# Не запускается ВМ Hosted Engine из-за проблемы с службой vdsm

## 1. Проблема

Не запускается ВМ HostedEngine. Служба **vdsmd** на хосте не стартует или перезапускается в течении продолжительного времени.

Статус службы **vdsmd** на хосте:

```
systemctl status vdsmd.service
```

```
vdsmd.service - Virtual Desktop Server Manager
Loaded: loaded (/usr/lib/systemd/system/vdsmd.service; enabled; vendor preset:
enabled)
Active: activating (start-pre) since Tue 2023-03-28 14:35:41 MSK; 150ms ago
Process: 133515 ExecStart=/usr/share/vdsm/daemonAdapter -0 /dev/null -1
/dev/null -2 /dev/null /usr/share/vdsm/vdsmd (code=exited, status=1/FAILURE)
Main PID: 133515 (code=exited, status=1/FAILURE); : 133540 (vdsmd_init_comm)
Tasks: 2
CGroup: /system.slice/vdsmd.service
control
/bin/sh /usr/libexec/vdsm/vdsmd_init_common.sh --pre-start
/usr/bin/python2 /usr/libexec/vdsm/wait_for_ipv4s

Mar 28 14:35:41 host1.orionsoft.ru systemd[1]: Starting Virtual Desktop Server
Manager...
Mar 28 14:35:41 host1.orionsoft.ru vdsmd_init_common.sh[133540]: vdsm: Running
mkdirs
Mar 28 14:35:41 host1.orionsoft.ru vdsmd_init_common.sh[133540]: vdsm: Running
configure_coredump
Mar 28 14:35:41 host1.orionsoft.ru vdsmd_init_common.sh[133540]: vdsm: Running
configure_vdsm_logs
Mar 28 14:35:41 host1.orionsoft.ru vdsmd_init_common.sh[133540]: vdsm: Running
wait_for_network
```

```
systemctl status vdsmd.service
```

```
в—Џ vdsmd.service - Virtual Desktop Server Manager
Loaded: loaded (/usr/lib/systemd/system/vdsmd.service; enabled; vendor preset:
enabled)
```

```
Active: active (running) since Tue 2023-03-28 15:47:36 MSK; 7min ago
Process: 174633 ExecStartPre=/usr/libexec/vdsm/vdsmd_init_common.sh --pre-start
(code=exited, status=0/SUCCESS)
Main PID: 174720 (vdsmd)
Tasks: 77
CGroup: /system.slice/vdsmd.service
/usr/bin/python2 /usr/share/vdsm/vdsmd
/usr/libexec/ioprocess --read-pipe-fd 93 --write-pipe-fd 92 --max-threads 10 --
max-queued-requests 10
/usr/libexec/ioprocess --read-pipe-fd 92 --write-pipe-fd 91 --max-threads 10 --
max-queued-requests 10
/usr/libexec/ioprocess --read-pipe-fd 88 --write-pipe-fd 86 --max-threads 10 --
max-queued-requests 10

Mar 28 15:48:45 host1.orionsoft.ru vdsm[174720]: ERROR failed to retrieve Hosted
Engine HA score
Traceback (most recent call last):
File "/usr/lib/python2.7/site-packages/vdsm/host/api.py", line 182, in
_getHaInfo...
Mar 28 15:48:45 host1.orionsoft.ru vdsm[174720]: ERROR failed to retrieve Hosted
Engine HA score
Traceback (most recent call last):
File "/usr/lib/python2.7/site-packages/vdsm/host/api.py", line 182, in
_getHaInfo...
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN File:
/var/lib/libvirt/qemu/channels/478f3d55-44da-45a9-a4c9-b25991c8b0dc.ovirt-guest-
agent.0 already removed
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN Attempting to remove a non
existing network: ovirtmgmt/478f3d55-44da-45a9-a4c9-b25991c8b0dc
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN Attempting to remove a non
existing net user: ovirtmgmt/478f3d55-44da-45a9-a4c9-b25991c8b0dc
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN Attempting to remove a non
existing network: ovirtmgmt/478f3d55-44da-45a9-a4c9-b25991c8b0dc
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN Attempting to remove a non
existing net user: ovirtmgmt/478f3d55-44da-45a9-a4c9-b25991c8b0dc
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN File:
/var/lib/libvirt/qemu/channels/478f3d55-44da-45a9-a4c9-
b25991c8b0dc.org.qemu.guest_agent.0 already removed
Mar 28 15:49:48 host1.orionsoft.ru vdsm[174720]: WARN File: /var/run/ovirt-
vmconsole-console/478f3d55-44da-45a9-a4c9-b25991c8b0dc.sock already removed
Mar 28 15:49:50 host1.orionsoft.ru vdsm[174720]: WARN Attempting to add an
existing net user: ovirtmgmt/478f3d55-44da-45a9-a4c9-b25991c8b0dc
Hint: Some lines were ellipsized, use -l to show in full.
```

В журнале `journalctl -u vdsmd.service` есть следующие сообщения:

```
vdsmd[52920]: VDSM failed to start: Vdsm user could not manage to run sudo
operation: (stderr: sudo: a password is required). Verify sudoer rules
configuration
```

## 2. Решение

Необходимо проверить конфигурацию в файле **/etc/sudoers** на корректность.

Системная учётная запись **vdsm** наделяется расширенными правами с помощью конфигурационных файлов расположенных в каталоге **/etc/sudoers.d/**.

Строка, которая определяет расположение "дельта-файлов" должна выглядеть так:

```
## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includedir /etc/sudoers.d
```

Необходимо обратить внимание, что одиночный символ `#` в данном файле конфигурации, не является комментирующим строку.

# Grafana error "Origin not allowed"

Если при обращении Grafana к источнику данных PostgreSQL возникает ошибка **Origin not allowed**, то необходимо сделать следующее:

1. Открыть конфигурационный файл **/etc/httpd/conf.d/ovirt-engine-grafana-proxy.conf**;
2. В данном файле в блоке **<Location /ovirt-engine-grafana>** нужно добавить строку **ProxyPreserveHost on**. В итоге блок должен выглядеть следующим образом:

```
<Location /ovirt-engine-grafana>
ProxyPass http://127.0.0.1:3000 retry=0 disablereuse=On
ProxyPassReverse http://127.0.0.1:3000/ovirt-engine-grafana
ProxyPreserveHost on
</Location>
```

3. Перезапустить службу **httpd**:

```
systemctl restart httpd
```

# При проверке соединения в окне "Загрузить образ" ошибка "Connection to ovirt-imageio-proxy service has failed"

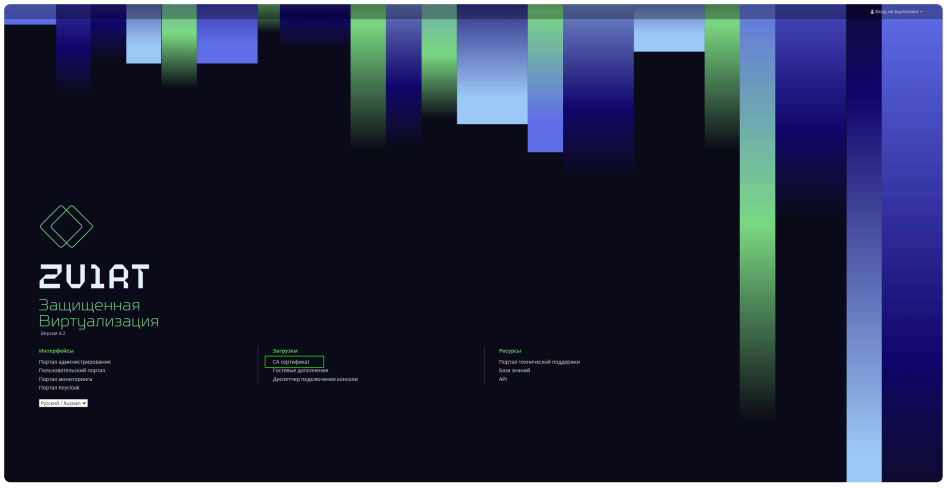
## 1. Вопрос

При проверке соединения в окне **Загрузить образ** ошибка:

```
"Connection to ovirt-imageio-proxy service has failed. Make sure the service is installed, configured, and ovirt-engine certificate is registered as a valid CA in the browser"
```

## 2. Решение

Необходимо установить сертификат менеджера управления (который можно скачать на главной странице) в **Доверенные корневые центра сертификации**.



# TPM модуль не работает

## 1. Проблема

Нерабочий TPM модуль из-за **rngd** сервиса

Статус сервиса:

```
rngd.service - Hardware RNG Entropy Gatherer Daemon
   Loaded: loaded (/usr/lib/systemd/system/rngd.service; enabled)
   Active: failed (Result: exit-code) since Thu 2014-09-11 12:48:40 UTC; 46min
ago
  Process: 28069 ExecStart=/sbin/rngd -f (code=exited, status=1/FAILURE)
 Main PID: 28069 (code=exited, status=1/FAILURE)

 systemd[1]: Started Hardware RNG Entropy Gatherer Daemon.
 rngd[28069]: Unable to open file: /dev/tpm0
 rngd[28069]: can't open any entropy source
rngd[28069]: Maybe RNG device modules are not loaded
systemd[1]: rngd.service: main process exited, code=exited, status=1/FAILURE
 systemd[1]: Unit rngd.service entered failed state.
```

## 2. Причины проблемы:

1. Процессор может не поддерживать **Intel 82802 Firmware Hub** (работает от процессоров **Ivy bridge** и далее)
2. Сервисный файл запущен некорректно
3. Процессор или ОС не работает с **rbrand** в

   ```
   cat /proc/cpuinfo | grep rdrand
   ```

4. Для виртуальных машин проверить состояние rngd можно по этой статье ссылка

## 3. Решения

### 3.1. Решение 1

1. Установить пакет **rng-tools** по ссылке ссылка или `dnf install rng-tools`

2. Проверить наличие **HRWNG** который использует **rngd**

```
[root 07-10 15:54:53 ~]# ll /dev/hwrng
crw-------. 1 root root 10, 183 Jun  8 19:09 /dev/hwrng
```

3. Остановить сервис **rngd** и запустить его:

```
systemctl stop rngd
systemctl start rngd
```

4. Проверить состояние сервиса командой `systemctl status rngd`

```
[root 07-10 15:51:46 ~]# systemctl status rngd
? rngd.service - Hardware RNG Entropy Gatherer Daemon
   Loaded: loaded (/usr/lib/systemd/system/rngd.service; enabled; vendor
preset: enabled)
   Active: active (running) since Mon 2023-07-10 15:50:28 AEST; 1min 19s ago
 Main PID: 807744 (rngd)
    Tasks: 3 (limit: 11155)
   Memory: 3.3M
   CGroup: /system.slice/rngd.service
           ??807744 /usr/sbin/rngd -f --fill-watermark=0 -x pkcs11 -x nist -
x qrypt -D daemon:daemon

Jul 10 15:50:28 censored.com rngd[807744]: Disabling 5: NIST Network Entropy
Beacon (nist)
Jul 10 15:50:28 censored.com rngd[807744]: Disabling 9: Qrypt quantum
entropy beacon (qrypt)
Jul 10 15:50:28 censored.com rngd[807744]: Initializing available sources
Jul 10 15:50:28 censored.com rngd[807744]: [hwrng ]: Initialization Failed
Jul 10 15:50:28 censored.com rngd[807744]: [rdrand]: Initialization Failed
Jul 10 15:50:28 censored.com rngd[807744]: [jitter]: JITTER timeout set to 5
sec
Jul 10 15:50:28 censored.com rngd[807744]: [jitter]: Initializing AES buffer
Jul 10 15:50:32 censored.com rngd[807744]: [jitter]: Enabling JITTER rng
support
Jul 10 15:50:32 censored.com rngd[807744]: [jitter]: Initialized
Jul 10 15:50:32 censored.com rngd[807744]: Process privileges have been
dropped to 2:2
```

Если решение 1 не помогло:

## 3.2. Решение 2

1. Переписываем для **systemd** , чтобы использовался по умолчанию **rngd.service** файл

```
cp /usr/lib/systemd/system/rngd.service   /etc/systemd/system
```

2. Изменяем значения в сервисном файле

   Было:

   ```
   ExecStart=/sbin/rngd -f
   ```

   Должно стать:

   ```
   ExecStart=/sbin/rngd -f -r /dev/urandom -o /dev/random
   ```

3. Перезаписываем демон сервиса

   ```
   systemctl daemon-reload
   ```

4. Запускаем сервис

   ```
   systemctl start rngd.service
   ```

5. Проверяем состояние сервиса

   ```
   systemctl status rngd.service
   ```

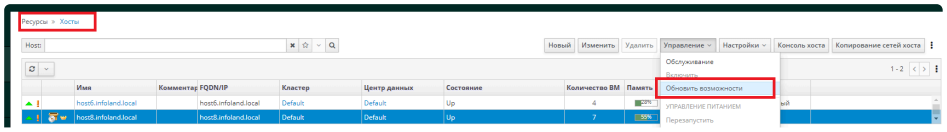
# Список USB-устройств на хосте zVirt не обновляются после добавления - извлечения USB устройства (USB-Flash)

## 1. Вопрос

Список USB-устройств на хосте zVirt не обновляются после добавления/извлечения USB устройства (USB-Flash)

## 2. Ответ

Подключение/извлечение устройства - это аппаратные изменения хоста, мониторинг хоста происходит с определенным периодом. Если необходимо форсировать эту процедуру можно воспользоваться функцией **Обновить возможности**.



# В Chrome не работает HTML5-консоль. Ошибка "WebSocket error Can’t connect to websocket on URL"

## 1. Вопрос

При использовании Chrome не работает HTML5-консоль. Ошибка:

```
WebSocket error: Can't connect to websocket on URL...
```

## 2. Решение

Перевыпустите TLS сертификат "Портала Администрирования" с соответствующим SAN (subject alternative name) или использовать Firefox.

# Аутентификация и безопасность в REST API

## 1. TLS/SSL-сертификация

API zVirt требует защищенного протокола передачи гипертекста (HTTPS) для безопасного взаимодействия с клиентским программным обеспечением, таким как компоненты SDK и CLI. Это включает в себя получение сертификата ЦС, используемого сервером, и его импорт в хранилище сертификатов вашего клиента.

### 1.1. Получение CA сертификата

Вы можете получить CA сертификат от Менеджера управления zVirt и передать его на клиентскую машину одним из следующих способов:

#### 1.1.1. Способ 1

Предпочтительный метод получения CA сертификата — использовать инструмент командной строки `openssl s_client` для выполнения реального рукопожатия TLS с сервером, а затем извлечь сертификаты, которые он представляет. Запустите команду следующим образом:

```
$ openssl s_client \
-connect myengine.example.com:443 \
-showcerts \
< /dev/null
```

Эта команда подключится к серверу и отобразит вывод, подобный следующему:

```
CONNECTED(00000003)
depth=1 C = US, O = Example Inc., CN = myengine.example.com.23416
verify error:num=19:self signed certificate in certificate chain
---
Certificate chain
 0 s:/C=US/O=Example Inc./CN=myengine.example.com
   i:/C=US/O=Example Inc./CN=myengine.example.com.23416
-----BEGIN CERTIFICATE-----
MIIEaTCCA1GgAwIBAgICEAQwDQYJKoZIhvcNAQEFBQAwSTELMAkGA1UEBhMCVVMx
FTATBgNVBAoTDEV4YW1wbGUgSW5jLjEjMCEGA1UEAxMaZW5naW5lNDEuZXhhbXBs
SVlJe7e5FTEtHJGTAeWWM6dGbsFhip5VXM0gfqg=
-----END CERTIFICATE-----
```

```
 1 s:/C=US/O=Example Inc./CN=myengine.example.com.23416
   i:/C=US/O=Example Inc./CN=myengine.example.com.23416
-----BEGIN CERTIFICATE-----
MIIDxjCCAq6gAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwSTELMAkGA1UEBhMCVVMx
FTATBgNVBAoTDEV4YW1wbGUgSW5jLjEjMCEGA1UEAxMaZW5naW5lNDEuZXhhbXBs
Pkyg1rQHR6ebGQ==
-----END CERTIFICATE-----
```

Текст между маркерами `-----BEGIN CERTIFICATE-----` и `-----END CERTIFICATE-----` показывает сертификаты, представленные сервером. Первый — это сертификат самого сервера, а последний — CA сертификат. Скопируйте CA сертификат вместе с метками в файл **ca.crt**. Результат должен выглядеть так:

```
-----BEGIN CERTIFICATE-----
MIIDxjCCAq6gAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwSTELMAkGA1UEBhMCVVMx
FTATBgNVBAoTDEV4YW1wbGUgSW5jLjEjMCEGA1UEAxMaZW5naW5lNDEuZXhhbXBs
Pkyg1rQHR6ebGQ==
-----END CERTIFICATE-----
```

> Это самый надежный способ получить CA сертификат, используемый сервером. Остальные описанные здесь способы сработают в большинстве случаев, но они не дадут правильный CA сертификат, если он был вручную заменен администратором сервера.

### 1.1.2. Способ 2

Если вы не можете использовать описанный выше метод `openssl s_client`, вы можете вместо этого использовать инструмент командной строки для загрузки CA сертификата из Менеджера управления zVirt.

Примеры инструментов командной строки включают `curl` и `wget`, оба из которых доступны на нескольких платформах.

При использовании `curl`:

```
$ curl \
--output ca.crt \
'http://myengine.example.com/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA'
```

При использовании wget:

```
$ wget \
--output-document ca.crt \
'http://myengine.example.com/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA'
```

### 1.1.3. Способ 3

С помощью веб-браузера перейдите к сертификату, расположенному по адресу:

```
https://myengine.example.com/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA
```

В зависимости от выбранного браузера сертификат загружается или импортируется в хранилище ключей браузера.

- Если браузер загружает сертификат: сохраните файл как **ca.crt**.
- Если браузер импортирует сертификат: экспортируйте его из параметров сертификации браузера и сохраните как **ca.crt**.

### 1.1.4. Способ 4

Войдите в Менеджер управления, экспортируйте сертификат из хранилища доверенных сертификатов и скопируйте его на свой клиентский компьютер.

1. Войдите на машину с Менеджером управления как *root*.
2. Экспортируйте сертификат из хранилища доверенных сертификатов с помощью утилиты `keytool`:

   ```
   keytool \
   -keystore /etc/pki/ovirt-engine/.truststore \
   -storepass mypass \
   -exportcert \
   -alias cacert \
   -rfc \
   -file ca.crt
   ```

   Это создает файл сертификата с именем **ca.crt**.
3. Скопируйте сертификат на клиентскую машину с помощью команды `scp`:

   ```
   $ scp ca.crt myuser@myclient.example.com:/home/myuser/.
   ```

Каждый из этих методов приводит к созданию файла сертификата с именем **ca.crt** на клиентском компьютере. Затем вы должны импортировать этот файл в хранилище сертификатов клиента.

## 1.2. Импорт сертификата в клиент

Импорт сертификата клиенту зависит от того, как клиент хранит и интерпретирует сертификаты. Дополнительную информацию об импорте сертификата см. в документации

вашего клиента.

# 2. Аутентификация

Любой пользователь с учетной записью Менеджера управления имеет доступ к API. Все запросы должны быть аутентифицированы с использованием **OAuth** или обычной аутентификации, как описано ниже.

## 2.1. OAuth-аутентификация

Предпочтительным механизмом аутентификации является OAuth 2.0, как описано в RFC 6749.

**OAuth** — это сложный протокол с несколькими механизмами получения токенов авторизации и доступа. Для использования с API поддерживается только предоставление учетных данных владельца ресурса, как описано в разделе 4.3 RFC 6749.

Сначала необходимо получить **токен**, отправив имя пользователя и пароль в службу единого входа Менеджера управления:

```
POST /ovirt-engine/sso/oauth/token HTTP/1.1
Host: myengine.example.com
Content-Type: application/x-www-form-urlencoded
Accept: application/json
```

Тело запроса должно содержать параметры `grant_type`, `scope`, `username` и `password`:

**Таблица 1. Параметры запроса токена OAuth**

| Имя | Значение |
|---|---|
| grant_type | password |
| scope | ovirt-app-api |
| username | api-user@internalsso |
| password | mypassword |

Эти параметры должны быть представлены в URL кодировке. Например, символ `@` в имени пользователя должен быть закодирован как `%40`. Результирующее тело запроса будет примерно таким:

```
grant_type=password&scope=ovirt-app-
api&username=admin%40zvirt%40internalsso&password=mypassword
```

Для работы с сервисами REST API настоятельно рекомендуем создать отдельного пользователя с правами, достаточными для выполнения необходимых операций.

Здесь и далее в качестве `username` используется **api-user@internalsso**.

Параметр `scope` описан как необязательный в OAuth RFC, но при использовании его с API он является обязательным, и его значение должно быть `ovirt-app-api`.

Если имя пользователя и пароль верны, служба единого входа Менеджера управления ответит документом JSON, подобным этому:

```
{
  "access_token": "fqbR1ftzh8wBCviLxJcYuV5oSDI=",
  "token_type": "bearer",
  "scope": "...",
  ...
}
```

Для целей аутентификации API единственной подходящей парой имя/значение является `access_token`. Ни в коем случае не манипулируйте этим; используйте его точно так, как это предусмотрено службой единого входа.

После получения токена его можно использовать для выполнения запросов к API, включив его в заголовок HTTP `Authorization` и используя схему `Bearer`. Например, чтобы получить список виртуальных машин, отправьте такой запрос:

```
GET /ovirt-engine/api/vms HTTP/1.1
Host: myengine.example.com
Accept: application/xml
Authorization: Bearer fqbR1ftzh8wBCviLxJcYuV5oSDI=
```

Токен можно использовать несколько раз для нескольких запросов, но в конечном итоге срок его действия истечет. По истечении этого срока сервер отклонит запрос с кодом ответа HTTP **401**:

```
HTTP/1.1 401 Unauthorized
```

В этом случае требуется новый токен, так как служба единого входа Менеджера управления в настоящее время не поддерживает обновление токенов. Новый токен можно запросить тем же способом, который описан выше.

## 2.2. Базовая аутентификация

Базовая аутентификация поддерживается только для обратной совместимости; она устарела и будет удалена в будущем.

Базовая аутентификация не поддерживается в среде zVirt с Keycloak.

Каждый запрос использует базовую аутентификацию HTTP для кодирования учетных данных. Если запрос не включает соответствующий заголовок `Authorization`, сервер отправляет ответ `401 Authorization Required`:

```
HEAD /ovirt-engine/api HTTP/1.1
Host: myengine.example.com

HTTP/1.1 401 Authorization Required
```

Запрос выдается с заголовком `Authorization` для указанной области. Закодируйте соответствующий домен и имя пользователя в предоставленных учетных данных с помощью соглашения `username@domain:password`.

В следующей таблице показан процесс кодирования учетных данных в **Base64**.

**Таблица 2. Кодирование учетных данных для доступа к API**

| Элемент | Значение |
|---|---|
| Имя пользователя | admin@zvirt |
| Домен | internalsso |
| Пароль | mypassword |
| Незакодированные учетные данные | api-user@internalsso:mypassword |
| Учетные данные в кодировке Base64 | YWRtaW5AaW50ZXJuYWw6bXlwYXNzd29yZA== |

Укажите учетные данные в кодировке **Base64**, как показано ниже:

```
HEAD /ovirt-engine/api HTTP/1.1
Host: myengine.example.com
Authorization: Basic YWRtaW5AaW50ZXJuYWw6bXlwYXNzd29yZA==

HTTP/1.1 200 OK
```

Базовая аутентификация включает потенциально конфиденциальную информацию, такую как пароли, отправляемые в виде обычного текста. Для API требуется безопасный протокол передачи гипертекста (HTTPS) для шифрования на транспортном уровне простых текстовых запросов.

Некоторые библиотеки Base64 разбивают результат на несколько строк и заканчивают каждую строку символом новой строки. Это ломает заголовок и вызывает ошибочный запрос. Для заголовка `Authorization` требуются закодированные учетные данные в одной строке заголовка.

## 2.3. Сеансы аутентификации

API также обеспечивает поддержку сеанса аутентификации. Отправьте первоначальный запрос с данными аутентификации, а затем отправьте все последующие запросы, используя файл cookie сеанса для аутентификации.

***Запрос аутентифицированного сеанса***

1. Отправьте запрос с заголовками `Authorization` и `Prefer: persistent-auth`:

   ```
   HEAD /ovirt-engine/api HTTP/1.1
   Host: myengine.example.com
   Authorization: Basic YWRtaW5AaW50ZXJuYWw6bXlwYXNzd29yZA==
   Prefer: persistent-auth

   HTTP/1.1 200 OK
   ...
   ```

   Он возвращает ответ со следующим заголовком:

   ```
   Set-Cookie: JSESSIONID=5dQja5ubr4yvI2MM2z+LZxrK; Path=/ovirt-engine/api;
   Secure
   ```

   Обратите внимание на значение `JSESSIONID=`. В этом примере значение равно `5dQja5ubr4yvI2MM2z+LZxrK`.

2. Отправляйте все последующие запросы с заголовками `Prefer: persistent-auth` и `Cookie` с соответствующим значением `JSESSIONID=`. Заголовок `Authorization` больше не нужен при использовании аутентифицированного сеанса.

   ```
   HEAD /ovirt-engine/api HTTP/1.1
   Host: myengine.example.com
   Prefer: persistent-auth
   Cookie: JSESSIONID=5dQja5ubr4yvI2MM2z+LZxrK

   HTTP/1.1 200 OK
   ...
   ```

3. Когда сеанс больше не требуется, выполните запрос к серверу без заголовка `Prefer: persistent-auth`.

```
HEAD /ovirt-engine/api HTTP/1.1
Host: myengine.example.com
Authorization: Basic YWRtaW5AaW50ZXJuYWw6bXlwYXNzd29yZA==

HTTP/1.1 200 OK
...
```