**Руководство администратора Агента ПОДД СМЭВ**

**ПОДД СМЭВ**

Листов 36

Москва 2022

**СОДЕРЖАНИЕ**

[Термины и сокращения 4](#_Toc110347160)

[1. Введение 6](#_Toc110347161)

[1.1. Область применения 6](#_Toc110347162)

[1.2. Краткое описание возможностей 6](#_Toc110347163)

[1.3. Уровень подготовки пользователя 6](#_Toc110347164)

[1.4. Перечень документации для работы с ПОДД СМЭВ 6](#_Toc110347165)

[2. Назначение и условия применения 7](#_Toc110347166)

[2.1. Назначение 7](#_Toc110347167)

[2.2. Требования к среде для развертывания Агента ПОДД СМЭВ 7](#_Toc110347168)

[3. Описание конфигурационного файла 8](#_Toc110347169)

[3.1. Состав и содержание дистрибутивного пакета 8](#_Toc110347170)

[3.2. Формирование конфигурационного файла 8](#_Toc110347171)

[3.2.1. Общие настройки Агента 8](#_Toc110347172)

[3.2.2. Настройка CryptoPro 8](#_Toc110347173)

[3.2.3. Настройка регистрации Витрин данных (только для агента-поставщика) 9](#_Toc110347174)

[3.2.4. Настройка параметров подключения к Kafka (только для агента-поставщика) 9](#_Toc110347175)

[3.2.5. Настройка подключения к Pulsar, настройки подключения и авторизации в Keycloak для получения JWT 9](#_Toc110347176)

[3.2.6. Настройка модуля подписания печатных форм 11](#_Toc110347178)

[3.2.7. Настройка использования S3 для передачи блобов 11](#_Toc110347179)

[3.2.8. Настройка aгентов для работы в гео-распределенной конфигурации 12](#_Toc110347180)

[3.2.9. Настройки разбиения получаемой информации на чанки (опционально) 13](#_Toc110347181)

[3.2.10. Настройка организации информационного обмена через API Gateway по HTTPS (только для агента-поставщика) 14](#_Toc110347182)

[3.2.11. Настройка подключения к агентам 14](#_Toc110347183)

[4. Подготовка и настройка системы для запуска агента без использования docker 16](#_Toc110347184)

[4.1. Состав и содержание дистрибутивного пакета 16](#_Toc110347185)

[4.2. Порядок загрузки данных и программ 16](#_Toc110347186)

[4.3. Предварительные операции (установка «пре-реквизитов») 16](#_Toc110347187)

[4.4. Запуск Агента ПОДД СМЭВ 18](#_Toc110347201)

[5. Настройка и запуск агента ПОДД с использованием docker 20](#_Toc110347202)

[5.1. Предварительные условия 20](#_Toc110347203)

[5.2. Настройка ротации лог-файлов в docker (опционально) 20](#_Toc110347204)

[5.3. Состав и содержание дистрибутивного пакета 20](#_Toc110347205)

[5.4. Подготовка к запуску 21](#_Toc110347206)

[5.5. Запуск агента 21](#_Toc110347207)

[5.6. Информация по лицензированию CryptoPro 22](#_Toc110347208)

[6. Типовые вопросы и проблемы 23](#_Toc110347210)

[Приложение 1 Добавление промежуточных сертификатов в ключ CryptoPro 25](#_Toc110347211)

[Приложение 2 Особенности запуска под kubernetes 30](#_Toc110347212)

[Приложение 3 Установка компонента prohibitor (опционально) 31](#_Toc110347213)

[1.1 Состав и содержание дистрибутивного пакета 31](#_Toc110347214)

[1.2 Порядок загрузки данных и программ 31](#_Toc110347215)

[1.3 Требования к среде для развёртывания Сервиса проверки полномочий 31](#_Toc110347216)

[1.4 Предварительные операции (установка «пре-реквизитов») 31](#_Toc110347217)

[1.5 Настройка БД для работы Сервиса проверки полномочий 35](#_Toc110347218)

[1.6 Запуск Сервиса проверки полномочий 36](#_Toc110347219)

[6.1. 1.6 Особенности запуска под docker 37](#_Toc110347220)

# Термины и сокращения

|  |  |
| --- | --- |
| Агент ПОДД СМЭВ  также: Экземпляр ПО Агент (ПОДД СМЭВ) | Экземпляр ПО «Агент (ПОДД СМЭВ)», установленный в контуре ИС УВ и обеспечивающее сопряжение Экземпляров ПО «Витрина данных» и ИС УВ с Ядром ПОДД СМЭВ. |
| Витрина (данных) | Комплекс программных и технических средств в составе информационно-телекоммуникационной инфраструктуры участника взаимодействия, обеспечивающий хранение и предоставление данных другим участникам взаимодействия в соответствии с критериями, установленными Министерством цифрового развития, связи и массовых коммуникаций |
| ГБ | Гигабайт |
| ГОСТ-2012 | ГОСТ Р 34.10-2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи» |
| ЕИП НСУД | Единая информационная система национальной системы управления данными |
| ИС | Информационная система |
| ИЭП | Инфраструктура электронного правительства |
| МинЦифры | Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации |
| Мнемоника | Короткое символическое имя объектов модели данных (Витрин, сущностей, атрибутов) для использования в Запросах. Имеет более строгие правила формирования, чем наименования объектов (нельзя использовать пробелы, специальные символы и т. д.), что облегчает чтение Запросов человеком и позволяет изменять наименования объектов без влияния на работоспособность систем Потребителей |
| НСУД | Национальная система управления данными |
| ОС | Операционная система |
| ПОДД | Подсистема обеспечения доступа к данным |
| ПОДД СМЭВ | Подсистема обеспечения доступа к данным — подсистема СМЭВ, предназначенная для обеспечения взаимодействия СМЭВ с Витринами данных |
| Подзапрос (данных) | Вариант Запроса, который формируется ПОДД СМЭВ в целях обработки распределённого Запроса Потребителя |
| Поставщик (данных ПОДД) | Участник взаимодействия с полномочиями по предоставлению данных с использованием ПОДД в соответствии с загруженной в Ядро ПОДД Моделью государственных данных |
| Потребитель (данных ПОДД) | Участник взаимодействия, обрабатывающий данные, размещенные на Витринах данных, с использованием ПОДД |
| СМЭВ | Единая система межведомственного электронного взаимодействия и Единая система нормативной справочной информации |
| УВ | Участник взаимодействия – федеральный орган исполнительной власти, государственный внебюджетный фонд и иной орган или организация, участвующие в предоставлении государственных и муниципальных услуг (функций) |
| УЦ | Удостоверяющий центр |
| ЦПУ | Центральное процессорное устройство |
| Ядро ПОДД СМЭВ | Централизованная часть ПОДД, предназначенная для маршрутизации запросов и контроля полномочий УВ |
| API | (англ. Application Programming Interface) — набор классов, процедур, функций, структур или констант, которыми одна компьютерная программа может взаимодействовать с другой программой |
| JDBC | (англ. Java DataBase Connectivity) — платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД |
| JWT | (англ. JSON Web Token) — открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON |
| OpenAPI, спецификация OpenAPI | Формализованная спецификация и экосистема множества инструментов, предоставляющая интерфейс между front-end системами, кодом библиотек низкого уровня и коммерческими решениями в виде API |
| SQL | (англ. Structured Query Language) — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных |

# Введение

## Область применения

ПОДД СМЭВ – подсистема обеспечения доступа к данным СМЭВ, направленная на автоматизацию процесса передачи данных и уведомлений об изменении данных между организациями или органами власти, ответственными за формирование и ведение информационных ресурсов, зарегистрированных в НСУД.

## Краткое описание возможностей

ПОДД СМЭВ обеспечивает:

* предоставление для Потребителей механизмов унифицированного доступа к данным, размещённым у Поставщиков, с использованием языка запросов, основанного на стандарте языка SQL;
* предоставление для Потребителей возможности получения данных, связанных по ключевым полям или вычисляемому критерию и размещённых у разных Поставщиков;
* контроль полномочий Потребителей по доступу к данным, размещённым у Поставщиков, и поддержку гибкой настройки полномочий с точностью до атрибута и критерия отбора строк.

## Уровень подготовки пользователя

Развертывание Агента ПОДД СМЭВ должно осуществляться в соответствии с настоящим Руководством администратора.

Специалист должен обладать следующими знаниями и опытом:

* администрирование CryptoPro;
* администрирование docker.

## Перечень документации для работы с ПОДД СМЭВ

Порядок подключения и использования ПОДД СМЭВ после установки Агента ПОДД СМЭВ отражён в следующих документах[[1]](#footnote-2):

* Методические рекомендации по работе с ПОДД СМЭВ;
* Регламент подключения к СМЭВ 4.

# Назначение и условия применения

## Назначение

ПОДД СМЭВ состоит из следующих компонентов:

* Ядро ПОДД СМЭВ – центральный компонент ПОДД СМЭВ, размещенный в ИЭП;
* Агент ПОДД СМЭВ – компонент ПОДД СМЭВ, устанавливаемый в контуре участника взаимодействия.

Агент ПОДД СМЭВ представляет собой типовое программное обеспечение, устанавливаемое на стороне УВ и обеспечивающее сопряжение Витрин, хранилищ реплик, ИС УВ с ПОДД.

Данный документ содержит состав и содержание дистрибутивного пакета, порядок установки и описание настроек Агента ПОДД СМЭВ.

## Требования к среде для развертывания Агента ПОДД СМЭВ

Минимальные требования к характеристикам сервера для развёртывания Агента ПОДД СМЭВ приведены в таблице 1.

Таблица 1 – Минимальные требования к характеристикам сервера

| Характеристики | Агент ПОДД СМЭВ |
| --- | --- |
| ОС | CentOS 7.9 |
| ЦПУ | 4-х ядерный процессор |
| Оперативная память | 8 ГБ |
| Дисковое пространство | 20 ГБ |

# Описание конфигурационного файла

## Состав и содержание дистрибутивного пакета

Состав дистрибутива Агента ПОДД СМЭВ (/distr/einfahrt):

* /distr/einfahrt/certs/cp\_ca\_store – JKS с поддержкой корневых сертификатов Ростелекома;
* /distr/einfahrt/s3\_creds.properties – креденшелы для подключения к хранилищу S3;
* /distr/einfahrt/customLogLevels.xml – файл описания настроек логирования;
* /conf/\*.yml – заготовки конфигурационных файлов для Потребителя и Поставщика, для тестового и продакшен контуров.

В папке conf пакета находятся типовые заготовки конфигурационного файла.

Необходимо взять заготовку конфигурационного файла в соответствии с потребностями:

* шаблоны конфигурационного файла Агента для продуктивной среды:
  1. conf/prod-consumer-application.yml – для потребителя.
  2. conf/prod-producer-application.yml – для поставщика.
* шаблоны конфигурационного файла Агента для продуктивной среды (2-х ЦОДовая конфигурация (ФЦОД, РЦОД)):
  1. conf/prodMC-consumer-application.yml – для потребителя.
  2. conf/prodMC-producer-application.yml – для поставщика.
* шаблоны конфигурационного файла Агента для тестовой среды (ТПОДД):
  1. conf/test-consumer-application.yml – для потребителя.
  2. conf/test-producer-application.yml – для поставщика.

Скопировать выбранный файл в каталог /distr/einfahrt под именем application.yml.

В скопированном конфигурационном файле «application.yml» задать необходимые параметры (в заготовке обозначены звездочками «\*»). Более подробное описание необходимых настроек файла application.yml указаны в разделах ниже.

## Формирование конфигурационного файла

### Общие настройки Агента

Указать параметры Агента ПОДД:

|  |
| --- |
| # Общие настройки агента  agent:  # Идентификатор (мнемоника) агента  id: \*\*\* ИДЕНТИФИКАТОР (МНЕМОНИКА) АГЕНТА \*\*\*  ogrn: '\*\*\* ОГРН АГЕНТА \*\*\*' |

### Настройка CryptoPro

Указать идентификатор и пароль ключа CryptoPro:

|  |
| --- |
| keys:  alias: \*\*\* ИДЕНТИФИКАТОР КЛЮЧА CryptoPro \*\*\*  password: \*\*\* ПАРОЛЬ КЛЮЧА CryptoPro \*\*\* |

### Настройка регистрации Витрин данных (только для агента-поставщика)

Пример раздела конфигурации для настройки Витрины данных указан ниже:

|  |
| --- |
| # Общие настройки витрины  datamart:  # признак взаимодействия агента с витриной (true - интеграция включена, false - только REST-интерфейс)  enabled: true  # Настройки регистрации витрин  datamart-registration:    # Список витрин для регистрации. Можно задать пустой список []    datamarts:      - id: {{ vitrine\_id1 }}        # Способ получения профиля витрины, предустановленный профиль из локального файла,        #  либо запросом к витрине [DEFINED\_PROFILE | DATAMART\_REQUEST]        registrationFlow: DEFINED\_PROFILE        # Предустановленный профиль        definedProfile: datamart\_profile/postgresql.json      - id: {{ vitrine\_id2 }}        # Способ получения профиля витрины, предустановленный профиль из локального файла,        #  либо запросом к витрине [DEFINED\_PROFILE | DATAMART\_REQUEST]        registrationFlow: DEFINED\_PROFILE        # Предустановленный профиль        definedProfile: datamart\_profile/postgresql.json |

### Настройка параметров подключения к Kafka (только для агента-поставщика)

Указать параметры подключения к kafka (раздел «kafka:»).

* {{ kafka\_link }} – URL адрес Kafka;

Пример раздела конфигурации:

|  |
| --- |
| kafka:    consumer:      property:        bootstrap.servers: {{ kafka\_link }}         {...}     producer:      property:        bootstrap.servers: {{ kafka\_link }}         {...}    admin:      property:        bootstrap.servers: {{ kafka\_link }} |

### Настройка подключения к Pulsar, настройки подключения и авторизации в Keycloak для получения JWT

для tls-trust-store указать:

* + path – фактический путь до директории c файлом «cp\_ca\_store» из дистрибутива (в данном примере - /distr/einfahrt/certs);
  + password – пароль от «cp\_ca\_store». Убедитесь, что пользователь, запускающий приложение имеет доступ на чтение к этому файлу;

для tls-key указать:

* + alias – идентификатор ключа CryptoPro, установленного ранее (на шаге 3);
  + password – пароль ключа CryptoPro.
* keycloak-oauth.auth-server-url – адрес сервиса авторизации. Данные настройки уже прописаны в заготовке конфигурационного файла. При использовании нескольких серверов авторизации, они могут быть перечислены через запятую.

Пример раздела конфигурации указан ниже:

|  |
| --- |
| # Настройки подключения к Pulsar  pulsar:  client:  serviceUrl: pulsar://XXX.XX.XXX.XXX:XXXX  consumer:  # Имя подписки, также используется при формировании имени DLQ топика  subscriptionName: ${agent.id}  negativeAckRedeliveryDelayMicros: 0  # Авторизация в Pulsar c помощью JWT  auth:  # Включение авторизации  enabled: true  # Путь к статичному JWT используемому для авторизации, если не указан, JWT запрашивается у Keycloak  service-token: ''  # Подключение и авторизация в Keycloak для получения JWT  keycloak-connection:  tls-verify-hostname: false  tls-trust-store:  path: \*\*\* /ПУТЬ/К/cp\_ca\_store \*\*\*  password: XXXXXXX  tls-key:  alias: \*\*\* ИДЕНТИФИКАТОР КЛЮЧА CryptoPro \*\*\*  password: \*\*\* ПАРОЛЬ КЛЮЧА CryptoPro \*\*\*  keycloak-oauth:  realm: pulsar  auth-server-url: https:// XXX.XX.XXX.XXX/keycloak/auth  grant\_type: client\_credentials  client\_id: ${agent.id} |

### Настройка модуля подписания печатных форм

Для настройки модуля подписания печатных форм необходимо перечислить используемые сертификаты для данного модуля и данные о них. При отсутствии используемых печатных форм, данный раздел в конфигурации должен отсутствовать. Пример раздела конфигурации:

|  |
| --- |
| printable-form:    signature:      printable-form-keys:        -          keystoreType: JCP          certificateAlias: {{ key1 }}          privateKeyAlias: {{ key1 }}          privateKeyPass: {{ key1\_password }}          signatureAlgorithm: {{ $.Values.keys.signaturealgorithm }}          signatureURI: {{ $.Values.keys.signatureuri }}          digestMethod: {{ $.Values.keys.digestmethod }}          useSmevTransform: false           -          keystoreType: JCP          certificateAlias: {{ key2 }}          privateKeyAlias: {{ key2 }}          privateKeyPass: {{ key2\_password }}          signatureAlgorithm: {{ $.Values.keys.signaturealgorithm }}          signatureURI: {{ $.Values.keys.signatureuri }}          digestMethod: {{ $.Values.keys.digestmethod }}          useSmevTransform: false          {...}    #Пары значения "имя\_печатной\_формы": "алиас сертификата"    forms:      {{ form1\_name }}: {{ key1 }}      {{ form2\_name }}: {{ key2 }}          {...}  # Максимальный размер данных для подписания  max-content-length: 268435456 |

### Настройка использования S3 для передачи блобов

*Раздел применим только при использовании S3 storage*

Для настройки использования S3 для передачи блобов, следует указать адрес точки доступа и ссылку на файл с учётными данными для подключения к S3. Bucket с соответствующим именем должен быть предварительно создан на стороне S3. Также в файле «s3\_creds.properties» необходимо прописать корректные значения логина и пароля для подключения к хранилищу S3 согласно образцу в файле /distr/einfahrt/s3\_creds.properties. В случае, если S3 не используется, адрес, bucket и путь могут быть любыми. Пример раздела конфигурации:

|  |
| --- |
| query:    # Настройки получения BLOB по ссылке    load-reference-data:      # Источник получения данных для BLOB ссылок,      # S3 - данные загружаются агентом из хранилища S3, DATAMART - запрос пересылается витрине      blob-source: DATAMART      # Настройки подключения к хранилищу S3      s3-storage:        endpoint: http://{{ s3\_url }}/        # Имя бакета        bucket-name: {{ bucket name }}        # Путь к файлу с данными для авторизации        pathToCredentialFile: {{ /absolute/path/to/s3\_creds.properties }} |

### Настройка aгентов для работы в гео-распределенной конфигурации

Для настройки aгентов для работы в гео-распределенной конфигурации необходимо перечислить все развернутые ядра с указанием ссылки на pulsar и сервис авторизации ядра.

*Примечание: данные настройки уже прописаны в заготовке конфигурационного файла.*

*Важно!  
Масштабируемый агент (см. раздел 3.2.11) должен работать только с одним ядром.  
Немасштабированный может работать с двумя ядрами.*

Пример раздела конфигурации:

|  |
| --- |
| # Настройки подключения ко всем ЦОД с установленным ПО ядра  data-center:   …   nodes:      - node-id: {{ node1\_name }}        # Настройки подключения к Pulsar        pulsar:          # Включение Round-Robin балансировки по спику брокеров          enable-balancing: true          # Максимально допустимое время передачи сообщения одному из брокеров          send-timeout: 35s          # Длительность периода вывода брокеров из пула, при его недоступности          max-dismissal-delay: 5s          # Список подключений к брокерам  # есть возможность указания нескольких адресов pulsar, для этого необходимо в переменн«ю "service»rl" через запятую прописать все адреса        # прим–р - serviceUrl: pulsar://{{ address1:port }}, pulsar://{{ address2:port }}          client:            serviceUrl: pulsar://{{ pulsar\_kernel1\_link }}  ...          auth:            enabled: ${pulsar.auth.enabled}            service-token: ${pulsar.auth.service-token}            keycloak-connection:              tls-verify-hostname: ${pulsar.auth.keycloak-connection.tls-verify-hostname}              tls-trust-store:                path: ${pulsar.auth.keycloak-connection.tls-trust-store.path}                password: ${pulsar.auth.keycloak-connection.tls-trust-store.password}              tls-key:                alias: ${pulsar.auth.keycloak-connection.tls-key.alias}                password: ${pulsar.auth.keycloak-connection.tls-key.password}            keycloak-oauth:              realm: ${pulsar.auth.keycloak-oauth.realm}              auth-server-url: {{ keycloak\_kernel1\_link }}              grant\_type: ${pulsar.auth.keycloak-oauth.grant\_type}              client\_id: ${pulsar.auth.keycloak-oauth.client\_id}            cached-token-max-expiration-lag: ${pulsar.auth.cached-token-max-expiration-lag}          # Пул потоков для обработчиков принимающих сообщения из ядра        consumer-pool:          # Размер пула          size: 4          # Таймаут на блокировку потока, не прерывает поток, пишет предупреждения в лог          execution-lock-timeout: 1M        # Пул потоков для обработчиков отправляющих сообщения в ядро        producer-pool:          # Размер пула          size: 4          # Таймаут на блокировку потока, не прерывает поток, пишет предупреждения в лог          execution-lock-timeout: 1M         - node-id: {{ node2\_name }}        pulsar:        {...}          client:            serviceUrl: pulsar://{{ pulsar\_kernel2\_link }}        {...}            keycloak-oauth:               auth-server-url: {{ keycloak\_kernel2\_link }}        {...} |

### Настройки разбиения получаемой информации на чанки (опционально)

*Конфигурационный файл имеет настройку по умолчанию для данного параметра. Применять изменения следует только в том случае, если значения по умолчанию не подходят для данного агента.*

Доступна возможность указать способ разбиения получаемой информации на чанки.

Для настройки разбиения получаемой информации на чанки для REST сервера необходимо в конфигурационном файле Агента задать параметр tableParamChunkType, который может принимать значения:

* EXACTLY\_CUT – позволяет разбивать информацию на чанки исходя из размера чанка (размер чанка указывается параметром tableParamChunkSize, например tableParamChunkSize: 100KB);
* EVEN\_ROWS – запрещает разбиение информации между чанками (автономно десериализуемые чанки), не гарантируется соблюдение размера чанка.

Пример раздела конфигурации для настройки REST сервера:

|  |
| --- |
| # Настройки REST сервера исполнения SQL запросов  rest-query-endpoint:    ...    # Размер чанка пользовательского табличного параметра    tableParamChunkSize: 900KB    # Способ разбиения пользовательского табличного параметра на чанки    tableParamChunkType: EXACTLY\_CUT |

Для настройки разбиения получаемой информации на чанки для JDBC драйвера необходимо добавить в свойства драйвера параметр tableParamChunkType, который может принимать значения:

* EXACTLY\_CUT – позволяет разбивать информацию на чанки исходя из размера чанка (размер чанка указывается дополнительным свойством в настройках драйвера tableParamChunkSize, пример tableParamChunkSize =100KB);
* EVEN\_ROWS – запрещает разбиение информации между чанками (автономно десериализуемые чанки), не гарантируется соблюдение размера чанка.

### Настройка организации информационного обмена через API Gateway по HTTPS (только для агента-поставщика)

Если планируется организация информационного обмена через API Gateway по HTTPS, то необходимо выполнить:

1. Скопировать файл cacerts java в локальный каталог

|  |
| --- |
| cp /distr/einfahrt/jdk-17.0.3/lib/security/cacerts /distr/einfahrt/certs/cacerts |

1. Импорт сертификата в java cacerts. Пример раздела конфигурации:

|  |
| --- |
| keytool -importcert -keystore /distr/einfahrt/certs/cacerts -storepass changeit -alias api-gateway -file *cert.crt* |

* 'changeit' - пароль по умолчанию для cacerts;
* *cert.crt* – файл с сертификатом используемым на стороне https сервера.

1. В параметры запуска агента добавить ключи «-Djavax.net.ssl.keyStore=certs/cacerts -Djavax.net.ssl.trustStore=certs/cacerts». При использовании [systemd](#_Запуск_Агента_ПОДД) данные параметры добавить в описание «Environment="JDK\_JAVA\_OPTIONS=…»
2. Произвести настройку модуля для работы по HTPS. Пример раздела конфигурации:

|  |
| --- |
| # Настройка модуля API gateway  api-gateway:  # Параметры HTTP-сервера для приёма запросов на стороне потребителя  server:  host: 0.0.0.0  port: 8171  # Параметры HTTP-клиента для отправки запросов на стороне поставщика  client:  impl: APACHE  options:  # указать адрес и порт https сервера  default-host: *nlb-auth*  default-port: *443*  ssl: true  verifyHost: false  pipelining: true  pipeliningLimit: 100  maxPoolSize: 100 |

### Настройка подключения к агентам

1. Указать URL подключения к агентам:

Для **масштабированного** агента-потребителя, пример раздела конфигурации:

|  |
| --- |
| # Настройки REST сервера исполнения SQL запросов  rest-query-endpoint:    # URL подключения к агентам по JDBC (только для implementation = JDBC)    jdbc-url: jdbc:podd://<instance-1-ip-address>:${query-server.port},<instance-2-ip-address>:${query-server.port},<instance-N-ip-address>:${query-server.port} |

где, <instance-\*-ip-address> - адреса всех экземпляров масштабированного агента.

Для **немасштабированного** агента-потребителя, а также для агента-поставщика, указать значение *jdbc:podd://localhost:${query-server.port}.*

1. Указать идентификатор экземпляра Агента ПОДД. В случае использования масштабируемого агента, значение параметра должно быть уникальным для каждого экземпляра агента. Пример раздела конфигурации:

|  |
| --- |
| # Настройки модуля "информация об агенте"  agent-info:    # идентификатор экземпляра агента (HOSTNAME используется только при запуске под kubernetes, в остальных случаях следует использовать уникальные идентификаторы)    instanceId: ${HOSTNAME:instance-1} |

*В случае запуска под kubernetes с помощью helm chart, данный параметр будет подставлен автоматически.*

# Подготовка и настройка системы для запуска агента без использования docker

## Состав и содержание дистрибутивного пакета

Состав дистрибутива Агента ПОДД СМЭВ (/distr/einfahrt):

* /distr/einfahrt/app/app.jar – исполняемый файл;
* /distr/einfahrt/csp-5.0.11455.tar.gz – дистрибутив CryptoPro CSP;
* /distr/einfahrt/java-csp-5.0.42119-A.zip– дистрибутив CryptoPro JСP;
* /distr/einfahrt/bellsoft-jdk17.0.3-linux-amd64.tar.gz– дистрибутив Java;

Путь /distr/einfahrt указан в качестве примера. Есть возможность использовать любой другой путь, скорректировав соответствующим образом упоминаемые ниже команды.

## Порядок загрузки данных и программ

Администратор УВ осуществляет развёртывание, запуск и настройку Агента ПОДД СМЭВ с помощью данного руководства.

## Предварительные операции (установка «пре-реквизитов»)

Перед запуском Агента ПОДД СМЭВ выполнить от имени пользователя «root» следующие действия:

1. Создать пользователя {{user}} и группу {{user\_group}}, под которым будет работать Агент ПОДД СМЭВ.
2. Установить java

|  |
| --- |
| cd /distr/einfahrt  tar zxvf bellsoft-jdk17.0.3-linux-amd64.tar.gz |

1. Установить системные переменные

|  |
| --- |
| export JAVA\_HOME= /distr/einfahrt/jdk-17.0.3  export PATH=$PATH:$JAVA\_HOME/bin |

Рекомендуется внести данную настройку в profile пользователя.

1. Установить CryptoPro JCP:

|  |
| --- |
| cd /distr/einfahrt  mkdir -p /distr/einfahrt/cryptopro  unzip java-csp-5.0.42119-A.zip  cd java-csp-5.0.42119-A  ./configure.sh  cp -f \*.jar /distr/einfahrt/cryptopro/  cd /distr/einfahrt  rm -rf java-csp-5.0.42119-A |

При наличии лицензионного кода CryptoPro JCP выполнить **от имени пользователя {{user}}** команду.

|  |
| --- |
| java -cp :/distr/einfahrt/cryptopro/\* ru.CryptoPro.JCP.tools.License -serial "{{ jcp\_serial }}" -store |

В случае отсутствия лицензионного кода, CryptoPro JCP будет работать в режиме trial лицензии

1. Установить CryptoPro CSP:

|  |
| --- |
| cd /distr/einfahrt  tar xzvf csp-5.0.11455.tar.gz  rm -f csp-5.0.11455.tar.gz  cd csp-5.0.11455 yum -y install lsb-cprocsp-base\*.rpm lsb-cprocsp-rdr-64-\*.rpm lsb-cprocsp-kc1-64-\*.rpm lsb-cprocsp-capilite-64-\*.rpm lsb-cprocsp-devel-\*.rpm lsb-cprocsp-kc2-64-\*.rpm cprocsp-curl-64-\*.rpm  ./install.sh  cd .. |

При наличии лицензионного кода CryptoPro CSP выполнить пользователем root команду

|  |
| --- |
| /opt/cprocsp/sbin/amd64/cpconfig -license -set {{cprocsp\_license}} |

В случае отсутствия лицензионного кода, CryptoPro CSP будет работать в режиме trial лицензии

1. Установить контейнер с ключами CryptoPro в директорию /var/opt/cprocsp/keys/{{user}}/.

Пользователь {{user}} должен быть **владельцем** данной директории и файлов в ней.

1. Для установки TLS соединения, который использует алгоритмы в соответствии с ГОСТ‑2012, требуется:

* использовать доверенное хранилище с корневыми сертификатами удостоверяющих центров (УЦ), которое находится в дистрибутиве с сертификатом МинЦифры;
* если ключи CryptoPro выданы не МинЦифры, то следует добавить корневые сертификаты данного УЦ, выполнив нижеприведенную команду, где /path/to/some.cer –путь к файлу корневого сертификата УЦ, выдавшего ключи, ‘cryptopro/’ – путь к jar файлам развернутого CryptoPro JCP:

|  |
| --- |
| $JAVA\_HOME/bin/keytool -J-Dkeytool.compat=true \                                           -J-Duse.cert.stub=true \                                           -import \                                           -provider ru.CryptoPro.JCP.JCP \  -providerpath cryptopro/JCSP.jar:cryptopro/JCP.jar:JCPRequest.jar:cryptopro/JCPRevCheck.jar:cryptopro/asn1rt.jar:cryptopro/ASN1P.jar \                                           -storetype HDImageStore \                                           -keystore /distr/einfahrt/certs/cp\_ca\_store \                                           -storepass *changeit* \                                           -file /path/to/some.cer |

*Важно! Ключ CryptoPro должен содержать цепочку сертификатов промежуточных УЦ (кроме корневого). Приложение 1 настоящего документа содержит сведения о добавлении промежуточных сертификатов в ключ.*

1. Выложить в каталог /distr/enfahrt конфигурационный файл application.yml, подготовленный согласно разделу 3.

## Запуск Агента ПОДД СМЭВ

Убедитесь, что пользователь {{user}} имеет доступ к директории, в которой установлен Агент ПОДД СМЭВ (/distr/einfahrt).

Пример раздела конфигурации скрипта для запуска Агента ПОДД СМЭВ с использованием сервиса «systemd»:

|  |
| --- |
| [Unit]  Description="Service for einfahrt"  After=syslog.target    [Service]  Type=simple    WorkingDirectory=/distr/einfahrt  # Важно! Запуск должен производиться от имени того пользователя, для которого ранее были выложены ключи КриптоПро!  User={{user}}  Group={{user\_group}}    Environment="JAVA\_HOME=/distr/einfahrt/jdk-17.0.3"  Environment="JDK\_JAVA\_OPTIONS=--add-exports=java.base/sun.security.util=ALL-UNNAMED \  --add-exports=java.base/sun.security.x509=ALL-UNNAMED" \  --add-exports=java.base/sun.security.pkcs=ALL-UNNAMED" \  --add-exports=java.base/sun.security.provider=ALL-UNNAMED" \  --add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED" \      -Dsaffron.default.charset=UTF-16LE \  -Dsaffron.default.collation.name='UTF-16LE$en\_US' \      -Dsaffron.default.nationalcharset=UTF-16LE \  -XX:MaxRAMPercentage=80.0 \  -XX:+UnlockExperimentalVMOptions \  -XX:InitiatingHeapOccupancyPercent=16 \  -XX:+UseStringDeduplication \  -XX:+G1EagerReclaimHumongousObjects \  -XX:+G1EagerReclaimHumongousObjectsWithStaleRefs \  -XX:G1HeapWastePercent=2 \  -XX:G1MaxNewSizePercent=25 \  -XX:G1MixedGCLiveThresholdPercent=15 \  -XX:+UseG1GC"  ExecStart=/distr/einfahrt/jdk-17.0.3/bin/java -cp app/\*:cryptopro/\* -jar app/app.jar    [Install]  WantedBy=multi-user.target |

Соответствующий файл скрипта, для примера с названием «einfahrt.service», необходимо положить в каталог /etc/systemd/system/.

Далее от имени пользователя root выполнить команду по включению, запуску сервиса и проверке его статуса:

|  |
| --- |
| systemctl daemon-reload  systemctl enable einfahrt  systemctl start einfahrt  systemctl -l status einfahrt |

# Настройка и запуск агента ПОДД с использованием docker

## Предварительные условия

1. На сервере должен быть установлен docker версии не ниже 20.10.12.
2. Создан пользователь, который будет запускать агент.
3. Этот пользователь должен иметь права на загрузку докер-образов и запуск/остановку контейнеров (т.е. добавлен в группу docker).

## Настройка ротации лог-файлов в docker (опционально)

Во избежание переполнения диска лог-файлами, рекомендуется настроить ротацию лог-файлов контейнеров в docker.

Настройка логирования в Docker осуществляется с помощью файла конфигурации /etc/docker/daemon.json. Если этого файла не существует, его необходимо создать. Добавьте в файл следующие настройки логирования:

|  |
| --- |
| {  "log-opts": {  "max-file": "3",  "max-size": "100m"  }  } |

где

- **max-file** – ограничение по количеству файлов (настройки ротации). Максимальное количество файлов журнала, которые могут быть созданы. При превышении количества, самый старый файл удаляется. Действует только при установленном параметре max-size. Положительное целоечисло. По умолчанию 1.

- **max-size** – устанавливает ограничение по размеру лог-файла (k, m или g). По умолчанию -1 (неограниченно)

## Состав и содержание дистрибутивного пакета

Ниже приведён состав дистрибутива Агента ПОДД СМЭВ для docker.

/distr/einfahrt – Агент ПОДД СМЭВ:

* /distr/einfahrt/einfahrt.tgz – docker image для загрузки в систему. Включает в себя все также необходимые версии CryptoPro;
* /distr/einfahrt/run\_agent.sh – скрипт для запуска и перезапуска агента;
* /distr/einfahrt/stop.sh – скрипт для остановки агента;
* /distr/einfahrt/log\_save.sh – скрипт для сохранения лога агента в файл;
* /distr/einfahrt/log.sh – скрипт для просмотра лога агента в реальном времени;
* /distr/einfahrt/diag.sh – скрипт для сбора диагностической информации при обращении в службу поддержки;

Путь /distr/einfahrt использован для примера, не является фиксированным, и может быть изменен на любой другой по желанию пользователя, с соответствующим внесением корректировок при его упоминаниях ниже. Ниже будет использоваться данный путь как пример.

## Подготовка к запуску

Для запуска агента, необходимо после распаковки пакета, сформировать конфигурационный файл application.yml, как указано в разделе 3, и выложить его в каталог /distr/einfahrt

***Важно!*** *При использовании скрипта из пакета поставки для запуска агента, параметр pulsar.auth.keycloak-connection.tls-trust-store.path необходимо задать* ***строго /egov/java/certs/cp\_ca\_store***

*Пример:*

|  |
| --- |
| # Настройки подключения к Pulsar  pulsar:  auth:  keycloak-connection:  tls-trust-store:  path: /egov/java/certs/cp\_ca\_store |

В каталоге /distr/einfahrt:

* создать подкаталог keys, выложить в него полученный ключ CryptoPro в распакованном виде
* создать подкаталог certs, выложить в него файл cp\_ca\_store из пакета поставки
* *(только при использовании https API gateway)* подготовить файл cacerts согласно разделу 3.2.10, сформированный файл выложить в каталог /distr/einfahrt/certs

При наличии файла расширенных настроек логирования customLogLevels.xml, выложить его в каталог /distr/einfahrt

Для запуска агента под docker, полученный docker образ необходимо загрузить в локально установленный docker, на машине, где будет работать агент. Для этого необходимо выложить архив с образом на локальный диск этой машины и выполнить команду:

|  |
| --- |
| docker load < *einfahrt.tgz* |

где *einfahrt.tgz* – имя полученного архива с docker image агента из пакета поставки.

Загрузка должна завершиться без ошибок.

Проверить успешность загрузки можно командой:

|  |
| --- |
| docker images | grep einfahrt |

Вывод будет содержать примерно следующую информацию:

***einfahrt latest*** *c7511824117e 5 days ago 1.2G*

## Запуск агента

Для запуска агента используется скрипт run\_agent.sh, входящий в пакет поставки. Скрипт не требует указания параметров. При выполнении всех шагов, перечисленных в п. 6.3, агент будет успешно запущен, с сообщением о доступных портах. При повторном выполнении скрипта, контейнер будет остановлен и запущен заново.

Для сохранения лога агента в файл, запустите скрипт log-save.sh. Лог будет сохранен в файле log-*дата-время*.txt

Для просмотра лога в режиме реального времени, запустите скрипт log.sh.

В случае необходимости сбора диагностической информации для отправки службе поддержки, запустите скрипт diag.sh

Остановка контейнера осуществляется скриптом stop.sh

*Примечание: в ряде случаев может наблюдаться остановка контейнера через несколько секунд после запуска, с сообщением в логе «java.lang.IllegalStateException: Ошибка вызова функции acquireContext:* ***0x8009001a****».*

*В этом случае следует изменить владельца каталога keys и вложенных подкаталогов и файлов на пользователя с id=1000:*

|  |
| --- |
| chown -R 1000 keys |

*после чего повторить выполнение скрипта запуска контейнера.*

## Информация по лицензированию CryptoPro

Входящие в поставку версии CryptoPro не имеют предустановленной лицензии, и могут работать как trial в течение трех месяцев с момента создания образа (не с момента первого запуска!)

**Для использования trial версии CryptoPro не требуется каких-либо дополнительных действий, касающихся лицензирования**.

В случае, **если у Вас есть полная лицензия** на CryptoPro CSP и CryptoPro JCP, предусмотрена возможность передачи номеров лицензий CSP и JCP при запуске контейнера, при передаче валидных лицензий соответствующие компоненты CryptoPro будут лицензированы на этапе запуска.

Для этого необходимо, чтобы текстовый файл, содержащий сроку с лицензией на CryptoPro CSP был при запуске контейнера доступен внутри контейнера по пути /egov/csp.lic. Аналогично, лицензия для CryptoPro JCP должна быть доступна внутри контейнера в текстовом файле /egov/jcp.lic

При запуске под docker-ом можно для этого монтировать внешние файлы, содержащие номера лицензий, по указанным выше путям.

Для того, чтобы прилагаемый скрипт запуска применил лицензии CryptoPro, **при их наличии**, их необходимо поместить в текстовые файлы licenses/jcp.txt и licenses/csp.txt соответственно.

# Типовые вопросы и проблемы

|  |  |  |
| --- | --- | --- |
| № | Вопрос/Проблема | Решение |
|  | Как скачать актуальный JDBC драйвер? | 1. Драйвер можно получить по адресу http://<ваш\_хост>:8182/api/v1/jdbc-driver 2. C текущими настройками, агент будет доступен:  * на порту 8183 для sql запросов по JDBC * на порту 8192 для sql запросов по REST |
|  | Как проверить соединение Агента с Ядром ПОДД? | 1. Воспользуйтесь запросом "SELECT 1". 2. Воспользуйтесь командой для проверки через REST с использованием утилиты curl:   *curl -X POST -H "Accept-Version:1" -H "Content-Type: application/json" -d '{"sql": {"sql": "select 1"}}' http://<ваш\_хост>:8192/query --silent -m 30* |
|  | В лог-файле присутствует ошибка "Ошибка вызова функции acquireContext: 0x80090016" | 1. При запуске агента без использования docker, убедитесь, что контейнер с ключами выложен в /var/opt/cprocsp/keys/{{user}}/ 2. Пользователь {{user}} имеет доступ на чтение к /var/opt/cprocsp/keys/{{user}}/ 3. Убедитесь, что агент запускается от пользователя {{user}} при запуске без использования docker 4. При использовании docker, убедитесь, что при запуске указан ключ «--user=1000» |
|  | В лог-файле присутствует ошибка "Ошибка вызова функции acquireContext: 0x80090016" | 1. При использовании docker, установите владельца id=1000 на каталог keys рекурсивно chown -R 1000 keys 2. При запуске без использования docker, сделайте пользователя {{user}} владельцем каталога /var/opt/cprocsp/keys/{{user}}/  рекурсивно |
|  | В логе присутствуют сообщения «No such provider: JCP» | Убедитесь, что CryptoPro JCS и CryptoPro CSP установлены корректно, а также путь к jar файлам cryptoPro JCP присутствует в classpath при запуске java |
|  | В логе присутствуют сообщения «unable to find valid certification path to requested target» | Убедитесь, что файл cp\_ca\_store содержит нужную цепочку сертификатов |
|  | В логе присутствуют сообщения «No required SSL certificate was sent» | 1. Убедитесь, что используется верный ключ 2. Убедитесь, что указан корректный адрес {{ gost\_nlb }} 3. Убедитесь, что используется cp\_ca\_store, содержащий необходимые цепочки для подключения к используемому ядру |

# Добавление промежуточных сертификатов в ключ CryptoPro

Для добавления промежуточных сертификатов в ключ CryptoPro следует:

1. Поместить контейнер ключа в папку пользователя, под которым далее будут исполняться команды, например /var/opt/cprocsp/keys/<user> или C:\Users\<user>\Local Settings\Application Data\Crypto Pro\.
2. Создать запрос на сертификат с помощью Java утилиты keytool. Для подписи запроса ГОСТ алгоритмом в Java должно быть установлено CryptoPro JCP или JCSP (см. документацию на CryptoPro).

Пример команды для Java17 с установленным CryptoPro JCP (где «cryptopro/» - имя каталога в котором находятся jar файлы CryptoPro JCSP):

|  |
| --- |
| keytool -J-Dkeytool.compat=true -J-Duse.cert.stub=true -provider ru.CryptoPro.JCP.JCP -providerpath cryptopro/JCSP.jar:cryptopro/JCP.jar:JCPRequest.jar:cryptopro/JCPRevCheck.jar:cryptopro/asn1rt.jar:cryptopro/ASN1P.jar -certreq -alias client\_01 -keysize 256 -sigalg GOST3411\_2012\_256withGOST3410\_2012\_256 -storetype HDImageStore -keystore NONE -storepass 1 -keypass 1234567890 -file client\_01.csr -ext KeyUsage=keyEncipherment,keyAgreement -ext ExtendedkeyUsage=clientAuth |

* Для ключей длиной 256 указывается «-sigalg GOST3411\_2012\_256withGOST3410\_2012\_256», для ключей длиной 512 указывается «-sigalg GOST3411\_2012\_512withGOST3410\_2012\_512».
* Если контейнер ключа имеет русские буквы в алиасе, утилита keytool не увидит его, тогда контейнер можно переименовать, изменив алиас обычным редактором в файле контейнера «name.key». При этом целостность ключа не нарушится.

1. Запрос на сертификат передать в соответствующий УЦ для выпуска сертификата, например тестовый УЦ CryptoPro выдает сертификат на Web-форме https://testgost2012.cryptopro.ru/certsrv/certrqxt.asp.
2. Полученный сертификат импортировать в контейнер ключа. Для этого в первую очередь подготовить файл в формате PKCS#7 (\*.p7b). Данную операцию можно сделать через GUI утилиту CryptoPro Tools (cptools) либо через Windows GUI утилиту «certmgr». Сертификат устанавливается в персональное хранилище пользователя, далее экспортируется в файл PKCS#7 (\*.p7b). Подробнее на рисунках 1 – 5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 – Установка сертификата

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Установка сертификата в персональное хранилище пользователя

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 3 – Экспорт сертификата в CryptoPro Tools

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Экспорт сертификата в certmgr

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Выбор формата PKCS#7 (\*.p7b) в certmgr

1. Импортировать файл \*.p7b в контейнер ключа командой (где «cryptopro/» - имя каталога в котором находятся jar файлы CryptoPro JCSP):

|  |
| --- |
| keytool   -J-Dkeytool.compat=true -J-Duse.cert.stub=true -provider ru.CryptoPro.JCP.JCP -providerpath cryptopro/JCSP.jar:cryptopro/JCP.jar:JCPRequest.jar:cryptopro/JCPRevCheck.jar:cryptopro/asn1rt.jar:cryptopro/ASN1P.jar -importcert -alias client\_01 -storetype HDImageStore -keystore NONE -storepass 1 -keypass 1234567890 -file client\_01.p7b |

Аналогичным образом можно выгрузить несколько сертификатов (цепочку) в файл PKCS#7 (\*.p7b), чтобы потом загрузить их в контейнер. Сертификаты из контейнера ключа передаются другой стороне для проверки при установке TLS соединения, соответственно, если в доверенном хранилище второй стороны (сервера или клиента) не хватает промежуточных сертификатов для проверки подписи по цепочке, сертификаты могут быть переданы из контейнера ключа. При этом в контейнер можно класть не всю цепочку, а только сертификаты, которых не хватает у второй стороны для построения всей цепочки. Стороны посылают сертификаты для проверки только из контейнера своего ключа и проверяют подпись по цепочке для полученных от второй стороны сертификатов по сертификатам в своем хранилище доверенных. Проверить наличие цепочки в контейнере можно через утилиты CryptoPro (рисунки 6, 7).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Проверка наличия нескольких сертификатов в контейнере

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Проверка наличия нескольких сертификатов в контейнере

Для промежуточных сертификатов сервер nginx делает проверку атрибута BasicConstraints=ca:true.... , если атрибут отсутствует, nginx отклонит соединение с ошибкой в логе: «client SSL certificate verify error: (24:invalid CA certificate)».

# Особенности запуска под kubernetes

**Использование лицензий CryptoPro при запуске агента под kubernetes**

*Данная информация актуальна только при наличии и использовании полных лицензий на CryptoPro.* ***Неприменимо при использовании trial лицензий****.*

При запуске под kubernetes возможно использование secret-ов для создания внутри контейнера указанных файлов. Пример secret-а:

|  |
| --- |
| apiVersion: v1 data:   csp.lic: *XXXXX*   jcp.lic: *YYYYY* kind: Secret metadata:   name: cryptopro type: Opaque |

где *XXXXX* – **закодированный в base64** лицензионный номер CSP, *YYYYY* – **закодированный в base64** лицензионный номер JCP.

Пример использования данного secret в helm chart Агента

*Указанный раздел должен отсутствовать в чарте в случае использования trial лицензий!*

|  |
| --- |
| volumes:  - name: cryptopro-license  secret:  secretName: cryptopro  volumeMounts:  - mountPath: /egov/csp.lic  name: cryptopro-license  subPath: csp.lic  - mountPath: /egov/jcp.lic  name: cryptopro-license  subPath: jcp.lic |

# Установка компонента prohibitor (опционально)

## 1.1 Состав и содержание дистрибутивного пакета

Состав дистрибутива Сервиса проверки полномочий:

/distr/prohibitor – Сервис проверки полномочий:

* /distr/prohibitor/app/app.jar – исполняемый файл;
* /distr/prohibitor/csp-5.0.11455.tar.gz – дистрибутив CryptoPro CSP;
* /distr/prohibitor/java-csp-5.0.42119-A.zip– дистрибутив CryptoPro JСP;
* /distr/prohibitor/bellsoft-jdk17.0.3-linux-amd64.tar.gz– дистрибутив Java;
* /distr/prohibitor/cp\_ca\_store – JKS с поддержкой корневых сертификатов Ростелекома;
* /dist/prohibitor/application.yml – заготовка конфигурационного файла;
* /distr/prohibitor/customLogLevels.xml – файл описания настроек логирования
* /dist/prohibitor/liquibase/\* – файлы, описывающие все операции, которые необходимо выполнить над базой Сервиса проверки полномочий.

Путь /distr/prohibitor указан в качестве примера, можно использовать любой другой путь, скорректировав соответствующим образом упоминаемые ниже команды.

## 1.2 Порядок загрузки данных и программ

Администратор УВ осуществляет развёртывание, настройку и запуск Сервиса проверки полномочий с помощью данного руководства.

## 1.3 Требования к среде для развёртывания Сервиса проверки полномочий

Минимальные требования к характеристикам сервера для развёртывания Сервиса проверки полномочий приведены в Таблица 2.

Таблица 2 – Минимальные требования к характеристикам сервера для развёртывания Сервиса проверки полномочий

| Характеристики | Агент ПОДД СМЭВ |
| --- | --- |
| ОС | CentOS 7.9 |
| ЦПУ | 4-х ядерный процессор |
| Оперативная память | 8 ГБ |
| Дисковое пространство | 20 ГБ |
| Требование к БД | БД PostgreSQL версии 11.14 или выше |

## 1.4 Предварительные операции (установка «пре-реквизитов»)

Перед запуском Сервиса проверки полномочий выполнить от имени пользователя «root» следующие действия:

1. Создать пользователя {{user}} и группу {{user\_group}}, под которым будет работать Агент.
2. Установить java

|  |
| --- |
| cd /distr/prohibitor  tar zxvf bellsoft-jdk17.0.3-linux-amd64.tar.gz |

1. Установить системные переменные

|  |
| --- |
| export JAVA\_HOME= distr/prohibitor/jdk-17.0.3  export PATH=$PATH:$JAVA\_HOME/bin |

Рекомендуется внести данную настройку в profile пользователя.

1. Установить CryptoPro JCP:

|  |
| --- |
| cd /distr/prohibitor  mkdir -p /distr/prohibitor/cryptopro  unzip java-csp-5.0.42119-A.zip  cd java-csp-5.0.42119-A  ./configure.sh  cp -f \*.jar /distr/prohibitor/cryptopro/  cd /distr/prohibitor  rm -rf java-csp-5.0.42119-A |

При наличии лицензионного кода CryptoPro JCP выполнить **от имени пользователя {{user}}** команду.

|  |
| --- |
| java -cp :/distr/prohibitor/cryptopro/\* ru.CryptoPro.JCP.tools.License -serial "{{ jcp\_serial }}" -store |

В случае отсутствия лицензионного кода, CryptoPro JCP будет работать в режиме trial лицензии

1. Установить CryptoPro CSP:

|  |
| --- |
| cd /distr/prohibitor  tar xzvf csp-5.0.11455.tar.gz  rm -f csp-5.0.11455.tar.gz  cd csp-5.0.11455 yum -y install lsb-cprocsp-base\*.rpm lsb-cprocsp-rdr-64-\*.rpm lsb-cprocsp-kc1-64-\*.rpm lsb-cprocsp-capilite-64-\*.rpm lsb-cprocsp-devel-\*.rpm lsb-cprocsp-kc2-64-\*.rpm cprocsp-curl-64-\*.rpm  ./install.sh  cd .. |

При наличии лицензионного кода CryptoPro CSP выполнить команду

|  |
| --- |
| /opt/cprocsp/sbin/amd64/cpconfig -license -set {{cprocsp\_license}} |

В случае отсутствия лицензионного кода, CryptoPro CSP будет работать в режиме trial лицензии

1. Установить контейнер с ключами CryptoPro в директорию /var/opt/cprocsp/keys/{{user}}/ (ключ следует использовать от Агента ПОДД СМЭВ).

Пользователь {{user}} должен иметь доступ на чтение к данной директории и файлам в ней.

1. Для установки TLS соединения, который использует алгоритмы в соответствии с ГОСТ-2012, требуется:

* использовать доверенное хранилище с корневыми сертификатами удостоверяющих центров (УЦ), которое находится в дистрибутиве с сертификатом МинЦифры.
* если ключи CryptoPro выданы не МинЦифры, то следует добавить корневые сертификаты данного УЦ, выполнив нижеприведённую команду, где /path/to/some.cer путь к файлу корневого сертификата УЦ, выдавшего ключи:

|  |
| --- |
| $JAVA\_HOME/bin/keytool -J-Dkeytool.compat=true \                                           -J-Duse.cert.stub=true \                                           -import \                                           -provider ru.CryptoPro.JCP.JCP \                                           -providerpath cryptopro/JCSP.jar:cryptopro/JCP.jar:JCPRequest.jar:cryptopro/JCPRevCheck.jar:cryptopro/asn1rt.jar:cryptopro/ASN1P.jar \  -storetype HDImageStore \                                           -keystore /distr/prohibitor/cp\_ca\_store \                                           -storepass changeit \                                           -file /path/to/some.cer |

*Важно! Ключ CryptoPro должен содержать цепочку сертификатов промежуточных УЦ (кроме корневого). Приложение 1 настоящего документа содержит сведения о добавлении промежуточных сертификатов в ключ.*

1. В конфигурационном файле «application.yml» задать переменные (в заготовке обозначены звездочками):

* {{ agentid }} – мнемоника ИС;
* {{ key\_alias }} – алиас контейнера CryptoPro, установленного ранее (на шаге 3);
* {{ key\_password }} – пароль контейнера CryptoPro;
* {{ ca\_store\_path }} – фактический путь до директории c файлом «cp\_ca\_store» из дистрибутива (в данном примере - /distr/einfahrt);
* {{ ca\_pass }} – пароль от «cp\_ca\_store», по умолчанию на поставляемое в дистрибутиве хранилище установлен пароль changeit, который рекомендуется заменить в процессе установки. Убедитесь, что пользователь, запускающий приложение, имеет доступ на чтение к этому файлу;
* {{ gost\_nlb }} – адрес сервиса авторизации. Данные настройки уже прописаны в заготовке конфигурационного файла.
* {{ postgres\_host }} – адрес БД Postgres;
* {{ postgres\_port }} – порт БД Postgres;
* {{ postgres\_pass }} – пароль от БД Postgres.

1. Переменные "postgres\_\*" указываются в разделе настроек подключения к PostgreSQL:

|  |
| --- |
| # параметры подключения к базе и настройки пулла коннектов  db:  host: {{ postgres\_host }}  port: {{ postgres\_port }}  user: dbadmin\_prohibitor  password: {{ postgres\_pass }}  database: prohibitor  pool:  max-size: 10 |

Пример раздела конфигурации для настройки Сервиса проверки полномочий для работы в геораспределённой конфигурации (необходимо перечислить все развёрнутые Ядра с указанием ссылки на pulsar Ядра).

*Примечание: данные настройки уже прописаны в заготовке конфигурационного файла.*

|  |
| --- |
| # Настройки подключения ко всем ЦОД с установленным ПО ядра  data-center:   ...   nodes:      - node-id: {{ node1\_name }}        # Настройки подключения к Pulsar        pulsar:          # Включение Round-Robin балансировки по спику брокеров          enable-balancing: true          # Максимально допустимое время передачи сообщения одному из брокеров          send-timeout: 35s          # Длительность периода вывода брокеров из пула, при его недоступности          max-dismissal-delay: 5s          # Список подключений к брокерам          client:            serviceUrl: pulsar://{{ pulsar\_kernel1\_link }}          consumer:            subscriptionName: ${pulsar.consumer.subscriptionName}          producer:            maxPendingMessages: 1000            blockIfQueueFull: false          auth:            enabled: true            service-token: ''            keycloak-connection:              tls-verify-hostname: false              tls-trust-store:                path: {{ ca\_store\_path }}                password: {{ ca\_pass }}              tls-key:                alias: {{ key\_alias }}                password: {{ key\_password }}            keycloak-oauth:              realm: pulsar              auth-server-url: {{ keycloak\_kernel1\_link }}              grant\_type: client\_credentials              client\_id: {{ agent }}            cached-token-max-expiration-lag: 5s          # Пул потоков для обработчиков принимающих сообщения из ядра        consumer-pool:          # Размер пула          size: 4          # Таймаут на блокировку потока, не прерывает поток, пишет предупреждения в лог          execution-lock-timeout: 1M        # Пул потоков для обработчиков отправляющих сообщения в ядро        producer-pool:          # Размер пула          size: 4          # Таймаут на блокировку потока, не прерывает поток, пишет предупреждения в лог          execution-lock-timeout: 1M         - node-id: {{ node2\_name }}        pulsar:        {...}          client:            serviceUrl: pulsar://{{ pulsar\_kernel2\_link }}        {...}            keycloak-oauth:               auth-server-url: {{ keycloak\_kernel2\_link }}        {...} |

## 1.5 Настройка БД для работы Сервиса проверки полномочий

Подключится к БД Postgresql с административным аккаунтом и выполнить SQL команды ниже:

|  |
| --- |
| CREATE DATABASE prohibitor;  CREATE USER dbadmin\_prohibitor WITH ENCRYPTED PASSWORD '<postgres\_pass>' NOSUPERUSER CREATEROLE CREATEDB INHERIT LOGIN NOREPLICATION;  ALTER DATABASE prohibitor OWNER TO dbadmin\_prohibitor;    Команды ниже следует выполнять подключенным к БД prohibitor с административным аккаунтом  GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO dbadmin\_prohibitor;  GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO dbadmin\_prohibitor; |

С помощью инструмента liquidbase развернуть в созданной БД скрипты.

Для этого в консоли выполнить следующие команды:

|  |
| --- |
| export LIQUIBASE\_URL=jdbc:postgresql://{{ postgres\_host }}:{{ postgres\_port }}/prohibitor;  export LIQUIBASE\_USERNAME=dbadmin\_prohibitor;  export LIQUIBASE\_PASSWORD={{ postgres\_pass }};  export LIQUIBASE\_CHANGELOG=/dist/prohibitor/liquibase/changelog.xml;  liquibase update |

Если отсутствует Liquidbase, его необходимо установить по приложенной инструкции (<https://docs.liquibase.com/install/liquibase-linux.html>).

## 1.6 Запуск Сервиса проверки полномочий

Убедитесь, что пользователь {{user}} имеет доступ к директории, в которой установлен Агент (/distr/prohibitor).

Пример скрипта для запуска Сервиса проверки полномочий с использованием сервиса «systemd»:

|  |
| --- |
| [Unit]  Description="Service for prohibitor"  After=syslog.target    [Service]  Type=simple    WorkingDirectory=/distr/prohibitor  User={{user}}  Group={{user\_group}}    Environment="JAVA\_HOME=/distr/prohibitor/jdk-17.0.3"  Environment="JDK\_JAVA\_OPTIONS=--add-exports=java.base/sun.security.util=ALL-UNNAMED \  --add-exports=java.base/sun.security.x509=ALL-UNNAMED" \  --add-exports=java.base/sun.security.pkcs=ALL-UNNAMED" \  --add-exports=java.base/sun.security.provider=ALL-UNNAMED" \  --add-exports=java.base/sun.security.tools.keytool=ALL-UNNAMED" \      -Dsaffron.default.charset=UTF-16LE \  -Dsaffron.default.collation.name='UTF-16LE$en\_US' \      -Dsaffron.default.nationalcharset=UTF-16LE \  -XX:MaxRAMPercentage=80.0 \  -XX:+UnlockExperimentalVMOptions \  -XX:InitiatingHeapOccupancyPercent=16 \  -XX:+UseStringDeduplication \  -XX:+G1EagerReclaimHumongousObjects \  -XX:+G1EagerReclaimHumongousObjectsWithStaleRefs \  -XX:G1HeapWastePercent=2 \  -XX:G1MaxNewSizePercent=25 \  -XX:G1MixedGCLiveThresholdPercent=15 \  -XX:+UseG1GC"  ExecStart=/distr/prohibitor/jdk-17.0.3/bin/java -cp app/\*:cryptopro/\* -jar app/app.jar  [Install]  WantedBy=multi-user.target |

Соответствующий файл скрипта, для примера с названием prohibitor.service, необходимо разместить в каталоге /etc/systemd/system/.

Далее выполнить команду по включению, запуску сервиса и проверить его статус:

|  |
| --- |
| systemctl daemon-reload  systemctl enable prohibitor  systemctl start prohibitor  systemctl -l status prohibitor |

## 1.6 Особенности запуска под docker

prohibitor, упакованный в docker image, уже имеет предустановленными все необходимые компоненты (java, CryptoPro CSP, CryptoPro JCP).

Входящие в поставку версии CryptoPro не имеют предустановленной лицензии, и могут работать как trial в течение трех месяцев с момента создания образа (не с момента первого запуска!)

**Для использования trial версии CryptoPro не требуется каких-либо дополнительных действий, касающихся лицензирования**.

|  |
| --- |
| В случае, **если у Вас есть полная лицензия** на CryptoPro CSP и CryptoPro JCP, предусмотрена возможность передачи номеров лицензий CSP и JCP при запуске контейнера, при передаче валидных лицензий соответствующие компоненты CryptoPro будут лицензированы на этапе запуска.  Для этого необходимо, чтобы текстовый файл, содержащий сроку с лицензией на CryptoPro CSP был при запуске контейнера доступен по пути /egov/csp.lic. Аналогично, лицензия для CryptoPro JCP должна быть доступна при запуске контейнера в текстовом файле /egov/jcp.lic  При запуске под docker-ом можно для этого монтировать внешние файлы, содержащие номера лицензий, по указанным выше путям. |

*Важно! Для использования приведенного ниже скрипта запсука агента под докером, ключ CryptoPro должен находиться в подкаталоге keys, файл cp\_ca\_store – в каталоге certs. Их наличие обязательно.*

*Лицензии CryptoProCryptoPro,* ***при их наличии****, необходимо поместить в файлы licenses/jcp.txt и licenses/csp.txt соответственно.*

Для того, чтобы prohibitor можно было запустить под docker, полученный docker образ необходимо загрузить в локально установленный docker, на машине, где он будет работать. Для этого необходимо выложить архив с образом на локальный диск этой машины и выполнить

|  |
| --- |
| docker load < *prohibitor.tgz* |

где *prohibitor.tgz* – имя полученного архива с docker image

Загрузка должна завершиться без ошибок.

Проверить успешность загрузки можно командой

|  |
| --- |
| docker images | grep prohibitor |

Вывод будет содержать примерно следующую информацию:

***prohibitor latest*** *c7511824117e 5 days ago 1.2GB*

Скрипт для запуска prohibitor под docker:

|  |
| --- |
| #!/bin/bash  IMG=prohibitor:latest  NAME=$(echo ${IMG} |cut -d: -f1)  echo "Checking for already started container, stop it if running"  [ ! -z $(docker ps |awk '{print $NF}'|grep "^${NAME}$") ] && docker stop ${NAME}  [ ! -z $(docker ps -a |awk '{print $NF}'|grep "^${NAME}$") ] && docker rm ${NAME}  DIR=$(pwd)  # установка прав на объекты  for OBJECT in certs keys; do  find ${OBJECT} \( -type f -exec chmod 0666 {} + \) -o \( -type d -exec chmod 0777 {} + \)  done  # **Если есть** лицензии КриптоПро, они должны находиться в текстовом виде в файлах licenses/**csp**.lic и licenses/**jcp**.lic  # При отсутствии внешних файлов с лицензией, будет использовано trial лицензирование cryptoPro  for OBJECT in jcp csp; do  [ -f licenses/${OBJECT}.txt ] && LICENSE=" ${LICENSE} --mount type=bind,source=${DIR}/licenses/${OBJECT}.txt,target=/egov/${OBJECT}.lic,readonly"  done  # порты, по которым доступен агент  for OBJECT in 8692; do  PORTS="${PORTS} -p ${OBJECT}:${OBJECT}"  done  echo "Starting docker container"  # **Важно!** Для корректного использования ключей CryptoPro,  # процесс в контейнере docker должен выполняться пользователем app (id=1000)  # использование ключа «--user=1000» обязательно!  docker run \  -d \  --user=1000 \  ${PORTS} \  ${LICENSE} \  --name ${NAME} \  --mount type=bind,source=${DIR}/certs,target=/egov/java/certs,readonly \  --mount type=bind,source=${DIR}/keys,target=/var/opt/cprocsp/keys/app \  --mount type=bind,source=${DIR}/application.yml,target=/egov/java/application.yml,readonly \  --env JDK\_JAVA\_OPTIONS=" -XX:+UnlockExperimentalVMOptions \  -XX:InitiatingHeapOccupancyPercent=16 \  -XX:+UseStringDeduplication \  -XX:+G1EagerReclaimHumongousObjects \  -XX:+G1EagerReclaimHumongousObjectsWithStaleRefs \  -XX:G1HeapWastePercent=2 \  -XX:G1MaxNewSizePercent=25 \  -XX:G1MixedGCLiveThresholdPercent=15 \  -XX:+UseG1G" \  ${IMG}  [ $? -eq 0 ] && echo "Application started. Container name: ${NAME}. Available port: 8692" || echo "Error starting docker." |

Получить лог-файл агента, запущенного под docker данным скриптом, можно командой

|  |
| --- |
| docker logs einfahrt |

1. Перечисленные документы размещены на технологическом портале СМЭВ3 – <https://smev3.gosuslugi.ru/portal/> [↑](#footnote-ref-2)