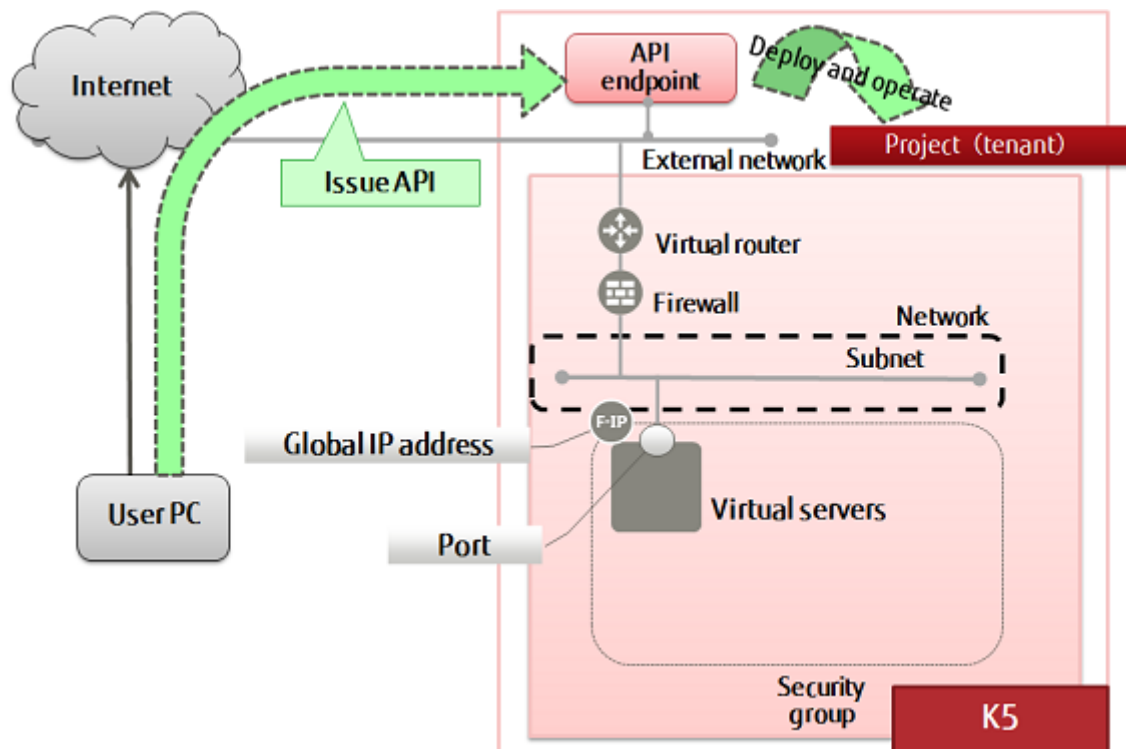
2.1 Building the API operating environment

2.1.1 Environment

This section provides an overview of the API usage environment that is to be built.

Access the API usage environment via the Internet from a user PC (a client PC. The figure below assumes that CentOS user environment is being used. If you are using Windows, install Cygwin to enable execution of Linux commands.) Use an external endpoint, a virtual server, etc. to access the API usage environment.

For details on installing Cygwin, refer to the appendix "[Configuring the API access environment \(Windows\)](#)" on page 113".



2.1.2 Creating the setup shells

Create the shells that will perform setup for running the API.

This section explains how to create a working directory and two scripts for setting up environment variables and authenticating services.

1. Create the working directory.

Create the working directory under the user's home directory, and change to it.

```
$ mkdir <anyName>
```

```
$ cd <anyName>
```

2. Create init.sh and init_global.sh.

Creates the scripts "init.sh" and "init_global.sh" that set account information as environment variables of the current directory.



Important

"init.sh" and "init_global.sh" configure environment variables to use regional services and global services respectively. For the explanations of regional services and global services, refer to the "Features Handbook".

init.sh

```
#!/bin/bash
# Account information.
DOMAIN_NAME=<contractNum(Domain)>
DOMAIN_ID=<domainID>
TENANT_ID=<projID>
PROJECT_ID=$TENANT_ID
USER_NAME=<userName>
USER_PW=<password>

# Endpoint shortcut.
echo "EP initial setup."

TOKEN=https://identity.<Region Identifier>.cloud.global.fujitsu.com
IDENTITY=$TOKEN
NETWORK=https://networking.<Region Identifier>.cloud.global.fujitsu.com
COMPUTE=https://compute.<Region Identifier>.cloud.global.fujitsu.com
CEILOMETER=https://telemetry.<Region Identifier>.cloud.global.fujitsu.com
TELEMETRY=$CEILOMETER
DB=https://database.<Region Identifier>.cloud.global.fujitsu.com
BLOCKSTORAGE=https://blockstorage.<Region Identifier>¥
.cloud.global.fujitsu.com
HOST_BLOCKSTORAGEV2=$BLOCKSTORAGE
OBJECTSTORAGE=https://objectstorage.<Region Identifier>¥
.cloud.global.fujitsu.com
ORCHESTRATION=https://orchestration.<Region Identifier>¥
.cloud.global.fujitsu.com
LB=https://loadbalancing.<Region Identifier>.cloud.global.fujitsu.com
AUTOSCALE=https://autoscale.<Region Identifier>.cloud.global.fujitsu.com
IMAGE=https://image.<Region Identifier>.cloud.global.fujitsu.com
MAILSERVICE=https://mail.<Region Identifier>.cloud.global.fujitsu.com
NETWORK_EX=https://networking-ex.<Region Identifier>¥
.cloud.global.fujitsu.com
DNS=https://dns.gls.cloud.global.fujitsu.com
COMPUTE_SAP=https://compute-w.<Region Identifier>¥
.cloud.global.fujitsu.com
KEYMANAGEMENT=https://keymanagement.<Region Identifier>¥
.cloud.global.fujitsu.com
SOFTWARE=https://software.<Region Identifier>.cloud.global.fujitsu.com
VMIMPORT=https://vmimport.<Region Identifier>.cloud.global.fujitsu.com
VMEXPORT=https://import-export.<Region Identifier>¥
.cloud.global.fujitsu.com

# Initial setup
NAME_FORMAT="TES_$(date "+%m%d")_$(who am I | cut -d " " -f1)_"
# Other
alias curl='curl --tlsv1.2'
SCRIPT_PATH=`pwd`
RES_DIR=response
RES_PATH=$SCRIPT_PATH/$RES_DIR
```

init_global.sh

```
#!/bin/bash
# Account information.
DOMAIN_NAME=<contractNum(Domain)>
DOMAIN_ID=<domainID>
```

```
TENANT_ID=<projID>
PROJECT_ID=$TENANT_ID
USER_NAME=<userName>
USER_PW=<password>

# Endpoint shortcut.
echo "EP initial setup."
TOKEN=https://identity.gls.cloud.global.fujitsu.com
IDENTITY=$TOKEN
CONTRACT=https://contract.gls.cloud.global.fujitsu.com
BILL=https://billing.gls.cloud.global.fujitsu.com
DNS=https://dns.gls.cloud.global.fujitsu.com
CATALOG=https://catalog.gls.cloud.global.fujitsu.com

# Initial setup
NAME_FORMAT="TES_$(date "+%m%d")_$(who am I | cut -d " " -f1)_"
# Other
alias curl='curl --tlsv1.2'
SCRIPT_PATH='pwd'
RES_DIR=response
RES_PATH=$SCRIPT_PATH/$RES_DIR
```



Important

- For the information necessary for this shell, such as the domain ID, project name, project ID, user ID, etc., refer to "K5 Portal"- "IaaS Management" and then enter the information.
- For Region Identifier, refer to the "Features Handbook", and replace it with a Region Identifier corresponding to the region you use.
- This guide uses the command line tool "cURL". This guide has been verified for operation with version 7.45.0 of cURL.
- When using a DNS service, the following must also be performed.
 - Create a project in "Eastern Japan Region 1 (jp-east-1)", and register the users who will use the DNS service in that project.
 - Use a regional token.
- The email delivery service is only provided in "Eastern Japan Region 1 (jp-east-1)".

3. Create get_token.sh and get_global_token.sh.

Create the scripts "get_token.sh" and "get_global_token.sh", which will be used to retrieve the tokens for service authentication.



Important

- The script "get_token.sh" obtains a regional token, and the script "get_global_token.sh" obtains a global token. Select the script to execute based on the service you are using.
- When using "certificate and password authentication" as the authentication method, add the option below to the cURL command line.

```
--cert <Client Certificate Name> --key <Name of the Private Key for the Client Certificate>
```

The "get_token.sh" and "get_global_token.sh" explained in this section both contain descriptions for the "password authentication" and the "certificate and password authentication" patterns. Ensure that you comment out the unnecessary pattern.



Note

When using "certificate and password authentication", it is necessary to change the authentication method from the K5 portal and issue certificates in advance. For details, refer to the "K5 Portal User Guide".

In addition, when using a certificate for cURL commands, it is necessary to convert it to PEM format. For details on converting certificate formats, refer to ["Converting certificate formats"](#) on page 112".

This procedure assumes that the private key used in the cURL description is an "Unencrypted private key", as described in ["Converting certificate formats on page 112"](#).

get_token.sh

```
#!/bin/bash

## Script to retrieve token
. ~/<anyName>/init.sh
TMPFILE=~/<anyName>/token.txt

echo ""
echo "*****"
echo "**          Retrieve token          **"
echo "** (Display key after retrieving token) **"
echo "*****"
echo ""
echo ' ■Setting content'
echo '   endpoint':$TOKEN
echo '   domain_name':$DOMAIN_NAME
echo '   domain_id':$DOMAIN_ID
echo '   user_name':$USER_NAME
echo '   user_pw':$USER_PW
echo '   project_id':$PROJECT_ID

echo " ■CURL"
echo ' curl -X POST '$TOKEN'/v3/auth/tokens -H "Content-Type:application/json"
-H "Accept:application/json" -d'
echo ' {"auth":{"identity":{"methods":["password"],"password":{"user":{"domain":
{"name":"' $DOMAIN_NAME' "', "name": "' $USER_NAME' "', "password": "' $USER_PW' "'}}},
"scope": { "project": {"id": "' $PROJECT_ID' "'}}}}' | jq .

echo -n "***** Hit Enter Key *****"

read

# When using password authentication
curl -X POST -sS $TOKEN/v3/auth/tokens -H "Content-Type: application/json" \
-H "Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"' $DOMAIN_NAME' "', "name": "' $USER_NAME' "', "password":
"' $USER_PW' "'}}}, "scope": { "project": {"id": "' $PROJECT_ID' "'}}}}' | \
awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

# When using certificate and password authentication
curl -X POST -s $TOKEN/v3/auth/tokens --cert <Client Certificate Name>.pem \
--key <Name of the Private Key for the Client Certificate>.pem -H "Content-Type:
application/json" \
-H "Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"' $DOMAIN_NAME' "', "name": "' $USER_NAME' "', "password":
"' $USER_PW' "'}}}, "scope": { "project": {"id": "' $PROJECT_ID' "'}}}}' | awk '/X-Subject-
Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

OS_AUTH_TOKEN=`cat $TMPFILE | tr -d '¥r¥n'`

echo "=== Retrieved authentication token starts from here ==="
echo $OS_AUTH_TOKEN
echo "=== Retrieved authentication token ends here ==="
```

get_global_token.sh

```
#!/bin/bash
## Script to retrieve token
. ~/<anyName>/init_global.sh
```

```

TMPFILE=~/<anyName>/token.txt

echo ""
echo "*****"
echo "**          Retrieve token          **"
echo "** (Display key after retrieving token) **"
echo "*****"
echo ""
echo '■Setting content'
echo '  endpoint':$TOKEN
echo '  domain_name':$DOMAIN_NAME
echo '  domain_id':$DOMAIN_ID
echo '  user_name':$USER_NAME
echo '  user_pw':$USER_PW
echo '  project_id':$PROJECT_ID

echo "■CURL"
echo 'curl -X POST '$TOKEN'/v3/auth/tokens -H "Content-Type:application/json" -H
"Accept:application/json" -d'
echo '{"auth":{"identity":{"methods":["password"],"password":{"user":{"domain":
{"name":"$DOMAIN_NAME"},"name":"$USER_NAME","password":"$USER_PW"}}},
"scope":{"project":{"id":"$PROJECT_ID"}}}}' | jq .

echo -n "***** Hit Enter Key *****"

read

# When using password authentication
curl -X POST -ssi $TOKEN/v3/auth/tokens -H "Content-Type: application/json" -H
"Accept:application/json" -d '{"auth":{"identity":{"methods":["password"],"password":
{"user":{"domain":{"name":"$DOMAIN_NAME"},"name":"$USER_NAME","password":
"$USER_PW"}}}, "scope":{"project":{"id":"$PROJECT_ID"}}}}' | awk '/X-Subject-
Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

# When using certificate and password authentication
curl -X POST -si $TOKEN/v3/auth/tokens --cert <Client Certificate Name>.pem --
key <Name of the Private Key for the Client Certificate>.pem -H "Content-Type:
application/json" -H "Accept:application/json" -d '{"auth":{"identity":{"methods":
["password"],"password":{"user":{"domain":{"name":"$DOMAIN_NAME"},"name":
"$USER_NAME","password":"$USER_PW"}}}, "scope":{"project":{"id":
"$PROJECT_ID"}}}}' | awk '/X-Subject-Token/ {print $2}' > $TMPFILE | tr -d '¥r¥n'

OS_AUTH_TOKEN=`cat $TMPFILE | tr -d '¥r¥n'`

echo "=== Retrieved authentication token starts from here ==="
echo $OS_AUTH_TOKEN
echo "=== Retrieved authentication token ends here ==="

```

4. List the contents of the current directory.

List the content to confirm that the scripts have been created in the current directory.

```

$ ls -l
get_global_token.sh get_token.sh init.sh init_global.sh

```

2.1.3 Retrieving the authentication token

This section explains how to retrieve the token necessary for authentication, in order to use API.



It is assumed that the working directory and the setup scripts have been created.

Note

1. Change to the working directory.

```

$ cd <anyName>

```

2. Retrieve the token.

The script assigns the token to the OS_AUTH_TOKEN environment variable.

```
$ . ./get_token.sh
```

A response such as the following is output to the console window.

EP initial setup.

```
*****
**          Retrieve token          **
**(Display key after retrieving token)**
*****

- Setting content
  endpoint:https://identity.<Region Identifier>.cloud.global.fujitsu.com
  domain_name:<contractNum(Domain)>
  domain_id:<domainId>
  user_name:<userName>
  user_pw:<userPassword>
  project_id:<projectId>

- CURL
curl -X POST https://identity.<Region Identifier>.cloud.global.fujitsu.com/v3/auth/
tokens -H "Content-Type:application/json" -H "Accept:application/json" -d
{
  "auth": {
    "identity": {
      "methods": [
        "password"
      ],
      "password": {
        "user": {
          "domain": {
            "name": "<contractNumber (Domain)>"
          },
          "name": "<userName>",
          "password": "<userPassword>"
        }
      }
    },
    "scope": {
      "project": {
        "id": "<projectId>"
      }
    }
  }
}

=== Retrieved authentication token starts here ===
<anyString> (up to 32 halfwidth alphanumeric characters)
=== Retrieved authentication token ends here ===
```

Provided the authentication token can be confirmed, API operations can be performed in this environment.

2.1.4 Cautionary notes regarding the use of APIs

This section describes cautionary notes regarding the use of APIs.



Note

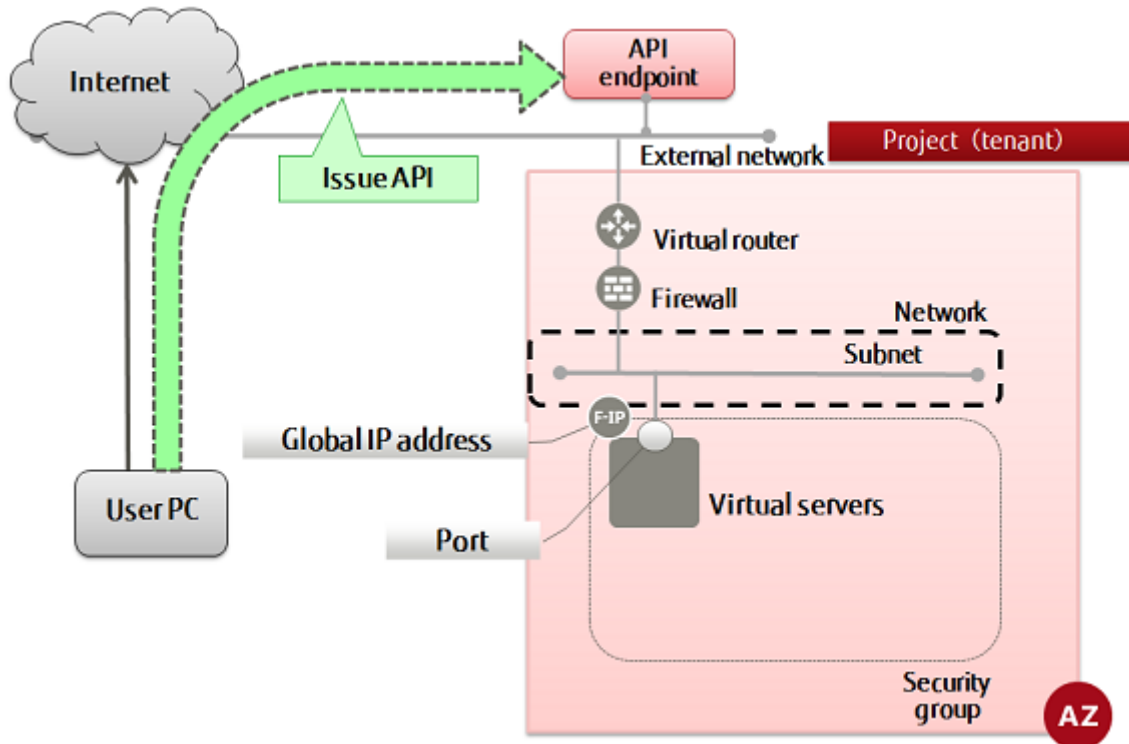
If a 5xx error is returned during execution of an API, wait a while and then execute the API again.

2.2 Managing projects

2.2.1 Creating a project

This section explains how to create a project and then check that it was created properly. In K5, the virtual resources, groups and users to be used in a contract are divided up and managed by project.

Note that a project can also be referred to as a "tenant" in the API.



You cannot delete projects that have been created.

Note

1. Set the environment variables below as follows:

```
$ TENANT_NAME=<projName>
```

```
$ DESCRIPTION=<projDesc>
```

2. Execute the following API:

```
$ curl -X POST -Ss $IDENTITY/v3/projects ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
-d '{"project": {"name": "'$TENANT_NAME'",  
"description": "'$DESCRIPTION'", "domain_id": "'$DOMAIN_ID'",  
"enabled": true}}' | jq .
```

3. The following response is output:

```
{  
  "project": {  
    "description": "<projDesc>",  
    "links": {  
      "self": "http://identity.cloud.global.fujitsu.com/v3/projects/<createdProjId>"
```

```

    },
    "enabled": true,
    "id": "<createdProjId>",
    "domain_id": "<contractOrgNoToWhichUserBelongs>",
    "name": "<createdProjName>"
  }
}

```

4. Check the created project:

```

$ curl -X GET -Ss $IDENTITY/v3/projects?domain_id=$DOMAIN_ID ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | ¥
jq .

```

5. If the project, including the specified project name, is output as follows, creation is complete.

```

{
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/projects",
    "previous": null,
    "next": null
  },
  "projects": [
    ...
    {
      "description": "<projDesc>",
      "links": {
        "self": "http://identity.cloud.global.fujitsu.com/v3/projects/<createdProjId>"
      },
      "enabled": true,
      "id": "<createdProjId>",
      "domain_id": "<contractOrgNoToWhichUserBelongs>",
      "name": "<createdProjName>"
    },
    ...
  ]
}

```


2.3 Managing groups

2.3.1 Creating a group

This section explains how to create a group (which can contain multiple users and is useful for role management) and then check that it was created properly.



A domain can contain multiple groups, but a group cannot belong to another group.

Note



It is necessary to use a global token when executing this API.

Note

1. Set the environment variables below as follows:

```
$ TEMP_GROUP_NAME=<newGroupName>
```

```
$ DOMAIN_ID=<domainName>
```

```
$ DESCRIPTION=<groupDesc>
```

2. Execute the following API:

```
$ curl -X POST -Ss $IDENTITY/v3/groups ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
-d '{"group": {"description": "' $DESCRIPTION' ",  
"domain_id": "' $DOMAIN_ID' ", "name": "' $TEMP_GROUP_NAME' "}}' | ¥  
jq .
```

The following response is output:

```
{  
  "group": {  
    "domain_id": "<domainNameThatGroupBelongsTo>",  
    "description": "<groupDesc>",  
    "id": "<newGroupId>",  
    "links": {  
      "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<newGroupId>"  
    },  
    "name": "<newGroupName>"  
  }  
}
```

3. List the groups to confirm that it has been created. Only groups of the same domain can be retrieved.

```
$ curl -X GET -Ss $IDENTITY/v3/groups?domain_id=$DOMAIN_ID ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
| jq .
```

If a list including the group names you specified, like the following, is returned, that means the creation of the group was successful.

```
{  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",  
    "previous": null,  
    "next": null  
  },  
  "groups": [  
    {  
      "domain_id": "<domainNameThatGroupBelongsTo>",  
      "description": "<groupDesc>",  
      "id": "<newGroupId>",  
      "links": {  
        "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<newGroupId>"  
      },  
      "name": "<newGroupName>"  
    }  
  ]  
}
```

```

...
{
  "domain_id": "<domainId>",
  "description": "<groupDesc>",
  "id": "<groupId>",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<groupId>"
  },
  "name": "<groupName>"
},
...
]
}

```

2.3.2 Adding a user to a group

This section explains how to add a user to a group and then check that it was added properly. Users in a group inherit the roles assigned to it.



It is necessary to use a global token when executing this API.

Note

1. Set the environment variables below as follows:

```
$ TMP_USER_ID=<userIdToBeAdded>
```

```
$ TMP_GROUP_ID=<groupId>
```

2. Execute the following API:

```
$ curl -i -X PUT -Ss $IDENTITY/v3/groups/$TMP_GROUP_ID/users/¥
$TMP_USER_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json"
```

The following response is output:

```
HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<nineDigitNum>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT
```

3. Set the environment variable below as follows, and list the users in the group.

```
$ TMP_GROUP_ID=<groupId>
```

```
$ curl -X GET -Ss $IDENTITY/v3/groups/$TMP_GROUP_ID/users ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | ¥
jq .
```

If a list including the users you added, like the following, is returned, that means the addition of users was successful.

```

{
  "users": [
    ...
    {
      "domain_id": "<domainId>",
      "name": "<userName>",

```

```

    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/users/<userId>"
    },
    "locale": "ja",
    "enabled": true,
    "id": "<userId>",
    "default_project_id": "<defaultProjId>",
    "description": "<userDesc>"
  },
  ...
],
"links": {
  "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<groupId>/users",
  "next": null,
  "previous": null
}
}

```

4. Users can also be deleted from a group. Follow the procedure below to delete a user from a group:

Set the environment variables below as follows:

```
$ TMP_USER_ID=<userIdToBeDeleted>
```

```
$ TMP_GROUP_ID=<groupId>
```

5. Execute the following API:

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TMP_GROUP_ID/users/$TMP_USER_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

The following response is output:

```

HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<nineDigitNum>_204
Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT

```

The method for checking if users have been successfully deleted is the same as for checking the addition of users.

2.3.3 Deleting a group

This section explains how to delete a group and then check that it was deleted properly.



It is necessary to use a global token when executing this API.

Note

1. Set the environment variable below as follows:

```
$ TEMP_GROUP_ID=<groupIdToBeDeleted>
```

2. Execute the following API:

```
$ curl -i -X DELETE -s $IDENTITY/v3/groups/$TEMP_GROUP_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

The following response is output:

```

HTTP/1.1 204 No Content
X-Fcx-Endpoint-Request: EXECUTED_REQ<nineDigitNum>_204

```

Vary: X-Auth-Token
Date: Tue, 17 Nov 2015 07:00:02 GMT

3. List the groups to confirm that the group has been successfully deleted.

```
$ curl -X GET -Ss $IDENTITY/v3/groups?domain_id=$DOMAIN_ID ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | ¥  
jq .
```

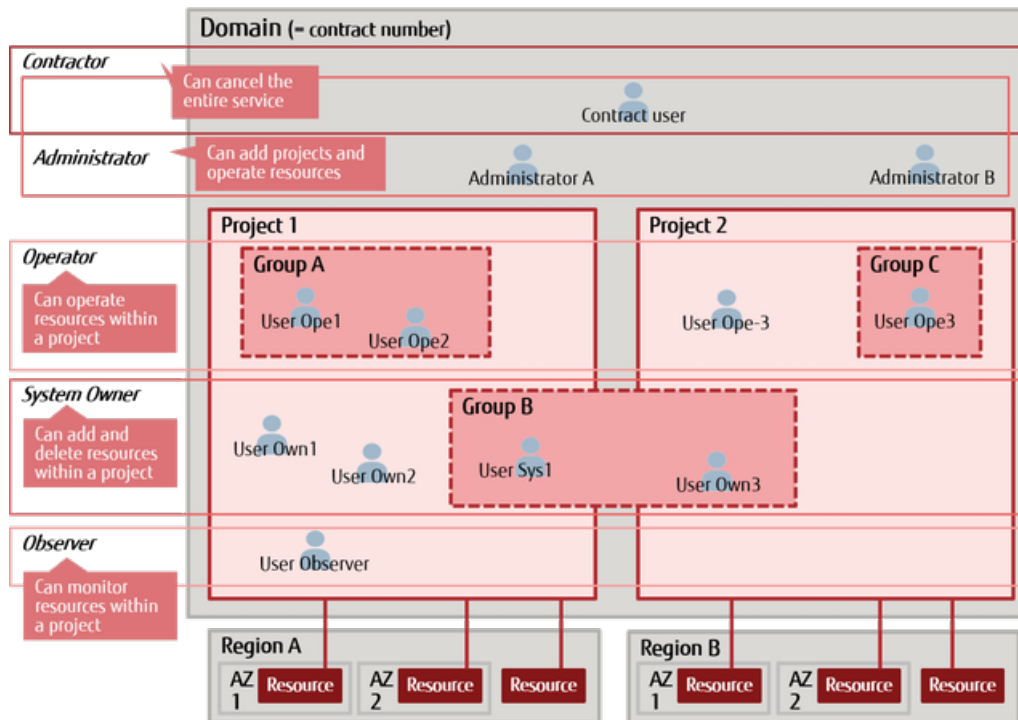
With a format such as the following, a list of groups in the same domain is displayed, and if the group targeted for deletion is not in the list, this indicates that deletion was successful.

```
{  
  "links": {  
    "self": "http://identity.cloud.global.fujitsu.com/v3/groups",  
    "previous": null,  
    "next": null  
  },  
  "groups": [  
    ...  
    {  
      "domain_id": "<domainId>",  
      "description": "<groupDesc>",  
      "id": "<groupId>",  
      "links": {  
        "self": "http://identity.cloud.global.fujitsu.com/v3/groups/<groupId>"  
      },  
      "name": "<groupName>"  
    },  
    ...  
  ]  
}
```

2.4 Managing users and groups

2.4.1 Listing the existing roles

This section explains how to list existing roles to determine the operating privileges of users.



Execute the following API:

```
$ curl -X GET -Ss $IDENTITY/v3/roles?domain_id=$DOMAIN_ID \
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | \
jq '.roles[] |select(.policy_role != true)'
```

The following response is output: Refer to the Service Specification for details on the privileges of each preset role.

```
{
  "id": "0739580a550d4a0f9c78f45a9f038c05",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/roles/0739580a550d4a0f9c78f45a9f038c05"
  },
  "name": "cpf_systemowner"
}
{
  "id": "20e572b88c544606863548f8abd4ff53",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/roles/20e572b88c544606863548f8abd4ff53"
  },
  "name": "cpf_operator"
}
{
  "id": "3af119c426a742999e7890f6d1f70b36",
  "links": {
    "self": "http://identity.cloud.global.fujitsu.com/v3/roles/3af119c426a742999e7890f6d1f70b36"
  },
  "name": "cpf_observer"
}
```

```

    "name": "cpf_admin"
  }
  {
    "id": "970ea0105b514c16828626fe4dd50960",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/970ea0105b514c16828626fe4dd50960"
    },
    "name": "cpf_observer"
  }
  {
    "id": "9fe2ff9ee4384b1894a90878d3e92bab",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/9fe2ff9ee4384b1894a90878d3e92bab"
    },
    "name": "_member_"
  }
  {
    "id": "df7d043a09d34a7c9e2bad15926ee097",
    "links": {
      "self": "http://identity.cloud.global.fujitsu.com/v3/roles/df7d043a09d34a7c9e2bad15926ee097"
    },
    "name": "cpf_org_manager"
  }
}

```

2.4.2 Assigning and referencing a role

This section explains how to assign a role to a user or group, and how to reference it.

To have users or groups participate in a project, select a role to assign.

1. Set the environment variables below as follows:

```
$ TMP_PROJECT_ID=<projIdThatUserOrGroupIsToParticipateIn>
```

```
$ TMP_USER_ID=<userToParticipateInProj>
```

or

```
$ TMP_GROUP=<groupToParticipateInProj>
```

```
$ TMP_ROLE_ID=<roleId>
```

2. Execute the following API:

```
$ curl -i -X PUT -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/¥
$TMP_USER_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json"
```

or

```
$ curl -i -X PUT -Ss $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/¥
$TMP_GROUP_ID/roles/$TMP_ROLE_ID -H "X-Auth-Token:$OS_AUTH_TOKEN" ¥
-H "Content-Type:application/json"
```

The following response is output:

```

HTTP/1.1 204 No Content
Vary: X-Auth-Token
Content-Length: 0
Date: Www, DD MMM yyyy hh:mm:ss GMT

```

3. Check the roles of users and groups.

To check the roles, the four following patterns are available.

- a. Project - Group
- b. Project - User
- c. Domain - Group
- d. Domain - User

Set the environment variables below as follows, according to the desired pattern:

1.

```
$ TMP_PROJECT_ID="<projId>"
```

2.

```
$ TMP_DOMAIN_ID="<domainId>"
```

3.

```
$ TMP_USER_ID="<userId>"
```

4.

```
$ TMP_GROUP_ID="<groupId>"
```

4. Execute the following API:

- ```
$ curl -X GET $IDENTITY/v3/projects/$TMP_PROJECT_ID/groups/$TMP_GROUP_ID/roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET $IDENTITY/v3/projects/$TMP_PROJECT_ID/users/$TMP_USER_ID/roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET $IDENTITY/v3/domains/$DOMAIN_ID/groups/$TMP_GROUP_ID/roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```
- ```
$ curl -X GET $IDENTITY/v3/domains/$DOMAIN_ID/users/$TMP_USER_ID/roles -H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" | jq .
```

Part 3: Creating resources

Topics:

- [Assumed system configuration](#)
- [Building a network environment](#)
- [Security group settings](#)
- [Creating a firewall](#)
- [Simple configuration for SSL-VPN connection \(V2 service\)](#)
- [Create a virtual server](#)
- [Confirming connection](#)
- [High-security configuration for SSL-VPN connection \(V2 service\)](#)

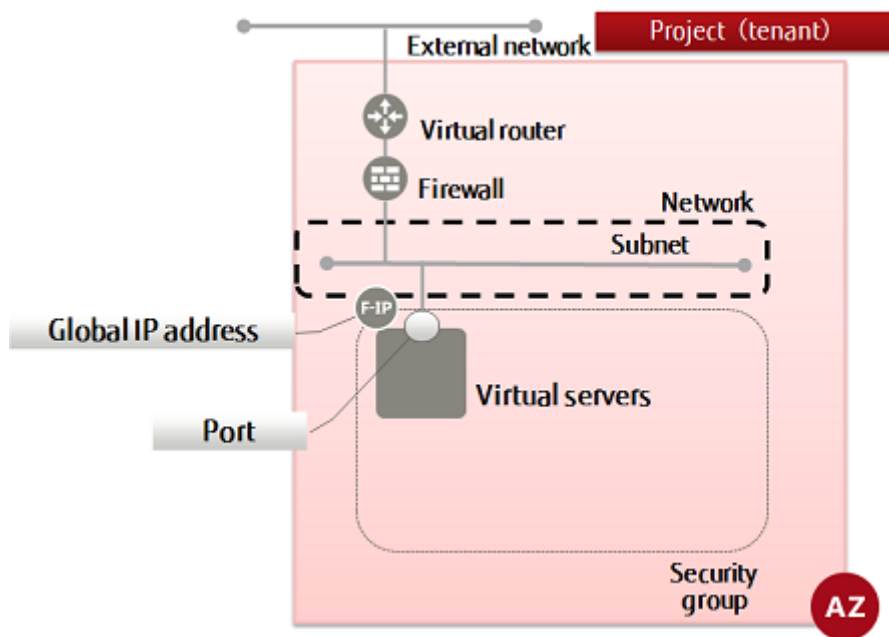
3.1 Assumed system configuration

3.1.1 System structure diagram

This section explains the system structures that can be created.

When releasing a virtual server on the Internet

This section describes the basic procedure in the K5 IaaS for configuring a system that will be released on the Internet.



Perform the following tasks in the K5 IaaS to create the system structure.

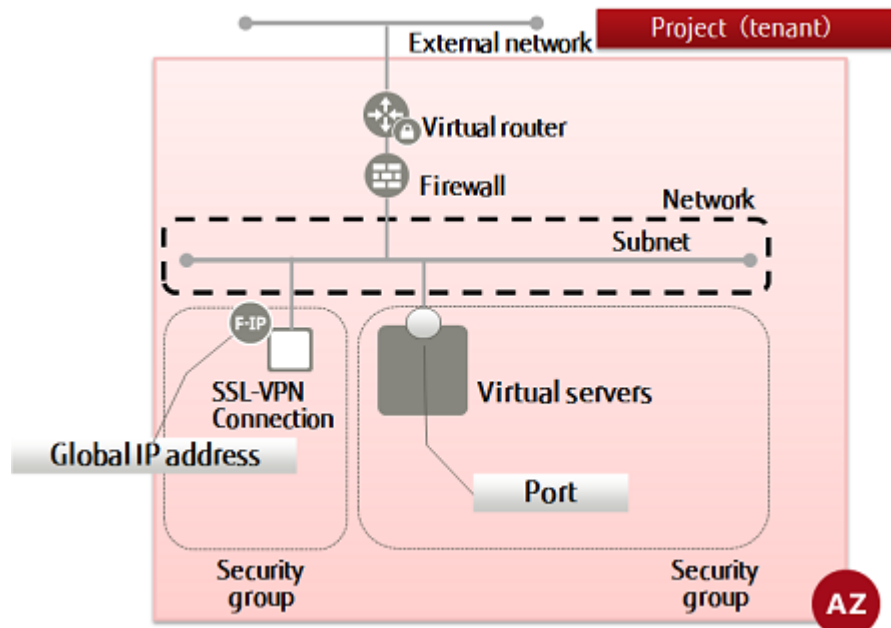
1. [Creating a network](#) on page 24
2. [Creating a subnet](#) on page 26
3. [Creating a virtual router](#) on page 30
4. [Changing the information of a virtual router \(attaching to an external network\)](#) on page 31
5. [Changing the information of a virtual router \(attaching to a subnet\)](#) on page 32
6. [Creating a security group](#) on page 34
7. [Creating a security group rule](#) on page 35
8. Create firewall rules
 - [Creating a firewall rule \(IP address specification\)](#) on page 39
 - [Creating a firewall rule \(port number specification\)](#) on page 41
 - [Creating a firewall rule \(ICMP permission\)](#) on page 42
 - [Creating a firewall rule \(deny rule\)](#) on page 44
9. [Creating a firewall policy](#) on page 46
10. [Creating a firewall](#) on page 48
11. [Creating a key pair](#) on page 75
12. [Creating a port](#) on page 79 (when creating a virtual server by specifying a port)
13. Create a virtual server
 - [Creating a virtual server \(CentOS and port specification\)](#) on page 80

- [Creating a virtual server \(CentOS and DHCP retrieval\)](#) on page 82
 - [Creating a virtual server \(Windows and port specification\)](#) on page 84
 - [Creating a virtual server \(Windows and DHCP retrieval\)](#) on page 87
14. [Retrieving a global IP address and assigning it to a virtual server](#) on page 91

When using the SSL-VPN connection function

This section describes the basic procedure to perform in the K5 IaaS to configure a system that can be connected to over SSL-VPN.

- V2 service method



A method in which SSL-VPN connections are established using virtual instances. For details regarding functionality, refer to the "Features Handbook".

There are two user authentication methods that can be used with the V2 service SSL-VPN connection function, "using a K5 client certificate" and "using a self-signed certificate". This document explains the configuration procedure for each user authentication method.



Note

A "K5 client certificate" refers to a client certificate that is issued by the K5 Portal and used for client authentication when the relevant user accesses the SSL-VPN service.

Perform the following tasks in the K5 IaaS to configure the system structure.

1. [Creating a network](#) on page 24
2. [Creating a subnet](#) on page 26
3. [Creating a virtual router](#) on page 30
4. [Changing the information of a virtual router \(attaching to an external network\)](#) on page 31
5. [Changing the information of a virtual router \(attaching to a subnet\)](#) on page 32
6. [Creating a security group](#) on page 34
7. [Creating a security group rule](#) on page 35
8. Create firewall rules
 - [Creating a firewall rule \(IP address specification\)](#) on page 39
 - [Creating a firewall rule \(port number specification\)](#) on page 41
 - [Creating a firewall rule \(ICMP permission\)](#) on page 42
 - [Creating a firewall rule \(deny rule\)](#) on page 44
9. [Creating a firewall policy](#) on page 46

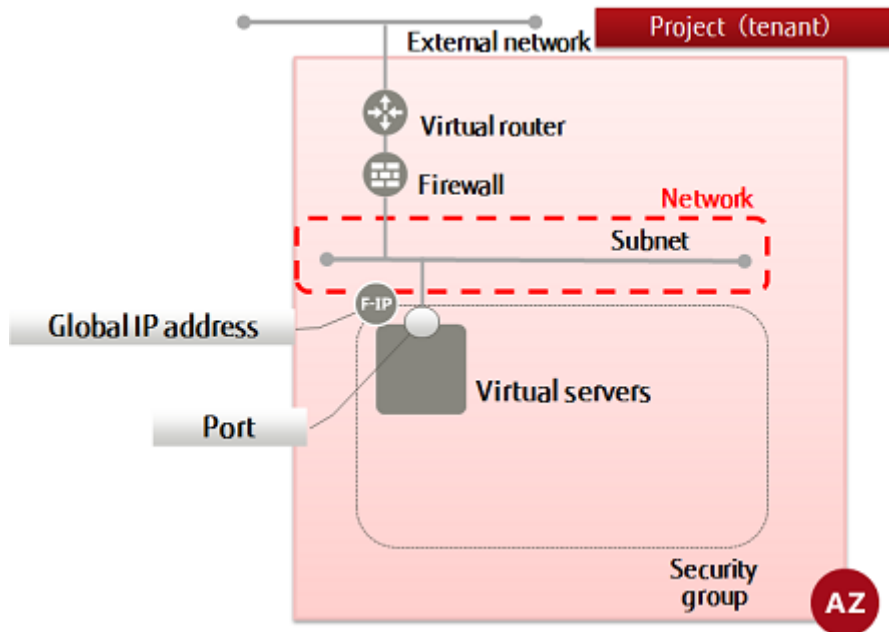
- 10.[Creating a firewall](#) on page 48
- 11.SSL-VPN connection
 - [SSL-VPN connection \(V2 service/K5 client certificate\)](#) on page 51
 - [SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 58
- 12.[Creating a key pair](#) on page 75
- 13.[Creating a port](#) on page 79(if creating a virtual server on which ports are specified)
- 14.Create a virtual server
 - [Creating a virtual server \(CentOS and port specification\)](#) on page 80
 - [Creating a virtual server \(CentOS and DHCP retrieval\)](#) on page 82
 - [Creating a virtual server \(Windows and port specification\)](#) on page 84
 - [Creating a virtual server \(Windows and DHCP retrieval\)](#) on page 87

3.2 Building a network environment

3.2.1 Creating a network

This section explains how to create a network within a project for the deployment of resources such as a virtual server.

You can create multiple networks within a project.



1. Set the environment variable below as follows:

```
$ NW_NAME=<newNetworkName> (any)
```

```
$ AZ=<availabilityZoneName>
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/networks -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"network": {"name": "'$NW_NAME'", "availability_zone": "'$AZ'"}}' \
| jq .
```

The following response is output:

```
{
  "network": {
    "status": "ACTIVE",
    "subnets": [],
    "name": "<specifiedNetworkName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "shared": false,
    "id": "<newNetworkId>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to confirm that the network was created properly.

```
$ curl -Ss $NETWORK/v2.0/networks -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If a list including the network name that you specified is output, as follows, that means the network was created successfully.

The network names "inf_az1_ext_net01" and "inf_az2_ext_net01" are external networks that exist by default.

```
{
  "networks": [
    {
      "status": "ACTIVE",
      "subnets": [
        "5079f324-5db0-44ee-92ac-3a6b7977b23f",
        "a56b6058-0479-43a1-8b27-01c1c05e96a2",
        "c1da3ee7-51c3-4801-bb97-aa03a4383ef0",
        "e96e55b8-84bb-4777-a782-a5d6e8340039",
        "f5e9ec37-88ec-494b-ac55-dae101a54cc1"
      ],
      "name": "inf_az2_ext-net01",
      "router:external": true,
      "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
      "admin_state_up": true,
      "shared": true,
      "id": "375c49fa-a706-4676-b55b-2d3554e5db6a",
      "availability_zone": "jp-east-1b"
    },
    ...
    {
      "status": "ACTIVE",
      "subnets": [],
      "name": "<networkName>",
      "router:external": false,
      "tenant_id": "<projId>",
      "admin_state_up": true,
      "shared": false,
      "id": "<networkId>",
      "availability_zone": "<availabilityZone>"
    },
    {
      "status": "ACTIVE",
      "subnets": [
        "37ca5225-6ca6-4ee6-a49d-d479400a632b",
        "3c2419a7-7745-452c-bee5-664db03129bb",
        "7c7feecd-082c-415b-865a-82d4e5de97e5",
        "d96cdd6c-6e1e-4331-9c8e-42b52588b767",
        "ff4a9bd6-37be-4c03-9b87-1f693f807b48"
      ],
      "name": "inf_az1_ext-net01",
      "router:external": true,
      "tenant_id": "31ceb599e8ff48aeb66f2fd748988960",
      "admin_state_up": true,
      "shared": true,
      "id": "af4198a9-b392-493d-80ec-a7c6e5a1c22a",
      "availability_zone": "jp-east-1a"
    }
  ]
}
```

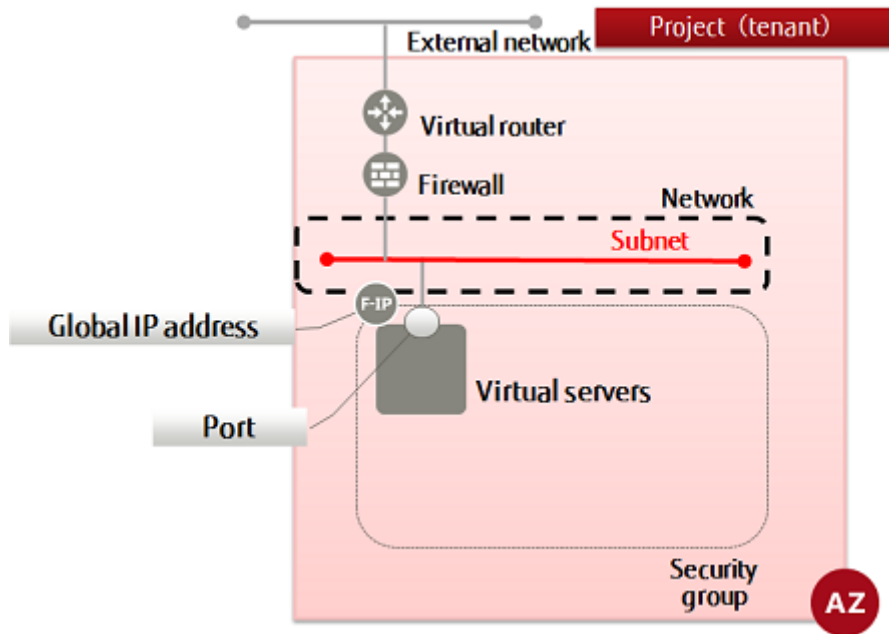
Create a subnet for the network you created, to deploy resources such as a virtual server.

3.2.2 Create a subnet

3.2.2.1 Creating a subnet

A subnet is required for managing private IP addresses for resources to be connected to a network, and for automatic setting of IP addresses using DHCP. This section explains how to create a subnet and then check that it was created properly.

Specify a network and create a subnet.



1. Set the environment variables below as follows:



Note

This procedure sets up a DNS server, but for a virtual server to communicate with the DNS server, it is necessary to allow outbound communication to the Internet. Check the settings of the security group feature or firewall service, and allow communication with the DNS server (protocol: TCP/UDP, port number: 53).

Note that the value specified for DNS will vary depending on the environment. When using the DNS provided by the K5 IaaS, refer to the appendix "[Common Network Services](#)" in the "Features Handbook".

```
$ SUBNET=<newSubnetName> (any)
```

```
$ NETWORK_ID=<networkId>
```

```
$ CIDR=<networkAddr> (any, using format XXX.XXX.XXX.0/24)
```

```
$ GATEWAY_ID=<defaultGatewayIpAddr> (any, using format XXX.XXX.XXX.1)
```

```
$ DNS=¥"133.162.XXX.XXX¥"¥, ¥"133.162.XXX.XXX¥"
```



Note

For the range of addresses specifiable for the CIDR parameter, refer to "[Subnet Management](#)" in the "Features Handbook".

Specify any desired IP address within the address range specified for the CIDR parameter for the GATEWAY_IP parameter.



Warning For the created subnet, if connecting using an SSL-VPN connection (the V2 service), it is necessary to specify the following values for the variables above.

- CIDR
Specify the mask value for the network address in the range of "16bit - 29bit".
Example: 192.168.1.0/24
- GATEWAY_IP
Specify the IP address of the virtual router specified for the VPN service.

2. Execute the following API:

```
$ curl -sS $NETWORK/v2.0/subnets -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"subnet": {"name": "'$SUBNET'",
"network_id": "'$NETWORK_ID'", "cidr": "'$CIDR'",
"dns_nameservers": ['$DNS'], "ip_version": 4,
"gateway_ip": "'$GATEWAY_IP'", "availability_zone": "'$AZ'"} }' | jq .
```

The following response is output:

```
{
  "subnet": {
    "name": "<newSubnetName>",
    "enable_dhcp": true,
    "network_id": "<specifiedNetworkId>",
    "tenant_id": "<projId>",
    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "start": "XXX.XXX.XXX.2",
        "end": "XXX.XXX.XXX.254"
      }
    ],
    "host_routes": [],
    "ip_version": 4,
    "gateway_ip": "<specifiedDefaultGatewayIpAddr>",
    "cidr": "<specifiedNetworkAddr>",
    "id": "<newSubnetId>",
    "availability_zone": "<sameAvailabilityZoneAsSpecifiedNetwork>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check that the subnet was created properly.

```
$ curl -sS $NETWORK/v2.0/subnets -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

A response is output in a format such as the following. If a list including the subnet name that you specified is output, that means the subnet was created successfully.

Subnets with names that begin with "inf_az1_fip-pool", "inf_az2_fip-pool", "inf_az1_ext-subnet", and "inf_az1_ext-subnet", are associated with external networks that exist by default.

```
{
  "subnets": [
```

```

...
{
  "name": "<subnetName>",
  "enable_dhcp": true,
  "network_id": "<networkId>",
  "tenant_id": "<projId>",
  "dns_nameservers": [
    "133.162.XXX.XXX",
    "133.162.XXX.XXX"
  ],
  "allocation_pools": [
    {
      "start": "XXX.XXX.XXX.2",
      "end": "XXX.XXX.XXX.254"
    }
  ],
  "host_routes": [],
  "ip_version": 4,
  "gateway_ip": "<defaultGatewayIpAddr>",
  "cidr": "<networkAddr>",
  "id": "<subnetId>",
  "availability_zone": "<sameAvailabilityZoneAsNetwork>"
},
...
]
}

```

3.2.2.2 Adding routing

Configure the necessary routing information for the created subnet. This setting is necessary for [Building a management network](#) on page 95. The following is an example procedure for the routing configuration that becomes necessary for SSL-VPN connections.

1. Perform the settings necessary for routing as indicated below.

```
$ SUBNET_ID=<ID of the subnet to add routing (host_routes) to>
```

```
$ HOST_ROUTES={¥"nexthop¥":¥"IP address of the default gateway of the subnet¥",
¥"destination¥":¥"Virtual network cidr of the VPN tunnel¥"}
```

Specify the network address of the virtual network cidr of the VPN tunnel using the format "XXX.XXX.XXX.XXX/XX".

Example: 192.168.246.0/24



Warning Specify a network address that does not conflict with the network address used by K5 or the local network addresses used by client PCs.

The cidr of the virtual network of the VPN tunnel must be set the same as the one set for CLIENT_ADDRESS_POOL_CIDR when creating the SSL-VPN connection ([Creating an SSL-VPN connection \(V2 service/K5 client certificate\)](#) on page 52 or [Creating an SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 70).

2. Execute the following API.

```
$ curl -Ss $NETWORK/v2.0/subnets/$SUBNET_ID -X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
-H "Content-Type: application/json" -d '{"subnet": { "host_routes": [ '$HOST_ROUTES' ] } }'
| jq .
```

The following response is output.

```
{
```



```

"subnet": {
  "availability_zone": "<Same availability zone as specified subnet>",
  "id": "<Subnet ID>",
  "cidr": "<Virtual network cidr of the specified VPN tunnel>",
  "gateway_ip": "<Default gateway IP address of specified subnet>",
  "name": "<Subnet name>",
  "enable_dhcp": true,
  "network_id": "<Network ID>",
  "tenant_id": "<Project ID>",
  "dns_nameservers": [
    "133.162.XXX.XXX",
    "133.162.XXX.XXX"
  ],
  "allocation_pools": [
    {
      "end": "XXX.XXX.XXX.254",
      "start": "XXX.XXX.XXX.2"
    }
  ],
  "host_routes": [
    {
      "destination": "<Virtual network cidr of specified VPN tunnel>",
      "nexthop": "<Default gateway IP address of specified subnet>"
    }
  ],
  "ip_version": 4
}
}

```

3. Execute the following API to check the routing that you set:

```
$ SUBNET_ID=<ID of the subnet that routing (host_routes) was added to>
```

```
$ curl -sS $NETWORK/v2.0/subnets/$SUBNET_ID -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

The following response is output. When a list containing the set routing name is output, creation is complete.

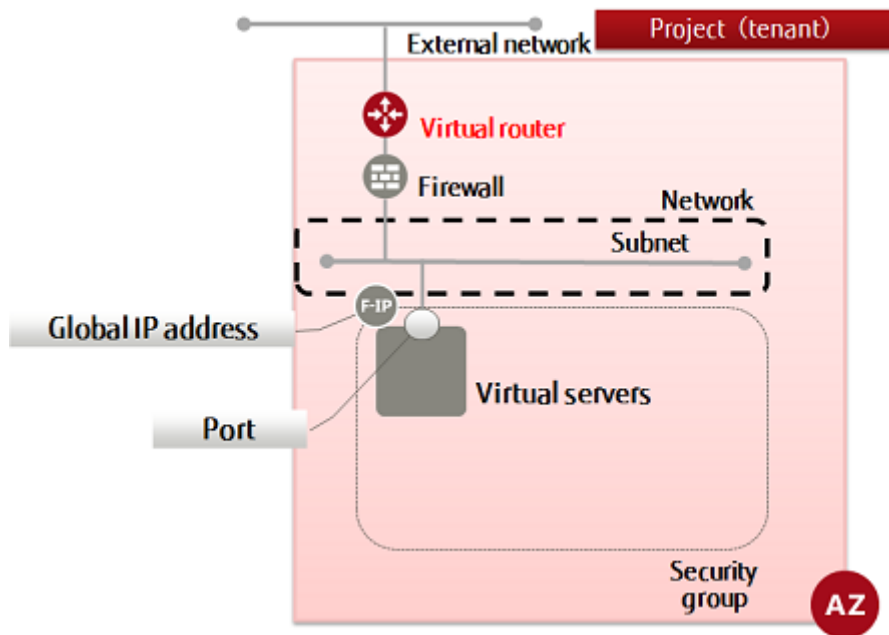
```

{
  "subnet": {
    "availability_zone": "<Same availability zone as specified subnet>",
    "id": "<Subnet ID>",
    "cidr": "<Virtual network cidr of the specified VPN tunnel>",
    "gateway_ip": "<Default gateway IP address of specified subnet>",
    "name": "<Subnet name>",
    "enable_dhcp": true,
    "network_id": "<Network ID>",
    "tenant_id": "<Project ID>",
    "dns_nameservers": [
      "133.162.XXX.XXX",
      "133.162.XXX.XXX"
    ],
    "allocation_pools": [
      {
        "end": "XXX.XXX.XXX.254",
        "start": "XXX.XXX.XXX.2"
      }
    ],
    "host_routes": [
      {
        "destination": "<Virtual network cidr of specified VPN tunnel>",
        "nexthop": "<Default gateway IP address of specified subnet>"
      }
    ],
    "ip_version": 4
  }
}

```

3.2.3 Creating a virtual router

This section explains how to create a virtual router to either connect a network with an external network or to connect networks internally.



1. Set the environment variables below as follows:

```
$ ROUTER_NAME=<newRouterName (any)>
```

```
$ AZ=<availabilityZoneName>
```

2. Execute the API.

```
$ curl -sS $NETWORK/v2.0/routers -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"router": {"name": "'$ROUTER_NAME'", "tenant_id": "'$TENANT_ID'",
"availability_zone": "'$AZ'"}}' | jq .
```

The following response is output:

```
{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": null,
    "name": "<newVirtualRouterName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "id": "<newVirtualRouterId>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check that the virtual router was created properly:

```
$ curl -sS $NETWORK/v2.0/routers -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

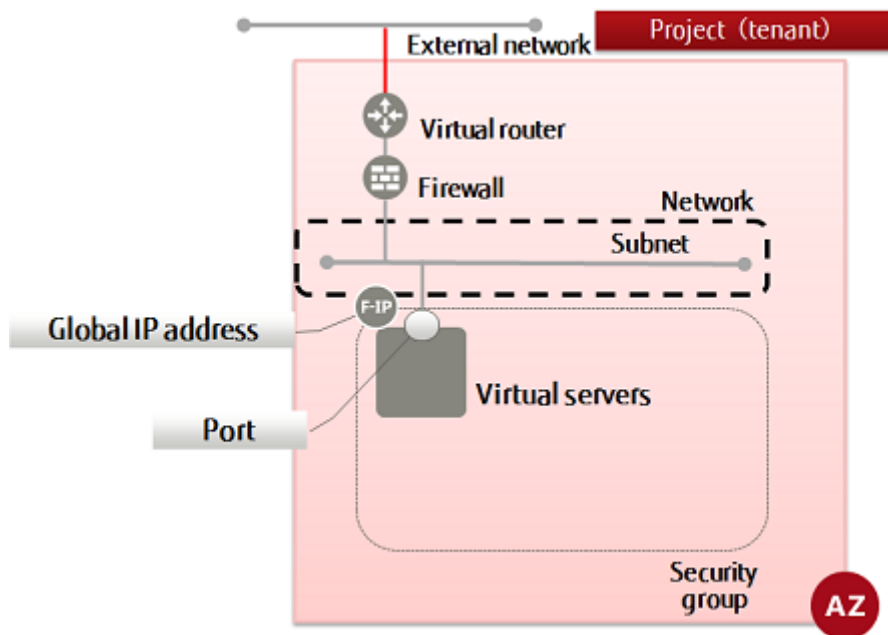
If a list including the virtual router name that you specified is output, as follows, that means the virtual router was created successfully.

```
{
  "routers": [
    ...
    {
      "status": "ACTIVE",
      "external_gateway_info": null,
      "name": "<newVirtualRouterName>",
      "admin_state_up": true,
      "tenant_id": "<projId>",
      "id": "<newVirtualRouterId>",
      "availability_zone": "<specifiedAvailabilityZone>"
    },
    ...
  ]
}
```

Refer to the information changes of the virtual router (attached) for information on how to add a subnet using the information change feature of the virtual router.

3.2.4 Changing the information of a virtual router (attaching to an external network)

This section explains how to connect a virtual router to an external network by updating its settings.



1. Set the environment variables below as follows:

```
$ ROUTER_ID=<targetVirtualRouterId>
```

```
$ EXT_NET_ID=<externalNetworkIdToBeConnected>
```

The external network ID can be checked using the API executed in step 3 of [Creating a network](#) on page 24.

```
$ curl -Ss $NETWORK/v2.0/networks -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

Subnets with names that start with "inf_az1_ext_net01" and "inf_az2_ext_net01" are created associated with the default external network.

2. Execute the following API:

```
$ curl -sS $NETWORK/v2.0/routers/$ROUTER_ID -X PUT \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"router": {"external_gateway_info": {"network_id": "'$EXT_NET_ID'"}}, {"name": "'$ROUTER_NAME'", "admin_state_up": true, "tenant_id": "'$TENANT_ID'", "id": "'$ROUTER_ID'", "availability_zone": "'$AZ'"}}
```

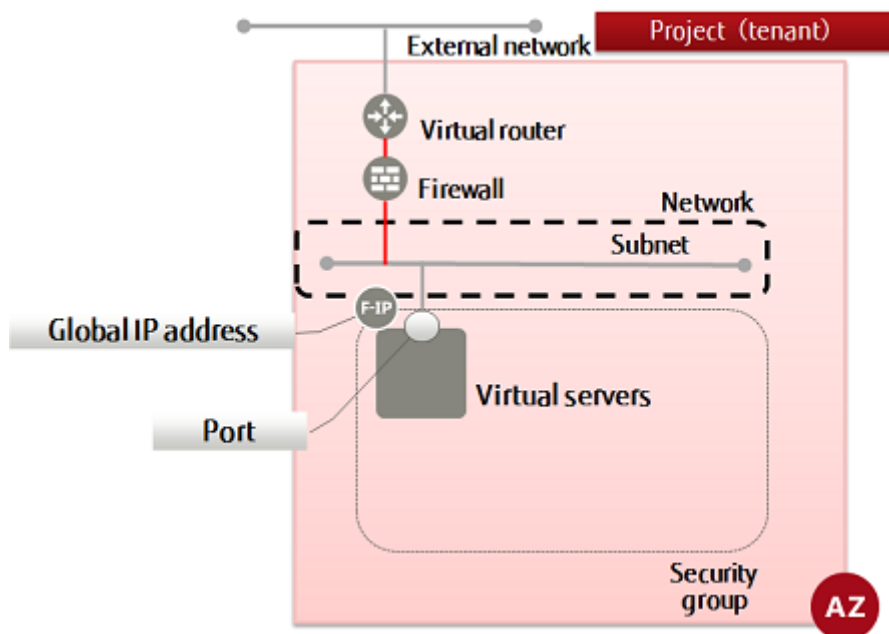
The following response is output:

```
{
  "router": {
    "status": "ACTIVE",
    "external_gateway_info": {
      "network_id": "<specifiedExternalNetworkId>",
      "enable_snat": true
    },
    "name": "<specifiedVirtualRouterName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "routes": [],
    "id": "<specifiedVirtualRouterId>",
    "availability_zone": "<availabilityZone>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3.2.5 Changing the information of a virtual router (attaching to a subnet)

This section explains how to connect a virtual router to a subnet on K5 by updating its settings.



1. Set the environment variables below as follows:

```
$ ROUTER_ID=<targetVirtualRouterId>
```

```
$ SUBNET_ID=<subnetIdToBeConnected>
```

2. Execute the following API:

```
$ curl -sS $NETWORK/v2.0/routers/$ROUTER_ID/add_router_interface -X PUT \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"subnet_id": "'$SUBNET_ID'" }' | jq .
```

The following response is output:

```
{
  "subnet_id": "<specifiedSubnetId>",
  "tenant_id": "<projId>",
  "port_id": "<portId>",
  "id": "<virtualRouterId>",
  "availability_zone": "<availabilityZone>"
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3.3 Security group settings

3.3.1 Creating a security group

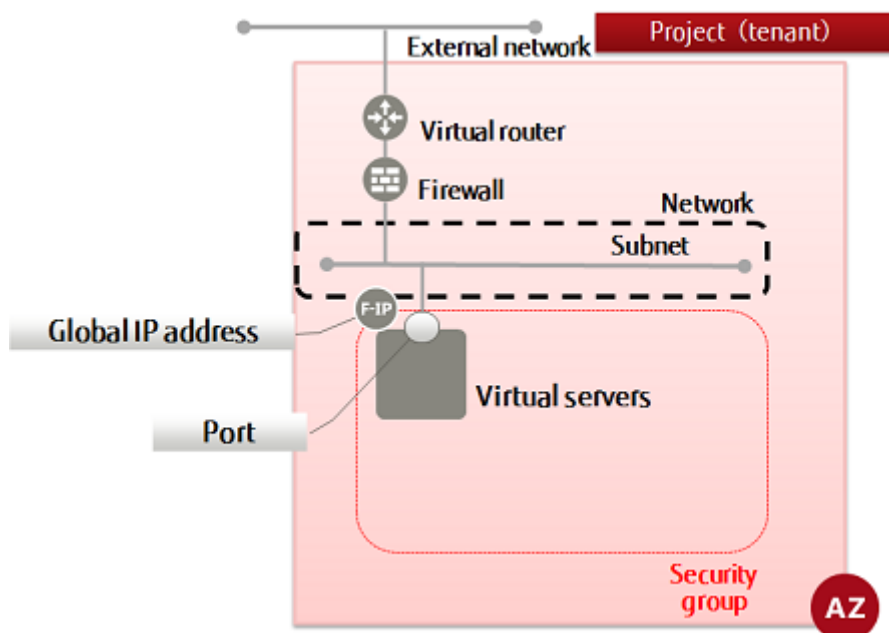
Security groups allow you to group, define and configure rules to filter packets for ports that virtual servers are connected to. Follow the procedure below to create a security group.

You can set multiple rules for a security group. Of the security groups set for a single port, Communication on a port is permitted only for packets that match at least one rule in any one of the security groups assigned to it. All other packets are intercepted. (whitelist method, OR condition)



Security groups cannot be set for ports used by a virtual router or DHCP server.

Note



A default security group that automatically intercepts communication is set for ports. To permit communication, create a security group.

1. Set the environment variable below as follows:

Security groups are not associated with availability zones, and therefore, it is not necessary to specify one.

```
$ SG_NAME=<newSecGroupName> (any)
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/security-groups -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"security_group": {"name": "'$SG_NAME'"}}' | jq .
```

The following response is output:

```
{
  "security_group": {
    "tenant_id": "<projId>",
    "description": "",
    "name": "<newSecGroupName>",
    "security_group_rules": [
```

```
{
  "remote_group_id": null,
  "direction": "egress",
  "remote_ip_prefix": null,
  "protocol": null,
  "ethertype": "IPv6",
  "port_range_max": null,
  "security_group_id": "<newSecGroupId>",
  "port_range_min": null,
  "tenant_id": "<projId>",
  "id": "<defaultSecGroupRuleId>"
},
{
  "remote_group_id": null,
  "direction": "egress",
  "remote_ip_prefix": null,
  "protocol": null,
  "ethertype": "IPv4",
  "port_range_max": null,
  "security_group_id": "<newSecGroupId>",
  "port_range_min": null,
  "tenant_id": "<projId>",
  "id": "<defaultSecGroupRuleId>"
}
],
"id": "<newSecGroupId>"
}
```

3. List the security groups to confirm that it has been created.

```
$ curl -Ss $NETWORK/v2.0/security-groups -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | ¥
jq '.security_groups[] | .name, .description, .id'
```

If a list including the security group name that you specified is output, as follows, that means the security group was created successfully.

```
"default"
"default"
"<defaultSecGroupIdOfProj>"
...
"<newSecGroupName>"
""
"<newSecGroupId>"
...
```

3.3.2 Creating a security group rule

This section explains how to create a security group rule.



Tip

For communication between virtual servers on which a security group can be used by both, it is generally recommended that you specify the communicating destination using the security group ID.



Tip

A rule permitting communications must be explicitly specified for communication (including communication that comes back to a server) between virtual servers that have the same security group set. Resolve this by specifying your own security group ID.



Tip

For ingress, specify the source IP address, and for egress, specify the destination IP address.

1. Set the environment variables below as follows:

```
$ DIRECTION=<communicDirection> (specify ingress or egress)
```

```
$ PROTOCOL=<communicProtocol> (specify tcp, udp, icmp or 0-65535)
```

```
$ MIN_PORT_NUM=<minPortNum> (specify 0-65535)
```

```
$ MAX_PORT_NUM=<maxPortNum> (specify 0-65535)
```

```
$ SG_ID=<secGroupIdToAddRuleTo>
```

```
$ REMOTE_IP=<ipAddressToAllow> (specify using a format such as XXX.XXX.XXX.0/24)
```

or

```
$ REMOTE_GROUP_ID=<secGroupIdToAllow>
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"security_group_rule":{"direction": "'$DIRECTION'",
"port_range_min": '$MIN_PORT_NUM',
"port_range_max": '$MAX_PORT_NUM', "protocol": "'$PROTOCOL'",
"remote_ip_prefix": "'$REMOTE_IP'", "security_group_id": "'$SG_ID'"}}' \
| jq .
```

or

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{"security_group_rule":{"direction": "'$DIRECTION'",
"port_range_min": '$MIN_PORT_NUM', "port_range_max": '$MAX_PORT_NUM', "protocol":
"'$PROTOCOL'",
"remote_group_id": "'$REMOTE_GROUP_ID'",
"security_group_id": "'$SG_ID'"}}' | jq .
```

The following response is output:

```
{
  "security_group_rule": {
    "remote_group_id": "<secGroupIdToAllow>",
    "direction": "<communicDirection>",
    "remote_ip_prefix": "<ipAddressToAllow>",
    "protocol": "<protocol>",
    "tenant_id": "<projId>",
    "port_range_max": <maxPortNum>,
    "security_group_id": "<secGroupIdWithNewRule>",
    "port_range_min": <minPortNum>,
    "ethertype": "IPv4",
    "id": "<secGroupRuleId>",
    "availability_zone": null
  }
}
```

3. List the security group rules to confirm that it has been specified.

```
$ curl -Ss $NETWORK/v2.0/security-group-rules -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If a list including set security group rules is output, as follows, that means creation was successful.

```
{
  "security_group_rules": ]
```



```

...
{
  "remote_group_id": "<secGroupIdToAllow>",
  "direction": "<commDirection>",
  "remote_ip_prefix": "<ipAddressToAllow>",
  "protocol": "<protocol>",
  "tenant_id": "<projId>",
  "port_range_max": <maxPortNum>,
  "security_group_id": "<secGroupIdWithNewRule>",
  "port_range_min": <minPortNum>,
  "ethertype": "IPv4",
  "id": "<secGroupRuleId>",
}
...
]
}

```

3.3.3 Examples of security group rules

This section provides examples of security group rules, specified for different purposes.

Configure the following settings:

- ingress: PING (ICMP: 0-255)
- ingress: SSH (TCP: 22)
- ingress: HTTP (TCP: 80)
- ingress: HTTPS (TCP: 443)
- ingress: NTP (UDP: 123)
- ingress: KMS (TCP: 1688)
- ingress: RDP (TCP: 3389)

The default values will be used for egress communication direction.

ingress: PING (ICMP: 0-255)

```

$ DIRECTION=ingress
$ PROTOCOL=icmp
$ MIN_PORT_NUM=0
$ MAX_PORT_NUM=255
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)
or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>

```

ingress: SSH (TCP: 22)

```

$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=22
$ MAX_PORT_NUM=22
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)
or
$ REMOTE_GROUP_ID=<secGroupId>
$ SG_ID=<secGroupIdToBeSet>

```

ingress: HTTP (TCP: 80)

```

$ DIRECTION=ingress
$ PROTOCOL=tcp
$ MIN_PORT_NUM=80
$ MAX_PORT_NUM=80
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)
or

```

```
$ REMOTE_GROUP_ID=<secGroupId>  
$ SG_ID=<secGroupIdToBeSet>
```

ingress: HTTPS (TCP: 443)

```
$ DIRECTION=ingress  
$ PROTOCOL=tcp  
$ MIN_PORT_NUM=443  
$ MAX_PORT_NUM=443  
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)  
or  
$ REMOTE_GROUP_ID=<secGroupId>  
$ SG_ID=<secGroupIdToBeSet>
```

ingress: NTP (UDP: 123)

```
$ DIRECTION=ingress  
$ PROTOCOL=udp  
$ MIN_PORT_NUM=123  
$ MAX_PORT_NUM=123  
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)  
or  
$ REMOTE_GROUP_ID=<secGroupId>  
$ SG_ID=<secGroupIdToBeSet>
```

ingress: KMS (TCP: 1688)

```
$ DIRECTION=ingress  
$ PROTOCOL=tcp  
$ MIN_PORT_NUM=1688  
$ MAX_PORT_NUM=1688  
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)  
or  
$ REMOTE_GROUP_ID=<secGroupId>  
$ SG_ID=<secGroupIdToBeSet>
```

ingress: RDP (TCP: 3389)

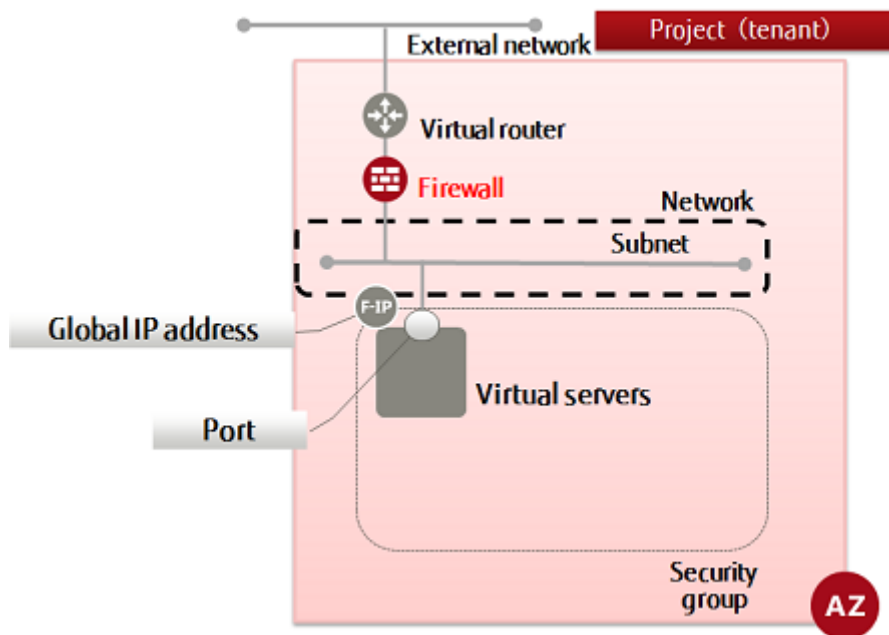
```
$ DIRECTION=ingress  
$ PROTOCOL=tcp  
$ MIN_PORT_NUM=3389  
$ MAX_PORT_NUM=3389  
$ REMOTE_IP=<remoteIpAddr> (specify using format XXX.XXX.XXX.0/24)  
or  
$ REMOTE_GROUP_ID=<secGroupId>  
$ SG_ID=<secGroupIdToBeSet>
```

Refer to the "Service Specification" for other advanced setting items.

3.4 Creating a firewall

3.4.1 Creating a firewall rule (IP address specification)

This section explains how to create a firewall rule (IP address specification) to use the firewall service for setting packet filters for virtual routers



Tip The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

All traffic is intercepted by default (whitelist format), and only traffic that is defined using permission rules (AC=allow) passes through the firewall.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router

Configure the settings as below to create firewall rules.

- The source or destination IP address
- The protocol to use for communication
- The port number to use for communication
- Whether to allow or deny communication

This item explains how to create specific rules such as those shown below.

- Allow: Source IP address: TCP
- Allow: Source IP address: UDP
- Allow: Source IP address: ICMP

While you can do advanced settings for a firewall, this guide shows how to create commonly used rules. Refer to the Service Specification and API Reference Manual for details on the advanced setting items and how to configure them.

1. Set the environment variables below as follows:

```
$ FWR_NAME=<firewallRuleName> (any)
```

```
$ AC=<accessControl> (allow)
```

```
$ SOURCE_IP=<ipAddressToAllow> (specify using a format such as XXX.XXX.XXX.0/24)
```

```
$ PROTOCOL=<protocol> (specify tcp, udp, icmp)
```

```
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"firewall_rule": { "name": "'$FWR_NAME'",  
"action": "'$AC'", "source_ip_address": "'$SOURCE_IP'",  
"protocol": "'$PROTOCOL'",  
"availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

The following response is output:

```
{  
  "firewall_rule": {  
    "protocol": "<protocol>",  
    "description": "",  
    "ip_version": 4,  
    "tenant_id": "<projId>",  
    "enabled": true,  
    "source_ip_address": "<specifiedIpAddr>",  
    "destination_ip_address": null,  
    "firewall_policy_id": null,  
    "action": "allow",  
    "shared": false,  
    "source_port": null,  
    "position": null,  
    "destination_port": null,  
    "id": "<newFirewallRuleId>",  
    "name": "<newFirewallRuleName>",  
    "availability_zone": "<specifiedAvailabilityZone>"  
  }  
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the firewall rules that you created:

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name, .id'
```

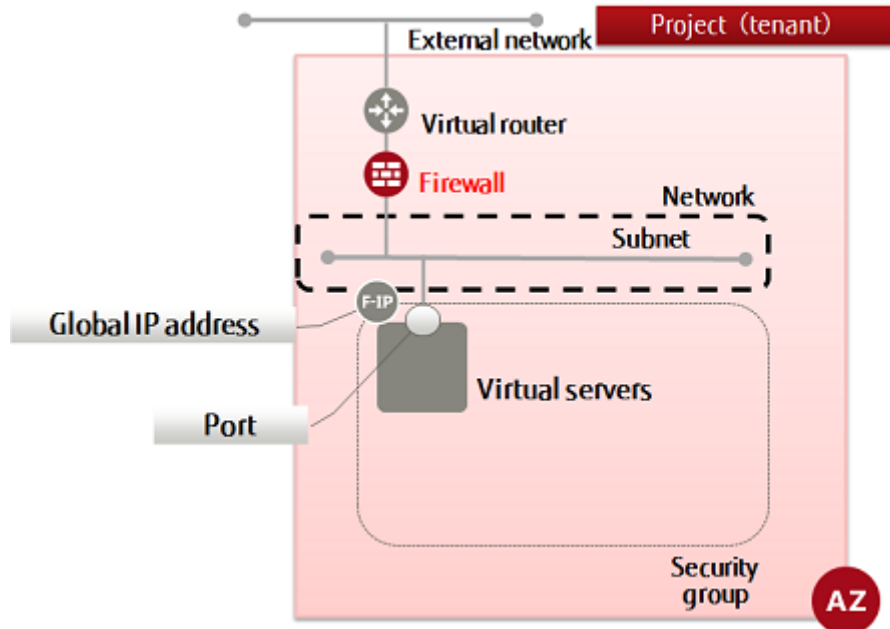
If a list including the firewall rule names that you specified is output, as follows, that means the firewall rules were created successfully.

```
...  
"<newFirewallRuleName>"  
"<newFirewallRuleId>"  
...
```

When finished creating firewall rules, proceed with creation of a firewall policy to which those rules are bound.

3.4.2 Creating a firewall rule (port number specification)

This section explains how to create a firewall rule (port number specification) to use the firewall service for setting packet filters for virtual routers.



Tip The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

All traffic is intercepted by default (whitelist format), and only the traffic that is defined using permission rules (AC=allow) passes through the firewall.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router

This items explains how to create specific rules such as those shown below.

- Allow: SSH (TCP: 22)
- Allow: HTTP (TCP: 80)
- Allow: HTTPS (TCP: 443)
- Allow: NTP (UDP: 123)
- Allow: KMS (TCP: 1688)
- Allow: RDP (TCP: 3389)

While you can do advanced settings for a firewall, this guide shows how to create commonly used rules. Refer to the Service Specification and API Reference Manual for details on the advanced setting items and how to configure them.

1. Set the environment variables below as follows:

```
$ FWR_NAME=<firewallRuleName> (any)
```

```
$ AC=<accessControl> (allow)
```

```
$ DESTINATION_PORT=<destinationPortNumOrRange> (for example, "0 - 255", 22, etc.)
```

```
$ PROTOCOL=<protocol> (specify tcp, udp, icmp)
```

```
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"firewall_rule": { "name": "'$FWR_NAME'", "action": "'$AC'",  
"destination_port": "'$DESTINATION_PORT'", "protocol": "'$PROTOCOL'",  
"availability_zone": "'$AVAILABILITY_ZONE'" }}' | jq .
```

The following response is output:

```
{  
  "firewall_rule": {  
    "protocol": "<protocol>",  
    "description": "",  
    "ip_version": 4,  
    "tenant_id": "<projId>",  
    "enabled": true,  
    "source_ip_address": "null",  
    "destination_ip_address": null,  
    "firewall_policy_id": null,  
    "action": "allow",  
    "shared": false,  
    "source_port": null,  
    "position": null,  
    "destination_port": "<specifiedPortNum>",  
    "id": "<newFirewallRuleId>",  
    "name": "<newFirewallRuleName>",  
    "availability_zone": "<specifiedAvailabilityZone>"  
  }  
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the firewall rules that you created:

```
$ curl -Ss $NETWORK/v2.0/fw/firewall_rules -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'
```

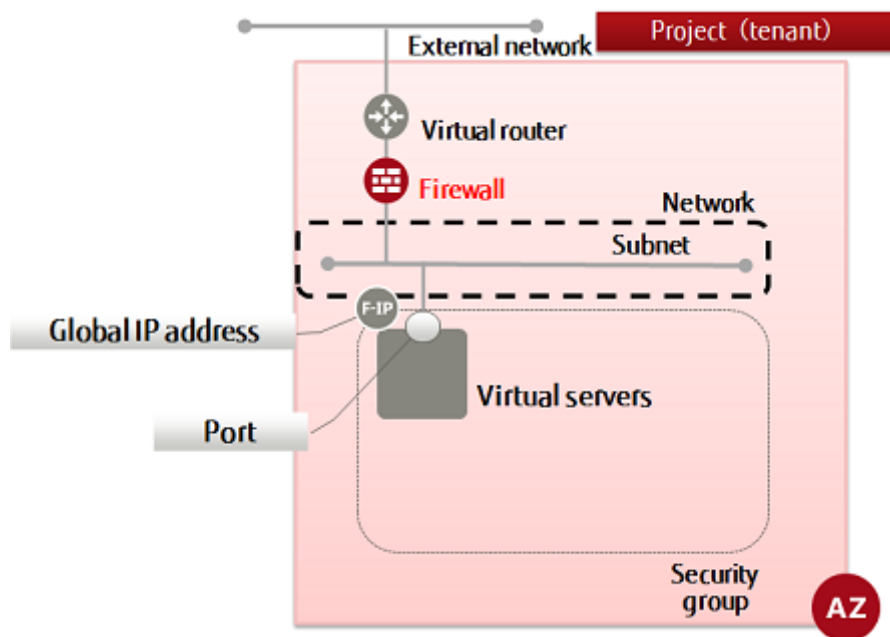
If a list including the firewall rule names that you specified is output, as follows, that means the firewall rules were created successfully.

```
...  
"<newFirewallRuleName>"  
"<newFirewallRuleId>"  
...
```

When finished creating firewall rules, proceed with creation of a firewall policy to which those rules are bound.

3.4.3 Creating a firewall rule (ICMP permission)

This section explains how to create a firewall rule (ICMP permission) to use the firewall service for setting packet filters for virtual routers.



Tip The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

All traffic is intercepted by default (whitelist format), and only traffic that is defined using permission rules (AC=allow) passes through the firewall.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router

This item explains how to create specific rules such as those shown below.

- Allow: PING (ICMP: 0 to 255)

While you can do advanced settings for a firewall, this guide shows how to create commonly used rules. Refer to the Service Specification and API Reference Manual for details on the advanced setting items and how to configure them.

1. Set the environment variables below as follows:

```
$ FWR_NAME=<firewallRuleName> (any)
```

```
$ AC=<accessControl> (allow)
```

```
$ PROTOCOL=icmp
```

```
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"firewall_rule": {"name": "$FWR_NAME", "action": "$AC",
"protocol": "$PROTOCOL",
"availability_zone": "$AVAILABILITY_ZONE" }}' | jq .
```

The following response is output:

```
{
  "firewall_rule": {
    "protocol": "icmp",
    "description": "",

```

```

    "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "allow",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the firewall rules that you created:

```

$ curl -X GET -Ss $NETWORK/v2.0/fw/firewall_rules ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'

```

If a list including the firewall rule names that you specified is output, as follows, that means the firewall rules were created successfully.

```

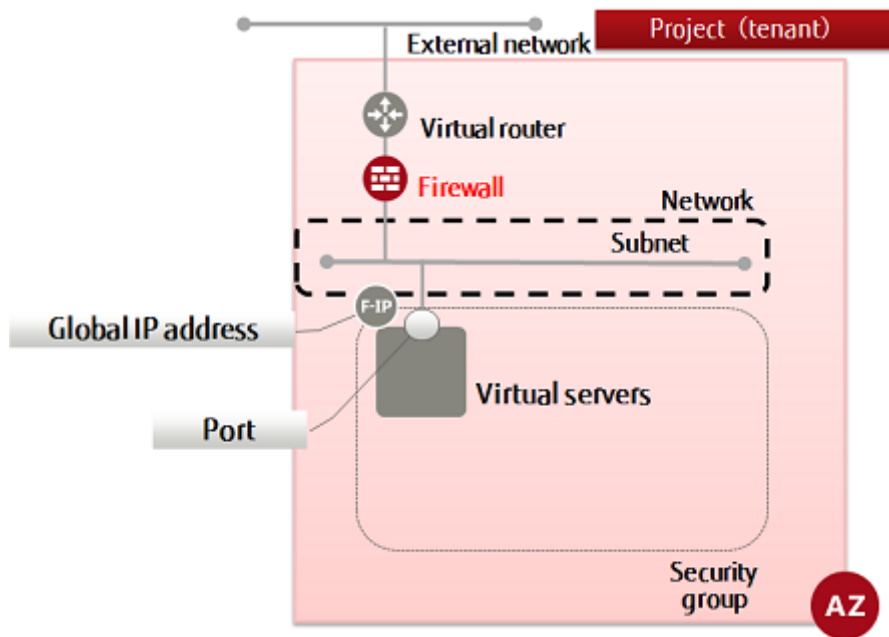
...
"<newFirewallRuleName>"
"<newFirewallRuleId>"
...

```

When finished creating firewall rules, proceed with creation of a firewall policy to which those rules are bound.

3.4.4 Creating a firewall rule (deny rule)

This section explains how to create a firewall rule (deny all rules other than those that have already been set) to use the firewall service for setting packet filters for virtual routers, and for checking the rules have been created.



Tip The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

All traffic is intercepted by default (whitelist format), and only traffic that is defined using permission rules (AC=allow) passes through the firewall.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router

This item explains how to create specific rules such as those shown below.

- Close all ports that have not been set

While you can do advanced settings for a firewall, this guide shows how to create commonly used rules. Refer to the Service Specification and API Reference Manual for details on the advanced setting items and how to configure them.

1. Set the environment variables below as follows:

```
$ FWR_NAME=<firewallRuleName> (any)
```

```
$ AC=deny
```

```
$ AVAILABILITY_ZONE=<creationDestinationAvailabilityZoneName>
```

2. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_rules ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"firewall_rule": {"name": "$FWR_NAME", "action": "$AC",
"availability_zone": "$AVAILABILITY_ZONE"}}' | jq .
```

The following response is output:

```
{
  "firewall_rule": {
    "protocol": null,
    "description": "",
    "ip_version": 4,
    "tenant_id": "<projId>",
    "enabled": true,
```

```

    "source_ip_address": "null",
    "destination_ip_address": null,
    "firewall_policy_id": null,
    "action": "deny",
    "shared": false,
    "source_port": null,
    "position": null,
    "destination_port": null,
    "id": "<newFirewallRuleId>",
    "name": "<newFirewallRuleName>",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the firewall rules that you created:

```

$ curl -X GET -Ss $NETWORK/v2.0/fw/firewall_rules ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name,.id'

```

If a list including the firewall rule names that you specified is output, as follows, that means the firewall rules were created successfully.

```

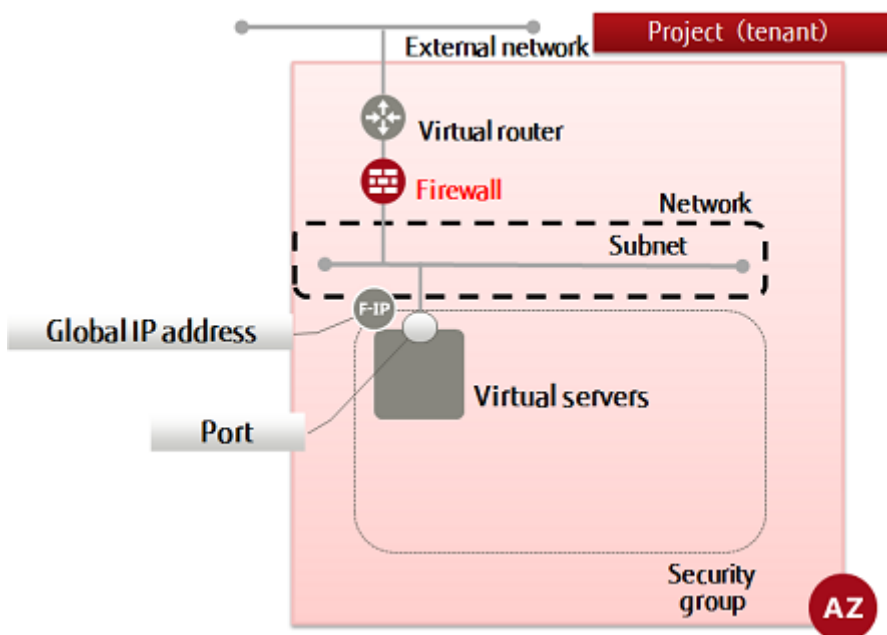
...
"<newFirewallRuleName>"
"<newFirewallRuleId>"
...

```

When finished creating firewall rules, proceed with creation of a firewall policy to which those rules are bound.

3.4.5 Creating a firewall policy

This section explains how to create a firewall policy to bind multiple firewall rules.



Rules are prioritized according to their sequence in the list, they are then verified sequentially from the top of the list, and communication permission is controlled.



Tip A "DENY ALL" rule is automatically added to the end of the policy. This ensures that all traffic without a permission rule defined will be intercepted by default. (Whitelist method) However, it becomes implicit, and does not appear in the policy.

The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router

1. Execute the following API to retrieve a list of the firewall rules:

```
$ curl -X GET -Ss $NETWORK/v2.0/fw/firewall_rules ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.firewall_rules[] | .name, .id'
```

A list of the firewall rules is displayed in a format such as that below. Select the rules you require for the policy from this list.

```
...  
"<firewallRuleName>"  
"<firewallRuleId>"  
...
```

2. Set the environment variables below as follows:

```
$ NAME=<firewallPolicyName> (any)
```

```
$ FWR1=<firewallRuleId1>
```

```
$ FWR2=<firewallRuleId2>
```

```
$ FWR3=<firewallRuleId3>
```

Note: Specify as many firewall rule IDs as required by the firewall policy.

```
$ AVAILABILITY_ZONE=<availabilityZoneSetInFirewallRules>
```

3. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewall_policies ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"firewall_policy":{"firewall_rules": [ "'$FWR1'", "'$FWR2'",  
"'$FWR3'", "'<remainingSpecifiedRuleIdEnvVars>'"],  
"name": "'$NAME'", "availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

The following response is output: The firewall rule IDs are prioritized according to the sequence order in which they are displayed.

```
{  
  "firewall_policy": {  
    "name": "<newFirewallPolicyName>",  
    "firewall_rules": [  
      "<specifiedFirewallRuleId>(1)",  
      "<specifiedFirewallRuleId>(2)",  
      ...  
      "<specifiedFirewallRuleId>(n)"  
    ]  
  }  
}
```

```

    ],
    "tenant_id": "<projId>",
    "audited": false,
    "shared": false,
    "id": "<newFirewallPolicyId>",
    "description": "",
    "availability_zone": "<specifiedAvailabilityZone>"
  }
}

```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

4. Execute the following API to check the firewall policy that you created:

```

$ curl -X GET -Ss $NETWORK/v2.0/fw/firewall_policies ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .

```

If a list including the firewall policy names that you specified is output, as follows, that means the firewall policies were created successfully.

```

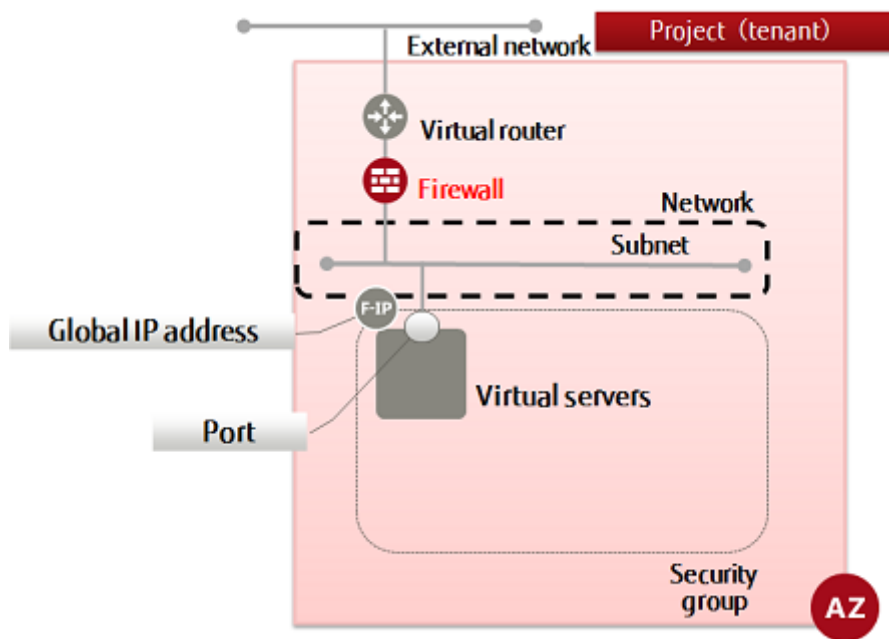
{
  "firewall_policies": [
    ...
    {
      "name": "<newFirewallPolicyName>",
      "firewall_rules": [
        "<specifiedFirewallRuleId1>",
        "<specifiedFirewallRuleId2>",
        ...
        "<specifiedFirewallRuleIdN>"
      ],
      "tenant_id": "<projId>",
      "audited": false,
      "shared": false,
      "id": "<newFirewallPolicyId>",
      "description": "",
      "availability_zone": "<specifiedAvailabilityZone>"
    },
    ...
  ]
}

```

When finished creating firewall policies, proceed with creation of a firewall to associate firewall policies with virtual routers.

3.4.6 Creating a firewall

This section explains how to specify a firewall policy (for which rules have been registered) to create a firewall for a virtual router.



The firewall service settings comprise the following elements, with the filtering information being configured in number order. By associating a firewall with a virtual router, the specified filtering will be performed.

1. Create firewall rules
2. Create a firewall policy and register rule groups
3. Specify a policy, create a firewall, and associate it with the virtual router
1. Set the environment variables below as follows:

```
$ FW_NAME=<newFirewallName> (any)
```

```
$ FWP_ID=<firewallPolicyIdToBeSet>
```

```
$ ROUTER_ID=<virtualRouterIdToBeApplied>
```

```
$ AVAILABILITY_ZONE=<sameAvailabilityZoneAsSetInTheFirewallPolicy>
```

2. Execute the following API:

```
$ curl -X POST -Ss $NETWORK/v2.0/fw/firewalls ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"firewall": {"name": "'$FW_NAME'",
"firewall_policy_id": "'$FWP_ID'", "router_id": "'$ROUTER_ID'",
"availability_zone": "'$AVAILABILITY_ZONE'"}}' | jq .
```

The following response is output:

```
{
  "firewall": {
    "status": "PENDING_CREATE",
    "router_id": "<specifiedVirtualRouterId>",
    "name": "<newFirewallName>",
    "admin_state_up": true,
    "tenant_id": "<projId>",
    "firewall_policy_id": "<specifiedFirewallPolicyId>",
    "id": "<newFirewallId>",
    "description": "",
    "availability_zone": "<availabilityZone>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the firewall that you created:

```
$ curl -X GET -Ss $NETWORK/v2.0/fw/firewalls ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If a list including the firewall policy names that you specified is output, as follows, that means the firewall policies were created successfully.

```
{  
  "firewalls": [  
    ...  
    {  
      "status": "PENDING_CREATE",  
      "router_id": "<specifiedVirtualRouterId>",  
      "name": "<newFirewallName>",  
      "admin_state_up": true,  
      "tenant_id": "<projId>",  
      "firewall_policy_id": "<specifiedFirewallPolicyId>",  
      "id": "<newFirewallId>",  
      "description": "",  
      "availability_zone": "<availabilityZone>"  
    },  
    ...  
  ]  
}
```



Warning

If a firewall policy or rules outside of your project are set, the created firewall will not work properly. When this state occurs, deletions by the user will no longer be possible, and it will be necessary for the K5 platform administrator to perform deletions. Create the firewall and its policies/rules so that they have the same project ID.

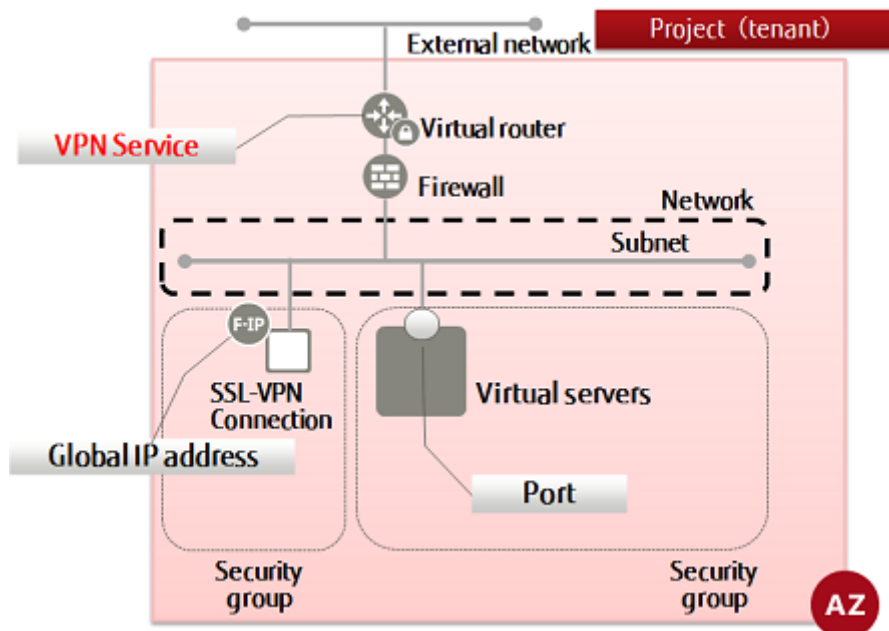
3.5 Simple configuration for SSL-VPN connection (V2 service)

3.5.1 SSL-VPN connection (V2 service/K5 client certificate)

This section explains the procedure for SSL-VPN connections that use K5 client certificates. For the procedure for SSL-VPN connections that use self-signed certificates, refer to [SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 58.

3.5.1.1 Creating a VPN service (V2 service/K5 client certificate)

This section explains the procedure for creating a VPN service.



1. Execute the following commands to perform the necessary configuration.

```
$ SUBNET_ID=<Desired Subnet ID for SSL-VPN Connection>
```

```
$ ROUTER_ID=<Router ID to Be Used for SSL-VPN>
```

```
$ VPN_SERVICE_NAME=<VPN Service Name (Optional)>
```

```
$ ADMIN_STATE_UP=true
```

```
$ AZ=<Availability Zone>
```



Warning

The parameter definition of the subnet for connecting using an SSL-VPN connection must be specified as follows:

- cidr

The mask value for the network address is specified in the range of "16bit - 29bit".

Example: 192.168.1.0/24

- gateway_ip

The IP address of the virtual router specified for the VPN service is specified.

-
2. Execute the following API.

```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"vpnservice": {"subnet_id": "'$SUBNET_ID'",
"router_id": "'$ROUTER_ID'", "name": "'$VPN_SERVICE_NAME'",
"admin_state_up": "'$ADMIN_STATE_UP'", "availability_zone": "'$AZ'" } }' ¥
| jq .
```

The following response will be returned.

```
{
  "vpnservice": {
    "availability_zone": "<Availability Zone>",
    "router_id": "<Router ID for SSL-VPN Connection>",
    "status": "PENDING_CREATE",
    "name": "<VPN Service Name>",
    "admin_state_up": true,
    "subnet_id": "<Subnet ID for SSL-VPN Connection>",
    "tenant_id": "<Project ID for SSL-VPN Connection>",
    "id": "<VPN Service ID>",
    "description": ""
  }
}
```

3. Execute the following API to confirm the status of the created VPN service.

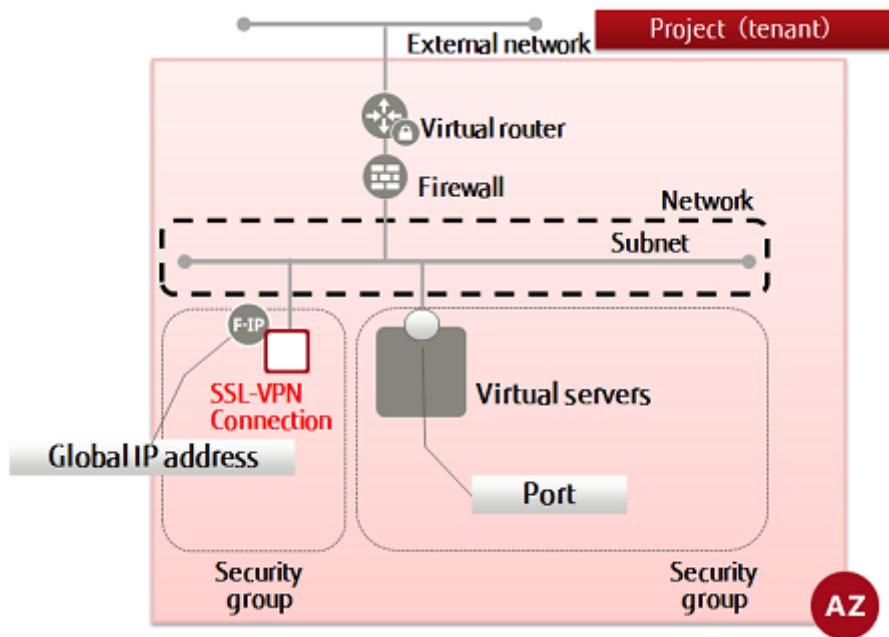
```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
| jq .
```

If the "status" of the created VPN service is "PENDING_CREATE", as shown below, then creation of the SSL-VPN connection is complete.

```
{
  "vpnservices": [
    {
      "availability_zone": "<Availability Zone>",
      "router_id": "<Router ID for SSL-VPN Connection>",
      "status": "PENDING_CREATE",
      "name": "<VPN Service Name>",
      "admin_state_up": true,
      "subnet_id": "<Subnet ID for SSL-VPN Connection>",
      "tenant_id": "<Project ID for SSL-VPN Connection>",
      "id": "<VPN Service ID>",
      "description": ""
    }
  ]
}
```

3.5.1.2 Creating an SSL-VPN connection (V2 service/K5 client certificate)

This section explains the procedure for creating a (V2 service) SSL-VPN connection.



1. Execute the following commands to perform the necessary configuration.

```
$ NAME=<Connection Name (Optional)>
```

```
$ VPNSERVICE_ID=<VPN Service ID>
```

```
$ PROTOCOL=tcp
```

```
$ AZ=<Availability Zone>
```

```
$ CLIENT_ADDRESS_POOL_CIDR=<Virtual Network cidr of VPN Tunnel (Optional)>
```

For "CLIENT_ADDRESS_POOL_CIDR", specify the network address using the format XXX.XXX.XXX.XXX/XX.

Example: 192.168.246.0/24



Warning

Specify a network address that does not conflict with the network address used by K5 or the local network addresses used by client PCs.



Note

When an SSL-VPN connection is created using this procedure, the following settings are configured automatically.

a. Firewall rules

A firewall rule (443/TCP) allowing access to the SSL-VPN connection is added automatically for use during dedicated SSL-VPN connections.

When you want to change the firewall rules that are added automatically, make changes after creating the SSL-VPN connection.

b. Static routing information

The routing information to "client_address_pool_cidr" configured in the SSL-VPN connection is added to virtual routers for use during dedicated SSL-VPN connections.

c. Security groups

A security group in which permissions for all types of communication are configured is created for use during dedicated SSL-VPN connections.

It is also possible to use a security group that has been created by a user by specifying it in the "security_groups" parameter when creating the SSL-VPN connection.

For details, refer to the "Fujitsu Cloud Service K5 IaaS API Reference - Network".

d. Global IP address

If the "floatingips" parameter is not specified, a global IP address to be allocated to the SSL-VPN connection is generated.

When using a global IP address that has been generated in advance, specify the "floatingips" parameter.

For details, refer to the "Fujitsu Cloud Service K5 IaaS API Reference - Network".



Note

In addition to the settings that are configured automatically, it is necessary to define permissions in the security groups and firewall rules so that the virtual network that is configured in the "client_address_pool_cidrs" parameter of the SSL-VPN connection can access the connection destination virtual server.

2. Execute the following API.

```
$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{ "ssl_vpn_v2_connection": { "name": "'$NAME'",
  "client_address_pool_cidrs": [ "'$CLIENT_ADDRESS_POOL_CIDR'" ],
  "admin_state_up": true, "vpnservice_id": "'$VPNSERVICE_ID'",
  "availability_zone": "'$AZ'", "protocol": "'$PROTOCOL'" } }' | jq .
```

The following response will be returned.

```
{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "<Virtual Network cidr of VPN Tunnel>",
          "internal_gateway": "<Private IP Address for SSL-VPN Connection>",
          "external_address": "<Global IP Address for SSL-VPN>"
        }
      ],
      "security_groups": [
        "<Dedicated Security Group for SSL-VPN Connection>"
      ],
      "protocol": "tcp",
      "availability_zone": "<Availability Zone>",
      "tenant_id": "<Project ID>",
      "name": "<SSL-VPN Connection Name>",
      "admin_state_up": true,
      "client_address_pool_cidrs": [
        "<Virtual Network cidr of VPN Tunnel>"
      ],
      "credential_id": null,
      "vpnservice_id": "<VPN Service Name>",
      "id": "<SSL-VPN Connection ID>",
      "status": "PENDING_CREATE"
    }
  ]
}
```

3. Execute the following API to confirm the status of the SSL-VPN connection.

```
$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
| jq .
```

Confirm that the "status" of the SSL-VPN connection is "ACTIVE" as shown in the following response.

```
{  
  "ssl_vpn_v2_connections": [  
    {  
      "access_points": [  
        {  
          "floatingip": null,  
          "client_address_pool_cidr": "<Virtual Network cidr of VPN Tunnel>",  
          "internal_gateway": "<Private IP Address for SSL-VPN Connection>",  
          "external_address": "<Global IP Address for SSL-VPN>"  
        }  
      ],  
      "security_groups": [  
        "<Dedicated Security Group for SSL-VPN Connection>"  
      ],  
      "protocol": "tcp",  
      "availability_zone": "<Availability Zone>",  
      "tenant_id": "<Project ID>",  
      "name": "<SSL-VPN Connection Name>",  
      "admin_state_up": true,  
      "client_address_pool_cids": [  
        "<Virtual Network cidr of VPN Tunnel>"  
      ],  
      "credential_id": null,  
      "vpnservice_id": "<VPN Service Name>",  
      "id": "<SSL-VPN Connection ID>",  
      "status": "ACTIVE"  
    }  
  ]  
}
```

Immediately after creation, in some cases the status may still be PENDING_CREATE. In such cases, wait a while and then execute the confirmation API again.

4. Execute the following API to confirm the status of the VPN service.

```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
| jq .
```

If the "status" of the VPN service is "ACTIVE" as shown in the following response, then creation of the SSL-VPN connection is complete.

```
{  
  "vpnservices": [  
    {  
      "availability_zone": "<Availability Zone>",  
      "router_id": "<Router ID for SSL-VPN Connection>",  
      "status": "ACTIVE",  
      "name": "<VPN Service Name>",  
      "admin_state_up": true,  
      "subnet_id": "<Subnet ID for SSL-VPN Connection>",  
      "tenant_id": "<Project ID for SSL-VPN Connection>",  
      "id": "<VPN Service ID>",  
      "description": ""  
    }  
  ]  
}
```

3.5.1.3 Configuring a connection to a VPN client (V2 service/K5 client certificate)

This section explains the procedure for configuring a (V2 service) connection to a VPN client.



Note

This procedure has been confirmed using the following SSL-VPN client PC environment.

- OS: Windows 7 Professional 64-bit, Japanese Version
- VPN Client: OpenVPN 2.3.12



Note

In this procedure, OpenSSL 1.1.0 is used to convert the format of the certificates.

1. Download the intermediate certificate and the root certificate to be used to create the CA certificates.

Download the intermediate certificate and the root certificate from <http://rms-digicert.ne.jp/howto/basis/digicert-root-certificates.html>.

Intermediate certificate: [DigiCert SHA2 High Assurance Server CA]

Root certificate: [DigiCert High Assurance EV Root CA]

2. Execute the following commands to convert the format of the downloaded certificates.

```
$ openssl x509 -in <Intermediate Certificate Name> -inform DER ¥  
-out <Intermediate Certificate Name (Optional)>
```

```
$ openssl x509 -in <Root Certificate Name> -inform DER ¥  
-out <Root Certificate Name (Optional)>
```

3. Execute the following command to confirm that the conversion was successful.

```
# ls -l
```

Confirm that the certificates are displayed as follows.

```
<Intermediate Certificate Name>  
<Root Certificate Name>
```

4. Merge the intermediate certificate and the root certificate.

Copy the content of the certificates to a text editor.



Note

- Using a text editor, open the two PEM format certificate files that you created.
- Copy the displayed information which starts with "-----BEGIN CERTIFICATE-----" and ends with "-----END CERTIFICATE-----" from each file, and paste the information from both files into a new text file.
- Confirm that the content of the created text file is as follows.

```
-----BEGIN CERTIFICATE-----  
Content of the Intermediate Certificate File (in PEM Format)  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Content of the Root Certificate File (in PEM Format)  
-----END CERTIFICATE-----
```

Save the text file using an arbitrary file name. Save using the file extension ".crt".

5. Issue the client certificate.

For details on issuing a client certificate, refer to "3.3.5.1 Issue a Certificate for Authentication" in the "K5 Portal User Guide".



Warning

Be sure to create a unique client certificate for SSL-VPN connection for each user.

It is not possible for multiple users to use the same certificate for SSL-VPN connection at the same time.

When there are multiple users, obtain as many certificates as there are users from the K5 portal.

6. Converting client certificate formats

For details on converting client certificate formats, refer to "[Converting certificate formats](#) on page 112".

The SSL-VPN client (OpenVPN) explained in this chapter can use the "Encrypted private keys" and the "Unencrypted private keys" described in "[Converting certificate formats](#) on page 112".

7. Install OpenVPN.

Download the installer from <https://www.openvpn.jp/download/>, and then perform installation.

For details on the installation procedure, refer to the "Features Handbook" > "Appendix" > "[Setup of an OpenVPN Client \(Windows\)](#)".

8. Edit the client configuration file.

In the sample-config folder in the installation directory, copy the file client.ovpn to the config folder.

It is necessary to save the CA certificates, the client certificate, and the private key that are created in this procedure to the config folder.

Open client.ovpn in the config folder using a text editor, and edit it as follows.

- Lines beginning with "proto"

```
proto tcp
```

- Lines beginning with "remote"

```
remote [Connection Target Server Address (Global IP Address of the SSL-VPN Connection Resource)] [Connection Target Port (443)]
```

- Lines beginning with "ca"

```
ca <CA Certificate Name>
```

- Lines beginning with "cert"

```
cert <Client Certificate Name>
```

- Lines beginning with "key"

```
key <Client_private_key_name>
```

- Lines beginning with "comp-lzo"

```
#comp-lzo
```

- Lines beginning with "cipher"

```
cipher AES-128-CBC
```

- Lines beginning with "http-proxy" (Specified when connecting using an HTTP proxy server)

```
http-proxy <HTTP Proxy Server Address> <HTTP Proxy Server Port Number> stdin basic
```

stdin: When connecting to the HTTP proxy server, entry of a user name and password will be requested.

basic: The authentication method will be basic authentication.



Example:

```
proto tcp
remote xxx.xxx.xxx.xxx 443
ca ca.crt
cert client.crt
key client.key
#comp-lzo
cipher AES-128-CBC
http-proxy xxx.xxx.xxx.xxx 8080 stdin basic
```

9. Start the OpenVPN client.

Right-click the OpenVPN client icon, and select "Run as administrator" to start the OpenVPN client with administrator privileges.

10. Connect over SSL-VPN.



To connect to a virtual server after establishing a SSL-VPN connection, it is necessary for the following to be configured to allow VPN tunnel network addresses to access the relevant virtual server.

- The firewall of a virtual router for which SSL-VPN functionality has been configured
- The security group allocated to the virtual server

Start the client PC terminal, right-click the OpenVPN icon in the system tray, and then select [Connect].

If the OpenVPN icon in the system tray turns green, then the SSL-VPN connection has been established.

It is possible to connect to the virtual server by specifying its private IP address on the SSL-VPN client PC.

* If you wish to terminate the SSL-VPN connection

To terminate the connection, right-click the OpenVPN icon in the system tray, and select [Disconnect].

If the OpenVPN icon turns grey, then the SSL-VPN connection has been terminated.

3.5.2 SSL-VPN connection (V2 service/self-signed certificate)

This section explains the procedure for SSL-VPN connections that use self-signed certificates. For the procedure for SSL-VPN connections that use K5 client certificates, refer to [SSL-VPN connection \(V2 service/K5 client certificate\)](#) on page 51.

3.5.2.1 Creating certificates for SSL-VPN (V2 service/self-signed certificate)

This section explains the procedure for creating certificates for SSL-VPN.

This procedure describes creation when using a Linux environment (OS: CentOS 6.8, tool: EasyRSA 3.0.1).

This procedure must be performed by a superuser.

* This procedure assumes that all operations are performed in the following directory.

Work directory: /root/EasyRSA-3.0.1/

1. Initialize the CA.

```
# ./easyrsa init-pki
```

The environment for certificate creation is created.

Certificates are generated in the following directories.

- CA certificate, DH key: ./pki/.
- Server and client certificates: ./pki/issued/.
- Private key: ./pki/private/.

2. Create the CA certificate.

```
# ./easyrsa build-ca
```

A message like the following is displayed. Follow the on-screen instructions to create the CA certificate interactively.

```
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/ca.key.0MjyMXpDsW'
Enter PEM pass phrase: "<passphraseForCaCertificate(Enter)>"
Verifying - Enter PEM pass phrase: "<passphraseForCaCertificate(Re-enter)>"
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]: "<commonName(Optional)>"
CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/root/EasyRSA-3.0.1/pki/ca.crt
```

3. Execute the following command to confirm that the CA certificate has been created.

```
# ls -l /root/easyrsa/pki/
```

If the created CA certificate is output as shown below, then creation is complete.

```
-rw-----. 1 root root 1180 MM DD hh:mm ca.crt
```

4. Create the server certificate and the private key.

```
# ./easyrsa build-server-full <serverCertificateAndPrivateKeyName(Optional)> nopass
```

A message like the following is displayed. Follow the on-screen instructions to create the server certificate and private key interactively.

```

Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/server1.key.LCqVP0rjSV'
-----
Using configuration from /root/EasyRSA-3.0.1/openssl-1.0.cnf
Enter pass phrase for /root/EasyRSA-3.0.1/pki/private/ca.key:
"<passphraseForCaCertificate(Re-enter)>"
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :PRINTABLE:'<serverCertificateAndPrivateKeyName>'
Certificate is to be certified until  MM DD hh:mm:ss YYYY GMT (3650 days)

Write out database with 1 new entries
Data Base Updated

```

5. Execute the following commands to confirm that the server certificate and the private key have been created.

Execute the following command to confirm that the server certificate has been created.

```
# ls -l /root/easyrsa/pki/issued/
```

If the created server certificate is output in the following format, then creation is complete.

```
-rw-r--r--. 1 root root 4391 MM DD hh:mm <serverCertificateAndPrivateKeyName>.crt
```

6. Execute the following command in order to confirm that the private key for the server certificate has been created.

Execute the following command to confirm that the private key for the server certificate has been created.

```
# ls -l /root/easyrsa/pki/private/
```

If the created private key is output as shown below, then creation is complete.

```
-rw-r--r--. 1 root root 1704 MM DD hh:mm <serverCertificateAndPrivateKeyName>.key
```

7. Create the DH key.

```
# ./easyrsa gen-dh
```

The following message is displayed.

```

Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....
+.....
+.....
+...+.....+.....+.....+.....
+.....+.....
.....+++++*
DH parameters of size 2048 created at /root/EasyRSA-3.0.1/pki/dh.pem

```

8. Confirm that the DH key has been created.

```
# ls -l /root/easyrsa/pki/
```

If the created DH key is output as shown below, then creation is complete.

```
-rw-r--r--. 1 root root 424 MM DD hh:mm dh.pem
```


9. Create the client private key and certificate.



Warning

Be sure to create a unique client certificate for SSL-VPN connection for each user. It is not possible for multiple users to use the same certificate for SSL-VPN connection at the same time.

```
# ./easyrsa build-client-full <clientPrivateKeyAndCertificateName(Optional)> nopass
```

A message like the following is displayed. Follow the on-screen instructions to create the CA certificate interactively.

Generating a 2048 bit RSA private key

```
+++
.....+++
writing new private key to '/root/EasyRSA-3.0.1/pki/private/bps021.key.a5TcULwXAN'
-----
Using configuration from /root/EasyRSA-3.0.1/openssl-1.0.cnf
Enter pass phrase for /root/EasyRSA-3.0.1/pki/private/ca.key:
"<passphraseForCaCertificate(Re-enter)>"
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName      :PRINTABLE:'<clientPrivateKeyAndCertificateName(Optional)>'
Certificate is to be certified until MM DD hh:mm:ss YYYY GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
```

10. Execute the following command to confirm that the client certificate has been created.

```
# ls -l ./root/easyrsa/pki/issued/
```

If the created client certificate is output as shown below, then creation is complete.

```
-rw-----. 1 root root 1704 MM DD hh:mm <clientPrivateKeyAndCertificateName>.crt
```

11. Execute the following command to confirm that the client private key has been created.

```
# ls -l ./root/easyrsa/pki/private/
```

If the created client private key is output as shown below, then creation is complete.

```
-rw-----. 1 root root 1704 MM DD hh:mm <clientPrivateKeyAndCertificateName>.key
```



Note

Transfer created certificates to client PCs as necessary. Transfer the following certificates to the following client PCs.

- CA certificate
API execution environment client PCs, SSL-VPN client PCs
- Server certificate and private key
API execution environment client PCs
- DH key
API execution environment client PCs
- Client private keys and certificates

3.5.2.2 Registering SSL-VPN certificates (V2 service/self-signed certificate)

This section explains the procedure for registering certificates for SSL-VPN in the K5 IaaS key management function.

1. Register the CA certificate.

```
$ CA_NAME=ca (Fixed value)
```



Specify the above fixed value for CA_NAME.
 Note If the fixed value is not specified, an error will occur when creating the SSL-VPN connection.

```
$ EXPIRATION="<CA Certificate Term of Validity (Optional)>"
```



Specify the **<CA_Certificate_Term_of_VValidity_(Optional)>** using the format YYYY-MM-DDThh:mm:ss.
 Note As the parameter contains metacharacters, enclose it in double quotes.

```
$ CONTENT_TYPE=text/plain
```

2. Execute the following API.

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"name": "$CA_NAME", "expiration": "$EXPIRATION",  
"payload": "<CA_Certificate_Content>",  
"payload_content_type": "$CONTENT_TYPE" }'
```



Configure the **<CA_Certificate_Content>** using the following procedure.

- Display the information of the CA certificate using the above cat command.
- Copy the retrieved information which starts with "-----BEGIN CERTIFICATE-----" and ends with "-----END CERTIFICATE-----".
- Use a text editor, etc. to replace the line feed codes with the character string "\n".
- Use the edited information for the **<CA_Certificate_Content>**.

If the response code 201 is output as shown below, then registration is complete.

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000256040_201
```

```
Location: http://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/<Project ID>/secrets/<CA Certificate ID>
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<Project ID>/secrets/<CA Certificate ID>"}
```

3. Register the server certificate.

```
$ SV_CERT_NAME=server_certificate (Fixed value)
```



Note

Specify the above fixed value for SV_CERT_NAME.

If the fixed value is not specified, an error will occur when creating the SSL-VPN connection.

```
$ EXPIRATION="<Server Certificate Term of Validity (Optional)>"
```



Note

Specify the <Server_Certificate_Term_of_Validity(Optional)> using the format YYYY-MM-DDThh:mm:ss.

By default, the term of validity of the server certificate is configured as 10 years after the time of creation of the certificate. Specify a term of validity less than or equal to 10 years.

As the parameter contains metacharacters, enclose it in double quotes.

```
$ CONTENT_TYPE=text/plain
```

4. Execute the following API.

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{ "name": "'$SV_CERT_NAME'", "expiration": "'$EXPIRATION'",  
  "payload": "<Server_Certificate_Content>",  
  "payload_content_type": "'$CONTENT_TYPE'" }'
```



Note

Configure the <Server_Certificate_Content> using the following procedure.

- Display the information of the server certificate using the above cat command.
- Copy the retrieved information which starts with "-----BEGIN CERTIFICATE-----" and ends with "-----END CERTIFICATE-----".
- Use a text editor, etc. to replace the line feed codes with the character string "\n".
- Use the edited information for the <Server_Certificate_Content>.

If the response code 201 is output as shown below, then registration is complete.

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000225866_201
```

```
Location: http://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/<Project ID>/secrets/<Server Certificate ID>
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/  
<Project ID>/secrets/<Server Certificate ID>"}
```

5. Register the private key for the server certificate.

```
$ SV_KEY_NAME=server_key (Fixed value)
```



Note

Specify the above fixed value for SV_KEY_CERT_NAME.

If the fixed value is not specified, an error will occur when creating the SSL-VPN connection.

```
$ EXPIRATION="<Private Key for Server Certificate Term of Validity (Optional)>"
```



Note

Specify the **<Private_Key_for_Server_Certificate_Term_of_Validity_(Optional)>** using the format YYYY-MM-DDThh:mm:ss.

By default, the term of validity of the private key for the server certificate is configured as 10 years after the time of creation of the private key. Specify a term of validity less than or equal to 10 years.

As the parameter contains metacharacters, enclose it in double quotes.

```
$ CONTENT_TYPE=text/plain
```

6. Execute the following API.

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{ "name": "'$SV_KEY_NAME'", "expiration": "'$EXPIRATION'",  
      "payload": "<Private Key for Server Certificate Content>",  
      "payload_content_type": "'$CONTENT_TYPE'" }
```



Note

Configure the **<Private_Key_for_Server_Certificate_Content>** using the following procedure.

- Display the information of the private key for the server certificate using the above cat command.
- Copy the retrieved information which starts with "-----BEGIN PRIVATE KEY-----" and ends with "-----END PRIVATE KEY-----".
- Use a text editor, etc. to replace the line feed codes with the character string "\n".
- Use the edited information for the **<Private_Key_for_Server_Certificate_Content>**.

If the response code 201 is output as shown below, then registration is complete.

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fcx-Endpoint-Request: EXECUTED_REQ000257600_201
```

```
Location: http://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/<Project ID>/secrets/<ID of Private Key for Server Certificate>
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/  
<Project ID>/secrets/<ID of Private Key for Server Certificate>"}
```

7. Register the DH key.

```
$ DH_NAME=dh (Fixed value)
```



Note

Specify the above fixed value for DH_NAME.

If the fixed value is not specified, an error will occur when creating the SSL-VPN connection.

```
$ EXPIRATION="<DH Key Term of Validity (Optional)>"
```



Note Specify the <DH_Key_Term_of_Validity_(Optional)> using the format YYYY-MM-DDThh:mm:ss.

As the parameter contains metacharacters, enclose it in double quotes.

```
$ CONTENT_TYPE=text/plain
```

8. Execute the following API.

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"name":"' $DH_NAME '", "expiration": "' $EXPIRATION '",
"payload":"' <DH Key Content>', "payload_content_type": "' $CONTENT_TYPE' " }'
```



Note Configure the <DH_Key_Content> using the following procedure.

- Display the information of the DH key using the above cat command.
- Copy the retrieved information which starts with "-----BEGIN DH PARAMETERS-----" and ends with "-----END DH PARAMETERS-----".
- Use a text editor, etc. to replace the line feed codes with the character string "\n".
- Use the edited information for the <DH_Key_Content>.

If the response code 201 is output as shown below, then registration is complete.

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 201 Created
```

```
X-Fox-Endpoint-Request: EXECUTED_REQ000229580_201
```

```
Location: http://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/<Project ID>/secrets/<DH Key ID>
```

```
Content-Type: application/json; charset=UTF-8
```

```
Content-Length: 155
```

```
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

```
{"secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<Project ID>/secrets/<DH Key ID>"}
```

9. Execute the following API to confirm the certificates that were registered using this procedure.

```
$ curl -X GET -sS $KEYMANAGEMENT/v1/$PROJECT_ID/secrets ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If the registered certificates are output as shown below, then creation is complete.

```
{
  "total": 4,
  "secrets": [
    {
      "expiration": "<CA Certificate Term of Validity>",
      "bit_length": null,
      "status": "ACTIVE",
      "secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<projectId>/secrets/<caCertificateID>",
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",
      "name": "ca",
      "algorithm": null,
      "created": "YYYY-MM-DDThh:mm:ss.SSS",
      "content_types": {
        "default": "text/plain"
      },
      "mode": null
    }
  ]
}
```

```

    },
    {
      "expiration": "<Server Certificate Term of Validity>",
      "bit_length": null,
      "status": "ACTIVE",
      "secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<projectId>/secrets/<serverCertificateId>",
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",
      "name": "server_certificate",
      "algorithm": null,
      "created": "YYYY-MM-DDThh:mm:ss.SSS",
      "content_types": {
        "default": "text/plain"
      },
      "mode": null
    },
    {
      "bit_length": null,
      "status": "ACTIVE",
      "secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<projectId>/secrets/<idOfPrivateKeyForServerCertificate>",
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",
      "name": "server_key",
      "algorithm": null,
      "created": "YYYY-MM-DDThh:mm:ss.SSS",
      "content_types": {
        "default": "text/plain"
      },
      "mode": null
    },
    {
      "expiration": "<DH Key Term of Validity (Optional)>",
      "bit_length": null,
      "status": "ACTIVE",
      "secret_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<projectId>/secrets/<dhKeyId>",
      "updated": "YYYY-MM-DDThh:mm:ss.SSS",
      "name": "dh",
      "algorithm": null,
      "created": "YYYY-MM-DDThh:mm:ss.SSS",
      "content_types": {
        "default": "text/plain"
      },
      "mode": null
    }
  ]
}

```



Note

Take a note of the "secret_ref" values for the certificates registered using this procedure, as these values will be used during ["Creating an SSL-VPN connection \(V2 service/self-signed certificate\)"](#) on page 70.

3.5.2.3 Creating a key container for SSL-VPN (V2 service/self-signed certificate)

This section explains the procedure for creating a key container for SSL-VPN.



Note

A container is an aggregate of a CA certificate, a server certificate, a private key, and a DH key. This container will later be used when creating an SSL-VPN connection. When creating this container, it is necessary to specify the following information as fixed values.

- type: "generic"
- name: "ca", "server_certificate", "server_key", "dh"

1. Execute the following commands to perform the necessary configuration.

```
$ CONTAINER_NAME=<Key Container Name (Optional)>
```

```
$ TYPE=generic (Fixed value)
```

```
$ CA_NAME=ca (Fixed value)
```

```
$ CA_URL=<secret_ref of CA Certificate>
```

```
$ SV_CERT_NAME=server_certificate (Fixed value)
```

```
$ SV_CERT_URL=<secret_ref of Server Certificate>
```

```
$ SV_KEY_NAME=server_key (Fixed value)
```

```
$ SV_KEY_URL=<secret_ref of Private Key for Server Certificate>
```

```
$ DH_NAME=dh (Fixed value)
```

```
$ DH_URL=<secret_ref of DH Key>
```

2. Execute the following API.

```
$ curl -X POST -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/containers ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"name": "'$CONTAINER_NAME'", "type": "'$TYPE'",  
  "secret_refs": [{"name": "'$CA_NAME'", "secret_ref": "'$CA_URL'",  
    {"name": "'$SV_CERT_NAME'", "secret_ref": "'$SV_CERT_URL'",  
    {"name": "'$SV_KEY_NAME'", "secret_ref": "'$SV_KEY_URL'",  
    {"name": "'$DH_NAME'", "secret_ref": "'$DH_URL'"} ] }'
```

A response like the following will be returned.

```
HTTP/1.1 201 Created  
X-Fcx-Endpoint-Request: EXECUTED_REQ000257862_201  
Location: http://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/<Project ID>/containers/<Key Container ID>  
Content-Type: application/json; charset=UTF-8  
Content-Length: 161  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

3. Execute the following API to confirm that the key container has been created.

```
$ curl -X GET -sS $KEYMANAGEMENT/v1/$PROJECT_ID/containers ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
| jq .
```

If the certificates registered to the container are output as shown below, then creation is complete.

```
{  
  "containers": [  
    {  
      "type": "generic",  
      "container_ref": "https://keymanagement.<Region Identifier>.cloud.global.fujitsu.com/v1/<Project ID>/containers/<Key Container ID>",  
      "created": "YYYY-MM-DDThh:mm:ss.SSS",  
      "secret_refs": [  
        {  
          "name": "ca",  
          "secret_id": "<CA Certificate ID>"  
        },  
        {  
          "name": "server_certificate",  
          "secret_id": "<Server Certificate ID>"  
        }  
      ]  
    }  
  ]  
}
```

```

    },
    {
      "name": "server_key",
      "secret_id": "<ID of Private Key for Server Certificate>"
    },
    {
      "name": "dh",
      "secret_id": "<DH Key ID>"
    }
  ],
  "name": "<Key Container Name>",
  "updated": "YYYY-MM-DDThh:mm:ss.SSS",
  "status": "ACTIVE"
}
],
"total": 1
}

```

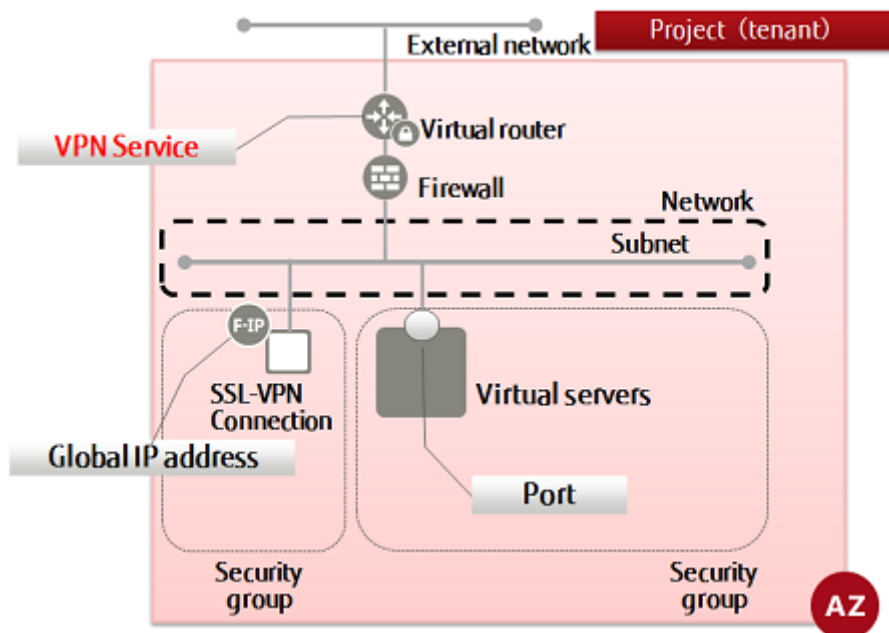


Note

Take a note of the <Key Container ID> that is given at the end of the "container_ref" parameter of the key container, as this value will be used in ["Creating an SSL-VPN connection \(V2 service/self-signed certificate\)"](#) on page 70".

3.5.2.4 Creating a VPN service (V2 service/self-signed certificate)

This section explains the procedure for creating a VPN service.



1. Execute the following commands to perform the necessary configuration.

```
$ SUBNET_ID=<Desired Subnet ID for SSL-VPN Connection>
```

```
$ ROUTER_ID=<Router ID to Be Used for SSL-VPN>
```

```
$ VPN_SERVICE_NAME=<VPN Service Name (Optional)>
```

```
$ ADMIN_STATE_UP=true
```

```
$ AZ=<Availability Zone>
```



Warning

The parameter definition of the subnet for connecting using an SSL-VPN connection must be specified as follows:

- cidr

The mask value for the network address is specified in the range of "16bit - 29bit".

Example: 192.168.1.0/24

- gateway_ip

The IP address of the virtual router specified for the VPN service is specified.

2. Execute the following API.

```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{"vpnservice": {"subnet_id": "'$SUBNET_ID'",
"router_id": "'$ROUTER_ID'", "name": "'$VPN_SERVICE_NAME'",
"admin_state_up": "'$ADMIN_STATE_UP'", "availability_zone": "'$AZ'" }}' ¥
| jq .
```

The following response will be returned.

```
{
  "vpnservice": {
    "availability_zone": "<Availability Zone>",
    "router_id": "<Router ID for SSL-VPN Connection>",
    "status": "PENDING_CREATE",
    "name": "<VPN Service Name>",
    "admin_state_up": true,
    "subnet_id": "<Subnet ID for SSL-VPN Connection>",
    "tenant_id": "<Project ID for SSL-VPN Connection>",
    "id": "<VPN Service ID>",
    "description": ""
  }
}
```

3. Execute the following API to confirm the status of the created VPN service.

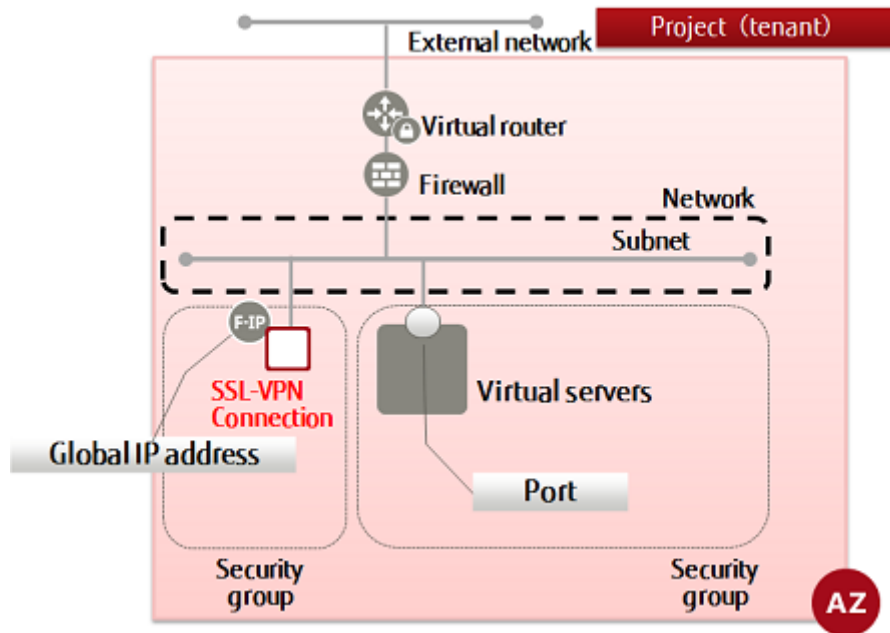
```
$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
| jq .
```

If the "status" of the created VPN service is "PENDING_CREATE", as shown below, then creation of the SSL-VPN connection is complete.

```
{
  "vpnservices": [
    {
      "availability_zone": "<Availability Zone>",
      "router_id": "<Router ID for SSL-VPN Connection>",
      "status": "PENDING_CREATE",
      "name": "<VPN Service Name>",
      "admin_state_up": true,
      "subnet_id": "<Subnet ID for SSL-VPN Connection>",
      "tenant_id": "<Project ID for SSL-VPN Connection>",
      "id": "<VPN Service ID>",
      "description": ""
    }
  ]
}
```

3.5.2.5 Creating an SSL-VPN connection (V2 service/self-signed certificate)

This section explains the procedure for creating a (V2 service) SSL-VPN connection.



1. Execute the following commands to perform the necessary configuration.

```
$ NAME=<Connection Name (Optional)>
```

```
$ CREDENTIAL_ID=<Key Container ID>
```

```
$ VPNSERVICE_ID=<VPN Service ID>
```

```
$ PROTOCOL=tcp
```

```
$ AZ=<Availability Zone>
```

```
$ CLIENT_ADDRESS_POOL_CIDR=<Virtual Network cidr of VPN Tunnel (Optional)>
```

For "CLIENT_ADDRESS_POOL_CIDR", specify the network address using the format XXX.XXX.XXX.XXX/XX.

Example: 192.168.246.0/24



Warning

Specify a network address that does not conflict with the network address used by K5 or the local network addresses used by client PCs.



Note

When an SSL-VPN connection is created using this procedure, the following settings are configured automatically.

- a. Firewall rules

A firewall rule (443/TCP) allowing access to the SSL-VPN connection is added automatically for use during dedicated SSL-VPN connections.

When you want to change the firewall rules that are added automatically, make changes after creating the SSL-VPN connection.

- b. Static routing information

The routing information to "client_address_pool_cidr" configured in the SSL-VPN connection is added to virtual routers for use during dedicated SSL-VPN connections.

c. Security groups

A security group in which permissions for all types of communication are configured is created for use during dedicated SSL-VPN connections.

It is also possible to use a security group that has been created by a user by specifying it in the "security_groups" parameter when creating the SSL-VPN connection.

For details, refer to the "Fujitsu Cloud Service K5 IaaS API Reference - Network".

d. Global IP address

If the "floatingips" parameter is not specified, a global IP address to be allocated to the SSL-VPN connection is generated.

When using a global IP address that has been generated in advance, specify the "floatingips" parameter.

For details, refer to the "Fujitsu Cloud Service K5 IaaS API Reference - Network".



Note

In addition to the settings that are configured automatically, it is necessary to define permissions in the security groups and firewall rules so that the virtual network that is configured in the "client_address_pool_cidrs" parameter of the SSL-VPN connection can access the connection destination virtual server.

2. Execute the following API.

```
$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X POST \
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" \
-d '{ "ssl_vpn_v2_connection": { "name": "'$NAME'",
  "client_address_pool_cidrs": [ "'$CLIENT_ADDRESS_POOL_CIDR' " ],
  "admin_state_up": true, "credential_id": "'$CREDENTIAL_ID'",
  "vpnservice_id": "'$VPNSERVICE_ID'", "availability_zone": "'$AZ'",
  "protocol": "'$PROTOCOL'" }}' | jq .
```

The following response will be returned.

```
{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "<Virtual Network cidr of VPN Tunnel>",
          "internal_gateway": "<Private IP Address for SSL-VPN Connection>",
          "external_address": "<Global IP Address for SSL-VPN>"
        }
      ],
      "security_groups": [
        "<Dedicated Security Group for SSL-VPN Connection>"
      ],
      "protocol": "tcp",
      "availability_zone": "<Availability Zone>",
      "tenant_id": "<Project ID>",
      "name": "<SSL-VPN Connection Name>",
      "admin_state_up": true,
      "client_address_pool_cidrs": [
        "<Virtual Network cidr of VPN Tunnel>"
      ],
      "credential_id": null,
    }
  ]
}
```

```

    "vpnservice_id": "<VPN Service Name>",
    "id": "<SSL-VPN Connection ID>",
    "status": "PENDING_CREATE"
  }
]
}

```

3. Execute the following API to confirm the status of the SSL-VPN connection.

```

$ curl -sS $NETWORK/v2.0/vpn/ssl-vpn-v2-connections -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥
| jq .

```

Confirm that the "status" of the SSL-VPN connection is "ACTIVE" as shown in the following response.

```

{
  "ssl_vpn_v2_connections": [
    {
      "access_points": [
        {
          "floatingip": null,
          "client_address_pool_cidr": "<Virtual Network cidr of VPN Tunnel>",
          "internal_gateway": "<Private IP Address for SSL-VPN Connection>",
          "external_address": "<Global IP Address for SSL-VPN>"
        }
      ],
      "security_groups": [
        "<Dedicated Security Group for SSL-VPN Connection>"
      ],
      "protocol": "tcp",
      "availability_zone": "<Availability Zone>",
      "tenant_id": "<Project ID>",
      "name": "<SSL-VPN Connection Name>",
      "admin_state_up": true,
      "client_address_pool_cids": [
        "<Virtual Network cidr of VPN Tunnel>"
      ],
      "credential_id": null,
      "vpnservice_id": "<VPN Service Name>",
      "id": "<SSL-VPN Connection ID>",
      "status": "ACTIVE"
    }
  ]
}

```

Immediately after creation, in some cases the status may still be PENDING_CREATE. In such cases, wait a while and then execute the confirmation API again.

4. Execute the following API to confirm the status of the VPN service.

```

$ curl -sS $NETWORK/v2.0/vpn/vpnservices -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥
| jq .

```

If the "status" of the VPN service is "ACTIVE" as shown in the following response, then creation of the SSL-VPN connection is complete.

```

{
  "vpnservices": [
    {
      "availability_zone": "<Availability Zone>",
      "router_id": "<Router ID for SSL-VPN Connection>",
      "status": "ACTIVE",
      "name": "<VPN Service Name>",
      "admin_state_up": true,

```

```

    "subnet_id": "<Subnet ID for SSL-VPN Connection>",
    "tenant_id": "<Project ID for SSL-VPN Connection>",
    "id": "<VPN Service ID>",
    "description": ""
  }
]
}

```

3.5.2.6 Configuring a connection to a VPN client (V2 service/ self-signed certificate)

This section explains the procedure for configuring a (V2 service) connection to a VPN client.



Note

This procedure has been confirmed using the following SSL-VPN client PC environment.

- OS: Windows 7 Professional 64-bit, Japanese Version
- VPN Client: OpenVPN 2.3.12

1. Install OpenVPN.

Download the installer from <https://www.openvpn.jp/download/>, and then perform installation.

For details on the installation procedure, refer to the appendix "[Setup of an SSL-VPN Client \(Windows\)](#)" in the "Features Handbook".

2. Edit the client configuration file.

In the sample-config folder in the installation directory, copy the file client.ovpn to the config folder.

It is necessary to save the CA certificate, the client certificates and private keys created in [Creating certificates for SSL-VPN](#) to the config folder.

Open client.ovpn in the config folder using a text editor, and edit it as follows.

- Lines beginning with "proto"

```
proto tcp
```

- Lines beginning with "remote"

```
remote [Connection Target Server Address (Global IP Address of the SSL-VPN Connection Resource)] [Connection Target Port (443)]
```

- Lines beginning with "ca"

```
ca <CA Certificate Name>
```

- Lines beginning with "cert"

```
cert <Client Certificate Name>
```

- Lines beginning with "key"

```
key <Client_private_key_name>
```

- Lines beginning with "comp-lzo"

```
#comp-lzo
```

- Lines beginning with "cipher"

```
cipher AES-128-CBC
```

- Lines beginning with "http-proxy" (Specified when connecting using an HTTP proxy server)

```
http-proxy <HTTP Proxy Server Address> <HTTP Proxy Server Port Number> stdin basic
```

stdin: When connecting to the HTTP proxy server, entry of a user name and password will be requested.

basic: The authentication method will be basic authentication.



Note

Example:

```
proto tcp
remote xxx.xxx.xxx.xxx 443
ca ca.crt
cert client.crt
key client.key
#comp-lzo
cipher AES-128-CBC
http-proxy xxx.xxx.xxx.xxx 8080 stdin basic
```

3. Start the OpenVPN client.

Right-click the OpenVPN client icon, and select "Run as administrator" to start the OpenVPN client with administrator privileges.

4. Connect over SSL-VPN.



Note

To connect to a virtual server after establishing a SSL-VPN connection, it is necessary for the following to be configured to allow VPN tunnel network addresses to access the relevant virtual server.

- The firewall of a virtual router for which SSL-VPN functionality has been configured
- The security group allocated to the virtual server

Start the client PC terminal, right-click the OpenVPN icon in the system tray, and then select [Connect].

If the OpenVPN icon in the system tray turns green, then the SSL-VPN connection has been established.

It is possible to connect to the virtual server by specifying its private IP address on the SSL-VPN client PC.

* If you wish to terminate the SSL-VPN connection

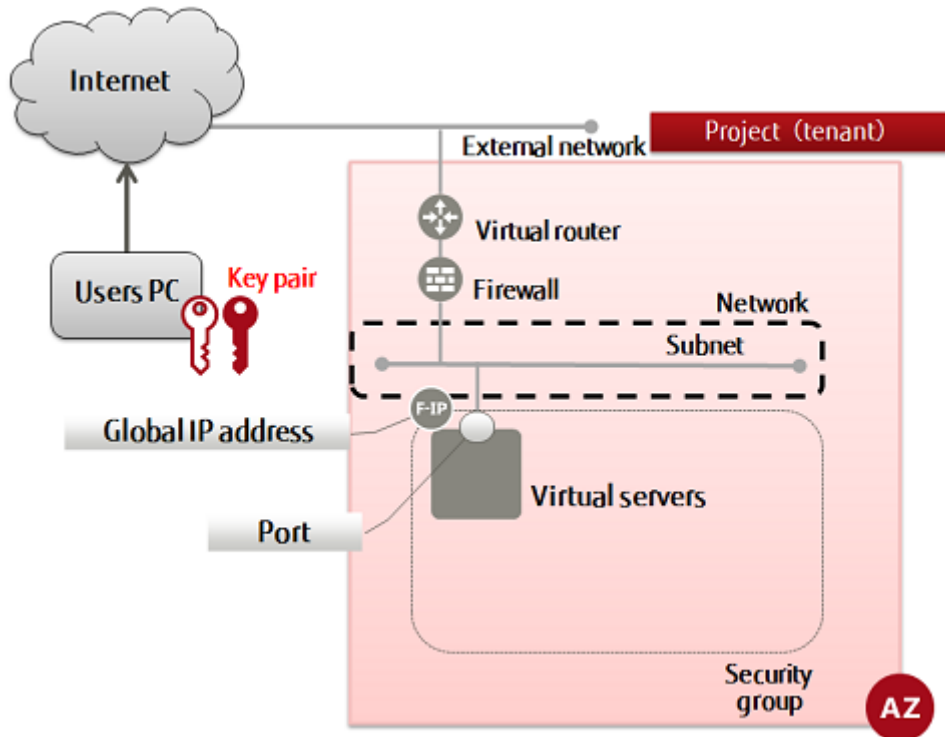
To terminate the connection, right-click the OpenVPN icon in the system tray, and select [Disconnect].

If the OpenVPN icon turns grey, then the SSL-VPN connection has been terminated.

3.6 Create a virtual server

3.6.1 Creating a key pair

This section explains how to create a key pair to be used when logging in to a virtual server using SSH.



When registering the key pair, create the key file (*.pem), which is used for SSH authentication. You can use this key pair to log in to a virtual server.



The key file should be managed with care.

Important

1. Set the environment variable below as follows:

```
$ KEYPAIR=<keyPairName> (any)
```

```
$ AZ=<availabilityZoneName>
```

2. Execute the following API:

```
$ curl -X POST -Ss $COMPUTE/v2/$PROJECT_ID/os-keypairs ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
-d '{"keypair": {"name": "'$KEYPAIR'", "availability_zone": "'$AZ'"}}' | ¥  
jq -r .keypair.private_key > $KEYPAIR.pem
```

3. Enable the key file:

```
$ chmod 600 $KEYPAIR.pem
```

4. Display the content of the key file:

```
cat $KEYPAIR.pem
```

The following response is output:

```
-----BEGIN RSA PRIVATE KEY-----  
~~omitted~~  
-----END RSA PRIVATE KEY-----
```

5. Execute the following API to check the key pair that you created:

```
$ curl -X GET -Ss $COMPUTE/v2/$PROJECT_ID/os-keypairs ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
| jq .
```

If a list including the key pair name that you specified is output, as follows, that means the key pair was created successfully.

```
{  
  "keypairs": [  
    ...  
    {  
      "keypair": {  
        "public_key": "ssh-rsa  
        ~~content omitted~~  
        == Generated by Nova¥n",  
        "name": "<keyPairName>",  
        "fingerprint": "64:96:18:f9:f2:96:e9:7d:f2:b3:dd:ee:bc:45:eb:a0"  
      }  
    },  
    ...  
  ]  
}
```

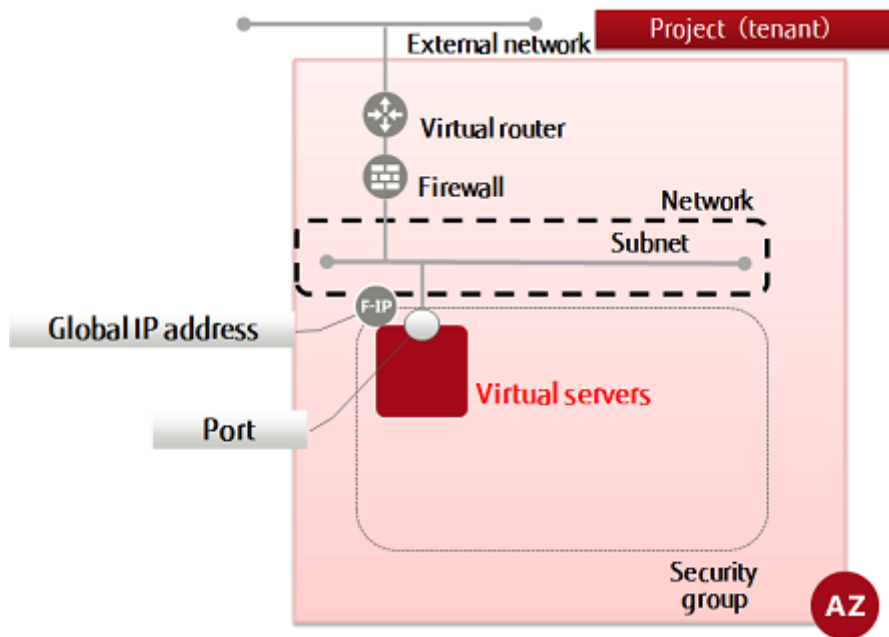


Tip

Apart from generating key pairs, it is also possible to import key pairs created externally.

3.6.2 Preparing to create a virtual server (listing virtual server images)

This section explains how to list virtual server OS images provided by the OS provision service.



Select the OS of a virtual server from the following list. Select the OS image you want to use when deploying a virtual server.

Refer to "[OS Provision Service](#)" in the "Features Handbook" for the OS images provided by K5.

To display the list, execute the following API:

```
$ curl -X GET -sS $COMPUTE/v2/$PROJECT_ID/images/detail ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq '.images[] | .name, .id, .status'
```

A response is output in a format such as the following.

```
"<imgName>"
"<imgId>"
"<imgStatus>"
```

The following response is output:

```
"Ubuntu Server 14.04 LTS (English) 01"
"ffa17298-537d-40b2-a848-0a4d22b49df5"
"ACTIVE"
"FJK5-TemplateBuilder-V02"
"14117885-4104-45a1-9bcd-6dc572e9ee5f"
"ACTIVE"
"Windows Server 2012 SE 64bit (Japanese) 01"
"30718484-1002-43bf-9cf5-f1777c6ed4cb"
"ACTIVE"
"Windows Server 2012 R2 SE 64bit (Japanese) 03"
"5ab16551-c229-4611-834b-a16e074c187e"
"ACTIVE"
"Red Hat Enterprise Linux 6.5 64bit (English) 02"
"db9766f0-c95c-4f1c-bb29-304a90405e3e"
"ACTIVE"
"Windows Server 2008 R2 EE 64bit SP1 (Japanese) 02"
"dcbd4261-e5d1-4d49-9e4f-7434a14dbf4e"
"ACTIVE"
"CentOS 6.5 64bit (English) 03"
"839c1db6-738c-4e2b-9a1d-c14977564203"
"ACTIVE"
"Windows Server 2008 R2 SE 64bit SP1 (Japanese) 02"
"0e2bd896-0ede-4e00-bf71-248ef92c2202"
"ACTIVE"
```

When creating a virtual server, specify the image ID.

3.6.3 Preparing to create a virtual server (listing virtual server types)

This section explains how to list of virtual server types (flavors) to be provided.

Refer to ["Virtual Server"](#) in the "Features Handbook" for the types (flavors) of virtual servers provided.

To display the list, execute the following API:

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/flavors/detail -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.flavors[] | {name: .name, id: .id}'
```

A response is output in a format such as the following:

```
{
  "name": "<flavorName>",
  "id": "<flavorId>"
}
```

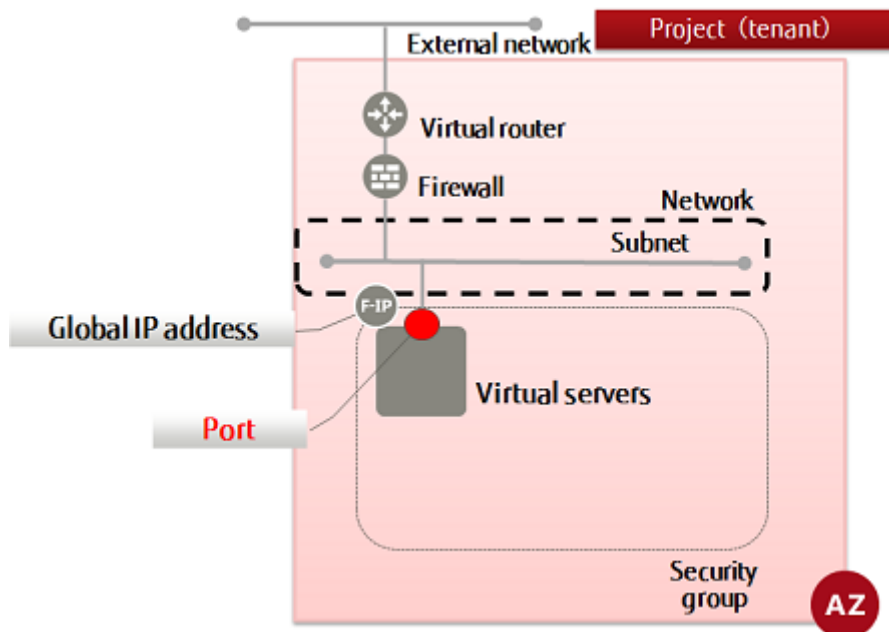
The following response is output:

```
{
  "name": "S-1",
  "id": "1101"
},
{
  "name": "S-2",
  "id": "1102"
},
{
  "name": "S-4",
  "id": "1103"
},
{
  "name": "S-8",
  "id": "1104"
},
{
  "name": "S-16",
  "id": "1105"
},
{
  "name": "M-1",
  "id": "1201"
},
{
  "name": "M-2",
  "id": "1202"
},
{
  "name": "M-4",
  "id": "1203"
},
{
  "name": "M-8",
  "id": "1204"
},
{
  "name": "M-16",
  "id": "1205"
}
```

When creating a virtual server, specify the flavor ID.

3.6.4 Creating a port

This section explains how to create a port (network interface) to be associated with an IP address, for connecting resources such as virtual servers to a network.



If only the subnet is specified when creating the resources below, the system will automatically create a port and assign an IP address to it.

- Virtual server



Note

To manually specify an IP address instead of having one automatically assigned using DHCP when deploying a virtual server, create a port associated with that IP address in advance, and then assign it to the virtual server.



Tip

You can add multiple ports to a virtual server.

- Virtual router



Note

A port is assigned automatically only when created for a default gateway (x.x.x.1). When adding a virtual router to a network that already has a virtual router connected with the x.x.x.1 address, it is necessary to configure the port manually.

1. Set the environment variables below as follows:

```
$ PORT_NAME=<portName> (any)
```

```
$ NETWORK_ID=<networkId>
```

```
$ SUBNET_ID=<subnetId>
```

```
$ FIXED_IP_ADDRESS=<ipAddrToBeSpecified> (specify using format "XXX.XXX.XXX.XXX" within a range specified during creation of subnet)
```

```
$ SG_ID=<securityGroupId>
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/ports -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
```

```
-d '{"port": {"network_id": "'$NETWORK_ID'", "name": "'$PORT_NAME'",
"availability_zone": "'$AZ'", "fixed_ips":
[{"subnet_id": "'$SUBNET_ID'", "ip_address": "'$FIXED_IP_ADDRESS'"}],
"security_groups": ["'$SG_ID'"]}]' | jq .
```

The following response is output:

```
{
  "port": {
    "status": "<status>",
    "name": "<portName>",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "<networkId>",
    "tenant_id": "<projId>",
    "binding:vnic_type": "normal",
    "device_owner": "",
    "mac_address": "<macAddr>",
    "fixed_ips": [
      {
        "subnet_id": "<subnetId>",
        "ip_address": "<ipAddrToBeSpecified>XXX.XXX.XXX.XXX"
      }
    ],
    "id": "<portId>",
    "security_groups": [
      null
    ],
    "device_id": "",
    "availability_zone": "<availabilityZone>"
  }
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the port that you created:

```
$ curl -sS $NETWORK/v2.0/ports -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" ¥
| jq '.ports[] | .name, .status, .id, .fixed_ips[]'
```

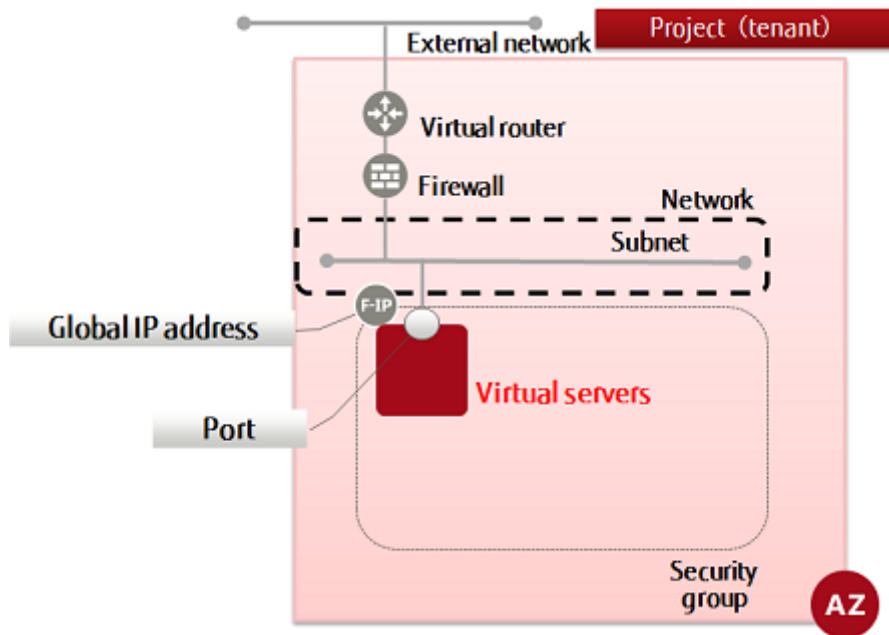
If a list including the port name that you specified is output, as follows, that means the port was created successfully.

```
...
""
"<status>"
"<portId>"
{
  "subnet_id": "<subnetId>",
  "ip_address": "<privateIpAddr>>"
}
...
```

3.6.5 Creating a virtual server (CentOS and port specification)

This section explains how to specify a port (private IP address) and create a virtual server (CentOS).

Refer to "Creating a virtual server (CentOS and DHCP retrieval)" for details on retrieving a port (private IP address) using DHCP.



1. Set the environment variables below as follows:

```
$ VM_NAME=<virtualServerName> (any)
$ FLAVOR_ID=<specOfVirtualServer> (listed flavor ID)
$ VOL_SIZE=30
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
$ IMAGE_REF_ID=839c1db6-738c-4e2b-9a1d-c14977564203<imgIdOfCentos>
$ SOURCE=<blockStorageType>
$ DESTINATION=<connectionDestinationVol>
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of the virtual server, or 0 otherwise)
$ KEYNAME=<keyPairName>
$ INSTANCE_MAX=<maxNumOfServers>
$ INSTANCE_MIN=<minNumOfServers>
$ PORT_ID=<portIdToBeSpecified>
```

2. Execute the following API:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{ "server": { "name": "'$VM_NAME'", "imageRef": "",
"flavorRef": "'$FLAVOR_ID'", "block_device_mapping_v2":
[ { "boot_index": "0", "uuid": "'$IMAGE_REF_ID'",
"volume_size": "'$VOL_SIZE'", "device_name": "'$DEVICE_NAME'",
"source_type": "'$SOURCE'", "destination_type": "'$DESTINATION'",
"delete_on_termination": '$ISDELETE' } ], "key_name": "'$KEYNAME'",
"max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',
"networks": [ { "port": "'$PORT_ID'" } ] } }' | jq .
```

The following response is output:

```
{
  "server": {
    "security_groups": [
      {
        "name": "default"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<newVirtualServerId>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<projId>/servers/<newVirtualServerId>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. Execute the following API to check the virtual server that you created:

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | ¥
jq '.servers[] | {status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] |
.name, name: .name, }'
```

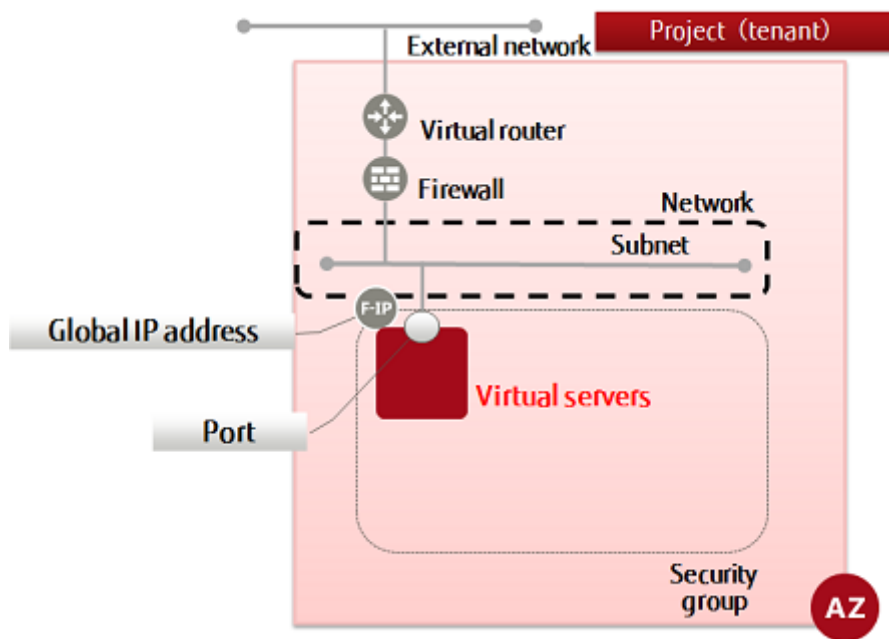
If a list including the virtual server name that you specified is output, as follows, that means the virtual server was created successfully.

```
...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<secGroupName>",
  "name": "<virtualServerName>"
}
...
```

3.6.6 Creating a virtual server (CentOS and DHCP retrieval)

This section explains how to retrieve a port (private IP address) using DHCP and create a virtual server (CentOS).

Refer to "[Creating a virtual server \(CentOS and port specification\)](#) on page 80" for details on specifying a port (private IP address).



1. Set the environment variables below as follows:

```
$ VM_NAME=<virtualServerName> (any)
```

```
$ FLAVOR_ID=<specOfVirtualServer> (listed flavor ID)
```

```
$ VOL_SIZE=30
```

```
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
```

```
$ SOURCE=<blockStorageType>
```

```
$ IMAGE_REF_ID=839c1db6-738c-4e2b-9a1d-c14977564203<imgIdOfCentos>
```

```
$ DESTINATION=<connectionDestinationVol> (specify volume)
```

```
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of the virtual server, or 0 otherwise)
```

```
$ KEYNAME=<keyPairName>
```

```
$ INSTANCE_MAX=<maxNumOfServers>
```

```
$ INSTANCE_MIN=<minNumOfServers>
```

```
$ NETWORK_ID=<networkIdThatYouWantToConnectTo>
```

```
$ SG_NAME=<secGroupNameToBeSpecified>
```

2. Execute the following API:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{ "server": { "name": "'$VM_NAME'", "availability_zone": "'$AZ'",
  "imageRef": "", "flavorRef": "'$FLAVOR_ID'", "block_device_mapping_v2":
  [ { "boot_index": "0", "uuid": "'$IMAGE_REF_ID'",
    "volume_size": "'$VOL_SIZE'", "device_name": "'$DEVICE_NAME'",
    "source_type": "'$SOURCE'", "destination_type": "'$DESTINATION'",
    "delete_on_termination": '$ISDELETE' } ], "key_name": "'$KEYNAME'",
  "max_count": '$INSTANCE_MAX', "min_count": '$INSTANCE_MIN',
  "networks": [ { "uuid": "'$NETWORK_ID'" } ],
  "security_groups": [ { "name": "'$SG_NAME'" } ] } }' | jq .
```

The following response is output:

```
{
  "server": {
    "security_groups": [
      {
        "name": "<specifiedSecGroupName>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<newVirtualServerId>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<projId>/servers/<newVirtualServerId>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. Execute the following API to check the virtual server that you created:

```
$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.servers[] | {status: .status, network_name: .addresses |
  keys, key_name: .key_name, flavor_id: .flavor |
  .id, vm_id: .id, security_group: .security_groups[] |
  .name, name: .name, }'
```

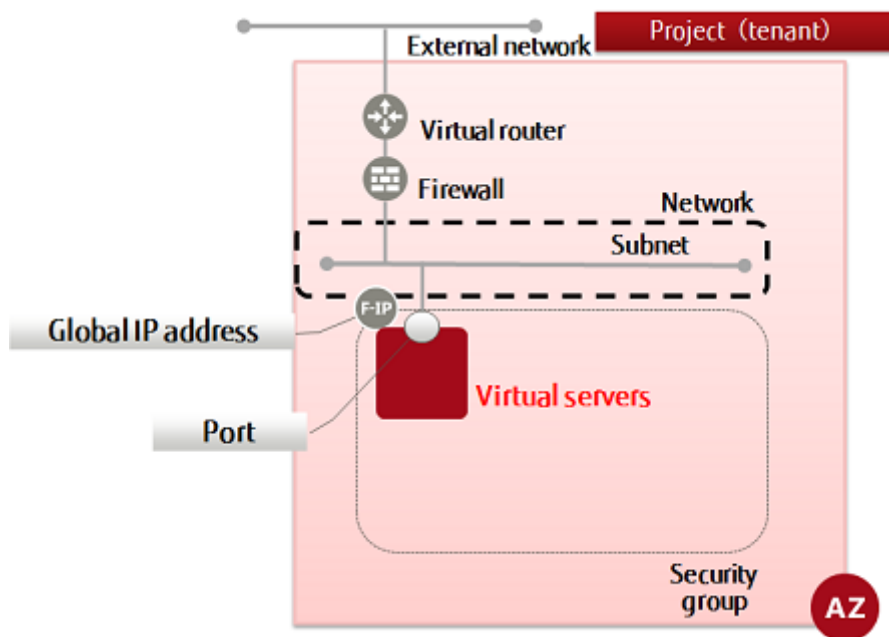
If a list including the virtual server name that you specified is output, as follows, that means the virtual server was created successfully.

```
...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<specifiedSecGroupName>",
  "name": "<virtualServerName>"
}
...
```

3.6.7 Creating a virtual server (Windows and port specification)

This section explains how to specify a port (private IP address) and create a virtual server (Windows).

Refer to "Creating a virtual server (Windows and DHCP retrieval)" for details on retrieving a port (private IP address) using DHCP.



Follow the procedure below to create a virtual server and obtain a password. A password is required when logging in to the remote desktop.

- Set key pairs for password encryption and decryption, and create a virtual server
- Retrieve the encrypted password
- Use the key pair to decrypt and retrieve the password

⚠ Danger It is possible for other users to reference the passwords when using a Windows virtual server that has been deployed with admin_pass specified in the metadata, so you should change the password after deployment.

1. Set key pairs for password encryption and decryption, and create a virtual server.

Set the environment variables below as follows:

```
$ VM_NAME=<virtualServerName> (any)
```

```
$ IMAGE_REF_ID=<windowsimgId>
```

```
$ FLAVOR_ID=<specOfVirtualServer> (listed flavor ID)
```

```
$ VOL_SIZE=80
```

```
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
```

```
$ SOURCE=<blockStorageType>
```

```
$ DESTINATION=<connectionDestinationVol> (specify volume)
```

```
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of the virtual server, or 0 otherwise)
```

```
$ KEYNAME=<keyPairName>
```

```
$ INSTANCE_MAX=<maxNumOfServers>
```

```
$ INSTANCE_MIN=<minNumOfServers>
```

```
$ PORT_ID=<portIdToBeSpecified>
```

2. Execute the following API:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥
-d '{ "server": { "name": "'$VM_NAME'", "imageRef": "",
"flavorRef": "'$FLAVOR_ID'", "block_device_mapping_v2":
[ { "boot_index": "0", "uuid": "'$IMAGE_REF_ID'",
"volume_size": "'$VOL_SIZE'", "source_type": "'$SOURCE'",
"destination_type": "'$DESTINATION'",
"delete_on_termination": '$ISDELETE' } ], "max_count": '$INSTANCE_MAX',
"min_count": '$INSTANCE_MIN', "key_name": "'$KEYNAME'",
"networks": [ { "port": "'$PORT_ID'" } ] } }' | jq .
```

The following response is output:

```
{
  "server": {
    "security_groups": [
      {
        "name": "<specifiedSecGroupName>"
      }
    ],
    "OS-DCF:diskConfig": "MANUAL",
    "id": "<newVirtualServerId>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<projId>/servers/<newVirtualServerId>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
        "rel": "bookmark"
      }
    ]
  }
}
```

3. Execute the following API to check the virtual server that you created:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET ¥
-H "X-Auth-Token: $OS_AUTH_TOKEN" | ¥
jq '.servers[] | {status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] |
.name, name: .name, }'
```

If a list including the virtual server name that you specified is output, as follows, that means the virtual server was created successfully.

```
...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<specifiedSecGroupName>",
  "name": "<virtualServerName>"
}
...
```

- Retrieve the encrypted password.

After creating the virtual server, the interval before password retrieval can be performed is 10 minutes.

Set the environment variables below as follows:

```
$ SERVER_ID=<virtualServerId>
```

```
$ PROJECT_ID=<projId>
```

- Execute the following API:

```
$ curl -s $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-server-password ¥  
-X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

The response will be like the following:

```
{  
  "password": "~~~~ <encryptedPassword> ~~~~=="  
}
```

- Use the key pair to decrypt and retrieve the password.

Set the environment variable below as follows:

```
$ PASSWORD=<retrievedPassword>
```

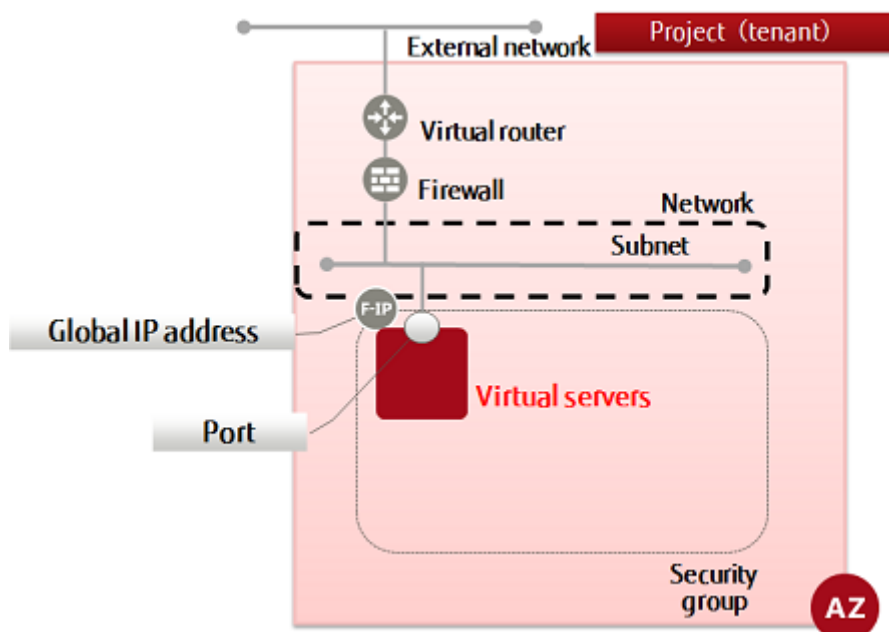
- Execute the following command:

```
$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt -inkey  
<pathToSpecifiedKeyFile>
```

3.6.8 Creating a virtual server (Windows and DHCP retrieval)

This section explains how to retrieve a port (private IP address) using DHCP and create a virtual server (Windows).

Refer to "[Creating a virtual server \(Windows and port specification\)](#) on page 84" for details on specifying a port (private IP address).



Follow the procedure below to create a virtual server and obtain a password. A password is required when logging in to the remote desktop.

- Set key pairs for password encryption and decryption, and create a virtual server
- Retrieve the encrypted password
- Use the key pair to decrypt and retrieve the password



It is possible for other users to reference the passwords when using a Windows virtual server that has been deployed with `admin_pass` specified in the metadata, so you should change the password after deployment.

1. Set key pairs for password encryption and decryption, and create a virtual server.

Set the environment variables below as follows:

```
$ VM_NAME=<virtualServerName> (any)
```

```
$ IMAGE_REF_ID=<windowsImgId>
```

```
$ FLAVOR_ID=<specOfVirtualServer> (listed flavor ID)
```

```
$ VOL_SIZE=80
```

```
$ DEVICE_NAME=<blockStoragePath> (specify using "/dev/vda" format)
```

```
$ SOURCE=<blockStorageType>
```

```
$ DESTINATION=<connectionDestinationVol> (specify volume)
```

```
$ ISDELETE=<volDeletion> (specify 1 to delete volumes created during creation of the virtual server, or 0 otherwise)
```

```
$ KEYNAME=<keyPairName>
```

```
$ INSTANCE_MAX=<maxNumOfServers>
```

```
$ INSTANCE_MIN=<minNumOfServers>
```

```
$ NETWORK_ID=<networkIdThatYouWantToConnectTo>
```

```
$ SG_NAME=<secGroupNameToBeSpecified>
```

2. Execute the following API:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"server": {"name": "'$VM_NAME'", "imageRef": "",  
"flavorRef": "'$FLAVOR_ID'", "block_device_mapping_v2":  
[ {"boot_index": "0", "uuid": "'$IMAGE_REF_ID'",  
"volume_size": "'$VOL_SIZE'", "source_type": "'$SOURCE'",  
"destination_type": "'$DESTINATION'",  
"delete_on_termination": '$ISDELETE' } ], "max_count": '$INSTANCE_MAX',  
"min_count": '$INSTANCE_MIN', "key_name": "'$KEYNAME'",  
"networks": [{"uuid": "'$NETWORK_ID'"}],  
"security_groups": [{"name": "'$SG_NAME'"}]} }' | jq .
```

The following response is output:

```
{  
  "server": {  
    "security_groups": [  
      {  
        "name": "<specifiedSecGroupName>"  
      }  
    ],  
  },  
}
```

```

    "OS-DCF:diskConfig": "MANUAL",
    "id": "<newVirtualServerId>",
    "links": [
      {
        "href": "http://10.3.0.201/v2/<projId>/servers/<newVirtualServerId>",
        "rel": "self"
      },
      {
        "href": "http://10.3.0.201/<projId>/servers/<newVirtualServerId>",
        "rel": "bookmark"
      }
    ]
  }
}

```

3. Execute the following API to check the virtual server that you created:

```

$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/detail -X GET \
-H "X-Auth-Token: $OS_AUTH_TOKEN" | \
jq '.servers[] | {status: .status, network_name: .addresses |
keys, key_name: .key_name, flavor_id: .flavor |
.id, vm_id: .id, security_group: .security_groups[] |
.name, name: .name, }'

```

If a list including the virtual server name that you specified is output, as follows, that means the virtual server was created successfully.

```

...
{
  "status": "<statusOfVirtualServer>",
  "network_name": [
    "<connectedNetworkName>"
  ],
  "key_name": "<keyPairName>",
  "flavor_id": "<flavorId>",
  "vm_id": "<virtualServerId>",
  "security_group": "<specifiedSecGroupName>",
  "name": "<virtualServerName>"
}
...

```

4. Retrieve the encrypted password.

After creating the virtual server, the interval before password retrieval can be performed is 10 minutes.

Set the environment variables below as follows:

```
$ SERVER_ID=<virtualServerId>
```

```
$ PROJECT_ID=<projId>
```

5. Execute the following API:

```

$ curl -sS $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/
os-server-password -X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .

```

The response will be like the following:

```

{
  "password": "~~~~~ <encryptedPassword> ~~~~=="
}

```

- Use the key pair to decrypt and retrieve the password.
Set the environment variable below as follows:

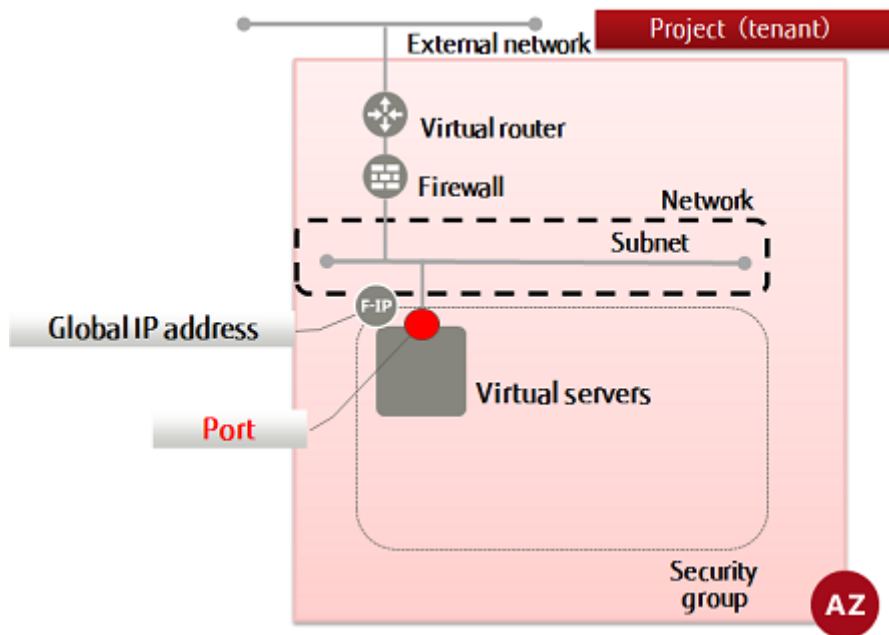
```
$ PASSWORD=<retrievedPassword>
```

- Execute the following command:

```
$ echo $PASSWORD | openssl base64 -d -A | openssl rsautl -decrypt ¥  
-inkey <pathToSpecifiedKeyFile>
```

3.6.9 Retrieving a virtual server port

This section explains how to retrieve the port ID of a virtual server, required for assigning global IP addresses to the virtual server.



- Set the environment variable below as follows:

```
$ SERVER_ID=<virtualServerIdToRetrievePortFrom>
```

- Execute the following API to retrieve the port:

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface ¥  
-X GET -H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

A response is output in a format such as the following:

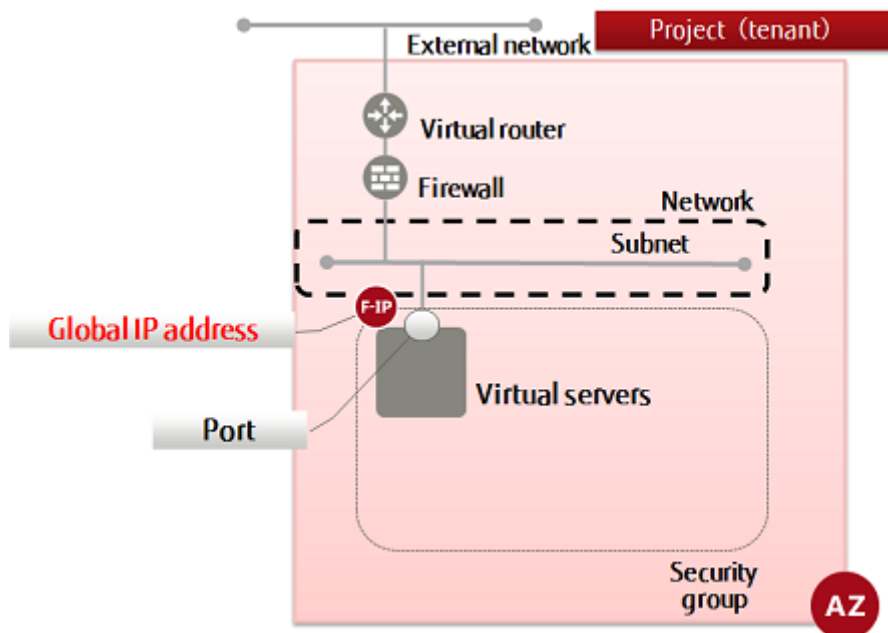
```
{  
  "interfaceAttachments": [  
    {  
      "port_state": "<status>",  
      "fixed_ips": [  
        {  
          "subnet_id": "<virtualServerId>",  
          "ip_address": "<privateIpAddr>"  
        }  
      ],  
      "port_id": "<portId>",  
      "net_id": "<networkIdThatVirtualServerIsConnectedTo>",  
      "mac_addr": "<macAddr>"  
    }  
  ]  
}
```

}

3.6.10 Retrieving a global IP address and assigning it to a virtual server

This section explains how to retrieve a global IP address for accessing virtual resources from the Internet and assign it to a virtual server.

A single call to the API can perform these two steps at the same time.



1. Set the environment variables below as follows:

```
$ NETWORK_ID=<networkId> (specify the external network ID of the availability zone where the virtual server is located)
```

```
$ VM_PORT_ID=<virtualServerId>
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/floatingips -X POST ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" ¥  
-d '{"floatingip":{"floating_network_id":"$NETWORK_ID",  
"port_id":"$VM_PORT_ID", "availability_zone":"$AZ"}}' | jq .
```

The following response is output:

```
{  
  "floatingip": {  
    "router_id": "<virtualRouterIdOfDefaultGateway>",  
    "status": "DOWN",  
    "tenant_id": "<projId>",  
    "floating_network_id": "<externalNetworkAddr>",  
    "fixed_ip_address": "<privateIpAddr>",  
    "floating_ip_address": "<globalIpAddr>",  
    "port_id": "<specifiedPortId>",  
    "id": "<globalIpAddrId>",  
    "availability_zone": "<availabilityZone>"  
  }  
}
```

For the availability zones, AZ1 is expressed as "jp-east-1a", and AZ2 is expressed as "jp-east-1b".

3. Execute the following API to check the global IP address that you created:

```
$ curl -sS $NETWORK/v2.0/floatingips.json -X GET \
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json" \
| jq .
```

If a list including the global IP address that you newly assigned is output, as follows, that means the address was created successfully.

```
{
  "floatingips": [
    ...
    {
      "router_id": "<virtualRouterIdOfDefaultGateway>",
      "status": "<status>",
      "tenant_id": "<projId>",
      "floating_network_id": "<externalNetworkAddr>",
      "fixed_ip_address": "<privateIpAddr>",
      "floating_ip_address": "<globalIpAddr>",
      "port_id": "<portId>",
      "id": "<globalIpAddrId>",
      "availability_zone": "<availabilityZone>"
    }
    ...
  ]
}
```


3.7 Confirming connection

3.7.1 Logging in to a virtual server (CentOS)

This section explains how to log in to a virtual server (CentOS).

Before logging in, check the following:

- The status of the global IP address assigned to the virtual server targeted for login must be "ACTIVE"
- The security group rules must be set correctly for the operating environment
- The firewall rules and policies must be set correctly for the operating environment
- A key file (.pem) must exist in the working directory

1. Execute the following command:

```
$ ssh -i <k5user>@<globalIpAddrAssignedToTargetVirtualServer>
```

```
The authenticity of host '<globalIpAddrAssignedToTargetVirtualServer>' can't be established.
```

```
RSA key fingerprint is: <fingerprint>.
```

```
Are you sure you want to continue connecting (yes/no)? <>
```

```
Warning: Permanently added '<globalIpAddrAssignedToTargetVirtualServer>' (RSA) to the list of known hosts.
```

2. Retrieve the root privileges.

In the CentOS provided by K5, a special command is required for retrieving the root privileges after login.

```
$ sudo su -
```

3.7.2 Logging in to a virtual server (Windows)

This section explains how to log in to a virtual server (Windows).

Before logging in, check the following:

- The status of the global IP address assigned to the virtual server targeted for login must be "ACTIVE"
- The security group rules must be set correctly for the operating environment
- The firewall rules and policies must be set correctly for the operating environment

Start the remote desktop from the client PC, and log in to connect.



Warning

After logging in to the virtual server, for security reasons it is necessary to configure the firewall on the virtual server.

After checking that the firewall has been configured, use the virtual server.

3.8 High-security configuration for SSL-VPN connection (V2 service)

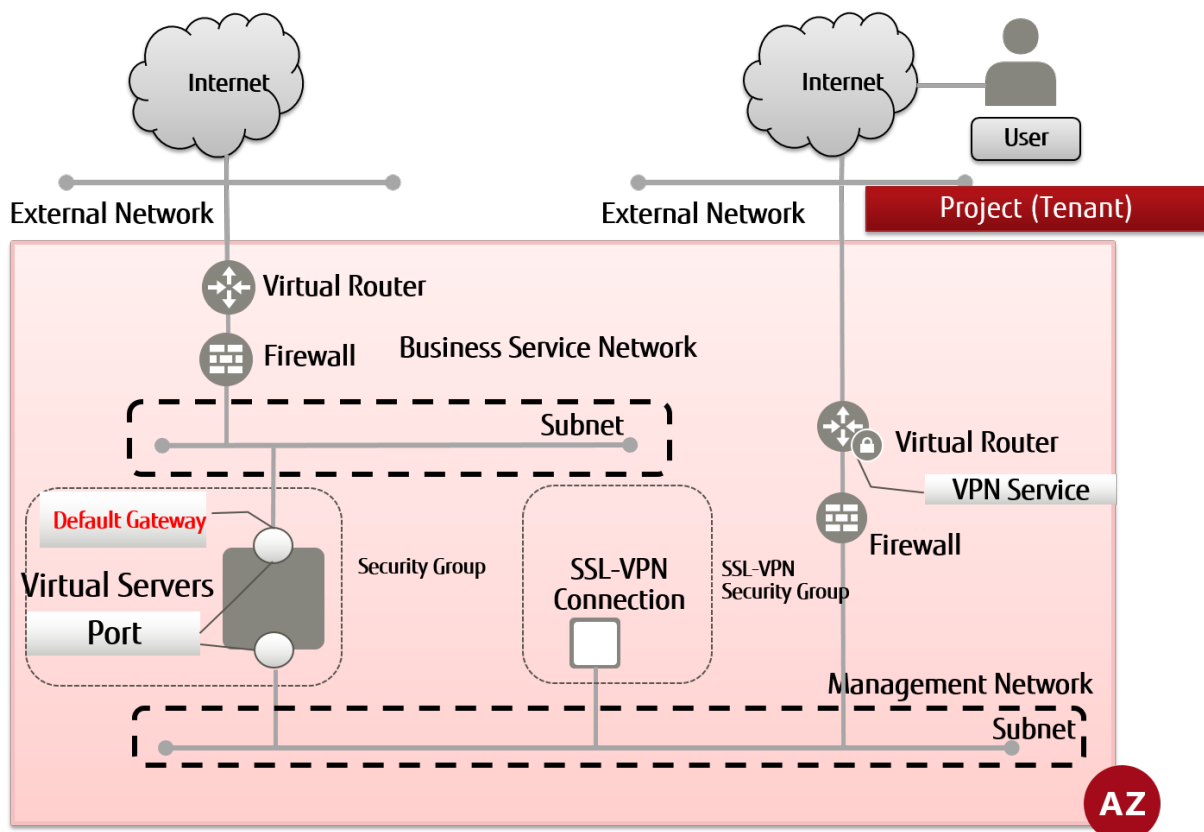
3.8.1 System structure diagram

This section explains an example of using the SSL-VPN connection function in a configuration (hereinafter a high-security configuration) with separate networks for business services and management.

Example structure for when using the SSL-VPN connection function in a high-security configuration

In this structure, it is possible to access the virtual servers from both the management network and the business service network.

- Access the management network using SSL-VPN.
- Use the business service network as the default gateway (default route).
- It is assumed Linux is used for the virtual server. In this case, it is necessary to create an interface file, and perform configuration specifying which of the two networks the OS will use as the default gateway (default route).

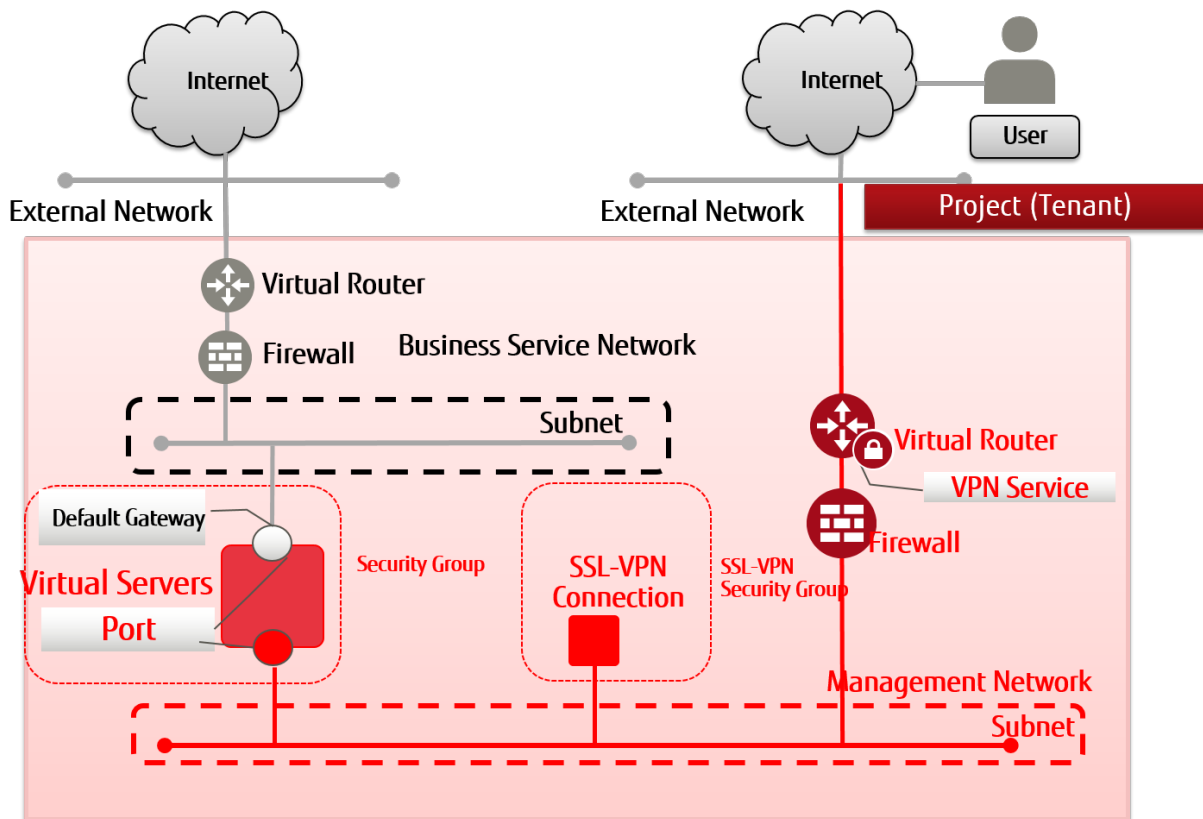


3.8.2 Building a network environment

This section explains the procedure for configuring a network environment with a management network and a business service network.

3.8.2.1 Building a management network

Use the following flow to build a network environment for management.

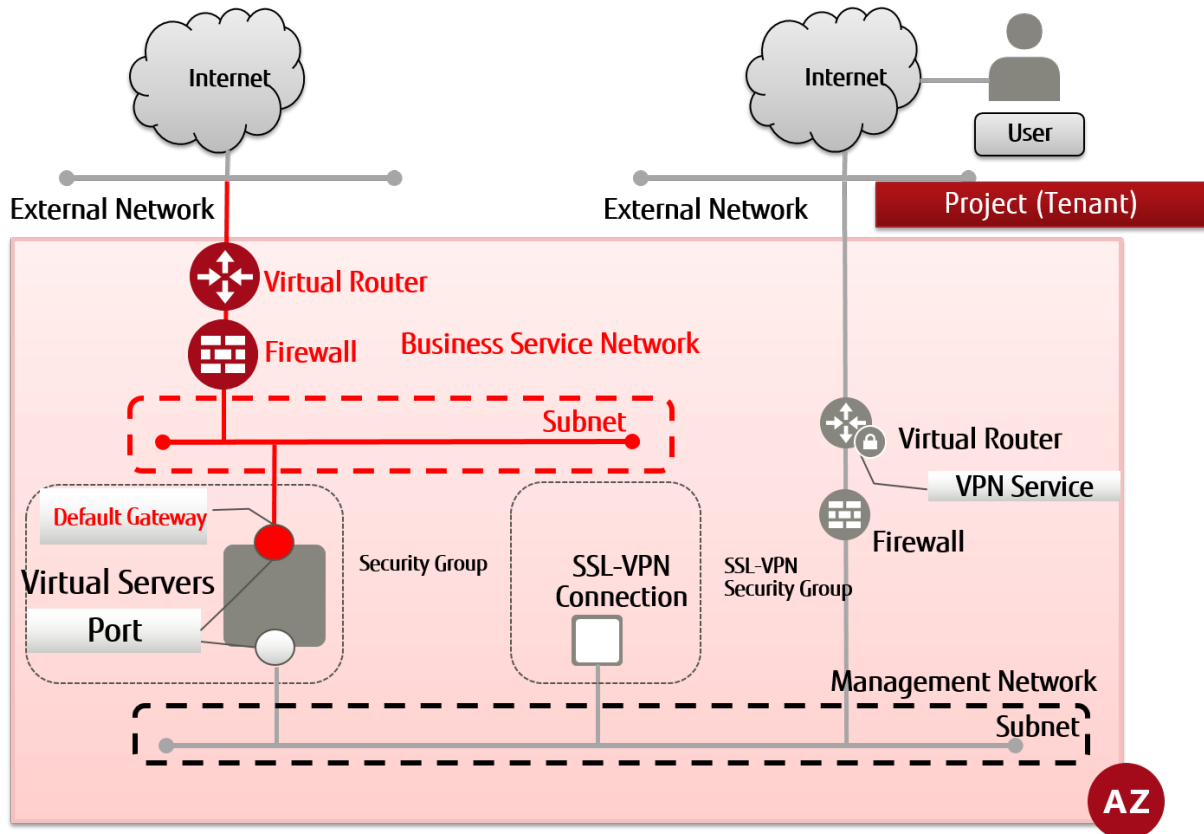


1. [Creating a network](#) on page 24
2. Create a subnet
 - [Creating a subnet](#) on page 26
 - [Adding routing](#) on page 28
3. [Creating a virtual router](#) on page 30
4. [Changing the information of a virtual router \(attaching to an external network\)](#) on page 31
5. [Changing the information of a virtual router \(attaching to a subnet\)](#) on page 32
6. [Creating a security group](#) on page 34
7. [Creating a security group rule](#) on page 35
8. Create firewall rules
 - [Creating a firewall rule \(IP address specification\)](#) on page 39
 - [Creating a firewall rule \(port number specification\)](#) on page 41
 - [Creating a firewall rule \(ICMP permission\)](#) on page 42
 - [Creating a firewall rule \(deny rule\)](#) on page 44
9. [Creating a firewall policy](#) on page 46
10. [Creating a firewall](#) on page 48
11. [SSL-VPN connection](#)
 - [SSL-VPN connection \(V2 service/K5 client certificate\)](#) on page 51
 - [SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 58
12. [Creating a key pair](#) on page 75
13. [Creating a port](#) on page 79 (Perform this when creating a virtual server on which ports are specified.)
14. Create a virtual server
 - [Creating a virtual server \(CentOS and port specification\)](#) on page 80

- [Creating a virtual server \(CentOS and DHCP retrieval\)](#) on page 82
- [Creating a virtual server \(Windows and port specification\)](#) on page 84
- [Creating a virtual server \(Windows and DHCP retrieval\)](#) on page 87

3.8.2.2 Building a business service network

Use the following flow to build a network environment for business services.



1. [Creating a network](#) on page 24
2. Create a subnet
 - [Creating a subnet](#) on page 26
3. [Creating a virtual router](#) on page 30
4. [Changing the information of a virtual router \(attaching to an external network\)](#) on page 31
5. [Changing the information of a virtual router \(attaching to a subnet\)](#) on page 32
6. [Creating a security group](#) on page 34
7. [Creating a security group rule](#) on page 35
8. Create firewall rules
 - [Creating a firewall rule \(IP address specification\)](#) on page 39
 - [Creating a firewall rule \(port number specification\)](#) on page 41
 - [Creating a firewall rule \(ICMP permission\)](#) on page 42
 - [Creating a firewall rule \(deny rule\)](#) on page 44
9. [Creating a firewall policy](#) on page 46
10. [Creating a firewall](#) on page 48
11. [Creating a port](#) on page 79
12. [Attaching ports](#) on page 97
13. [Configuring interface files \(Linux OS only\)](#) on page 98

3.8.2.2.1 Attaching ports

This section explains the procedure for attaching ports created in advance in the business service network to virtual servers. Refer to [Creating a port](#) on page 79 for the procedure for creating ports.

1. Perform the necessary settings indicated below.

```
$SERVER_ID=<ID of the virtual server to attach the port to>
```

```
$PORT_ID=<Port ID of the business service network to attach to the virtual server>
```

2. Execute the API.

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface -X POST ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" -H "Content-Type: application/json" ¥  
-d '{"interfaceAttachment": {"port_id": "'$PORT_ID'"}}' | jq .
```

The following response will be returned.

```
{  
  "interfaceAttachment": {  
    "mac_addr": "<MAC address of the interface for the business service network>",  
    "net_id": "<Network ID>",  
    "port_id": "<Network Port ID for the business service network to attach to the  
virtual server>",  
    "fixed_ips": [  
      {  
        "ip_address": "<IP address of the port of the business service network>",  
        "subnet_id": "<Subnet ID of the business service network>"  
      }  
    ],  
    "port_state": "DOWN"  
  }  
}
```

3. Execute the following API command to confirm that the interface of the virtual server and the port have been attached.

```
$ SERVER_ID=<ID of the virtual server the port was attached to>
```

```
$ curl -Ss $COMPUTE/v2/$PROJECT_ID/servers/$SERVER_ID/os-interface -X GET ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN" | jq .
```

If the following response is returned, attaching is complete.

```
{  
  "interfaceAttachments": [  
    {  
      "mac_addr": "<MAC address of the interface of the business service network>",  
      "net_id": "<Network ID>",  
      "port_id": "<Port ID of the business service network attached to the virtual  
server>",  
      "fixed_ips": [  
        {  
          "ip_address": "<IP address of the port of the business service network>",  
          "subnet_id": "<Subnet ID of the business service network>"  
        }  
      ],  
      "port_state": "ACTIVE"  
    },  
    {  
      "mac_addr": "<MAC address of the interface>",  
      "net_id": "<Network ID>",  
      "port_id": "<Port ID of the management network attached to the virtual server>",  
      "fixed_ips": [  
        {  
          "ip_address": "<IP address of the port of the management network>",  
          "subnet_id": "<Subnet ID of the management network>"  
        }  
      ],  
      "port_state": "ACTIVE"  
    }  
  ]  
}
```

```

        "ip_address": "<IP address of the port of the business service network>";",
        "subnet_id": "<Subnet ID of the management network>"
    }
  ],
  "port_state": "ACTIVE"
}
]
}

```

3.8.2.2.2 Configuring interface files (Linux OS only)

When adding ports to Linux OS virtual servers, perform configuration of interface files. This section explains the procedure.



This procedure uses the following environment.

- Note
- OS: CentOS 6.8
 - Default gateway of the virtual server: Port on the business service network side
 - eth0: Port of the management network
 - eth1: Port of the business service network

1. Copy the interface file (ifcfg-eth0) of the management network port, and create the interface file (ifcfg-eth1) for the business service network port attached to the virtual server. Also, describe in the interface file which port to use as the default gateway.

- /etc/sysconfig/network-scripts/ifcfg-eth0

Add "DEFROUTE=no".

```

DEVICE="eth0"
BOOTPROTO="dhcp"
IPV6INIT="yes"
MTU="1500"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
UUID="733285fa-1efe-4d9c-a70b-668922168a3f"
DEFROUTE=no

```

- /etc/sysconfig/network-scripts/ifcfg-eth1

Comment out the UUID and add "DEFROUTE=yes".

```

DEVICE="eth1"
BOOTPROTO="dhcp"
IPV6INIT="yes"
MTU="1500"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
#UUID="733285fa-1efe-4d9c-a70b-668922168a3f"
DEFROUTE=yes

```



Specify in DEFROUTE whether to use the interface as the default gateway.

Important

2. Perform reconfiguration of the network

Execute the following command and reconfigure the network settings of the OS.

```
/etc/init.d/network restart
```

Part 4: Deleting resources

Topics:

- [Deleting the basic system structure](#)
- [Deleting an SSL-VPN connection \(V2 service\)](#)
- [Deleting a firewall](#)
- [Deleting a virtual server](#)
- [Deleting a security group](#)
- [Deleting a network environment](#)

4.1 Deleting the basic system structure

4.1.1 The order of deletion

This section explains the appropriate order of deletion of the basic system structure.



Note

To avoid any potential security issues, first delete any sections of the structure that connect to external networks.

The basic procedure is the reverse of the procedure for creation.

4.2 Deleting an SSL-VPN connection (V2 service)

4.2.1 Deleting an SSL-VPN connection (V2 service)

This section explains the procedure for deleting a (V2 service) SSL-VPN connection.



Note

When an SSL-VPN connection is deleted, the following settings that were automatically configured during the creation of that SSL-VPN connection are also deleted.

1. Firewall rules

The firewall rules created for use during dedicated SSL-VPN connections are deleted.

2. Static routing information

The static routing information configured for use during dedicated SSL-VPN connections is deleted.

3. Security groups

The security group created for use during dedicated SSL-VPN connections is deleted.

4. Global IP address

The global IP address allocated to the SSL-VPN connection is de-allocated.

The basic procedure for deletion is the reverse of the procedure for creation.

1. Configure the following.

```
$ TMP_SSLCONNECT_ID=<SSL-VPN Connection ID>
```

2. Execute the following API.

```
$ curl -X DELETE -sS -i $NETWORK/v2.0/vpn/ssl-vpn-v2-connections/¥  
$TMP_SSLCONNECT_ID -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥  
-H "Content-Type: application/json"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100  
Content-Type: text/html; charset=UTF-8  
Content-Length: 0
```

4.2.2 Deleting a VPN service (V2 service)

This section explains the procedure for deleting a VPN service.

The basic procedure for deletion is the reverse of the procedure for creation.

1. Configure the following.

```
$ TMP_VPN_ID=<VPN Service ID>
```

2. Execute the following API.

```
$ curl -X DELETE -sS -i $NETWORK/v2.0/vpn/vpnservices/$TMP_VPN_ID ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<ID>  
Cache-Control: no-cache  
X-Request-Id: <ID>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100  
Content-Type: text/html; charset=UTF-8  
Content-Length: 0
```

4.2.3 Deleting a container for SSL-VPN keys (V2 service)

This section explains the procedure for deleting a key container for SSL-VPN.

Perform this procedure only if you previously referred to [SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 58 and created an SSL-VPN connection.

The basic procedure for deletion is the reverse of the procedure for creation.

1. Configure the following.

```
$ TMP_CON_NAME=<Key Container ID>
```

2. Execute the following API.

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/containers/¥  
$TMP_CON_NAME -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

4.2.4 Deleting certificates for SSL-VPN (V2 service)

This section explains the procedure for deleting certificates for SSL-VPN.

Perform this procedure only if you previously referred to [SSL-VPN connection \(V2 service/self-signed certificate\)](#) on page 58 and created an SSL-VPN connection.

The basic procedure for deletion is the reverse of the procedure for creation.



Warning

During deletion of certificates, it is necessary to individually delete the CA certificates, private keys for server certificates, server certificates, and DH keys.

1. Configure the following.

```
$ TMP_CRTKEY_NAME=<DH Key ID>
```

2. Execute the following API.

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/¥
```

```
$TMP_CRTKEY_NAME -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

3. Configure the following.

```
$ TMP_CRTKEY_NAME=<ID of Private Key for Server Certificate>
```

4. Execute the following API.

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/¥  
$TMP_CRTKEY_NAME -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

5. Configure the following.

```
$ TMP_CRTKEY_NAME=<Server Certificate ID>
```

6. Execute the following API.

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/¥  
$TMP_CRTKEY_NAME -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

7. Configure the following.

```
$ TMP_CRTKEY_NAME=<CA Certificate ID>
```

8. Execute the following API.

```
$ curl -X DELETE -sS -i $KEYMANAGEMENT/v1/$PROJECT_ID/secrets/¥  
$TMP_CRTKEY_NAME -H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
X-Fcx-Endpoint-Request: EXECUTED_REQ<ID>  
Date: Www, DD MMM yyyy hh:mm:ss GMT
```

4.3 Deleting a firewall

4.3.1 Deleting a firewall

This section explains how to delete a firewall.

Deletion of a firewall is basically the procedure for creating a firewall done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_FW_ID=<firewallIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/fw/firewalls/$TMP_FW_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runtime>
```

4.3.2 Deleting a firewall policy

This section explains how to delete a firewall policy.

Deletion of a firewall policy is basically the procedure for creating a firewall policy done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_FW_ID=<firewallPolicyIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/fw/firewall_policies/$TMP_FWP_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runtime>
```

4.3.3 Deleting a firewall rule

This section explains how to delete a firewall rule.

Deletion of a firewall rule is basically the procedure for creating a firewall rule done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_FW_ID=<firewallIRuleIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -X DELETE -i -s $NETWORK/v2.0/fw/firewall_rules/$TMP_FWR_ID \
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<reqId>
Cache-Control: no-cache
X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

4.4 Deleting a virtual server

4.4.1 Deleting a global IP address

This section explains how to delete a global IP address.

Deletion of a global IP address is basically the procedure for creating a global IP address done in reverse.



Note

Even if you delete a virtual server for which a global IP address has been specified, the global IP address will not be deleted automatically.

1. Set the environment variables below as follows:

```
$ TMP_FLOATINGIP_ID=<global IpAddr IdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/floatingips/$TMP_FLOATINGIP_ID -X DELETE ¥  
-H "X-Auth-Token:$OS_AUTH_TOKEN" -H "Content-Type:application/json"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runTime>  
Keep-Alive: timeout=5, max=100
```

4.4.2 Deleting a virtual server

This section explains how to delete a virtual server.

Deletion of a virtual server is basically the procedure for creating a virtual server done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_VM_ID=<virtualServerIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $COMPUTE/v2/$PROJECT_ID/servers/$TMP_VM_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runtime>  
Keep-Alive: timeout=5, max=100
```

4.4.3 Deleting a port

This section explains how to delete a port.

Deletion of a port is basically the procedure for creating a port done in reverse.

However, when a virtual server that the port is attached to is deleted, the port will be deleted at the same time.

1. Set the environment variable below as follows:

```
$ TMP_PORT_ID=<portIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/ports/$TMP_PORT_ID -X DELETE \
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<reqId>
Cache-Control: no-cache
X-Request-Id: <xReqId>
X-Runtime: <runtime>
Keep-Alive: timeout=5, max=100
```

4.4.4 Deleting a key pair

This section explains how to delete a key pair.

Deletion of a key pair is basically the procedure for creating a key pair done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_KEYPAIR_NAME=<keyPairIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -X DELETE -Ss -i $COMPUTE/v2/$PROJECT_ID/os-keypairs/$
TMP_KEYPAIR_NAME -H "X-Auth-Token:$OS_AUTH_TOKEN" \
-H "Content-Type:application/json"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<reqId>
Cache-Control: no-cache
X-Request-Id: <xReqId>
X-Runtime: <runtime>
Keep-Alive: timeout=5, max=100
Content-Type: text/html; charset=UTF-8
Content-Length: 0
```

4.5 Deleting a security group

4.5.1 Deleting a security group rule

This section explains how to delete a security group rule.



Tip

Deleting a security group does not delete its security group rules. Follow the procedure below to delete a security group rule.

1. Set the environment variable below as follows:

```
$ TMP_SGR_ID=<secGroupRuleIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/security-group-rules/$TMP_SGR_ID -X DELETE \
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<reqId>
Cache-Control: no-cache
X-Request-Id: <xReqId>
X-Runtime: <runtime>
```

4.5.2 Deleting a security group

This section explains how to delete a security group.

Deletion of a security group is basically the procedure for creating a security group done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_SGR_ID=<secGroupIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/security-groups/$TMP_SG_ID -X DELETE \
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content
Server: Apache
Date: Www, DD MMM yyyy hh:mm:ss GMT
x-compute-request-id: req-<reqId>
Cache-Control: no-cache
X-Request-Id: <xReqId>
X-Runtime: <runtime>
```


4.6 Deleting a network environment

4.6.1 Detaching a virtual router from a subnet

This section explains how to detach a virtual router from a subnet.

Detachment of a subnet is basically the procedure for creating a subnet done in reverse.

1. Set the environment variables below as follows:

```
$ TMP_ROUTER_ID=<virtualRouterIdToBeDetached>
```

```
$ TMP_SUBNET_ID=<subnetIdAttachedTo>
```

2. Execute the following API:

```
$ curl -Ss $NETWORK/v2.0/routers/$TMP_ROUTER_ID/remove_router_interface ¥  
-X PUT -H "X-Auth-Token: $OS_AUTH_TOKEN" ¥  
-H "Content-Type: application/json" ¥  
-d '{"subnet_id": "'$TMP_SUBNET_ID'" }' | jq .
```

```
{  
  "subnet_id": "<subnetIdTargetedForDetachment>",  
  "tenant_id": "<projId>",  
  "port_id": "<connectedPortId>",  
  "id": "<virtualRouterIdTargetedForDetachment>",  
  "availability_zone": "<availabilityZone>"  
}
```

4.6.2 Deleting a virtual router

This section explains how to delete a virtual router.

Deletion of a virtual router is basically the procedure for creating a virtual router done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_ROUTER_ID=<virtualRouterIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -Ss $NETWORK/v2.0/routers/$TMP_ROUTER_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-id  
Cache-Control: no-cache  
X-Request-Id: xReqId  
X-Runtime: <runtime>
```

4.6.3 Deleting a subnet

This section explains the how to deleted a subnet.

Deletion of a subnet is basically the procedure for creating a subnet done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_SUBNET_ID=<subnetIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -sS $NETWORK/v2.0/subnets/$TMP_SUBNET_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runtime>
```

4.6.4 Deleting a network

This section explains how to delete a network.

Deletion of a network is basically the procedure for creating a network done in reverse.

1. Set the environment variable below as follows:

```
$ TMP_NETWORK_ID=<networkIdTargetedForDeletion>
```

2. Execute the following API:

```
$ curl -i -sS $NETWORK/v2.0/networks/$TMP_NETWORK_ID -X DELETE ¥  
-H "X-Auth-Token: $OS_AUTH_TOKEN"
```

```
HTTP/1.1 204 No Content  
Server: Apache  
Date: Www, DD MMM yyyy hh:mm:ss GMT  
x-compute-request-id: req-<reqId>  
Cache-Control: no-cache  
X-Request-Id: <xReqId>  
X-Runtime: <runtime>
```

Appendix

A.1 Specification format for payloads during certificate registration

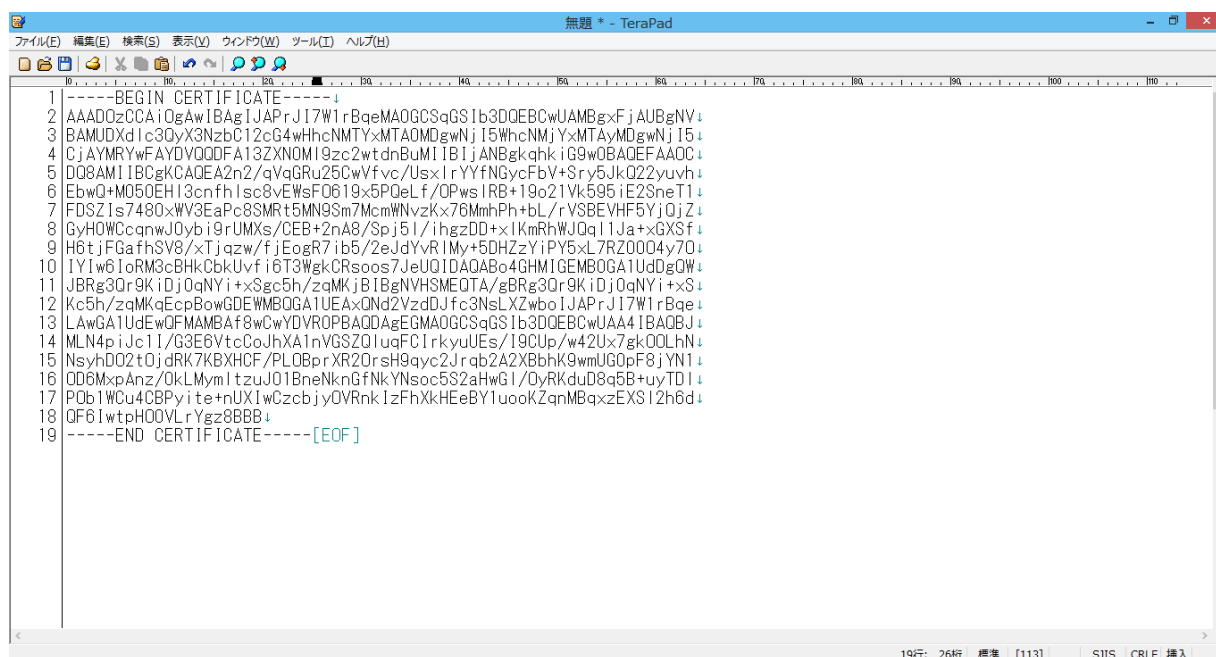
This section explains the replacement of line feed codes with LF's (\n) in the payloads of certificates. This replacement must be performed during "2.1.17 Registering SSL-VPN certificates".

When specifying a payload, it is necessary to change the line feed codes used in the relevant certificate to LF's (\n).

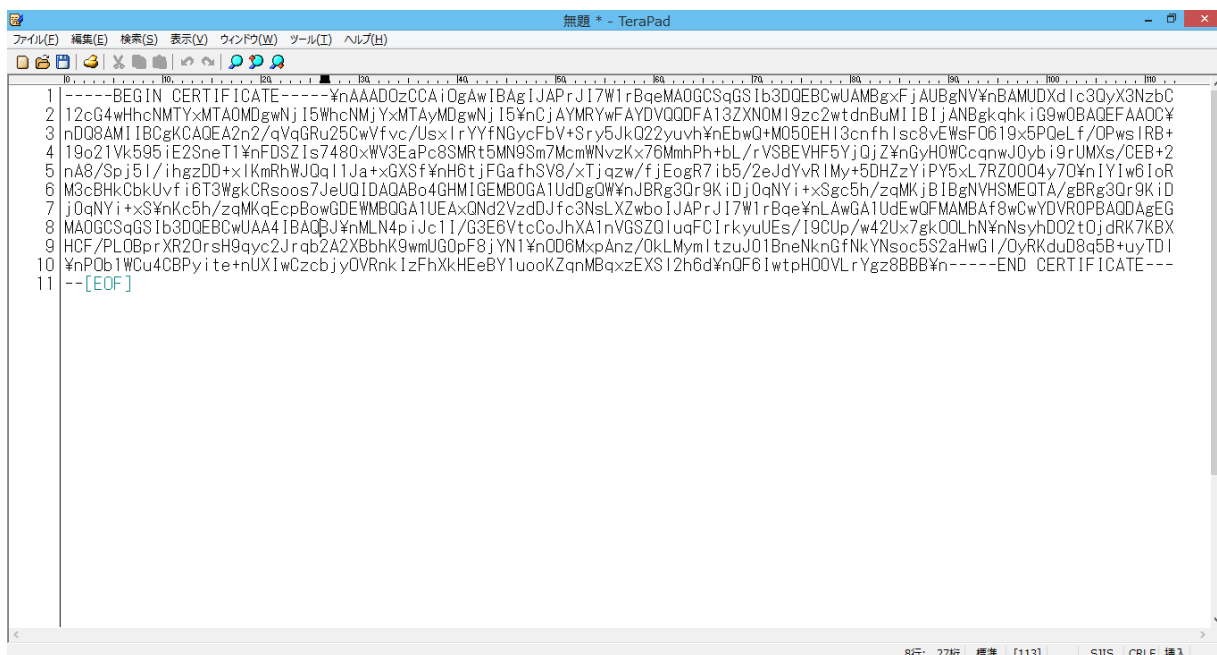
An example of the payload of a CA certificate before and after changing is shown below.

Specify the payloads of each other certificate using the same format.

[Before changing] Line feed codes are included.



[After changing] Line feed codes have been replaced with LF's (\n), and the text is displayed as a single line.



A.2 Converting certificate formats

Function used to perform conversion of certificate formats

When a client certificate is issued from the K5 portal, it is downloaded in the PKCS#12 format. Depending on the application that uses a client certificate, it may be necessary to convert the certificate to the PEM format.

Converting certificate formats

An example of executing a command to convert a PKCS#12 format certificate to a PEM format certificate is given below.

The example in this chapter uses Openssl.



Note

When operating a PKCS#12 format certificate, entry of a passphrase is necessary. For the <passphraseForPKCS#12> below, enter the passphrase for the certificate that was configured on the K5 portal.

- Converting client certificate formats

```
$ openssl pkcs12 -in <Client Certificate Name>.p12 -clcerts -nokeys -out <Client Certificate Name (User Defined)>
```

```
Enter Import Password:<passphraseForPKCS#12>
MAC verified OK
```

- Extracting the private key

There are two methods for extracting private keys, "Encrypted private keys" and "Unencrypted private keys".

Some applications may not be able to use "Encrypted private keys". Only use encrypted private keys after confirming that applications are able to use them.

- Encrypted private keys

```
$ openssl pkcs12 -in <Client Certificate>.p12 -nocerts -out <Name of the Private Key for the Client Certificate (User Defined)>
```

```
Enter Import Password:<passphraseForPKCS#12>
```

```
MAC verified OK
```

```
Enter PEM pass phrase:<passphraseForPrivateKeyEncryption>
```

```
Verifying - Enter PEM pass phrase:<passphraseForPrivateKeyEncryption>
```



Note

To create encrypted private keys, in addition to the passphrase to load the PKCS#12 format certificate, it is also necessary to enter a user defined passphrase for encryption.

As this passphrase must be entered when the application actually loads the private key it helps increase the level of security.

- Unencrypted private keys

```
$ openssl pkcs12 -in <Client Certificate>.p12 -nocerts -nodes -out <Name of the Private Key for the Client Certificate (User Defined)>
```

```
Enter Import Password:<passphraseForPKCS#12>
```

```
MAC verified OK
```

Confirming consistency of client certificates and private keys

It is necessary to use corresponding client certificates and private keys. If the client certificate and private key do not correspond, an error will occur.

The following is example of execution of the command to confirm that a client certificate and private key correspond.

1. Display the information of a PEM format client certificate

```
openssl x509 -modulus -noout -in <Client Certificate>
```

2. Display the information of a private key

```
openssl rsa -modulus -noout -in <Name of the Private Key for the Client Certificate>
```

When the information displayed for the above "1." and "2." matches the client certificate and private key correspond.

A.3 Configuring the API access environment (Windows)

It is necessary to execute cURL commands when using APIs to access the K5 IaaS. However, in Windows, cURL is not provided as a built-in OS function. This section explains an example procedure in which Cygwin is used to construct an execution environment to enable the use of cURL commands in a Windows environment. Although numerous cases of use of Cygwin have been reported, operations using Cygwin are not guaranteed. Perform these operations at your own risk.

1. Install Cygwin

Download the installer

Download the Cygwin installer from the following URL. Download the installer that is compatible with the Windows environment that you are using.

<https://cygwin.com/install.html>

When using a 64-bit OS, setup-x86_64.exe (Ver2.882)

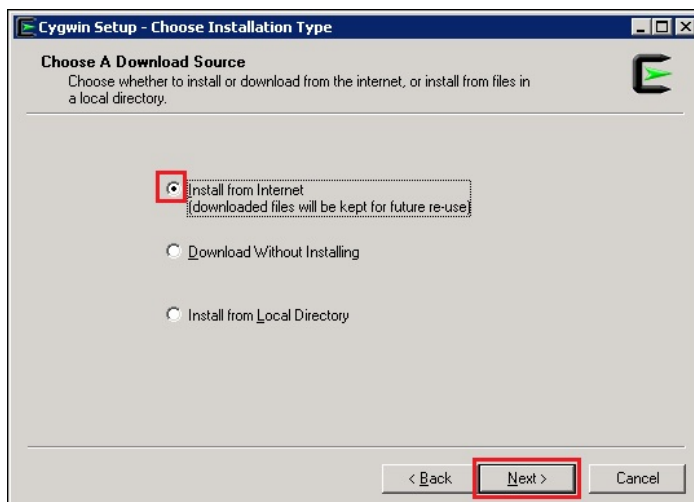
When using a 32-bit OS, setup-x86.exe (Ver2.882)



2. Download and install packages

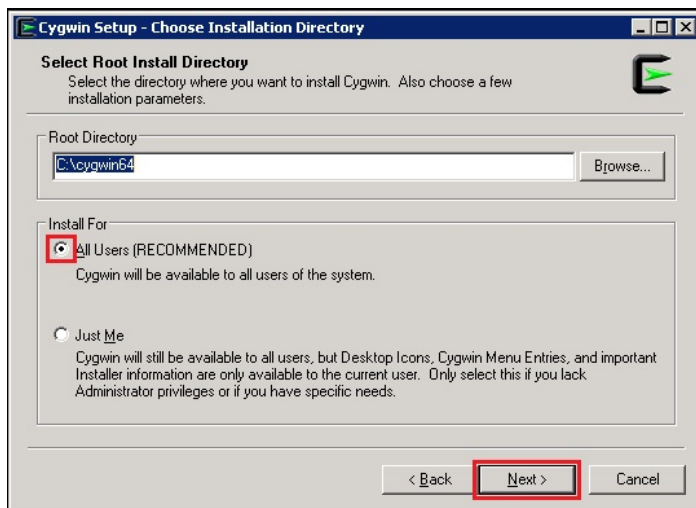
Execute the downloaded installer.

Execute installation over the Internet. "Install from Internet" is selected by default. Leave this selected, and click [Next].



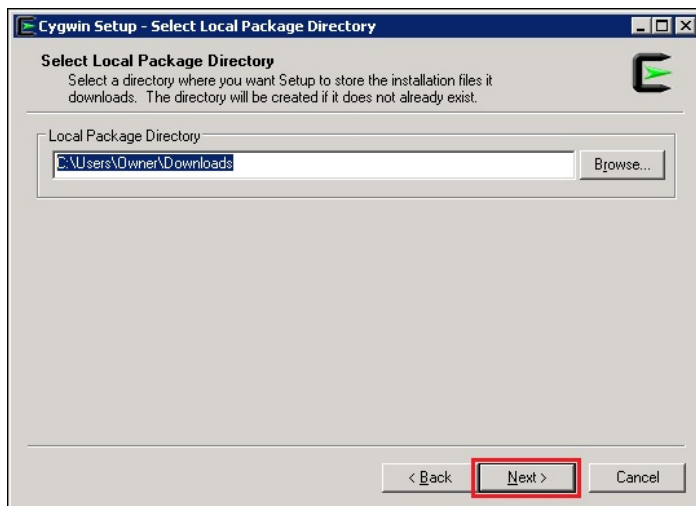
3. Specify the root directory and users of Cygwin

If you have no particular root directory in mind, leave the default value unchanged. For the users who can use Cygwin, select "All Users (RECOMMENDED)", and then click [Next].



4. Specify the storage directory for packages to install

If you have no particular directory in mind, leave the default value unchanged. Click [Next].



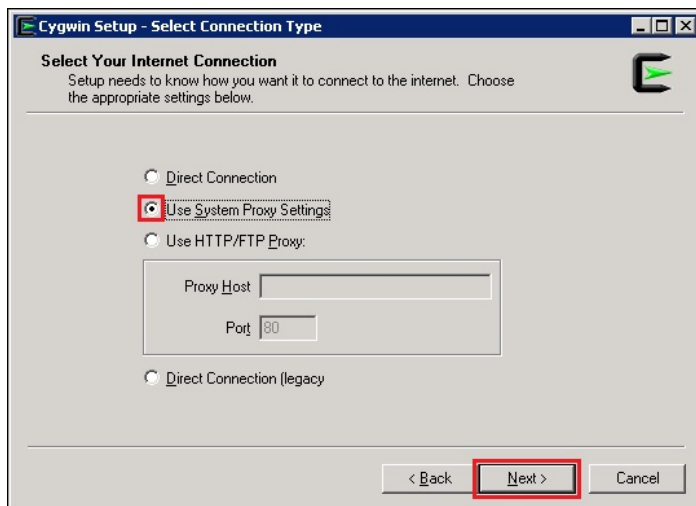
5. Specify the Internet connection method

When performing installation over the Internet in an environment in which it is necessary to specify a proxy in order to connect to an external network, select either "Use System Proxy Settings" or "Use HTTP/FTP Proxy". In this example procedure, "Use System Proxy Settings" is selected. After selecting the desired method, click [Next].

- Direct Connection (Connect without specifying a proxy)
- Use System Proxy Settings (Use the proxy settings specified in the Web browser)

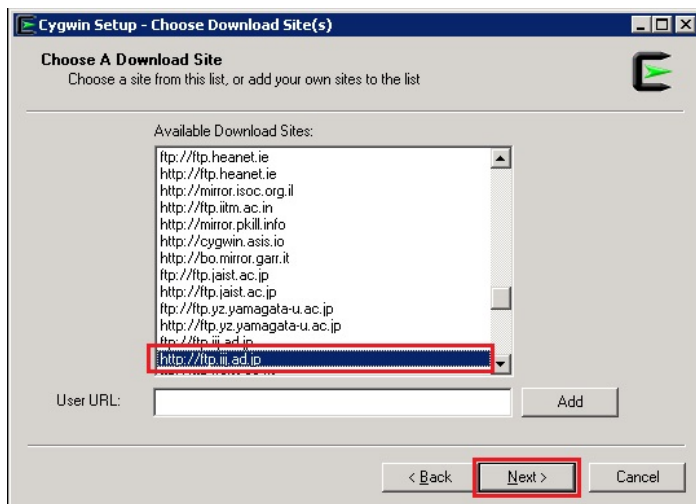
The settings in "Internet Options" > [Connections] tab > [LAN settings] > "Proxy server" are used

- Use HTTP/FTP Proxy (Specify a proxy directly)



6. Choose a download site

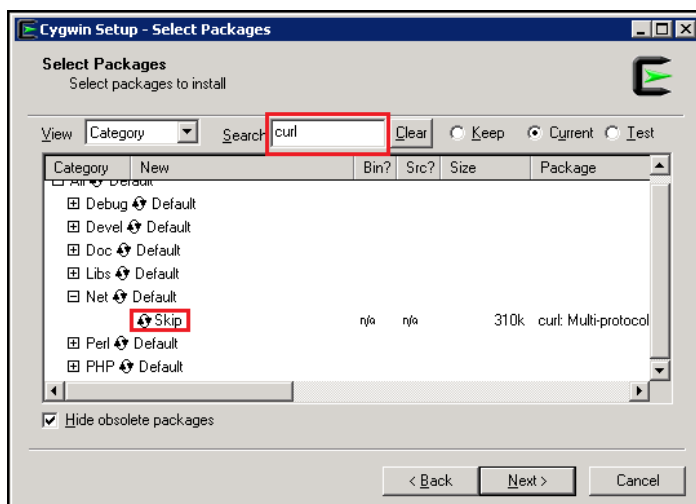
A list of download sites is displayed. Select a site, and then click [Next].

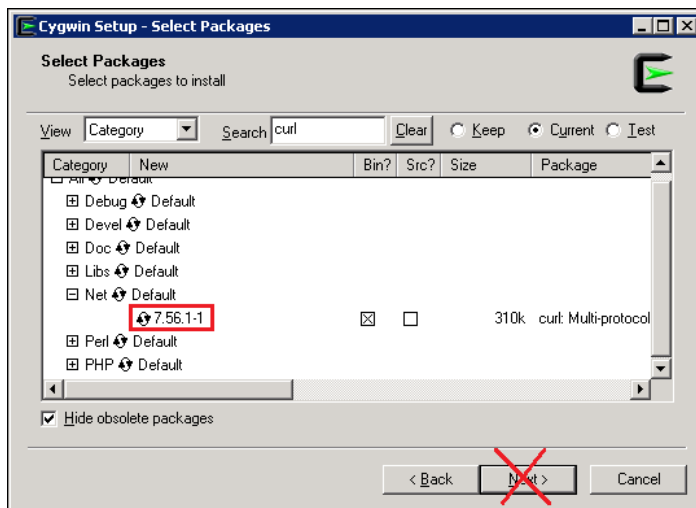


7. Enter "curl" in the search field for packages to install

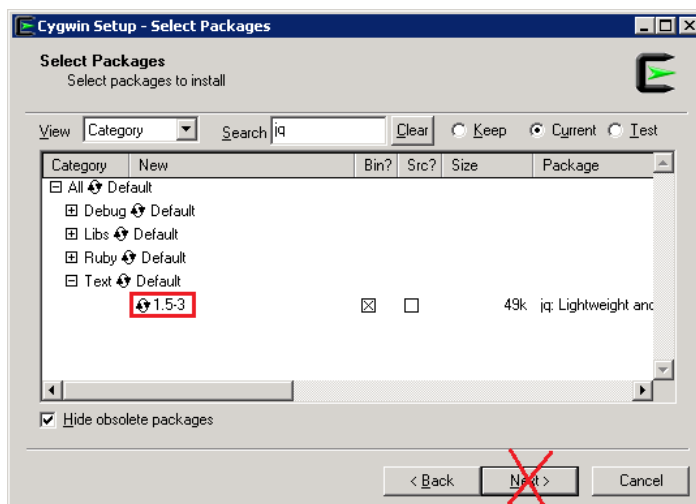
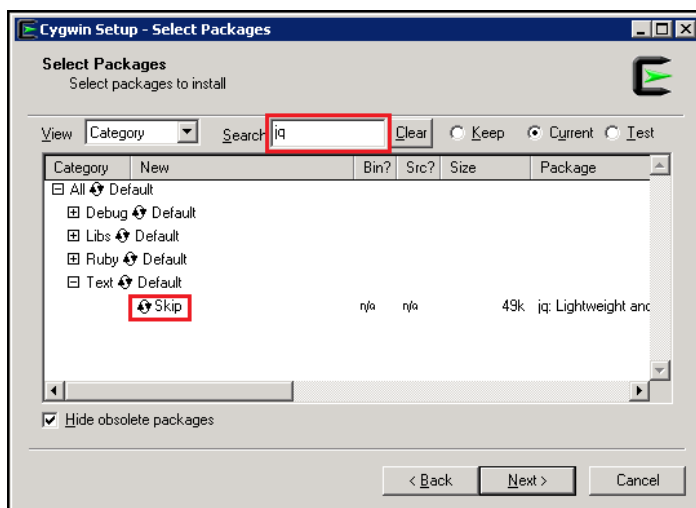
For "Net" - "curl", select version 7.49 or later. In the screenshot below, 7.56.1-1 is selected.

Clicking "Skip" displays the version number that will be installed.



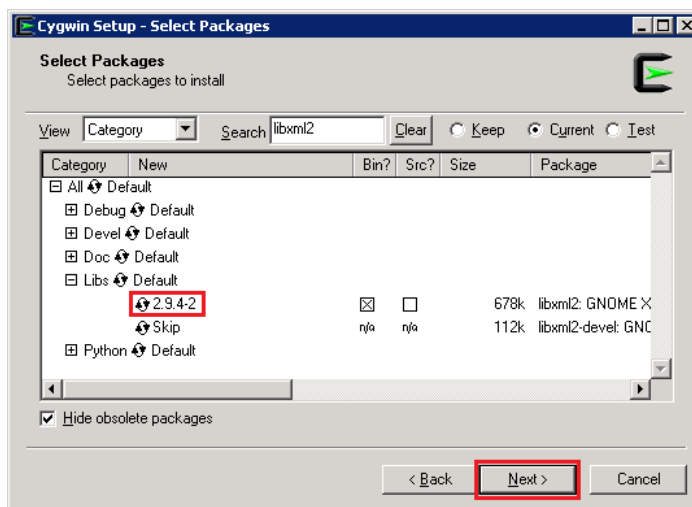
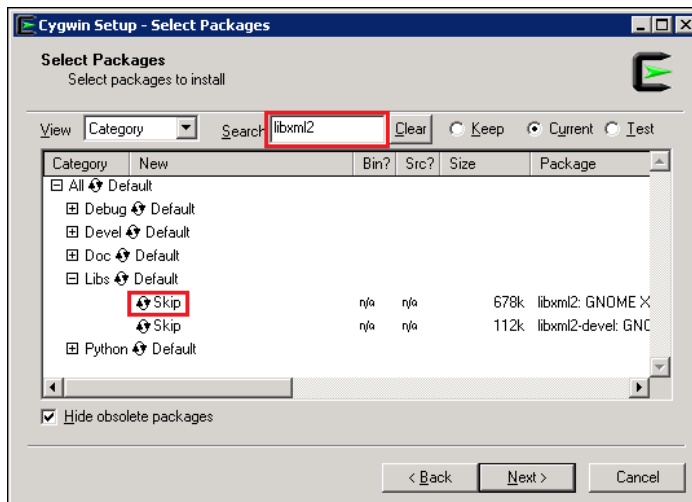


8. Enter "jq" in the search field for packages to install
 For "Text" - "jq", select version 1.5-1 or later. In the screenshot below, 1.5-3 is selected.
 Clicking "Skip" displays the version number that will be installed.

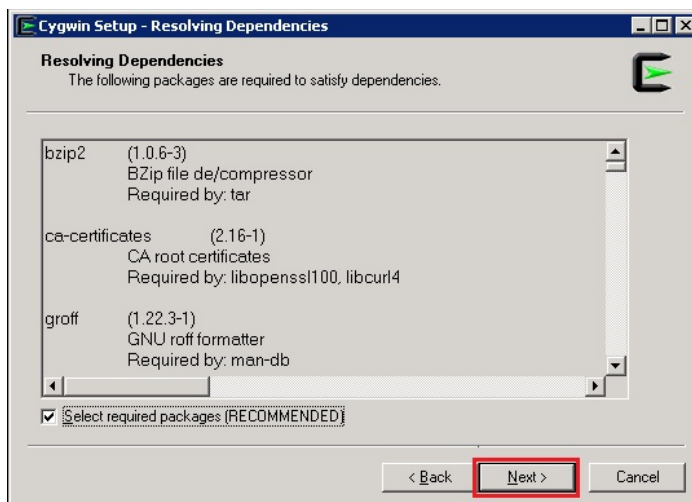


9. Enter "libxml2" in the search field for packages to install
 For "Libs" - "libxml2", select version 2.9.3-1 or later. In the screenshot below, 2.9.4-2 is selected.
 Install the "libxml2:GNOME XML library(runtime)" package.

Clicking "Skip" displays the version number that will be installed. Click [Next].

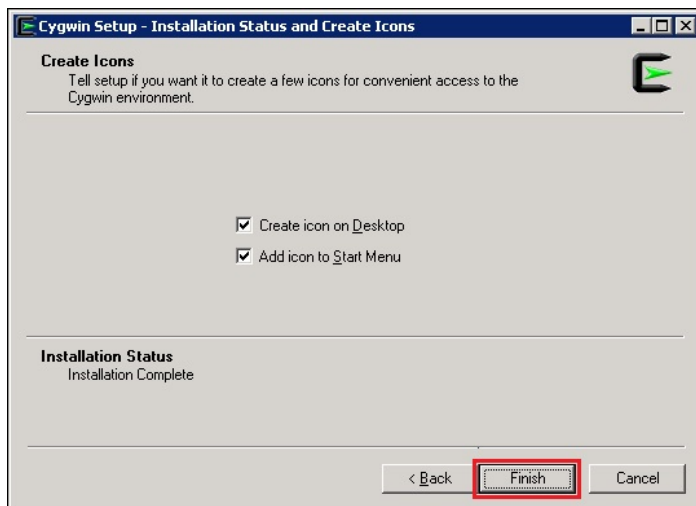


10.The screen for confirming dependencies is displayed. Click [Next].



11.Click "Finish" to close the installer.

Confirm that "Create icon on Desktop" is checked. A [Cygwin64 Terminal] icon is created on the desktop.

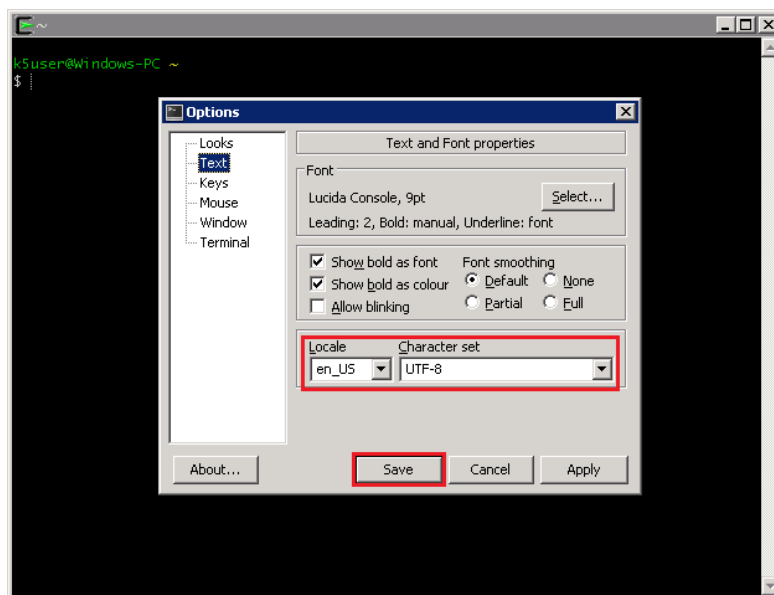


12. Set the language of the Cygwin Terminal

Double click the [Cygwin64 Terminal] icon that was created on the desktop to launch the terminal.

Right-click the title bar of the terminal, and select "Options".

In "Text", set "Locale" to [en_US] or [ja_JP], and set "Character set" to [UTF-8].



13. Confirm that curl, jq, and xmllint have been installed

Launch "Cygwin64 Terminal", and then execute the following commands.

```
$ curl --version
```

```
$ jq --version
```

```
$ xmllint --version
```

* If any of the above commands cannot be found, run the installer again, and reinstall the relevant package.

```
k5user@Windows-PC ~
$ curl --version
curl 7.56.1 (x86_64-unknown-cygwin) libcurl/7.56.1 OpenSSL/1.0.2m zlib/1.2.11 libidn2/
2.0.4 libpsl/0.18.0 (+libidn2/2.0.2) libssh2/1.7.0 nghttp2/1.23.1
Release-Date: 2017-10-23
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtsp
scp sftp smb smbs smtp smtps telnet tftp
Features: AsynchDNS Debug IDN IPv6 Largefile GSS-API Kerberos SPNEGO NTLM NTLM_WB SSL
libz TLS-SRP HTTP2 UnixSockets HTTPS-proxy Metalink PSL

k5user@Windows-PC ~
$ jq --version
jq-1.5

k5user@Windows-PC ~
$ xmlint --version
xmlint: using libxml version 20904
compiled with: Threads Tree Output Push Reader Patterns Writer SAXv1 FTP HTTP DTDVa
lid HTML Legacy C14N Catalog XPath XPointer XInclude Iconv ISO8859X Unicode Regexp
Autotomata Expr Schemas Schematron Modules Debug Zlib Lzma

k5user@Windows-PC ~
$ |
```

14. Configure the proxy settings for curl

If it is necessary to issue curl commands through a proxy, it is necessary to include the information of that proxy in the initial configuration file for curl.

Create the file ".curlrc" in the directory "C:\cygwin64\home\<userName>" using a text editor. Be sure to use LF (UNIX line feed) for the line feed code in this file.

Enter the following two lines of text in ".curlrc".

```
proxy="<proxyServerName>:<portNumber>"
```

```
proxy-user="<userName>:<password>"
```

FUJITSU Cloud Service K5
IaaS API User Guide 1.18.2 version

Published Date June 2018
All Rights Reserved, Copyright FUJITSU LIMITED 2015-2018

- The content of this document may be subject to change without prior notice.
- This document may not be reproduced without the written permission of Fujitsu Limited.