EDM Platform V Synapse Event-domain management

EDMS Сервис управления событийными доменами

Руководство по установке

ОГЛАВЛЕНИЕ

Руководство по установке

Термины и определения

Термин/Аббревиатура	Определение
EDMS	Четырехбуквенный код программного компонента. Event-domain management — сервис управления событийными доменами из состава продукта Platform V Synapse Event-domain management
ETCD	Event Discovery (ED) — высоконадежное распределенное хранилище топологии типа «ключ-значение» с открытым исходным кодом
EVTD	Четырехбуквенный код программного компонента. Event Transfer Service — сервис передачи событий из состава программного продукта Platform V Synapse Event Transfer Service, основанный на технологиях Apache Kafka
EVTP	Четырехбуквенный код программного компонента. Streaming Event Processing — сервис обработки событий, из состава программного продукта Platform V Synapse Streaming Event Processing, основанный на технологиях Apache Flink
mTLS	Mutual TLS — протокол взаимной TLS-аутентификации
SEDR	Четырехбуквенный код программного компонента. Event Replication Service — сервис для репликации событий между событийными доменами, в том числе

Термин/Аббревиатура	Определение
	при нахождении доменов в разных сетевых зонах, из состава программного продукта Platform V Synapse Event Replication Service, основанный на технологиях Kafka Connect
SSL	Secure Sockets Layer — уровень защищенных сокетов
LDAP	Lightweight Directory Access Protocol — это открытый и кроссплатформенный протокол, используемый для аутентификации служб каталогов
DN	Distinguished Name — Уникальное имя сертификата, должно быть уникальным в пределах дерева. В DName описывается содержимое атрибутов в дереве (так называемый путь навигации), требуемое для доступа к конкретной записи ИЛИ базовой (стартовой) записи поиска. DName состоит из серии RDN (Relative Distinguished Names, относительных уникальных имен), определяемых путем перемещения вверх по дереву в направлении его корневой записи (суффикса или базовой записи), и записываемых слева направо
БД	База данных PostgreSQL/ Platform V Pangolin SE (версия 4.5.)
ОС	Операционная система
УЦ	Удостоверяющий центр

Системные требования

Для корректной работы EDMS допускается использование браузеров:

1. Яндекс браузер версии 19 и выше.
2. Internet Explorer версии 11.
3. Microsoft Edge версии 14 и выше.

4. Google Chrome версии 49 и выше.
5. Safari версии 10 и выше.

Требования к серверам:

- EDMS:
    - Сервер 4-8-150 Гбайт.
    - ОС Linux с версией kernel не ниже 3.10.0-327.
    - На сервере установлены дистрибутивы:
        - OpenJDK 11.
        - Unzip.
- ETCD:
    - Сервер 4-8-150 Гбайт (один сервер или от трех серверов для кластерной конфигурации).
    - ОС Linux с версией kernel не ниже 3.10.0-327.
    - На сервере установлен Unzip.

- БД:

При необходимости строить конфигурацию с использованием БД, выполните следующие действия:

1. Установите отдельный экземпляр PostgreSQL (версия не ниже 11.7) только для нужд использования БД EDMS.
2. Разверните базу в PostgreSQL и создайте схему с названием «edms».
3. Создайте технологического пользователя для доступа к БД.
4. Настройте подключение к БД с использованием mTLS.
5. Предоставьте доступ для пользователя БД и сертификата доступа к БД.

- EVTD:
  При необходимости строить конфигурацию с использованием нескольких сетевых контуров (нескольких экземпляров EDMS), и настройки репликации между ними, выполните следующие действия:

    . Установите дистрибутив продукта EVTD.
    i. Создайте в нем служебные топики для репликации.
    ii. Выдайте права сертификату SSL.

Пререквизиты установки

- Ansible 2.9.
- Необходимо обеспечить доступ к нужным для использования элементам файловой системы.

При использовании Jenkins (опциональный способ), дополнительно:

- BitBucket (с созданным в нем проектом для размещения ролей и файлов настроек EDMS);
- Nexus (с размещенным в нем дистрибутивом EDMS);
- Jenkins (с нужными для использования сущностями: credentials для Bitbucket, vault, Nexus и задание Jenkins для установки);

Настройки для установки

- Задать настройки для процесса сборки дистрибутива по заявкам EDMS. Сборку и разворачивание дистрибутивов по заявкам можно осуществлять с помощью различных механизмов: прямой деплой или сборка и установка дистрибутивов с помощью Jenkins. При прямом деплое сборка дистрибутивов не выполняется, сразу происходит установка дистрибутива.
    - Если для развертывания дистрибутивов будет использоваться механизм прямого развертывания, предварительно его настройте:
        . Настройте сетевое взаимодействие между серверами EDMS и EVTD, SEDR.
        a. Настройте доступы к EVTD, SEDR.
        b. Добавьте сертификаты доступа к доверенным сертификатам.
        c. Пропишите настройки прямого развертывания в конфигурационном файле `application.yml`:
    - ```
      deploy:
        directDeploy:
          timeout: 5000
          properties:
            "[ssl.keystore.location]": <абсолютный путь до keystore>
            "[ssl.keystore.password]": <зашифрованный пароль>
            "[ssl.truststore.location]": <абсолютный путь до truststore>
            "[ssl.truststore.password]": <зашифрованный пароль>
      ```
    - Также возможно настроить сборку дистрибутивов с помощью Jenkins. В этом случае необходимо заполнить настройки в конфигурационном файле **application.yml**:
    - ```
      createDistr:
          token: <зашифрованный токен для доступа ТУЗ EDMS к Jenkins>
          createDistributiveUrl:<url задания Jenkins по сборке дистрибутива>
          login: <логин технологической записи EDMS для доступа к Jenkins с целью запуска задания по сборке дистрибутива>
      ```
- Возможно подключение аудита с помощью продукта Platform V Audit SE.
  Для использования в качестве системы аудита продукта Platform V Audit SE необходимо получить TLS-сертификаты для подключения к системе, а также прописать соответствующие настройки аудита в конфигурационном файле **application.yml**:

```
audit:
  url: <URL эндпойнта Platform V Audit SE>
  class: <класс, с помощью которого идет отправка в аудит>
  ssl:
    active: false
    keystore: <абсолютный путь до keystore>
    keystore-password: <зашифрованный пароль>
    truststore: <абсолютный путь до truststore>
    truststore-password: <зашифрованный пароль>
  tsaudit:
    metamodel: <абсолютный путь до метамодели>
    event-postfix:
    metamodel-postfix:
    redelivery:
      file:
```

При первом запуске EDMS указанная метамодель (параметр *tsaudit.metamodel*) регистрируется в Platform V Audit SE. При изменении версии метамодели необходимо поменять файл метамодели в указанном каталоге и перезапустить EDMS для ее перерегистрации в Platform V Audit SE. Метамодель событий передается в формате JSON.

- Возможны разные настройки способов аутентификации в EDMS:

1. С помощью LDAP.

При использовании LDAP необходимо предварительно его установить. При использовании защищенного подключения необходимо предварительно получить TLS-сертификаты для подключения к LDAP.

Настройки LDAP задаются в файле `application.yml` в блоке `authentication`:

```
  authentication:
    type: ldap
    url: <URL подключения к LDAP>
    port: 389
    ssl:
      active: false
      keystore: <абсолютный путь до keystore>
      keystore-password: <зашифрованный пароль>
      truststore: <абсолютный путь до truststore>
      truststore-password: <зашифрованный пароль>
    ldap:
      controller: CAB-VSP-DC00005
      base: <перечень DC через запятую>
      fields:
        displayName: fieldsOne
        employeeID: fieldsTwo
        mail: fieldThree
        thumbnailPhoto: fieldFour
```

- Пользователь вводит логин и пароль в форме аутентификации EDMS.
- Запрос отправляется в систему LDAP, проверяется наличие пары логин-пароль. В случае наличия данных в LDAP, происходит успешная аутентификация пользователя.

1. С помощью БД.

   Настройки аутентификации через БД задаются в файле `application.yml` в блоке `authentication`:

```
authentication:
  type: database
  url: <url расположения БД>
  port: <порт подключения к БД>
  ssl:
    active: <false or true - в зависимости от использования подключения по ssl>
    keystore: <абсолютный путь до keystore>
    keystore-password: <зашифрованный пароль от keystore>
    truststore: <абсолютный путь до truststore>
    truststore-password: <зашифрованный пароль от truststore>
  database:
    type: <тип БД, например, postgresql>
    instance: <Инстанс БД, например, cepdb>
    schema: <имя схемы БД>
    table: <имя таблицы, по которой производится идентификация пользователя>
    user: <пользователь подключения к БД>
    password: <зашифрованный пароль подключения к БД>
```

В данном случае имена пользователей и пароли хранятся в БД EDMS.

- Пользователь вводит логин и пароль в форме аутентификации EDMS.
- Запрос отправляется в БД EDMS, проверяется наличие пары логин-пароль в таблице, указанной в настройках файла `application.yml`. В случае наличия данных в БД, происходит успешная аутентификация пользователя.

Создание сервиса обслуживания на сервере размещения EDMS

Для автоматического перезапуска EDMS в случае перезагрузки сервера и корректной работы скриптов Ansible с сервисом выполните следующие действия:

1. Создайте на сервере установки EDMS пользователя с логином `emc`.
2. Выдайте права на чтение и запись пользователю `emc` в директории установки, журналирования и хранения данных EDMS.
3. Выдайте права пользователю (далее `emc`), от имени которого будет осуществляться работа с EDMS, на запуск, остановку и редактирование сервиса.
4. Под пользователем `root` добавьте строки в файл `/etc/sudoers`:

```
emc ALL= NOPASSWD: /bin/systemctl start emc
emc ALL= NOPASSWD: /bin/systemctl stop emc
emc ALL= NOPASSWD: /bin/systemctl status emc
emc ALL= NOPASSWD: /bin/systemctl restart emc
emc ALL= NOPASSWD: /bin/systemctl enable emc
emc ALL= NOPASSWD: /bin/systemctl disable emc
emc ALL= NOPASSWD: /bin/systemctl daemon-reload
emc ALL= NOPASSWD: /bin/sudoedit /etc/systemd/system/emc.service
```

13. Под пользователем `emc` выполните следующие действия:
    a. Создайте файл сервиса. Для этого выполните команду `sudo /bin/sudoedit /etc/systemd/system/emc.service`.
    i. Будет создан файл `emc.service`, наполните его следующим содержимым:

```
[Unit]
Description=EMC service
After=network.target local-fs.target

[Service]
Type=simple
User=emc
Group=emc
LimitNOFILE=65536
WorkingDirectory=/opt/control-plane/
ExecStart=/opt/control-plane/bin/emc run --
spring.config.location=file:///opt/control-plane/conf/application.yml
Restart=on-failure
RestartSec=30

[Install]
WantedBy=multi-user.target
```

    Здесь `/opt/control-plane/` — абсолютный путь до директории установки EDMS, зависит от параметров установки.
    xviii. Выполните команду `sudo systemctl daemon-reload`.

Создание сервиса обслуживания ETCD на серверах размещения ETCD

Для автоматического перезапуска ETCD в случае перезагрузки сервера и корректной работы скриптов Ansible с сервисом выполните следующие действия:

1. Создайте на сервере установки EDMS пользователя с логином `etcd`.
2. Выдайте права на чтение и запись пользователю `etcd` в директории установки и хранения данных ETCD.
3. Выдайте права пользователю (далее — `etcd`), от имени которого будет осуществляться работа с EDMS, на запуск, остановку и редактирование сервиса.
4. Под пользователем `root` добавьте строки в файл `/etc/sudoers`:

```
etcd ALL= NOPASSWD: /bin/systemctl start etcd
etcd ALL= NOPASSWD: /bin/systemctl stop etcd
etcd ALL= NOPASSWD: /bin/systemctl status etcd
etcd ALL= NOPASSWD: /bin/systemctl restart etcd
etcd ALL= NOPASSWD: /bin/systemctl enable etcd
etcd ALL= NOPASSWD: /bin/systemctl disable etcd
etcd ALL= NOPASSWD: /bin/systemctl daemon-reload
etcd ALL= NOPASSWD: /bin/sudoedit /etc/systemd/system/etcd.service
```

13. Под пользователем `etcd` выполните следующие действия:

. Создайте файл сервиса. Для этого выполните команду `sudo /bin/sudoedit /etc/systemd/system/etcd.service`.

i. При выполнении команды будет создан файл **etcd.service**, наполните его следующим содержимым:

```
ii.   [Unit]
iii.  Description=ETCD service
iv.   Requires=network.target remote-fs.target
v.    After=network.target local-fs.target
vi.
vii.  [Service]
viii. Type=simple
ix.   User=INSERT CORRECT USER HERE!!!!!!!!!!
x.    Group=INSERT CORRECT GROUP HERE!!!!!!!!!!
xi.   LimitNOFILE=65536
xii.  WorkingDirectory=/opt/discovery/etcd
xiii. ExecStart=/opt/discovery/etcd/etcd-run.sh
xiv.  Restart=on-failure
xv.   RestartSec=30
xvi.
xvii. [Install]
xviii. WantedBy=multi-user.target
```

`/opt/discovery/etcd` — абсолютный путь до директории установки ETC, зависящий от параметров установки.

xix. Выполните команду `sudo systemctl daemon-reload`.

Создание JKS хранилища с серверными сертификатами

Создайте JKS хранилища с серверными сертификатами, подписанными УЦ, доверенными для всех клиентов:

- для REST-интерфейса;
- для подключения к ETCD.

Для создания сертификата используется утилита **keytool.exe** из состава JDK. Для получения сертификата нужно:

1. Создать хранилище ключей и сертификатов:
   1.1. `keytool -genkey -keyalg RSA -alias Test -keystore [путь и имя файла с хранилищем ключей и сертификатов] -storepass [пароль для хранилища ключей и сертификатов] -validity 1440 -keysize 2048 -dname CN=xxx,OU=xxx,O=xx,L=xx,ST=xx,C=RU`
   Например: `keytool -genkey -keyalg RSA -alias ks -keystore D:\ks.jks -storepass 23101989 -validity 1440 -keysize 2048 -dname CN=00CA0001P.TestProducer.ZZ,OU=00CA,O=Org,L=Moscow,ST=Moscow,C=RU.`
   1.2. `keytool -certreq -alias Test -keyalg RSA -file [путь и имя файла с запросом на сертификат] -keystore [путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]`
2. Создать запрос на сертификат.
   Например: `keytool -certreq -alias ks -keyalg RSA -file D:\testProducer.csr -keystore D:\ks.jks`.
3. Отправить запрос на сертификат в УЦ.
4. Импортировать сертификаты в хранилище ключей.
   Полученные от УЦ файл с сертификатом и файлы с корневыми сертификатами необходимо импортировать в хранилище ключей и сертификатов при помощи утилиты **keytool.exe**.
   4.1. Первым необходимо импортировать корневой сертификат:

```
keytool -import -alias ks1 -file [путь и имя файла с корневым сертификатом,
полученным от УЦ] -keystore [путь и имя файла с хранилищем ключей и
сертификатов, созданного на шаге 1]
```

4.2. Вторым необходимо импортировать сертификат УЦ:

```
keytool -import -alias ks2 -file [путь и имя файла с сертификатом УЦ] -keystore
[путь и имя файла с хранилищем ключей и сертификатов, созданного на шаге 1]
```

4.3. Последним импортируется TLS-сертификат:

```
keytool -import -alias cmks -file [путь и имя файла с TLS-сертификатом,
полученным от УЦ] -keystore [путь и имя файла с хранилищем ключей и
сертификатов, созданного на шаге 1]
```

Формирование DN сертификата

CN = 00ZZ0001M.minitoringsystem.segment.contour.AS (пример)

Рекомендуется заполнять следующим образом:

- код ЦА - «00CA»;
- порядковый номер ключа (4 цифры) – до появления централизованной системы управления сертификатами не заполняется;
- тип ключа:
  - P - Producer (Продьюсер)
  - C - Consumer (Консьюмер)
  - S - Support monitoring (Инженер мониторинга)
  - M - Monitoring System (Система мониторинга)
  - A - Access manager (Администратор доступа)
  - E - Maintenance Engineer (Инженер сопровождения)
  - B - Broker (Брокер)
  - I - InfoSec Admin (Администратор безопасности)

  Для одного бизнес-сервиса допускается сертификат с несколькими ролями. Например сертификат системы, которая является одновременно поставщиком и потребителем событий, должен иметь тип ключа «PC» в DN сертификата.
- логин учетной записи пользователя (логин ТУЗ или КЭ для систем и УЗ для пользователей);
- сетевой сегмент;
- тип (контур) кластера (только для брокера и администрирования, **роли producer и consumer не должны включать данный раздел**);

OU = OrganizationalUnitName

O = Organization

L = LocalityName

ST = StateOrProvinceName

C = CountryName

Установка

Ручной способ

1. Проверить, что на сервер, с которого будет производиться установка, установлен Ansible и с него доступны хосты установки EDMS.
2. Распаковать дистрибутив и поместить содержимое директории *modules* в *scripts/Ansible*.
3. Все дальнейшие операции производить из директории *scripts/Ansible*.
4. Создать свой inventory (к примеру, *ID*), для этого создаем директорию в папке *inventories*.
5. Создать структуру файлов по примеру, указанному в таблице.

Файл конфигурации	**Секция**	**Параметры**	**Описание значения**
`group_vars/all/vars.yml`			
		`ansible_no_log:`	Включение опции скрытия сенситивной информации (пароли, токены и т. п.) в шагах выполнения скриптов `ansible`. Значения: `true` или `false`
		`contours:`	Контура EDMS. Например: DEV, IFT для IFT. Можно указать несколько через запятую без пробелов
	`etcd:`		Настройки ETCD
		`installdir:`	Абсолютный путь на конечном сервере до приложения. Например: `/logs/WF/discovery/etcd-data`
		`backup_installdir_to:`	Директория для создания архивной копии директории установки ETCD. По умолчанию данный параметр

Файл конфигурации	Секция	Параметры	Описание значения
			закомментирован, и может быть раскомментирован в случае необходимости создания архивной копии
		`backup_datadir_to:`	Директория для создания архивной копии данных, хранящихся в ETCD. По умолчанию данный параметр закомментирован, и может быть раскомментирован в случае необходимости создания архивной копии
		`cleanData:`	Очистка данных и создание нового кластера. Значения: `true` или `false`
		`initial_cluster_token:`	Токен кластера для начальной загрузки. Например: `my-etcd-token-fpss`
	`etcd: ssl:`		Обязательные параметры настройки SSL
		`sslKeystorePath:`	файл с сертификатами для подключения клиентов. Например: `ssl/event-discovery.jks`
		`sslKeystorePass:`	Пароль для `keystore`
	`etcd: access_control:`		Настройки доступа к ETCD
		`enable:`	Включение ролевой

Файл конфигурации	Секция	Параметры	Описание значения
			модели. Значения: `true` или `false`
		`root_password:`	Пароль пользователя `root`
	`etcd: access_control: roles: readwrite_all:`	`enable:`	Роль «readwrite_all» предназначена для чтения и записи. Если `false` то роль удаляется при переустановке ETCD, `true` — роль создается при переустановке ETCD. Значения: `true` или `false`
		`permissions: -`--prefix=true readwrite ""``	Настройки роли «readwrite_all» на чтение и запись. Параметр командной строки находится в `.
		`users: -00CA0000M.EventDiscovery.xxx.yyyy -00CA0000M.edms.Contour1.xxxx`	Список CN сертификатов пользователей для роли. CN сертификата для установки ETCD. Например: `00CA0000M.EventDiscovery.xxx.yyyy`. CN сертификата для пользователя EDMS при подключении к ETCD. Например:`00CA0000M.edms.xxx.yyyy`
	`etcd: access_control: roles: guest:`	`enable:`	Роль «guest» — роль пользователей по умолчанию. Настройка включения роли. Значения: `true` или `false`
		`permissions: -`--prefix=true read ""`	Настройки роли «guest» на

Файл конфигурации	Секция	Параметры	Описание значения
		`	чтение.
	etcd: access_control: roles: test:	enable:	Роль «test» — роль пользователей по умолчанию. Настройка включения роли. Значения: true или false. Значение по умолчанию:false.false— удалите роль «test».
	emc		Настройки EDMS
		installdir:	Абсолютный путь на конечном сервере до приложения, директория установки. Например: /opt/control-plane
		datadir:	Директория хранения данных. Например: /logs/WF/control-plane-data
		logdir:	абсолютный путь до директории с журналами. Например: /logs/WF/control-plane
		backup_installdir_to:	Директория для создания архивной копии директории установки EDMS. По умолчанию данный параметр закомментирован, и может быть раскомментирован в случае необходимости создания архивной копии

Файл конфигурации	Секция	Параметры	Описание значения
		`logname:`	Наименование файла журналов. Например: `emc`
		`cleanLog:`	Настройка очистки директории журналов `logdir` при установке. Значения: `true` или `false`
		`liquibase_migration:`	Настройка запуска миграции БД при запуске приложения EDMS. Значение `true` — запускаем миграцию БД. Значения: `true` или `false`
		`port:`	Ожидаемый порт для проверки запуска приложения. Значение по умолчанию: 8080
		`keystore_path:`	Сертификат EDMS для HTTPS и подключению к etcd и PostgreSQL (если указано `ssl=true` в строке подключения). Путь от `inventories/_стенд_/` до файла с `keyStore`. Например:`ssl/emc.jks`
		`keystore_password:`	Пароль от `keyStore`
		`authority:`	CN сертификата подключения к ETCD. Например: `00CA0000M.EventDiscovery.xxx.yyyy`
	`emc: audit:`		Настройки аудита

Файл конфигурации	Секция	Параметры	Описание значения
		`url:`	Префикс URL для отправки событий аудита.
		`metamodel:`	Суффикс URL для отправки метамодели. Например: `/v1/metamodel`
		`event:`	Суффикс URL для отправки событий. Например: `/v1/event`
	`emc: replication_settings:`		Настройки репликации
		`transport:`	Использование транспорта для репликатора. Например: `kafka`
		`kafka:`	Имя транспорта. Название данного параметра должно совпадать со значением параметра транспорт в предыдущем пункте.
	`emc: replication_settings: kafka:`	`ssl:`	Настройка использования SSL для подключения к транспорту. При значении `false` используется PLAINTEXT. Значения: `true` или `false`
		`bootstrap-servers:`	Настройки `bootstrap` серверов репликации. Например: `__use_from_inventory__`
	`emc:`	`deploy:`	Настройка видов развертывания: прямое

Файл конфигурации	Секция	Параметры	Описание значения
			развертывание, с помощью Jenkins. Значения: `jenkins,direct`
	`emc: authentication:`		Параметры аутентификации
		`url:`	Ссылка подключения. Например: `url.ru`
		`port:`	Порт подключения. Например: 389
	`emc: authentication: ssl:`		Настройки аутентификации через SSL
		`enabled:`	Настройка использования SSL. Значения: `true` или `false`
		`keystore_path:`	Абсолютный путь до JKS хранилища. Например: `ssl/emc_ldap.jks`
		`keystore_password:`	Пароль для JKS хранилища
		`truststore_path:`	Абсолютный путь до truststore. Например: `ssl/emc_ldap.jks`
		`truststore_password:`	Пароль от truststore
	`emc: authentication: type:`		Типы аутентификации. Значения: `ldap,database`

Файл конфигурации	Секция	Параметры	Описание значения
	`emc: mail:`		Данные для отправки писем
		`enabled:`	Доступность отправки писем. Значения: `true` или `false`
		`host:`	Хост отправки. Например: `mail.xxx.ru`
		`port:`	Порт отправки. Например: 465
		`emailFrom:`	Адрес отправителя. Например: `xxx@mail.ru`
		`username:`	Логин отправителя. Например: `TEST\xxx`
		`password:`	Пароль отправителя
	`emc: mail: smtp:`		Настройки SMTP
		`connectiontimeout:`	Значение по умолчанию: 2000
		`timeout:`	Значение по умолчанию: 1000
		`writetimeout:`	Значение по умолчанию: 2000
	`emc: mail: ssl:`		Настройки SSL для отправки почты
		`enabled:`	Доступность. Значения:

Файл конфигурации	Секция	Параметры	Описание значения
			`true` или `false`
		`keystore_path:`	Абсолютный путь до JKS-хранилища. Например: `ssl/emc_ldap.jks`
		`keystore_password:`	Пароль для JKS-хранилища
		`truststore_path:`	Абсолютный путь до truststore. Например: `ssl/emc_ldap.jks`
		`truststore_password:`	Пароль от truststore
	`emc: profile:`		Настройки подключения к БД
		`url:`	URL для подключения к БД. Например: `jdbc:postgresql://url.ru:5432`/control_plane
		`username:`	Имя пользователя БД. Например: `control_plane`
		`password:`	Пароль пользователя БД
		`ssl:`	Настройка подключение к базе данных с помощью SSL. При значении `false` подключение осуществляется без SSL, используется `plain-text`. Значения:`true` или `false`
`inventory`			Файл настроек host подключения

Файл конфигурации	Секция	Параметры	Описание значения
		[kafka]	Список host подключение к транспорту EVTD. Например:url.ru
		[emc]	Список host EDMS. Например:url.ru
		[etcd]	

Важно: Все параметры *inventory* аннотированы, приведенные выше параметры являются теми, на которые требуется обратить внимание (стендозависимые). В случае необходимости кастомизации рекомендуется читать аннотации параметров.
Пример заполненного файла vars.yml:

```
tmp_dir: /tmp/installer # путь до временной директории на конечных серверах
wait_for_start: 120 # время (в секундах) на корректный старт приложения (не запускается с ошибкой при превышении)
check_correct_start: true # проверка корректности запуска приложения по наличию строки в журнале
ansible_no_log: false # отключение вывода паролей в журнал
password_encoder_cli_path: password-encrypt-cli-1.3.jar # путь до утилиты для шифрования паролей (helper/encode_passwords.yml)

segment: SegmentA # идентификатор сегмента. Например: SegmentA, SegmentB,...
contours: Contour0 # контура EDMS. Например Contour0, Contour1,...

etcd:
  distr: etcd.tar.gz
  installdir: /opt/discovery/etcd # абсолютный путь на конечном сервере до приложения
  datadir: /logs/WF/discovery/etcd-data # абсолютный путь до директории с данными
  #backup_installdir_to: /path/to/backup # директория для создания архивной копии директории установки ETCD
  #backup_datadir_to: /path/to/data_backup # директория для создания архивной копии данных, хранящихся в ETCD
  cleanData: false # очистка данных и создание нового кластера
  client_port: 2379 # port for client traffic
  peer_port: 2380 # port for peer traffic
  initial_cluster_token: my-etcd-token-fpss # токен кластера для начальной загрузки
  ssl: # Обязательный параметр
    sslKeystorePath: ssl/event-discovery.jks # файл .jks с сертификатами для подключения клиентов
    sslKeystorePass: <password>
  additional_parameters: # дополнительные параметры
    auto-compaction-retention: 3 # сохранение истории в часах
  access_control:
    enable: false # включение ролевой модели
    root_password:<password> # пароль для `root` пользователя
    roles:
      readwrite_all: # наименование роли
        enable: true # при `false` роль удаляется без создания
        permissions:
          - `--prefix=true readwrite ""` # настройки роли (пустые ``)
        users: # пользователи роли, если пользователя нет, он удаляется из роли
```

```
          - 00XX0000X # укажите CN сертификата SSL, с которым пользователь будет
подключаться
      guest: # переопределение прав для группы «guest» по умолчанию
        enable: true
        permissions:
          - `--prefix=true read ""` # настройки роли (пустые кавычки)
#      test:
#        enable: false # всегда удаляйте роль «test»

emc:
  installdir: /opt/control-plane # абсолютный путь на конечном сервере до приложения,
директория установки
  datadir: /logs/WF/control-plane-data
  logdir: /logs/WF/control-plane # абсолютный путь до директории с журналами
  #backup_installdir_to: /path/to/backup # директория для создания архивной копии
директории установки EDMS
  logname: emc # наименование файла журналов
  loglevel: INFO
  #  log_total_size_cap: 2GB
  cleanLog: false # очистка `logdir` при установке
  distr: emc-boot.zip
  liquibase_migration: true # запускается ли миграция базы при запуске приложения EDMS
  port: 8080 # ожидаемый порт для проверки запуска приложения
  keystore_path: ssl/emc.jks # сертификат EDMS для HTTPS и подключения к etcd и PostgreSQL
(если указано ssl=true в строке подключения). Путь от `inventories/_стенд_/` до файла с
keyStore
  keystore_password: <password> # пароль от keyStore
  authority: 00XX0000X.EventDiscovery.xxx.yyyy # CN сертификата etcd
  audit:
   *** # блок для установки параметров аудита
  replication_settings: # настройки репликации
    transport: kafka # использование Kafka для репликатора
    kafka: # должно совпадать со значением в предыдущем пункте
      ssl: true # используется ли SSL для подключения к EVTD. В противном случае
используется PLAINTEXT
      bootstrap-servers: __use_from_inventory__
  #    to_application_yaml: # Настройки репликации запишите в application.yml (блок
replication)
  #      producers:
  #        - name: replication
  #          destinations:
  #            - name: replication
  #              active: false
  #              topic: EMC.REPLICATIONEVENT.V1
  #            - name: rollback
  #              active: true
  #              topic: EMC.ROLLBACKEVENT.V1
  #            - name: redelivery
  #              active: false
  #              topic: EMC.REPLICATIONEVENT.RETRY.V1
  #            - name: deadLetter
  #              active: false
  #              topic: EMC.DEADLETTEREVENT.V1
  #        - name: segment-replication
  #          destinations:
  #            - name: replication
  #              active: false
  #              topic: EMC.REPLICATIONEVENT.V1
  #            - name: redelivery
  #              active: false
  #              topic: EMC.REPLICATIONEVENT.RETRY.V1
  #            - name: deadLetter
  #              active: false
  #              topic: SEGMENT.DEADLETTEREVENT.V1
  #          properties:
  #            "[security.protocol]": PLAINTEXT
```

```
  #           "[bootstrap.servers]": localhost:9092
  #      consumers:
  #        - name: replication
  #          active: true
  #          groupId: cp_replica
  #          topic: EMC.REPLICATIONEVENT.V1
  #          pollTime: 1000                  # Необязательное поле
  #          schedulerTime: 2000             # Необязательное поле
  #        - name: retry-replication
  #          active: false
  #          groupId: cp_replica
  #          topic: EMC.REPLICATIONEVENT.RETRY.V1
  #        - name: rollback-replication
  #          active: false
  #          groupId: cp_replica
  #          topic: EMC.ROLLBACKEVENT.V1
  #        - name: SEGMENT-replication
  #          active: false
  #          groupId: cp_replica
  #          topic: SEGMENT.REPLICATIONEVENT.V1
  #          properties:                     # пример переопределения bootstrap-серверов
для этого consumer
  #            "[bootstrap.servers]": localhost:9092
  #        - name: SEGMENT-retry-replication
  #          active: false
  #          groupId: cp_replica
  #          topic: SEGMENT.REPLICATIONEVENT.RETRY.V1
  #          properties:                     # пример переопределения bootstrap-серверов и
типа шифрования для этого consumer
  #            "[security.protocol]": PLAINTEXT
  #            "[bootstrap.servers]": localhost:9092

  encoding_configs: # конфигурация шифрования паролей
    security.encoding.key: encrypt.pass
    security.encoding.class: **
  deploy: jenkins # direct or jenkins
  authentication: # настройки способа аутентификации пользователя
    url: **.ru
    port: 389
    ssl:
      enabled: false
      keystore_path: ssl/emc_ldap.jks
      keystore_password: <password>
      truststore_path: ssl/emc_ldap.jks
      truststore_password: <password>
    type: ldap # типы подключения
  #    ldap: # пример подключения ldap
  #      controller: CAB-VSP-DC00005
  #      base: DC=xx,DC=xxx,DC=ru
  #    database: # пример подключения database
  #      type: postgresql
  #      instance: control_plane
  #      schema: edms
  #      table: edms_user_name
  #      user: control_plane
  #      password: <password>
  mail: # данные для отправки писем
    enabled: false
    host: **.ru
    port: 465
    emailFrom: xxx@xxx.ru
    username: **
    password: <password>
    smtp:
      connectiontimeout: 2000
      timeout: 1000
```

```
      writetimeout: 2000
    ssl:
      enabled: true
      keystore_path: ssl/emc_ldap.jks
      keystore_password: <password>
      truststore_path: ssl/emc_ldap.jks
      truststore_password: <password>

  profile: # генерация профиля
    url: jdbc:postgresql://***.ru/***
    username: control_plane
    password: <password>
    ssl: true # включение SSL для подключения к базе
```

Параметры аудита в `vars.yml`:

```
audit:
    url: url.ru # префикс URL для отправки событий аудита
    metamodel: /v1/metamodel # суффикс для метамодели
    event: /v1/event # суффикс для событий
```

Пример заполненного файла `inventory`:

```
localhost ansible_connection=local

[kafka]
список host Kafka

[emc]
список host EDMS

[etcd]
список host etcd
```

Использование vault для шифрования паролей

При хранении сенситивной информации в Git ее рекомендуется шифровать. Для этого можно использовать утилиту **ansible-vault** (идет в комплекте с пакетом ansible).

Для шифрования пароля следует выполнить команду на сервере, с которого производится развертывание EDMS:

```
ansible-vault encrypt_string -n jks_password 'ENCRYPT_STRING'
```

где `ENCRYPT_STRING` - строка, которую необходимо зашифровать, а `jks_password` - имя переменной.

При запросе вводим пароль для шифрования, а на выходе получаем то, что нужно занести в inventory:

```
jks_password: !vault |
  $ANSIBLE_VAULT;1.1;AES256
  30323632346331616266363234303338663965366539343535353133626165316564633237626536
  39323338313537393561353764633233633261333333338340a33633862383730393739353831393 9
  37626531383432366662303466363761616566393638306564623661323133356133613863313032
  39666535316436316 60a666136623361613863643137396663653363316139316566393366653838
  3039
```

Аналогично, возможно шифрование файлов:

```
ansible-vault encrypt <имя файла>
```

Ручное шифрование паролей в конфигурационных файлах

Для шифрования паролей используется утилита **password-encrypt-cli-1.3.jar** в составе дистрибутива EDMS и пакет java, установленный на сервере.

Для шифрования вызывается команда:

```
java -jar password-encrypt-cli-1.3.jar --key <ключ> --password <пароль>
```

В результате выполнения команды будет выведен зашифрованный пароль:

```
Encrypted password: <зашифрованный пароль>
```

где <ключ> - содержимое файла *encrypt.pass*, путь к которому указывается в *encoding_configs/security.encoding.key*.

Запуск установки

Запустить установку командой:

```
ansible-playbook -i inventories/<ID>/inventory synapse_emc.yml --ask-vault-pass
```
где `ID` - имя недавно созданного inventory.

Установка будет производиться на все хосты из inventory. Для ограничения списка узлов используем команду:

```
ansible-playbook -i inventories/<ID>/inventory synapse_emc.yml --ask-vault-pass -l <узлы через запятую без пробелов>
```

Установка с помощью Jenkins (опциональный способ)

1. Распаковать дистрибутив и поместить содержимое папки *scripts* в BitBucket.
2. Создать и настроить inventory (см. раздел *Ручной способ установки* выше) и поместить изменения в BitBucket.
3. В Jenkins создать Jenkins Pipeline с получением скриптов развертывания из BitBucket.
    - Pipepline script from SCM
        - SCM - GIT
        - repository url - ссылка на репозиторий, куда поместили скрипты.
        - Выбираем или добавляем учетные данные для доступа к BitBucket
    - Script_path - относительный путь *SYN_custom.groovy*. Убедиться, что не стоит галочка *Lightweight checkout*.
4. Сохранить получившийся Jenkins Pipeline и запустить его.
5. Проверить, что после запуска подгрузились дополнительные параметры.
6. При необходимости, поменять для параметров значения по умолчанию. Например, изменить имя используемых *credentials*.

Запуск установки

При запуске задания Jenkins по установке в параметрах выбирать нужный inventory, playbook `synapse_emc.yml`, а в поле `customURL` указать ссылку на дистрибутив.

Варианты установки EDMS:

- Полная установка с помощью задания Jenkins `emc_custom` (поставляется в дистрибутиве) с выбором playbook `synapse_emc.yml` без указания тегов. Устанавливаются все компоненты в определенном порядке (сначала устанавливается ETCD, затем EDMS).
- Частичная установка с помощью задания Jenkins `emc_custom` (поставляется в дистрибутиве), с выбором playbook `etcd.yml` для установки ETCD и/или `emc.yml`

для установки EDMS. В данном варианте установки сначала запустите установку ETCD, затем установку EDMS без выбора тегов.

Обновление

Обновление всех компонентов EDMS выполняется путем переустановки.

PostgreSQL и Java считаются системным программным обеспечением и их обслуживание и обновление выполняется администраторами стендов, на которых размещается EDMS.

Удаление

Ручной способ

Для остановки компонентов EDMS на сервере, с которого производилась установка, выполните команду:

```
ansible-playbook -i inventories/<ID>/inventory synapse_emc.yml --ask-vault-pass -t stop
```

Удаление EDMS осуществляется сторонними администраторами, сопровождающими сервер, на котором размещен EDMS.

При помощи Jenkins (опциональный способ)

Остановку EDMS можно осуществить с помощью pipeline Jenkins используется задание **emc_custom** с выбором playbook `synapse_emc.yml` с тегом *stop*.

Удаление EDMS осуществляется аналогично ручному способу.

Проверка работоспособности

Проверка EDMS осуществляется в ручном режиме. Для этого проверьте работоспособность ссылки: https://host:port/emc.

Проверка работоспособности ETCD осуществляется тремя способами.

Ручной способ

На сервере, с которого производилась установка, выполнить команду:

```
ansible-playbook -i inventories/<ID>/inventory etcd.yml --ask-vault-pass -t status
```

где `ID` - имя созданного inventory. Данной операцией выполнится проверка статуса ETCD. Появится полная информация о состоянии кластера. При статусе *is healthy:* ETCD является работоспособным.

При помощи Jenkins (опциональный способ)

- Проверка работоспособности ETCD выполняется с помощью задания Jenkins `emc_custom` с параметром playbook `etcd.yml` с включенным тегом `status`. Выведется детальная информация о работоспособности кластера ETCD.

Автоматический способ

В автоматическом режиме — через компонент *Mayak*. Описание принципов работы системы мониторинга ETCD приведено в документации системы мониторинга *Mayak*.

Откат

Откат всех компонентов EDMS выполняется полной переустановкой предыдущей версии EDMS.

Перед переустановкой предыдущей версии EDMS необходимо выполнить откат БД до версии, соответствующей устанавливаемой версии UI EDMS. Подробнее данный вопрос описан в данном документе ниже.

Откат БД

```
Использование отката сопряжено с риском для БД, поэтому перед его выполнением важно
обратить внимание на возможные непредвиденные последствия. Рекомендуется создать
архивную копию текущей версии БД перед выполнением отката.
```

Откат БД необходимо выполнить **ДО** запуска переустановки UI EDMS. Откат БД осуществляется запуском команды *rollback* в терминале:

```
liquibase rollback <имя тега>
```

Здесь <имя тега> - имя тега БД, соответствующее версии БД, например *release_1.2*.

Команда tag используется для пометки состояния БД путем добавления тега к последней строке таблицы **edms_databasechangelog**. После установки тега становится возможным использовать команду отката (rollback) для отката всех изменений до этого тега.

Тег устанавливается в отдельном наборе изменений БД (в отдельном *changeset*), например:

```
<changeSet id="Add release 1.2 tag"
           author="Author"
           logicalFilePath="file.xml">
    <tagDatabase tag="release_1.2"/>
</changeSet>
```

Во время выполнения команды *rollback* производится последовательный откат всех изменений, указанных в наборе изменений *changeset* до строки **tag** в таблице **edms_databasechangelog**. Например, можно использовать команду *rollback*, когда необходимо отменить серию изменений, внесенных в БД, относящихся к определенному тегу, например, пронумерованному релизу.

Если указано несколько тегов, т.е. имеется несколько версий БД, то при попытке отката не на последнюю версию будет производится последовательный откат до нужной версии начиная с последней. Например, если существуют теги БД для версии 1, версии 2 и версии 3, и необходимо выполнить откат до версии 2, то сначала будет произведен откат до версии 3, затем до версии 2.

Также необходимо учесть, что нельзя произвести несколько раз откат к одной и той же версии БД, т.к. при откате БД тег версии, откат к которой был произведен, удаляется. При миграции БД в процессе обновления всех компонентов EDMS к имеющимся тегам добавляется новый тег с версией, к которой можно будет в дальнейшем осуществить откат.

Часто встречающиеся проблемы и пути их устранения

Не выявлены.

Чек-лист валидации установки

Проверка результатов установки EDMS

Для EDMS проверка корректности установки производится в рамках скриптов установки.

Ручная проверка успешности установки:

1. Наличие активного порта:
   1.1. Проверяется в рамках скриптов развертывания Ansible. 1.2. Проверьте работоспособность ссылки напрямую через браузер. Успешная загрузка страницы подтверждает наличие активного порта.
2. Содержание строки запуска приложения в журналах на сервере в директории:
   2.1. Убедитесь, что в файлах журнала присутствует запись о старте приложения. Для этого проверьте наличие файла журнала в каталоге `/logs/control-plane/emc.log` и убедитесь, что строка `ru.organization.controlplane.Main: Started Main in XX seconds (JVM running for XX)`, (где XX — количество секунд), входит в данный файл. 2.2. Проверьте успешный запуск EDMS по ссылке: https://host:port/emc.