ООО «Астра Консалтинг»

Астра ИС МД (Инфраструктурные Сервисы)

Модуль Astra Секреты

Документация, содержащая информацию, необходимую для эксплуатации экземпляра программного обеспечения, предоставленного для проведения экспертной проверки

Москва, 2026

Содержание

1. Введение

Документ содержит описание функциональных характеристик экземпляра программного обеспечения «Астра Секреты» (Модуль управления секретами), предоставленного для проведения экспертной проверки.

Система предоставляет REST API для управления секретами, политиками, аутентификацией и конфигурацией. Все операции доступны через HTTP-запросы с использованием токена аутентификации.

Базовый URL API: http://158.160.196.133:9200
Web UI: http:// 158.160.196.133:9200/ui

2. Аутентификация в Системе через REST API

Аутентификация в REST API осуществляется с использованием токена (Vault Token). Токен передаётся в HTTP-заголовке X-Vault-Token. Пример выполнения запроса через утилиту curl:

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/health \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

В случае неуспешной аутентификации возвращается ошибка 403 Forbidden:

```
HTTP/1.1 403 Forbidden

{"errors":["permission denied"]}
```

При успешной аутентификации возвращается результат обработки запроса, например:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"initialized":true,"sealed":false,"standby":false,
 "server_time_utc":1735689600,"version":"2.1.0",
 "cluster_name":"astra-secrets-cluster","cluster_id":"..."}
```

Также поддерживается аутентификация через заголовок Authorization с использованием LDAP, OIDC и других методов (см. п. 9).

3. Управление секретами (KV)

Секреты хранятся в модуле KV (Key-Value). По умолчанию используется KV версии 2 (с версионированием) по пути secret/.

3.1. Запись секрета

Для записи секрета необходимо выполнить POST-запрос:

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/secret/data/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn " \
  --header "Content-Type: application/json" \
  --data '{"data": {"username": "appuser", "password": "s3cr3t!",
           "db_host": "postgres.internal", "db_port": "5432"},
    "options": {"cas": 0}}'
```

, где

secret/data/myapp/config — путь к секрету (префикс data/ для KV v2);
data — объект с данными секрета;
cas=0 — Check-And-Set (0 = создавать только если не существует).

В случае успешного выполнения возвращается код 200 OK и JSON с метаданными:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"request_id":"xxxx-xxxx-xxxx","lease_id":"",
 "data":{"version":1,"destroyed":false},
 "wrap_info":null,"warnings":null,"auth":null}
```

3.2. Чтение секрета

Для чтения секрета необходимо выполнить GET-запрос:

```
curl --request GET \
  --url http://158.160.196.133:9200:8200/v1/secret/data/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
HTTP/1.1 200 OK
Content-Type: application/json

{"data":{"data":{"db_host":"postgres.internal",
  "db_port":"5432","password":"s3cr3t!","username":"appuser"},
  "metadata":{"created_time":"2026-01-15T10:30:00Z",
    "deletion_time":"","destroyed":false,
    "version":1}}}
```

Если секрет не найден, возвращается ошибка 404:

```
HTTP/1.1 404 Not Found
```

3.3. Список секретов

Для получения списка секретов по пути необходимо выполнить LIST-запрос:

```
curl --request LIST \
  --url http://158.160.196.133:9200/v1/secret/metadata/myapp \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
HTTP/1.1 200 OK

{"data":{"keys":["config","credentials","tls"]}}
```

3.4. Удаление секрета

Мягкое удаление (с возможностью восстановления):

```
curl --request DELETE \
  --url http://158.160.196.133:9200/v1/secret/data/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Полное уничтожение (без возможности восстановления):

```
curl --request DELETE \
  --url http://158.160.196.133:9200/v1/secret/destroy/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"versions": [1, 2, 3]}'
```

3.5. Просмотр истории версий

Для просмотра метаданных и истории версий секрета:

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/secret/metadata/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ содержит информацию о всех версиях, времени создания и статусе удаления.

3.6. Откат к предыдущей версии

Для отката к предыдущей версии секрета:

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/secret/undelete/myapp/config \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"versions": [2]}'
```

4. Управление PKI (сертификатами)

Модуль PKI обеспечивает генерацию и управление X.509-сертификатами.

4.1. Создание Root CA

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/pki/root/generate/internal \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"common_name": "Astra Secrets Root CA",
           "ttl": "87600h",
           "key_type": "rsa",
           "key_bits": 4096}'
```

Ответ содержит сертификат Root CA и серийный номер.

4.2. Выпуск сертификата

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/pki/issue/web-server \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"common_name": "app.example.com",
           "ttl": "720h",
           "alt_names": "app.example.com,www.example.com",
           "ip_sans": "192.168.1.100"}'
```

Ответ содержит certificate, private_key, ca_chain, serial_number.

4.3. Отзыв сертификата

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/pki/revoke \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"serial_number": "xxxx-xxxx-xxxx-xxxx"}'
```

4.4. Просмотр CRL

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/pki/crl \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

5. Управление Database (динамические учётные данные)

Модуль Database генерирует временные учётные данные для подключения к БД с автоматическим отзывом по истечении TTL.

5.1. Настройка подключения к БД

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/database/config/postgresql \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
```

```
  --header "Content-Type: application/json" \
  --data '{"plugin_name": "postgresql-database-plugin",
    "allowed_roles": ["readonly"],
    "connection_url": "postgresql://{{username}}:{{password}}@pg:5432/appdb",
    "username": "admin",
    "password": "<DB_PASSWORD>"}'
```

5.2. Генерация динамического credential

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/database/creds/readonly \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
HTTP/1.1 200 OK

{"data":{"username":"v-token-readonly-xxxx",
  "password":"yyyyyyyyyyyy"}}
```

Учётные данные автоматически отзываются по истечении TTL (по умолчанию 1 час).

5.3. Просмотр активных lease

```
curl --request LIST \
  --url
http://158.160.196.133:9200/v1/sys/leases/lookup/database/creds/readonly \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

5.4. Отзыв lease

```
curl --request PUT \
  --url http://158.160.196.133:9200/v1/sys/leases/revoke \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"lease_id": "database/creds/readonly/xxxxxxxx"}'
```

6. Управление Transit (шифрование как сервис)

Модуль Transit предоставляет шифрование данных без их хранения в Системе.

6.1. Создание ключа шифрования

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/transit/keys/my-key \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"type": "aes256-gcm96", "exportable": false}'
```

6.2. Шифрование данных

Данные должны быть закодированы в Base64:

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/transit/encrypt/my-key \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"plaintext": "bXkgc2VjcmV0IGRhdGE="}'
```

Ответ:

```
{"data":{"ciphertext":"vault:v1:xxxxxxxxxxxxxxxxxx"}}
```

6.3. Расшифрование данных

```
curl --request POST \
```

```
  --url http://158.160.196.133:9200/v1/transit/decrypt/my-key \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"ciphertext": "vault:v1:xxxxxxxxxxxxxxxxxx"}'
```

Ответ:

```
{"data":{"plaintext":"bXkgc2VjcmV0IGRhdGE="}}
```

6.4. Ротация ключа

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/transit/keys/my-key/rotate \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

После ротации старые версии ключа остаются доступными для расшифрования, но новое шифрование использует новую версию ключа.

7. Управление SSH

7.1. Выпуск SSH-ключа

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/ssh/issue/otp-key-role \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"public_key": "ssh-rsa AAAA...", "ttl": "1h"}'
```

7.2. Подпись SSH-ключа

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/ssh/sign/my-role \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"public_key": "ssh-rsa AAAA...", "ttl": "30m",
           "valid_principals": "ubuntu,admin",
           "cert_type": "user"}'
```

8. Управление политиками доступа

8.1. Создание политики

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/policies/acl/developer \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"policy": "path \"secret/data/myapp/*\" { capabilities =
[\"create\",\"read\",\"update\",\"delete\",\"list\"] }"}'
```

8.2. Просмотр списка политик

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/policies/acl \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
{"policies":["default","root","developer","admin"]}
```

8.3. Просмотр содержания политики

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/policies/acl/developer \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

8.4. Удаление политики

```
curl --request DELETE \
  --url http://158.160.196.133:9200/v1/sys/policies/acl/developer \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

9. Управление аутентификацией (Auth Methods)

9.1. Список включённых методов

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/auth \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
{"token/":{"accessor":"auth_token_xxx","type":"token"},
 "ldap/":{"accessor":"auth_ldap_xxx","type":"ldap"},
 "approle/":{"accessor":"auth_approle_xxx","type":"approle"}}
```

9.2. Включение/отключение метода

Включение метода LDAP:

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/auth/ldap \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"type": "ldap"}'
```

Отключение метода:

```
curl --request DELETE \
  --url http://158.160.196.133:9200/v1/sys/auth/ldap \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

10. Управление Namespaces

10.1. Создание Namespace

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/namespaces/team-alpha \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"path": "team-alpha/"}'
```

10.2. Список Namespace

```
curl --request LIST \
  --url http://158.160.196.133:9200/v1/sys/namespaces \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

10.3. Удаление Namespace

```
curl --request DELETE \
  --url http://158.160.196.133:9200/v1/sys/namespaces/team-alpha \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Примечание: удаление Namespace невозможно, если в нём есть активные секреты или политики. Необходимо сначала очистить Namespace.

11. Аудит и мониторинг

11.1. Просмотр аудит-логов

Для просмотра аудит-логов (при включённом File-аудите) выполните на сервере:

```
tail -f /opt/astra-secrets/logs/audit.log
```

Каждая запись аудит-лога содержит:

type — тип запроса (request/response);
auth — информация об аутентификации (кто обратился);
request — детали запроса (путь, операция, данные);
response — детали ответа (код, данные);
error — информация об ошибке (если есть).

Пример записи аудит-лога:

```
{"type":"request","auth":{"accessor":"xxxx","client_token":"s.xxxx",
  "display_name":"admin","entity_id":"xxxx","token_type":"service"},
 "request":{"client_token":"s.xxxx","operation":"read",
  "path":"secret/data/myapp/config","data":null},
 "response":null}
```

11.2. Включение аудит-устройств

Включение файлового аудит-лога:

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/audit/file-audit \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"type": "file", "options": {"file_path": "/opt/astra-
secrets/logs/audit.log"}}'
```

Просмотр активных аудит-устройств:

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/audit \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

11.3. Получение метрик Prometheus

Для получения метрик в формате Prometheus:

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/metrics \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Основные метрики:

```
vault_core_handle_request_count 1234
vault_token_create_count 567
vault_lease_create_count 890
vault_audit_log_request_count 4567
vault_audit_log_response_count 4567
vault_runtime_alloc_bytes 104857600
vault_runtime_sys_bytes 52428800
vault_core_unsealed 1
```

12. Управление кластером (HA)

12.1. Просмотр статуса кластера

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/leader \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

Ответ:

```
{"ha_enabled":true,"is_self":true,"leader_address":"http://node1:8200",
 "leader_cluster_address":"https://node1:8201",
 "active_time":"2026-01-15T10:00:00Z"}
```

12.2. Просмотр списка узлов

```
curl --request GET \
  --url http://158.160.196.133:9200/v1/sys/storage/raft/configuration \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn"
```

12.3. Удаление узла из кластера

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/storage/raft/remove-peer \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"server_id": "node3"}'
```

13. Управление Lease и TTL

13.1. Продление lease

```
curl --request PUT \
  --url http://158.160.196.133:9200/v1/sys/leases/renew \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"lease_id": "database/creds/readonly/xxxx", "increment": 3600}'
```

13.2. Массовый отзыв lease

```
curl --request PUT \
  --url http://158.160.196.133:9200/v1/sys/leases/revoke-prefix \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --header "Content-Type: application/json" \
  --data '{"prefix": "database/creds/readonly"}'
```

14. Резервное копирование и восстановление

14.1. Создание snapshot (Raft)

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/storage/raft/snapshot \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --output /opt/backup/snapshot-$(date +%Y%m%d).raft
```

14.2. Восстановление из snapshot

```
curl --request POST \
  --url http://158.160.196.133:9200/v1/sys/storage/raft/snapshot/restore \
  --header "X-Vault-Token: s.9xWwwHuteTdPycdCcDirVMyn" \
  --data-binary @/opt/backup/snapshot-20260115.raft
```

15. Проверка работоспособности Системы

Для проверки работоспособности Системы выполните следующие шаги:

1. Проверка статуса здоровья: curl http://158.160.196.133:9200/v1/sys/health — ожидается HTTP 200 и sealed=false;
2. Проверка лидера кластера: curl http://158.160.196.133:9200/v1/sys/leader — ожидается информация о лидере;
3. Проверка Secrets Engines: curl http://158.160.196.133:9200/v1/sys/mounts — ожидается список включённых движков;
4. Проверка аутентификации: curl http://158.160.196.133:9200/v1/sys/auth — ожидается список методов;
5. Запись и чтение тестового секрета: bao kv put secret/test key=value && bao kv get secret/test.

16. Работа через CLI (bao)

Утилита командной строки bao предоставляет удобный интерфейс для управления Системой. Основные команды:

Аутентификация:

```
export VAULT_ADDR="http://158.160.196.133:9200"
export VAULT_TOKEN="s.9xWwwHuteTdPycdCcDirVMyn"
```

Секреты (KV):

```
bao kv put secret/myapp/config username=user password=pass
bao kv get secret/myapp/config
bao kv list secret/myapp/
bao kv delete secret/myapp/config
```

Статус Системы:

```
bao status
bao operator unseal <KEY>
```

Политики:

```
bao policy write mypolicy policy.hcl
bao policy list
bao policy read mypolicy
```

Auth Methods:

```
bao auth enable ldap
bao auth list
```

Secrets Engines:

```
bao secrets enable kv-v2
bao secrets enable pki
bao secrets enable database
bao secrets enable transit
bao secrets list
```

Аудит:

```
bao audit enable file file_path=/var/log/vault-audit.log
bao audit list
```

Кластер:

```
bao operator raft list-peers
bao operator raft snapshot save /opt/backup/snapshot.snap
```