Using the Integration Server REST API in SVM management scenarios

Kaspersky Security for Virtualization 6.3 Light Agent

Contents

Using the Integration Server REST API

You can use the Integration Server REST API in SVM management scenarios to perform the following procedures:

- Configuring the connection between the Integration Server and the virtual infrastructure.
- Adding a virtual infrastructure certificate to the list of trusted certificates of the Integration Server.
- Removing settings for connecting to the virtual infrastructure from the Integration Server.
- Deploying SVMs in a virtual infrastructure, installing Kaspersky Security Center Network Agent on SVMs, and performing pre-deployment steps:
    - Registering SVM images on the Integration Server.
    - Registering the Network Agent package on the Integration Server.
    - Getting a list of virtual infrastructure objects required for SVM deployment.
- Reconfiguring deployed SVMs.
- SVM removal.

Interaction with the Integration Server REST API is based on requests and responses in JSON format and is carried out over the HTTPS protocol. To interact with the Integration Server REST API, authentication on the Integration Server using a bearer token is required.

For processing requests that are time consuming and run asynchronously, tasks are used. The task is created as an intermediate query result.

Authentication on the Integration Server

A server-signed token (bearer token) is used to authenticate on the Integration Server. In each REST API request, you need to specify the following header with the token:

```
Authorization: Bearer <accessToken>
```

To get the token, execute the following request:

```
GET /api/3.0/auth/tokens
```

In the header of this request, specify the user name "admin" and the password of the Integration Server administrator as follows:

```
Authorization: Basic <Base64-encoded username:password string>
```

If the request succeeds, a 200 code and a response body are returned:

```
{
  "accessToken": {
    "value": "<accessToken>"
  },
  "refreshToken": {
    "value": "<refresh_token>"
  },
  "session": {
    "id": "<session_id>",
    "createdAt": "<timestamp>",
    "updatedAt": "<timestamp>",
    "activeTo": "<timestamp>",
    "validTo": "<timestamp>"
  }
}
```

The token, which must be sent in the header of each request, is contained in the "value" field of the "accessToken" element.

Adding a virtual infrastructure certificate to the Integration Server

When connecting to the virtual infrastructure, the Integration Server verifies the authenticity of the virtual infrastructure objects it is connecting to. Depending on the type of virtual infrastructure, the Integration Server connects to a hypervisor, virtual infrastructure management server, or Keystone microservice.

For the Integration Server to successfully connect to the virtual infrastructure, you need to add the certificate received from the virtual infrastructure to the list of trusted certificates of the Integration Server.

> Authenticity is not verified for a Microsoft Windows Server (Hyper-V) hypervisor.

Obtaining a virtual infrastructure certificate

Depending on the type of virtual infrastructure, various requests are used to obtain a certificate.

- In infrastructures that you connect to using HTTPS, execute the following request to obtain a certificate:

  ```
  GET api/3.0/sslConfig/getCertificate?address=<address>:<port>
  ```

  where:

  - <address> is the address of the virtual infrastructure object to which the Integration Server will connect to interact with the virtual infrastructure. Depending on the type of virtual infrastructure, select a hypervisor, virtual infrastructure administration server, or Keystone microservice.
  - <port> is the port for connecting to the virtual infrastructure object.

  The following ports are commonly used depending on the type of infrastructure:

  - XenServer – 443.
  - Numa vServer – 443.
  - Proxmox VE – 8006.
  - VMware vSphere – 443.
  - Skala-R – 443.
  - Базис – 443.
  - Nutanix Acropolis – 9440.
  - HUAWEI FusionSphere – 7443.
  - TIONIX Cloud Platform / OpenStack / VK Cloud platform – 5000.

  > In an OpenStack platform, VK Cloud platform, or TIONIX Cloud Platform infrastructure, you only need to add a certificate if the Keystone microservice is using HTTPS.

- In infrastructures that use SSH for connections (KVM, Alt Virtualization Server, Astra Linux), execute the following request to get a certificate:

  ```
  GET api/3.0/sshConfig/getSshKey?address=<address>:<port>
  ```

  where:

  - <address> is the address of the virtual infrastructure object to which the Integration Server will connect to interact with the virtual infrastructure.
  - <port> is the port for connecting to the virtual infrastructure object.

  Port 22 is the standard port.
- In a Proxmox VE infrastructure, you must obtain a certificate for both HTTPS and SSH.

> If a port other than the standard port is used to connect to the infrastructure, the certificate must be requested through this port.

If the request succeeds, the following response is returned:

```
{
  "address": "<address>",
  "thumbprint": "<thumbprint>",
  "data": "<data>",
  "viisValidationResult": {
      "isAccepted": true,
      "validationWarnings": [],
      "validationErrors": []
  }
}
```

where:

- <address> is the address of the virtual infrastructure object specified in the request.
- <thumbprint> is the thumbprint of the certificate that you need to add to the list of trusted certificates of the Integration Server.
- <data> is the body of the received certificate, which is used to build and verify the received certificate.

Adding a certificate to the list of trusted certificates of the Integration Server

To add a certificate, execute the following request:

```
POST api/3.0/sslConfig/certificateValidator/rules
```

The following parameters must be specified in the request body:

```
{
  "address": "<address>:<port>",
  "thumbprint": "<thumbprint>"
}
```

where:

- <address>:<port> is the virtual infrastructure object's address and port that you specified in the certificate request.
- <thumbprint> is the certificate thumbprint received in response to the certificate request.

If the request succeeds, a 201 code and empty response body are returned:

Connecting the Integration Server and the virtual infrastructure

The procedure lets you add a virtual infrastructure to the list of infrastructures to which the Integration Server connects.

The procedure for connecting to OpenStack infrastructures differs from the standard procedure and is described separately.

Connecting to the virtual infrastructure

This section describes the procedure for connecting to virtual infrastructures based on the Microsoft Hyper-V, Numa vServer, VMware vSphere, KVM, Proxmox VE, Basis, Skala-R, HUAWEI FusionSphere, Nutanix Acropolis, ALT Virtualization Server, or Astra Linux platform.

To add the infrastructure connection settings to the Integration Server, execute the following request:

```
POST /api/3.0/infrastructures
```

The following parameters must be specified in the request body:

```
{
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "accounts": {
    "admin": {
      "name": "<admin login>",
      "password": "<admin password>"
    },
    "readOnly": {
      "name": "<read-only user login>",
      "password": "<read-only user password>"
    }
  }
}
```

where:

- <infrastructure type> is the type of virtual infrastructure. The infrastructure type is specified as follows:
  - HYPERV is a virtual infrastructure based on the Microsoft Hyper-V platform.
  - XEN is a virtual infrastructure based on the XenServer platform.
  - NUMAVSERVER is a virtual infrastructure based on the Numa vServer platform.
  - LIBVIRT is a virtual infrastructure based on the KVM (Kernel-based Virtual Machine) platform.
  - PROXMOX is a virtual infrastructure based on the Proxmox VE platform.

    - VMWARE is a virtual infrastructure based on the VMware vSphere platform.
    - ROSPLATFORMA is a virtual infrastructure based on the Skala-R platform.
    - BASISVCONTROL is a virtual infrastructure based on the Basis platform.
    - FUSIONCOMPUTE is a virtual infrastructure based on HUAWEI FusionSphere.
    - NUTANIXPRISM is a virtual infrastructure based on the Nutanix Acropolis platform.
    - ALTVIRTUALIZATIONSERVER is a virtual infrastructure based on the Alt Virtualization Server platform.
    - ASTRALINUX is a virtual infrastructure based on the Astra Linux platform.
- <infrastructure address> is the address of the virtual infrastructure object to which the Integration Server must connect. Depending on the type of virtual infrastructure, select a hypervisor or virtual infrastructure administration server.
- <admin login> is the name of an account with sufficient rights to deploy, remove, or reconfigure SVMs.
- <admin password> is the Base64-encoded password of the account.
- <read-only user login> is the name of an account with limited rights to perform actions in the virtual infrastructure (optional parameter). If no account with limited rights is specified, the Integration Server will use the account that has rights to deploy, remove, and reconfigure SVMs.
- <read-only user password> is the Base64-encoded password of the limited account.

If the request succeeds, the following response is returned:

```
{
  "infrastructureId": "<infrastructure ID>",
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "accounts": {
    "admin": {
      "name": "<admin login>"
    },
    "readOnly": {
      "name": "<read-only user login>"
    }
  },
  "connectionInfo": {
    "admin": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "errorMessage": ""
    },
    "readOnly": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "errorMessage": "",
      "updatePeriodSeconds": 10
    }
  }
}
```

where:

- <infrastructure ID> is the ID of the infrastructure to which the connection is being established.
- <connection status> is the current connection status. Possible values: CONNECTING | CONNECTED | DISCONNECTED.
- <error> is information about a connection error. Possible values: NO_ERROR | SERVER_ERROR | NETWORK_ERROR | INVALID_CERTIFICATE | ACCESS_DENIED | UNAUTHORIZED.

The process of connecting to the infrastructure takes some time. Please wait until the connection is successfully established.

To view the current connection status, execute the following request:

```
GET /api/3.0/infrastructures/<infrastructure ID>
```

The request uses the <infrastructure ID> obtained from the infrastructure connection request.

A connection status request returns the following response:

```
{
  "infrastructureId": "<infrastructure ID>",
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "name": "<infrastructure name>",
  "displayName": "<infrastructure name>",
  "version": "<infrastructure version>",
  "accounts": {
    "admin": {
      "name": "<admin login>"
    },
    "readOnly": {
      "name": "<read-only user login>"
    }
  },
  "connectionInfo": {
    "admin": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "errorMessage": ""
    },
    "readOnly": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "errorMessage": "",
      "updatePeriodSeconds": 10
    }
  }
}
```

The request must be repeated periodically until a successful connection is established or a connection error occurs. The infrastructure is considered to be successfully connected if the response has the following values in the "connectionInfo" -> "admin" and "connectionInfo" -> "readOnly" elements:

- "status": "CONNECTED"
- "connectionError": "NO_ERROR"

If an error occurs while connecting, the "status" field has the value "DISCONNECTED".

For Numa vServer and infrastructures based on the XenServer hypervisor, only connection to the primary hypervisor (primary server) is supported. If a secondary hypervisor (secondary server) was specified for the connection, a request to get the current connection status returns the address information of the primary server to which you need to connect.

```
{
  "infrastructureId": "<infrastructure ID>",
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "name": "<infrastructure name>",
  "displayName": "<infrastructure name>",
  "version": "<infrastructure version>",
  "accounts": {
    "admin": {
      "name": "<admin login>"
    },
    "readOnly": {
      "name": "<read-only user login>"
    }
  },
  "connectionInfo": {
    "admin": {
      "status": "DISCONNECTED"
      "connectionError": "NETWORK_ERROR"
      "errorMessage": "",
      "details": {
       "primaryServerAddress": "<primary server address>"
      }
    },
    "readOnly": {
      "status": "DISCONNECTED"
      "connectionError": "NETWORK_ERROR"
      "errorMessage": "",
      "updatePeriodSeconds": 10,
"details": {
      "primaryServerAddress": "<primary server address>"
      }
    }
  }
}
```

Connecting to the virtual infrastructure (OpenStack infrastructures)

This section describes the procedure for connecting to virtual infrastructures managed by the OpenStack platform, VK Cloud platform, or the TIONIX Cloud Platform.

Before connecting the Integration Server to the OpenStack virtual infrastructure, you need to check the connection to the microservices that are required for SVM operation.

Execute the following request:

`POST /api/3.0/infrastructures/testConnection`

The following parameters must be specified in the request body:

```
{
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "specificSettings": {
    "protocol": "<https | http>",
    "region": "<region id>"
  },
  "accounts": {
    "admin": {
      "domain": "<admin domain>",
      "name": "<admin login>",
      "password": "<admin password>"
    },
    "readOnly": {
      "domain": "<read-only user domain>",
      "name": "<read-only user login>",
      "password": "<read-only user password>"
    }
  }
}
```

where:

- <infrastructure type> is the type of virtual infrastructure. The infrastructure type is specified as follows:
  - TIONIXOPENSTACK is a virtual infrastructure based on the TIONIX Cloud Platform.
  - OPENSTACK is a virtual infrastructure based on the OpenStack platform.
  - VKCLOUDOPENSTACK is a virtual infrastructure based on the VK Cloud platform.
- <infrastructure address> is the address of the Keystone microservice to which the Integration Server should connect.
- <https | http> is the protocol used by the Keystone microservice.
- <region id> is the region in which the Integration Server must connect (optional parameter).
- <admin domain> is the name of the OpenStack domain to which an account belongs with sufficient rights to deploy, remove, or reconfigure the SVMs.
- <admin login> is the name of an account with sufficient rights to deploy, remove, or reconfigure SVMs.

- <admin password> is the Base64-encoded password of the account.
- <read-only user domain> is the name of the OpenStack domain to which the account with limited rights to perform actions in the virtual infrastructure belongs.
- <read-only user login> is the name of an account with limited rights to perform actions in the virtual infrastructure (optional parameter). If no account with limited rights is specified, the Integration Server will use the account that has rights to deploy, remove, and reconfigure SVMs.
- <read-only user password> is the Base64-encoded password of the limited account.

If the request succeeds, the following response is returned:

```
{
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "specificSettings": {
    "protocol": "<https | http>",
    "region": "<region id>"
  },
  "displayName": "<infrastructure name>",
  "name": "<infrastructure name>",
  "version": "<infrastructure version>",
  "edition": "<infrastructure edition>",
  "connectionInfo": {
    "admin": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "details": {
        "regions": [
          {
            "id": "<region id>"
            "endpoints": {
              "compute": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "glance": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "neutron": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "cinder": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "readonly": {
    "status": "<connection status>",
    "connectionError": "<error>",
    "updatePeriodSeconds": 0,
    "details": "..."
  }
}
```

The "endpoints" element contains a list of microservices required for the SVM to work.

If the microservice operates over HTTPS, a connection error may occur due to an invalid certificate:

- "status": "DISCONNECTED"
- "error": "INVALID_CERTIFICATE"

In this case, you need to add the microservice certificate to the trusted list using the requests described above. After that, you need to repeat the request to verify the connection to the microservices and make sure that all microservices have the "CONNECTED" status.

To add the infrastructure connection settings to the Integration Server, execute the following request:

`POST /api/3.0/infrastructures`

In the body of the request, specify the same parameters as in the request for verifying the connection to the microservices:

```
{
   "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "specificSettings": {
    "protocol": "<https | http>",
    "region": "<region id>"
  },
  "accounts": {
    "admin": {
      "domain": "<admin domain>",
      "name": "<admin login>",
      "password": "<admin password>"
    },
    "readOnly": {
      "domain": "<read-only user domain>",
      "name": "<read-only user login>",
      "password": "<read-only user password>"
    }
  }
}
```

If the request succeeds, the following response is returned:

```
{
  "infrastructureId": "<infrastructure ID>",
  "type": "<infrastructure type>",
  "address": "<infrastructure address>",
  "specificSettings": {
    "protocol": "<https | http>",
    "region": "<region id>"
  },
  "displayName": "<infrastructure name>",
  "name": "<infrastructure name>",
  "version": "<infrastructure version>",
  "edition": "<infrastructure edition>",
  "connectionInfo": {
    "admin": {
      "status": "<connection status>",
      "connectionError": "<error>",
      "details": {
        "regions": [
          {
            "id": "<region id>"
            "endpoints": {
              "compute": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "glance": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "neutron": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ],
              "cinder": [
                {
                  "address": "<IP>:<PORT>",
                  "protocol": "<https | http>",
                  "version": "<service version>",
                  "status": "<connection status>",
                  "error": "<error>"
                }
              ]
            }
          }
        ]
      }
    }
  },
  "readonly": {
    "status": "<connection status>",
    "connectionError": "<error>",
    "updatePeriodSeconds": 0,
    "details": "..."
  }
}
```

where:

- <infrastructure ID> is the ID of the infrastructure to which the connection is being established.
- <connection status> is the current connection status. Possible values: CONNECTING | CONNECTED | DISCONNECTED.
- <error> is information about a connection error. Possible values: NO_ERROR | SERVER_ERROR | NETWORK_ERROR | INVALID_CERTIFICATE | ACCESS_DENIED | UNAUTHORIZED.

The process of connecting to the infrastructure takes some time. Please wait until the connection is successfully established.

To view the current connection status, execute the following request:

```
GET /api/3.0/infrastructures/<infrastructure ID>
```

The request uses the <infrastructure ID> obtained from the infrastructure connection request.

As a result of executing a connection status request, the current status of the connection to the infrastructure and to each microservice is returned in the response body.

The request must be repeated periodically until a successful connection is established or a connection error occurs.

An OpenStack infrastructure is considered to be successfully connected if the following information is returned in the response body:

- the fields of the "connectionInfo" -> "admin" and "readonly" elements have the following values:
  - "status": "CONNECTED"
  - "connectionError": "NO_ERROR"
- the fields of the "endpoints" -> "<microservice>" elements have the following values:
  - "status": "CONNECTED"
  - "error": "NO_ERROR"

Registering an SVM image on the Integration Server

Deploying an SVM in a virtual infrastructure requires an SVM image file and an image description file (XML file). You can download archives containing SVM images and SVM image description files using the Kaspersky Security Components Installation Wizard or on the Kaspersky website (https://www.kaspersky.com/small-to-medium-business-security/downloads/virtualization-hybrid-cloud?utm_content=download). The obtained SVM image file and the image description file (XML file) must be placed in the same folder on the device where the Integration Server is installed.

The Integration Server must have access to the SVM image file and the image description file.

SVM image registration request

To register an SVM image on the Integration Server, execute the following request:

`POST /api/3.0/management/deployment/svm/images/register/manifest`

The following parameters must be specified in the request body:

```
{
  "manifestUrl": "<SVM image description file>",
  "hypervisors": [
    {
      "type": "<infrastructure type 1>"
    }
    {
      "type": "<infrastructure type 2>"
    }
  ]
}
```

where:

- <SVM image description file> is the absolute path to the image description file (*.xml) on the device with the Integration Server installed.
- <infrastructure type 1>, <infrastructure type 2> is the type of virtual infrastructure on which the SVM is to be deployed. You can specify one or more types of virtual infrastructure. The infrastructure type is specified as follows:
  - HYPERV is a virtual infrastructure based on the Microsoft Hyper-V platform.
  - XEN is a virtual infrastructure based on the XenServer platform.
  - NUMAVSERVER is a virtual infrastructure based on the Numa vServer platform.
  - LIBVIRT is a virtual infrastructure based on the KVM (Kernel-based Virtual Machine) platform.
  - PROXMOX is a virtual infrastructure based on the Proxmox VE platform.
  - VMWARE is a virtual infrastructure based on the VMware vSphere platform.

- ROSPLATFORMA is a virtual infrastructure based on the Skala-R platform.
- BASISVCONTROL is a virtual infrastructure based on the Basis platform.
- FUSIONCOMPUTE is a virtual infrastructure based on HUAWEI FusionSphere.
- NUTANIXPRISM is a virtual infrastructure based on the Nutanix Acropolis platform.
- TIONIXOPENSTACK is a virtual infrastructure based on the TIONIX Cloud Platform.
- OPENSTACK is a virtual infrastructure based on the OpenStack platform.
- VKCLOUDOPENSTACK is a virtual infrastructure based on the VK Cloud platform.
- ALTVIRTUALIZATIONSERVER is a virtual infrastructure based on the Alt Virtualization Server platform.
- ASTRALINUX is a virtual infrastructure based on the Astra Linux platform.

If the request succeeds, the ID of the registered SVM image (<SVM image ID>) and information about the image are returned in the response body:

```
{
  "id": "<SVM image ID>",
  "manifest": {
    "svmVersion": "<SVM image version>",
    "size": "<SVM image size>",
    "vendorInfo": {
      "en": {
        "productName": "<SVM image name>",
        "vendor": "AO Kaspersky Lab",
        "description": "<SVM image description_en>",
        "publisher": "AO Kaspersky Lab"
      },
      "ru": {
        "productName": "<SVM image name>",
        "vendor": "AO \"Kaspersky\"",
        "description": "<SVM image description_ru>",
        "publisher": "AO \"Kaspersky\""
      }
    }
  },
  "hypervisors": [
    {
      "type": "<infrastructure type>"
    }
  ]
}
```

Verifying the authenticity of the SVM image

After registering the image, it is recommended to verify the authenticity of the SVM image by using the following request:

`/api/3.0/management/deployment/svm/images/<SVM image ID>/validate`

The request uses the ID of the registered SVM image (<SVM image ID>) that is obtained as a result of the SVM image registration request.

The image verification request creates an image verification task:

```
{
  "id": "<task ID>",
  "created": "<timestamp>",
  "stateChanged": "<timestamp>",
  "changed": "<timestamp>",
  "state": "<task state>",
  "type": "ValidateSvmImage",
  "progress": 0,
  "children": [],
  "result": true
}
```

where:

- <task ID> is the task ID.
- <task state> is the task state. Possible values: Created, Queued, Starting, Running, Completed, Stopping, Failed, Cancelled.

The scan process takes some time. To receive the scan results, you will need to wait until the task is completed.

To view the current status of a task, execute the following request:

```
GET api/3.0/virtualization/tasks/<task ID>
```

A task is considered completed successfully if the "state" parameter in the response has the value "Completed". If an error occurred during verification, the "state" parameter in the response will have the value "Failed".

Example of a response for a successfully completed task for verifying the authenticity of an SVM image:

```
{
  "id": "<task ID>",
  "created": "<timestamp>",
  "stateChanged": "<timestamp>",
  "changed": "<timestamp>",
  "state": "Completed",
  "type": "ValidateSvmImage",
  "stage": null,
  "progress": 100,
  "children": [],
  "result": true
}
```

Registering the Network Agent package on the Integration Server

For installation on SVMs, you need Network Agent for Linux that is included in one of the supported versions of Kaspersky Security Center. Network Agent will be installed on SVMs during SVM deployment. Before deploying SVMs, you must prepare the Network Agent distribution package for installation:

1. Download the DEB package of Network Agent for Linux from the Kaspersky website (https://www.kaspersky.ru/small-to-medium-business-security/downloads/virtualization-hybrid-cloud?utm_content=downloads) in the **Kaspersky Security Center** section.
2. Place the Network Agent package on the device where the Integration Server is installed, in a folder to which the Integration Server has read access.
3. Read the terms and conditions of the End User License Agreement for Network Agent for Linux. The license.txt document is included in the Network Agent for Linux distribution package.
4. Create an answer file named answers.txt in the folder with the Network Agent package. In the file, specify:

   ```
   EULA_ACCEPTED=1
   ```

   The file name is case-sensitive. Use only lowercase letters.

   When installing on SVMs, other Network Agent installation settings are not supported.
5. Register the Network Agent package on the Integration Server.

To register the Network Agent package on the Integration Server, execute the following request:

```
POST /api/3.0/management/deployment/svm/components/register
```

The following parameters must be specified in the request body:

```
{
  "type": "Nagent",
  "category": "Package",
  "name": "<SVM tool name>",
  "source": {
    "type": "File",
    "sourceDetails": {
      "path": "<path to deb package>"
    }
  }
}
```

where:

- <SVM tool name> is the name with which the Network Agent package will be registered in the Integration Server database.
- <path to deb package> is the absolute path to the Network Agent package.

If the request succeeds, the ID of the registered package (<SVM tool ID>) and information about the package are returned in the response body:

```
{
  "id": "<SVM tool ID>",
  "type": "Nagent",
  "category": "Package",
  "name": "<SVM tool name>",
  "source": {
    "type": "File",
     "sourceDetails": {
       "path": "<path to deb package>"
     }
  }
}
```

Getting information about infrastructure objects required for SVM deployment

Before starting SVM deployment, you need to get information about available virtual infrastructure objects and select the objects that you will specify in the SVM deployment request.

It is recommended to update the infrastructure information before receiving infrastructure objects, especially if the infrastructure was added a long time ago.

To do this, execute the following request:

```
POST /api/3.0/infrastructures/<infrastructure ID>/refresh
```

If the request succeeds, a 200 code is returned.

After executing the request, it is recommended to wait for the connection to the infrastructure, just like after adding the infrastructure (see the "Connecting the Integration Server and the virtual infrastructure" section).

The procedure for getting information about OpenStack infrastructure objects differs from the standard procedure and is described separately.

Getting information about infrastructure objects

To get information about infrastructure objects, you can use the following requests:

- Request to get a list of infrastructure objects:

  ```
  GET /api/3.0/infrastructures/<infrastructure ID>/objects
  ```

  In this request, you can use the following parameters:

  - "objectTypes". You can use this parameter to query only objects of specific types. This parameter can take the following values: Host, VM, SVM. By default, the parameter is set to Host.

    For example, you can request a list of hypervisors and SVMs:

    ```
    GET /api/3.0/infrastructures/<infrastructure ID>/objects?objectTypes=Host+SVM
    ```

  - "outputType". You can use this parameter to query brief or full information about objects. This parameter can take the following values: GeneralInfo, DetailedInfo. By default, the parameter is set to GeneralInfo.
  - "parentId". You can use this parameter to query information about child objects of one of the objects. You must pass the ID of the parent object to the "parentId" parameter.

  For example, to get complete information about all hypervisors, virtual machines, and SVMs, execute the following request:

  ```
  GET /api/3.0/infrastructures/<infrastructure
  ID>/objects?objectTypes=Host+SVM+VM&outputType=DetailedInfo
  ```

  Result of successful execution of the request:

```
{
  "locations": [
    {
```

```
        "id": "<hypervisor ID>",
        "name": "<hypervisor name>",
        "parentId": "<infrastructure ID>",
        "state": "Started",
        "resources": {
          "networks": [
            {
              "id": "<network ID 1>",
              "name": "<network name 1>"
            },
            {
              "id": "<network ID 2>",
              "name": "<network name 2>"
            }
          ],
          "storages": [
            {
              "id": "<storage ID 1>",
              "name": "<storage name 1>",
              "freeSpace": <storage free space 1>,
            },
            {
              "id": "<storage ID 2>",
              "name": "<storage name 2>",
              "freeSpace": <storage free space 2>,
            }
          ]
        }
      }
    ],
    "vms": [
      {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<hypervisor ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID 1>",
            "name": "<network name 1>"
          }
        ],
        "isSvm": false
      }
      {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<hypervisor ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID 1>",
            "name": "<network name 1>"
          }
        ],
        "isSvm": true
        "svmVersion": "<svm version>"
      }
    }]
  }
```

- Request to get information about a single object by its ID:

  `GET /api/3.0/infrastructures/<infrastructure ID>/objects/<object ID>`

  If the request is successful, complete information about a single object is returned. For example, information about a virtual machine:

```
  {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<hypervisor ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID 1>",
            "name": "<network name 1>"
          }
        ],
        "isSvm": false
  }
```

If you request complete information about all objects at the same time in large infrastructures, the request may take a long time to complete. Therefore, we recommend using the following algorithm for querying infrastructure objects:

1. Request brief information about hypervisors:

   ```
   GET /api/3.0/infrastructures/<infrastructure
   ID>/objects?objectTypes=Host&outputType=GeneralInfo
   ```

2. For the hypervisors on which you plan to deploy SVMs, request complete information:

   `GET /api/3.0/infrastructures/<infrastructure ID>/objects/<hypervisor ID>`

3. If you need to find out which virtual machines or SVMs are deployed on this hypervisor, pass the hypervisor ID in the "parentId" parameter of your request:

   ```
   GET /api/3.0/infrastructures/<infrastructure ID>/objects?
   objectTypes=VM+SVM&outputType=GeneralInfo&parentId=<hypervisor ID>
   ```

4. If you want to reconfigure an SVM and need to get a list of network adapters configured on the SVM, request complete information about the SVM using a request to get a single object by ID:

   `GET /api/3.0/infrastructures/<infrastructure ID>/objects/<SVM ID>`

From the list of received infrastructure objects, you need to select the following settings to be used in the SVM deployment request:

- <hypervisor ID> is the ID of the hypervisor on which the SVM will run.

- <storage ID> in the "resources" -> " storages " element in the parameters of the selected hypervisor is the ID of the storage on this hypervisor, in which the SVM will be deployed.
- <network ID 1>, <network ID 2>, etc. in the "resources" -> "networks" element in the parameters of the selected hypervisor are the IDs of the virtual networks to be used by the SVM.

Getting information about infrastructure objects (OpenStack infrastructure)

To get information about infrastructure objects, you can use the following requests:

- Request to get a list of infrastructure objects:

  ```
  GET /api/3.0/infrastructures/<infrastructure ID>/objects
  ```

  In this request, you can use the following parameters:

  - "objectTypes". You can use this parameter to query only objects of specific types. This parameter can take the following values: OpenstackDomain, OpenstackProject, VM, SVM. By default, the parameter is set to OpenstackDomain+OpenstackProject.

    For example, you can request a list of OpenStack domains, OpenStack projects, and SVMs:

    ```
    GET /api/3.0/infrastructures/<infrastructure
    ID>/objects?objectTypes=OpenstackDomain+OpenstackProject+SVM
    ```

  - "outputType". You can use this parameter to query brief or full information about objects. This parameter can take the following values: GeneralInfo, DetailedInfo. By default, the parameter is set to GeneralInfo.
  - "parentId". You can use this parameter to query information about child objects of one of the objects. You must pass the ID of the parent object to the "parentId" parameter.

  For example, to get complete information about all projects, infrastructure domains, virtual machines, and SVMs, execute the following request:

  ```
  GET /api/3.0/infrastructures/<infrastructure
  ID>/objects?objectTypes=OpenstackDomain+OpenstackProject+SVM+VM&outputType=Detail
  edInfo
  ```

  Result of successful execution of the request:

```
{
  "locations": [
    {
      "id": "<domain ID>",
      "name": "<domain name>",
      "parentId": "<infrastructure ID>",
    },
    {
      "id": "<project ID>",
      "name": "<project name>",
      "parentId": "<domain ID>",
      "resources": {
        "networks": [
          {
            "id": "<network ID>",
```

```
              "name": "<network name>"
            }
          ],
          "volumeTypes": [
            {
              "id": "<volume type ID>",
              "name": "<volume type name>"
            }
          ],
          "securityGroups": [
            {
              "id": "<security group ID>",
              "name": "<security group name>"
            }
          ],
          "serverGroups": [
            {
              "id": "<server group ID>",
              "name": "<server group name>"
            }
          ],
          "flavors": [
            {
              "id": "<flavor ID>",
              "name": "<flavor name>",
              "vCpus": 1,
              "ramInMegabytes": 2048,
              "diskInGigabytes": 10
              "suitableForSvmVersions": [
                "5.2"
                "6.2"
              ]
            }
          ],
          "availabilityZones": [
            {
              "name": "<availability zone name>"
            }
          ]
        }
      }
    ],
    "vms": [
      {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<project ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID>",
            "name": "<network name>"
            "securityGroups": [
              {
                "id": "<security group ID>",
                "name": "<security group name>"
              }
            ]
          }
        ],
        "isSvm": false
    },
      {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<hypervisor ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID 1>",
            "name": "<network name 1>"
            "securityGroups": [
              {
                "id": "<security group ID>",
                "name": "<security group name>"
              }
            ]
          }
        ],
        "isSvm": true
        "svmVersion": "<svm version>"
      }
    ]
  }
```

- Request to get information about a single object by its ID:

  `GET /api/3.0/infrastructures/<infrastructure ID>/objects/<object ID>`

  If the request is successful, complete information about a single object is returned. For example, information about a virtual machine:

```
  {
        "id": "VM ID",
        "name": "VM name",
        "parentId": "<hypervisor ID>",
        "state": "Enabled",
        "networkAdapters": [
          {
            "id": "<network ID 1>",
            "name": "<network name 1>"
          }
        ],
        "isSvm": false
}
```

If you request complete information about all objects at the same time in large infrastructures, the request may take a long time to complete. Therefore, we recommend using the following algorithm for querying infrastructure objects:

1. Request brief information about domains and projects:

   ```
   GET /api/3.0/infrastructures/<infrastructure
   ID>/objects?objectTypes=OpenstackDomain+OpenstackProject&outputType=GeneralInfo
   ```

2. For projects in which you plan to deploy SVMs, request complete information:

   ```
   GET /api/3.0/infrastructures/<infrastructure ID>/objects/<project ID>
   ```

3. If you need to find out which virtual machines or SVMs exist in this project, pass the project ID in the "parentId" parameter of your request:

   ```
   GET /api/3.0/infrastructures/<infrastructure ID>/objects?
   objectTypes=VM+SVM&outputType=GeneralInfo&parentId=<project ID>
   ```

4. If you want to reconfigure an SVM and need to get a list of network adapters configured on the SVM, request complete information about the SVM using a request to get a single object by ID:

   ```
   GET /api/3.0/infrastructures/<infrastructure ID>/objects/<SVM ID>
   ```

From the list of received infrastructure objects, you need to select the following settings to be used in the SVM deployment request:

- <project ID> is the ID of the OpenStack project on which the SVMs will run.
- <flavor ID> in the "resources" -> " flavors " element in the parameters of the selected OpenStack project is the virtual machine type (instance type) ID that defines the amount of RAM, disk size, number of processor cores, and other parameters of the created virtual machine. Make sure that the selected virtual machine type complies with Kaspersky experts' recommendations on SVM resource allocation and that it meets the minimum requirements for SVM operation.
- <network ID> in the "resources" -> " networks " element in the parameters of the selected OpenStack project is the ID of the virtual network that the SVM will use to communicate with Light Agents, the Integration Server, and the Kaspersky Security Center Administration Server. You can select one or more networks. If necessary, you can select one or more Security Groups for each network (<security group ID> in the "resources" -> "securityGroups" element in the OpenStack project parameters).
- <volume type ID>, <server group ID>, <availability zone name> in the "resources" element in the parameters of the selected OpenStack project are optional parameters; you can specify them during SVM deployment, if required.

For more details on the settings for SVM deployment in OpenStack infrastructures, refer to the Kaspersky Security for Virtualization Light Agent Help (https://support.kaspersky.com/KSVLA/6.3/ru-RU/74377.htm).

Deploying SVMs in a virtual infrastructure

Before starting SVM deployment, you need to complete the following procedures:

1. Configure of the connection between the Integration Server and the virtual infrastructure.
2. Add the virtual infrastructure certificate to the list of trusted certificates of the Integration Server.
3. Get a list of virtual infrastructure objects required for SVM deployment.
4. Register SVM images on the Integration Server.
5. Register the Network Agent package on the Integration Server.

The SVM deployment procedure for OpenStack infrastructures differs from the standard procedure and is described separately.

SVM deployment request

To deploy SVMs, execute the following request:

```
POST /api/3.0/management/deployment/svm/
```

The following parameters must be specified in the request body:

```
{
  "threadsCount": "<number of SVMs>",
  "image": {
    "id": "<SVM image ID>",
    "skipCheckIntegrity": "<skip check: true | false>",
    "localization": "<SVM image localization>"
  },
  "components": [ { "id": "<SVM tool ID>" } ],
  "svms": [
    {
      "svmSettings": {
        "name": "<SVM name>",
        "ksc": {
          "address": "<KSC address>",
          "port": "<KSC port>",
          "sslPort": "<KSC SSL port>",
          "language": "<KSC localization>"
        },
        "users": {
          "root": {
            "allowSshAccess": "<SSH access: true | false>",
            "password": "<root user password>"
          },
          "klconfig": {
            "password": "<klconfig user password>"
          }
        },
        "dns": {
          "main": "<DNS IP>",
          "alternative": "<alternative DNS IP>"
        }
      },
      "deploymentInfo": {
        "type": "host",
        "target": {
          "infrastructureId": "<infrastructure ID>",
          "location": {
            "id": "<hypervisor ID>"
          }
        },
        "settings": {
          "storageId": "<storage ID>",
          "customStoragePath": "<custom storage path>",
          "networks": [
            {
              "id": "<network ID>",
              "vlanId": "<VLAN ID>",
              "isPrimary": <primary network: true | false>,
              "type": "<IP addressing: static | dynamic>",
              "ipAddress": "<SVM IP>",
              "mask": "<subnet mask>",
              "gateway": "<gateway>"
            }
          ]
        }
      }
    }
  ]
}
```

where:

- <number of SVMs> is the number of SVMs that will be deployed simultaneously.
- <SVM image ID> is the SVM image ID. You can obtain this ID by executing an SVM image registration request.
- <SVM tool ID> is the ID of the Network Agent package. You can obtain this ID by executing a Network Agent package registration request.
- <skip check: true | false> indicates whether to skip image authentication before starting SVM deployment: true – skip verification, false – perform verification. You can skip verification if you have executed a request to verify the authenticity of the SVM image after registering the SVM image on the Integration Server (see the "Verifying the authenticity of the SVM image" section).
- <SVM image localization> is the localization that will be used to register the image in the infrastructure.
- <SVM name> is the SVM name that will be used in the infrastructure.
- "ksc" – in this element, specify the settings for connecting the SVM to the Kaspersky Security Center Administration Server:

    - <KSC address> is the address of the device hosting the Kaspersky Security Center Administration Server.
    - <KSC port> is the port for connecting the SVM to the Kaspersky Security Center Administration Server.
    - <KSC SSL port> is the port for connecting an SVM to the Kaspersky Security Center Administration Server using an SSL certificate.
    - <KSC localization> is the localization of Kaspersky Security Center.
- "users" – in this element, specify the settings of accounts on the SVM:
    - <SSH access: true | false> indicates whether to allow remote access to the SVM via SSH under the root account: true – allow, false – deny.
    - <root user password> is the Base64-encoded password for the root user.
    - <klconfig user password> is the Base64-encoded password of the klconfig user.
- "dns" – in this element, you need to specify the IP addresses of DNS servers if static IP addressing is used for SVMs. This parameter is optional when using dynamic IP addressing (DHCP).
    - <DNS IP> is the IP address of the DNS server.
    - <alternative DNS IP> is the IP address of the alternative DNS server.
- "deploymentInfo" – in this element, specify the settings for the location of the new SVM in the infrastructure. You can get the infrastructure object IDs by executing a request to get a list of infrastructure objects.
    - "type" is the type of infrastructure object where the SVM is located. The parameter value must be "host".
    - <infrastructure ID> is the infrastructure ID.
    - <hypervisor ID> is the ID of the hypervisor on which the SVM will be deployed.
    - <storage ID> is the ID of the storage on this hypervisor .
    - <custom storage path> is the path where the SVM image will be located on the Hyper-V disk. The path must start with the storage ID on the hypervisor (<storage ID>). Optional parameter. The VLAN ID is displayed when you deploy SVM in a virtual infrastructure running on Microsoft Hyper-V platform.
    - "networks" – in this item, specify the list of networks that will be used by the SVM.
        - <network ID> is the network ID.
        - <VLAN ID> is the VLAN ID. Optional parameter. The VLAN ID is displayed when you deploy SVM in a virtual infrastructure running on Microsoft Hyper-V platform.
        - <primary network: true | false> indicates whether the network is the primary network. If you specified a single network, it must be the primary network. If multiple networks are specified, only one of them can be the primary network.
        - <IP addressing: static | dynamic> is the type of IP addressing for SVMs: static – static IP addressing, dynamic – dynamic IP addressing (DHCP).
        - <SVM IP>, <subnet mask>, and <gateway> are the static IP addressing settings: SVM IP address, subnet mask, and gateway. The parameters must be specified if you selected static IP addressing for the SVMs ("type": "static").

You can deploy multiple SVMs across multiple infrastructures using a deployment request. To deploy one SVM per hypervisor, you only need to specify one hypervisor and one SVM in the request body.

An SVM deployment request creates a DeployMultipleSvm task. The SVM deployment process takes some time. You will need to wait until the task finishes (see the "SVM deployment task" section).

SVM deployment request (OpenStack infrastructures)

To deploy SVMs in an OpenStack infrastructure, execute the following request:

`POST /api/3.0/management/deployment/svm/`

The following parameters must be specified in the request body:

```
{
  "threadsCount": "<number of SVMs>",
  "image": {
    "id": "<SVM image ID>",
    "skipCheckIntegrity": "<skip check: true | false>",
    "localization": "<SVM image localization>"
  },
  "components": [ { "id": "<SVM tool ID>" } ],
  "svms": [
    {
      "svmSettings": {
        "name": "<SVM name>",
        "ksc": {
          "address": "<KSC address>",
          "port": "<KSC port>",
          "sslPort": "<KSC SSL port>",
          "language": "<KSC localization>"
        },
        "users": {
          "root": {
            "allowSshAccess": "<SSH access: true | false>",
            "password": "<root user password>"
          },
          "klconfig": {
            "password": "<klconfig user password>"
          }
        },
        "dns": {
          "main": "<DNS IP>",
          "alternative": "<alternative DNS IP>"
        }
      },
      "deploymentInfo": {
        "type": "openstackProject",
        "target": {
          "infrastructureId": "<infrastructure ID>",
          "location": {
            "id": "<project ID>"
          }
        },
        "settings": {
          "flavorId": "<flavor ID>",
          "volumeTypeId": "<volume type ID>",
          "availabilityZoneName": "<availability zone name>",
          "serverGroupId": "<server group ID>",
          "networks": [
            {
              "id": "<network ID>",
              "ports": [
                {
                  "isPrimary": <primary network: true | false>,
                  "type": "<IP addressing: static | dynamic>",
                  "ipAddresses": [
                    "<network IP>"
                  ],
                  "mask": "<subnet mask>",
                  "gateway": "<gateway>"
                  "vlanId": "<VLAN ID>,
                  "portSecurityEnabled": true,
                  "securityGroupIds": [
                    "<security group ID>"
                  ]
                }
              ]
            }
          ]
        }
      }
    }
  ]
}
```

where:

- <number of SVMs> is the number of SVMs that will be deployed simultaneously.
- <SVM image ID> is the SVM image ID. You can obtain this ID by executing an SVM image registration request.
- <SVM tool ID> is the ID of the Network Agent package. You can obtain this ID by executing a Network Agent package registration request.
- <skip check: true | false> indicates whether to skip image authentication before starting SVM deployment: true – skip verification, false – perform verification. You can skip verification if you have executed a request to verify the authenticity of the SVM image after registering the SVM image on the Integration Server (see the "Verifying the authenticity of the SVM image" section).
- <SVM image localization> is the localization that will be used to register the image in the infrastructure.
- <SVM name> is the SVM name that will be used in the infrastructure.
- "ksc" – in this element, specify the settings for connecting the SVM to the Kaspersky Security Center Administration Server:

    - <KSC address> is the address of the device hosting the Kaspersky Security Center Administration Server.
    - <KSC port> is the port for connecting the SVM to the Kaspersky Security Center Administration Server.
    - <KSC SSL port> is the port for connecting an SVM to the Kaspersky Security Center Administration Server using an SSL certificate.
    - <KSC localization> is the localization of Kaspersky Security Center.
- "users" – in this element, specify the settings of accounts on the SVM:
    - <SSH access: true | false> indicates whether to allow remote access to the SVM via SSH under the root account: true – allow, false – deny.
    - <root user password> is the Base64-encoded password for the root user.
    - <klconfig user password> is the Base64-encoded password of the klconfig user.
- "dns" – in this element, you need to specify the IP addresses of DNS servers if static IP addressing is used for SVMs. This parameter is optional when using dynamic IP addressing (DHCP).
    - <DNS IP> is the IP address of the DNS server.
    - <alternative DNS IP> is the IP address of the alternative DNS server.
- "deploymentInfo" – in this element, specify the settings for the location of the new SVM in the infrastructure. You can get infrastructure object IDs by executing a request to get a list of OpenStack infrastructure objects.
    - "type" is the type of infrastructure object where the SVM is located. The parameter value must be "openstackProject".
    - <infrastructure ID> is the infrastructure ID.
    - <project ID> is the ID of the OpenStack project under which the SVM will be deployed.
    - <flavor ID> is the ID of the virtual machine type (instance type).
    - <volume type ID>, <availability zone name>, and <server group ID> are optional parameters. You can specify them when deploying SVMs, if required. For more details on the settings for SVM deployment in OpenStack infrastructures, refer to the Kaspersky Security for Virtualization Light Agent Help (https://support.kaspersky.com/KSVLA/6.3/en-US/74377.htm).
    - "networks" – in this item, specify the list of networks that will be used by the SVM.
        - <network ID> is the network ID.
        - "ports" – this element contains settings of network adapters for the network. You can define multiple network adapters for the same network.
            - <primary network: true | false> indicates whether the network is the primary network. If you specified a single network, it must be the primary network. If multiple networks are specified, only one of them can be the primary network.
            - <IP addressing: static | dynamic> is the type of IP addressing for SVMs: static – static IP addressing, dynamic – dynamic IP addressing (DHCP).
            - <network IP> is the list of IP addresses for the network adapter.
            - <VLAN ID> is the VLAN ID.
            - <subnet mask>, <gateway> are the static IP addressing settings: SVM IP address, subnet mask, gateway. The parameters must be specified if you selected static IP addressing for the SVMs ("type": "static").

- <security group ID> is the list of security groups to use for the network adapter. If you specified this parameter, "portSecurityEnabled": true must also be specified.

You can deploy multiple SVMs across multiple OpenStack projects using a deployment request. To deploy one SVM, you only need to specify one OpenStack project and one SVM in the request body.

An SVM deployment request creates a DeployMultipleOpenStackSvm task. The SVM deployment process takes some time. You will need to wait until the task finishes (see the "SVM deployment task" section).

SVM deployment task

In the "children" element of the SVM deployment task, for each SVM there is a subtask of the DeploySvm type (or DeployOpenStackSvm type for OpenStack infrastructures). There can be several subtasks, depending on the number of SVMs in the SVM deployment request. Subtasks run simultaneously or sequentially. The number of simultaneously started subtasks depends on the value of the `threadsCount` parameter.

After deployment starts, the deployment steps – tasks of the DeploySvmStep type that are performed sequentially – appear in the DeploySvm (or DeployOpenStackSvm) subtasks.

Example of an SVM deployment task in an OpenStack infrastructure:

```
{
  "id": "<TASK ID>",
  "created": "<timestamp>",
  "stateChanged": "<timestamp>",
  "changed": "<timestamp>",
  "state": "Running",
  "type": "DeployMultipleOpenStackSvm",
  "progress": 0,
  "children": [
    {
      "id": "fc3e6aeb-d16c-4c86-9756-cb6f5d63a46a",
      "created": "<timestamp>",
      "stateChanged": "<timestamp>",
      "changed": "<timestamp>",
      "state": "Running",
      "type": "DeployOpenStackSvm",
      "progress": 0,
      "parentId": "2a76b873-5d2e-4150-9111-ee2988fe5305",
      "parameters": [
        {
          "key": "infrastructureId",
          "value": "Infrastructure Id"
        },
        {
          "key": "hostId",
          "value": "Host Id"
        },
        {
          "key": "svmName"
```

```
            "value": "SVM name"
          },
          {
            "key": "svmId"
            "value": "SVM id"
          },
          {
            "key": "svmIpAddress"
            "value": "SVM IP address"
          }
        ],
        "result":{"svmId":"<SVM ID>"}
        "children": [
          {
            "id": "ebbd2180-d275-435e-95ec-5b9d8316332d",
            "created": "<timestamp>",
            "stateChanged": "<timestamp>",
            "changed": "<timestamp>",
            "state": "Running",
            "type": "DeploySvmStep",
            "progress": 100,
            "parentId": "fc3e6aeb-d16c-4c86-9756-cb6f5d63a46a",
            "parameters": [
              {
                "key": "deployStepName",
                "value": "UploadSvmImage"
              }
            ],
            "children": [],
            "result": true
          },
          ....
        ]
      }
    ]
  }
```

The deployment task takes some time. You will need to wait for it to complete.

To view the current status of a task, execute the following request:

```
GET api/3.0/virtualization/tasks/<task ID>
```

A task is considered completed successfully if the "state" parameter in the response has the value "Completed". After the task completes successfully, the ID of the created SVM in the infrastructure will appear in the "result": {"svmId": "<SVM ID>"} field.

Reconfiguring deployed SVMs

You can use the reconfiguration procedure to change the following settings of deployed SVMs:

- Mode for remote access to SVMs via SSH.
- List of virtual networks that SVMs use to connect to Light Agents, the Integration Server, and the Kaspersky Security Center Administration Server, as well as SVM IP addressing settings.
- IP addresses of DNS servers.
- Settings for connecting SVMs to the Kaspersky Security Center Administration Server.
- Configuration password and root account password.

The procedure for reconfiguring SVMs for OpenStack infrastructures differs from the standard procedure and is described separately.

SVM reconfiguration request

To reconfigure SVMs, execute the following request:

`PUT /api/3.0/management/deployment/svm/`

The following parameters must be specified in the request body:

```
{
  "threadsCount": "<number of SVMs>",
  "svms": [
    {
      "svmSettings": {
        "password": "<klconfig user password>",
        "ksc": {
          "address": "<new KSC address>",
          "port": "<new KSC port>",
          "sslPort": "<new KSC SSL port>",
          "language": "<KSC localization>"
        },
        "users": {
          "root": {
            "allowSshAccess": "<SSH access: true | false>",
            "password": "<new root user password>"
          },
          "klconfig": {
            "password": "<new klconfig user password>"
          }
        },
        "dns": {
          "main": "<new DNS IP>",
          "alternative": "<new alternative DNS IP>"
        }
      },
      "deploymentInfo": {
        "type": "host",
        "target": {
          "infrastructureId": "<infrastructure ID>",
          "svmId": "<SVM ID>",
          "location": {
            "id": "<hypervisor ID>"
          }
        },
        "settings": {
          "networks": [
            {
              "id": "<network ID>",
              "vlanId": "<VLAN ID>",
              "isPrimary": <primary network: true | false>,
              "type": "<IP addressing: static | dynamic>",
              "ipAddress": "<SVM IP>",
              "mask": "<subnet mask>",
              "gateway": "<gateway>"
            }
          ]
        }
      }
    }
  ]
}
```

where:

- <number of SVMs> - the number of SVMs that will be reconfigured simultaneously.
- <klconfig user password> is the current password of the klconfig user that was created during SVM deployment. The password must be Base64 encoded.
- "ksc" is an optional element. If you want to change the settings for connecting the SVM to the Kaspersky Security Center Administration Server, specify the new settings in this element:
  - <KSC address> is the address of the device hosting the Kaspersky Security Center Administration Server.
  - <KSC port> is the port for connecting the SVM to the Kaspersky Security Center Administration Server.
  - <KSC SSL port> is the port for connecting an SVM to the Kaspersky Security Center Administration Server using an SSL certificate.
  - <KSC localization> is the localization of Kaspersky Security Center.
- "users" is an optional element. If you want to change the settings of accounts on SVMs, specify the new settings in this element:
  - <SSH access: true | false> indicates whether to allow remote access to the SVM via SSH under the root account: true – allow, false – deny.
  - <root user password> is the new Base64-encoded root user password.
  - <klconfig user password> is the new Base64-encoded password for the klconfig user.
- "dns" – if you want to change the IP addresses of DNS servers, specify the new IP addresses in this element. The IP addresses of DNS servers must be specified if static IP

addressing is used for the SVMs. This parameter is optional when using dynamic IP addressing (DHCP).

- <DNS IP> is the IP address of the DNS server.
- <alternative DNS IP> is the IP address of the alternative DNS server.

- "deploymentInfo" – this element contains the settings for the location of SVMs in the infrastructure:
    - "type" is the type of infrastructure object where the SVM is located. The parameter value must be "host".
    - <infrastructure ID> is the ID of the infrastructure in which the SVM is deployed.
    - <SVM ID> is the SVM ID.
    - <hypervisor ID> is the ID of the hypervisor on which the SVM is deployed.
    - "networks" is an optional element. If you want to change the list of networks used by the SVM, specify new settings in this element:
        - <network ID> is the network ID.
        - <VLAN ID> is the VLAN ID. The VLAN ID is displayed when you deploy SVM in a virtual infrastructure running on Microsoft Hyper-V platform. Optional parameter.
        - <primary network: true | false> indicates whether the network is the primary network. If you specified a single network, it must be the primary network. If multiple networks are specified, only one of them can be the primary network.
        - <IP addressing: static | dynamic> is the type of IP addressing for SVMs: static – static IP addressing, dynamic – dynamic IP addressing (DHCP).
        - <SVM IP>, <subnet mask>, and <gateway> are the static IP addressing settings: SVM IP address, subnet mask, and gateway. The parameters must be specified if you selected static IP addressing for the SVMs ("type": "static").

An SVM reconfiguration request creates a ConfigMultipleSvm task. The SVM reconfiguration process takes some time. You will need to wait until the task finishes (see the "SVM reconfiguration task" section).

SVM reconfiguration request (OpenStack infrastructure)

To reconfigure SVMs in an OpenStack infrastructure, execute the following request:

```
PUT /api/3.0/management/deployment/svm/
```

The following parameters must be specified in the request body:

```
{
  "threadsCount": "<number of SVMs>",
  "svms": [
    {
      "svmSettings": {
        "password": "<klconfig user password>",
        "ksc": {
          "address": "<new KSC address>",
          "port": "<new KSC port>",
          "sslPort": "<new KSC SSL port>",
          "language": "<KSC localization>"
        },
        "users": {
          "root": {
            "allowSshAccess": "<SSH access: true | false>",
            "password": "<new root user password>"
          },
          "klconfig": {
            "password": "<new klconfig user password>"
          }
        },
        "dns": {
          "main": "<new DNS IP>",
          "alternative": "<new alternative DNS IP>"
        }
      },
      "deploymentInfo": {
        "type": "openstackProject",
        "target": {
          "infrastructureId": "<infrastructure ID>",
          "svmId": "<SVM ID>",
          "location": {
            "id": "<project ID>"
          }
        },
        "settings": {
          "networks": [
            {
              "id": "<network ID>",
              "ports": [
                {
                  "isPrimary": <primary network: true | false>,
                  "type": "<IP addressing: static | dynamic>",
                  "ipAddresses": [
                    "<network IP>"
                  ],
                  "mask": "<subnet mask>",
                  "gateway": "<gateway>"
                  "vlanId": "<VLAN ID>",
                  "portSecurityEnabled": true,
                  "securityGroupIds": [
                    "<security group ID>"
                  ]
                }
              ]
            }
          ]
        }
      }
    }
  ]
}
```

where:

- <number of SVMs> - the number of SVMs that will be reconfigured simultaneously.
- <klconfig user password> is the current password of the klconfig user that was created during SVM deployment. The password must be Base64 encoded.
- "ksc" is an optional element. If you want to change the settings for connecting the SVM to the Kaspersky Security Center Administration Server, specify the new settings in this element:
  - <KSC address> is the address of the device hosting the Kaspersky Security Center Administration Server.
  - <KSC port> is the port for connecting the SVM to the Kaspersky Security Center Administration Server.
  - <KSC SSL port> is the port for connecting an SVM to the Kaspersky Security Center Administration Server using an SSL certificate.
  - <KSC localization> is the localization of Kaspersky Security Center.
- "users" is an optional element. If you want to change the settings of accounts on SVMs, specify the new settings in this element:
  - <SSH access: true | false> indicates whether to allow remote access to the SVM via SSH under the root account: true – allow, false – deny.
  - <root user password> is the new Base64-encoded root user password.
  - <klconfig user password> is the new Base64-encoded password for the klconfig user.
- "dns" – if you want to change the IP addresses of DNS servers, specify the new IP addresses in this element. The IP addresses of DNS servers must be specified if static IP addressing is used for the SVMs. This parameter is optional when using dynamic IP addressing (DHCP).
  - <DNS IP> is the IP address of the DNS server.
  - <alternative DNS IP> is the IP address of the alternative DNS server.
- "deploymentInfo" – this element contains the settings for the location of SVMs in the infrastructure:
  - "type" is the type of infrastructure object where the SVM is located. The parameter value must be "openstackProject".
  - <infrastructure ID> is the ID of the infrastructure in which the SVM is deployed.
  - <SVM ID> is the SVM ID.
  - <project ID> is the ID of the OpenStack project under which the SVM has been deployed.
  - "networks" is an optional element. If you want to change the list of networks used by the SVM, specify new settings in this element:
    - <network ID> is the network ID.
    - "ports" – this element contains settings of network adapters for the network. You can define multiple network adapters for the same network.
      - <primary network: true | false> indicates whether the network is the primary network. If you specified a single network, it must be the primary network. If multiple networks are specified, only one of them can be the primary network.
      - <IP addressing: static | dynamic> is the type of IP addressing for SVMs: static – static IP addressing, dynamic – dynamic IP addressing (DHCP).
      - <network IP> is the list of IP addresses for the network adapter.

- <subnet mask>, <gateway> are the static IP addressing settings: subnet mask, gateway. The parameters must be specified if you selected static IP addressing for the SVMs ("type": "static").
- <VLAN ID> is the VLAN ID.
- <security group ID> is the list of security groups to use for the network adapter. If you specified this parameter, "portSecurityEnabled": true must also be specified.

An SVM reconfiguration request creates a ConfigMultipleOpenStackSvm task. The SVM reconfiguration process takes some time. You will need to wait until the task finishes (see the "SVM reconfiguration task" section).

SVM reconfiguration task

In the "children" element of the SVM reconfiguration task, for each SVM there is a subtask of the ConfigSvm type (or ConfigOpenStackSvm type for OpenStack infrastructures). There can be several subtasks, depending on the number of SVMs in the SVM reconfiguration request. Subtasks run simultaneously or sequentially. The number of simultaneously started subtasks depends on the value of the "threadsCount" parameter.

The SVM reconfiguration task is similar to the SVM deployment task (see the "SVM deployment task" section).

SVM removal

To remove SVMs, execute the following request:

```
DELETE /api/3.0/management/deployment/svm/
```

The following parameters must be specified in the request body:

```
{
  "svms": [
    {
      "deploymentInfo": {
        "type": "<infrastructure object type>",
        "target": {
          "infrastructureId": "<infrastructure ID>",
          "svmId": "<SVM ID>",
          "location": {
            "id": "<project ID | hypervisor ID>"
          }
        }
      }
    }
  ]
}
```

where:

- <infrastructure object type> is the type of infrastructure object where the SVM is located. Specify the parameter value:
  - host – for non-OpenStack infrastructures.
  - openstackProject – for OpenStack infrastructures.
- <infrastructure ID> is the ID of the infrastructure in which the SVM is deployed.
- <project ID | hypervisor ID> – depending on the type of infrastructure:
  - ID of the hypervisor on which the SVM is deployed
  - ID of the OpenStack project within which the SVM is deployed.

An SVM removal request creates a DeleteMultipleSvm task (or DeleteMultipleOpenStackSvm task for OpenStack infrastructures). In the "children" element of the task, for each SVM there is a subtask of the DeleteSvm type (or DeleteOpenStackSvm type for OpenStack infrastructures). There can be several subtasks, depending on the number of SVMs in the SVM removal request.

The SVM removal process takes some time. You will need to wait until the task finishes.

To view the current status of a task, execute the following request:

```
GET api/3.0/virtualization/tasks/<task ID>
```

A task is considered completed successfully if the "state" parameter in the response has the value "Completed".

Deleting infrastructure connection settings

To delete the infrastructure connection settings, execute the following request:

```
DELETE /api/3.0/infrastructures/<infrastructure ID>
```

If the request succeeds, a 200 code is returned.