Государственное автономное учреждение Республики Коми

«Центр информационных технологий»

|  |  |
| --- | --- |
| СОГЛАСОВАНО | УТВЕРЖДАЮ |
| ЗАКАЗЧИК: | ИСПОЛНИТЕЛЬ: |
|  |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 года | « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 года |
|  |  |

**РУКОВОДСТВО АДМИНИСТРАТОРА**

Наименование системы:

Модуль МФЦ Региональной комплексной информационной системы

«Госуслуги – Республика Коми»

На 34 листах

Сыктывкар, 2017

**СОДЕРЖАНИЕ**

[ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ 4](#_Toc499805034)

[1 ВВЕДЕНИЕ 5](#_Toc499805035)

[1.1 Область применения 5](#_Toc499805036)

[1.2 Краткое описание возможностей 5](#_Toc499805037)

[2 УСЛОВИЯ ПРИМЕНЕНИЯ 6](#_Toc499805038)

[2.1 Требования к клиентской конфигурации 6](#_Toc499805039)

[2.2 Требования к серверной конфигурации 6](#_Toc499805040)

[2.3 Требования к программному обеспечению Системы 7](#_Toc499805041)

[2.4 Программная поддержка рабочего места 7](#_Toc499805042)

[2.5 Требования к подготовке администраторов 8](#_Toc499805043)

[3 ПОДГОТОВКА К РАБОТЕ 9](#_Toc499805044)

[4 ОПИСАНИЕ ОПЕРАЦИЙ 10](#_Toc499805045)

[4.1 Установка сервисов 10](#_Toc499805046)

[4.1.1 case\_manager 10](#_Toc499805047)

[4.1.2 production\_calendar 12](#_Toc499805048)

[4.1.3 template\_service 13](#_Toc499805049)

[4.1.4 frgu-service 15](#_Toc499805050)

[4.1.5 service\_registry 17](#_Toc499805051)

[4.1.6 smev3\_response\_converter 18](#_Toc499805052)

[4.2 Установка клиента СМЭВ 3 20](#_Toc499805053)

[4.3 Установка Системы 23](#_Toc499805054)

[4.3.1 Установка базы данных 23](#_Toc499805055)

[4.3.2 Установка компонентов Системы 24](#_Toc499805056)

[4.4 Обновление сервисов 28](#_Toc499805057)

[4.5 Обновление Системы 29](#_Toc499805058)

[4.6 Проверка версии приложения Системы 29](#_Toc499805059)

[5 ПОДДЕРЖКА РАБОЧЕГО МЕСТА ОПЕРАТОРА СИСТЕМЫ 30](#_Toc499805060)

[5.1 Проблема подписания документов ЭП СП 30](#_Toc499805061)

[5.2 Проблема печати документов в браузере 33](#_Toc499805062)

[5.3 Памятка по настройке сканера 33](#_Toc499805063)

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

| **№ п/п** | **Сокращение** | **Расшифровка** |
| --- | --- | --- |
|  | ДУЛ | Документ, удостоверяющий личность |
|  | ЕСИА | Единая система идентификации и аутентификации в инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме |
|  | МВ запрос | Межведомственный запрос |
|  | МФЦ | Многофункциональный центр предоставления государственных и муниципальных услуг |
|  | РСИА | Региональная система идентификации и аутентификации |
|  | Система | Модуль МФЦ Региональной комплексной информационной системы «Госуслуги – Республика Коми» |
|  | ФИО | Фамилия, имя, отчество |
|  | ФРГУ | Федеральный реестр государственных и муниципальных услуг |
|  | ЦТО | Центр телефонного обслуживания |

# ВВЕДЕНИЕ

## Область применения

Система предназначена для автоматизации деятельности сотрудников многофункциональных центров предоставления государственных и муниципальных услуг на территории Республики Коми.

## Краткое описание возможностей

Основными возможностями Системы являются:

1. Прием и обработка запросов заявителей о предоставлении государственных или муниципальных услуг.
2. Прием и обработка информации из информационных систем органов, предоставляющих государственные услуги, и органов, предоставляющих муниципальные услуги.
3. Поиск и предоставление информации о государственных и муниципальных услугах.
4. Информирование заявителей и отслеживание статуса исполнения заказанной услуги.
5. Управление записями пользователей, справочниками и набором услуг.
6. УСЛОВИЯ ПРИМЕНЕНИЯ

## Требования к клиентской конфигурации

Для технической поддержки функционирования Системы в клиентской конфигурации рекомендуется следующее оборудование (или совместимое с ним).

Таблица 1. Технические характеристики рабочего места администратора Системы

| № п/п | Элемент КТС | Конфигурация |
| --- | --- | --- |
|  | Компьютер | Процессор Celeron 2GHz |
| Оперативная память 8 Гб и выше |
| Сетевая карта |
| Жесткий диск объемом не ниже 40 Гб |
|  | Монитор | LCD 17″ и выше разрешением 1024х768 |

## Требования к серверной конфигурации

Для технической поддержки функционирования Системы в серверной конфигурации требуются 2 сервера с рекомендуемыми параметрами (или аналогичными им), описанными ниже.

Для сервера баз данных рекомендуются следующие характеристики:

Таблица 2. Технические характеристики сервера баз данных

| № п/п | Элемент КТС | Конфигурация |
| --- | --- | --- |
|  | Компьютер | Процессор Xeon 2.2 GHz |
| Оперативная память 8 Гб и выше |
| Сетевая карта |
| Жесткий диск объемом не ниже 200 Гб |
|  | Монитор | LCD 17″ и выше разрешением 1024х768 |
|  | Операционная система | Linux Ubuntu Server 14.04 LTS 64-bit. |
|  | Инфраструктура | Доступ в интернет |

Для сервера приложений рекомендуются следующие характеристики:

Таблица 3. Технические характеристики сервера приложений

| № п/п | Элемент КТС | Конфигурация |
| --- | --- | --- |
|  | Компьютер | Процессор Xeon 2.2 GHz |
| Оперативная память 8 Гб и выше |
| Сетевая карта |
| Жесткий диск объемом не ниже 200 Гб |
|  | Монитор | LCD 17″ и выше разрешением 1024х768 |
|  | Операционная система | Linux Ubuntu Server 14.04 LTS 64-bit. |
|  | Инфраструктура | Доступ в интернет |

## Требования к программному обеспечению Системы

Программное обеспечение Системы состоит из:

1. Программного обеспечения, представляющего собой совокупность программных средств, разработанных вне связи с созданием Системы – общее программное обеспечение (ОПО). В состав ОПО входят следующие программы:

* RVM 1.25.x (установка для одного пользователя: Single-User Install Location: ~/.rvm/)
* Ruby 2.3.5
* nginx 1.4.6+
* java (JDK 1.8.0\_151)
* Сервер MySQLверсии не ниже 5.5
* Сервер PostgreSQL
* Wkhtmltopdf (версия приложения, работающая без X.org сервера)
* ruby\_brightbox\_bundler\_version 1.11.2+

1. Программного обеспечения, представляющего собой совокупность программ, разработанных при создании Системы – специальное программное обеспечение (СПО).

## Программная поддержка рабочего места

Для программной поддержки функционирования рабочего места оператора требуется:

* Операционная система Microsoft Windows 7 и выше.
* Браузер Google Chrome последней официальной версии.
* cryptoPro jcp 1.0.54
* КриптоПро ЭЦП Browser Plug-in версии не ниже 2.0.12888
* Java 8 версии не ниже 8.0.1310.11

Для программной поддержки функционирования рабочего места администратора требуется:

* Операционная система Ubuntu Server 14.04 LTS 64-bit.
* Браузер Google Chrome последней официальной версии.
* Git последней официальной версии.
* docker 1.12.4+
* Ansible 2.2.0.0+
* Общесистемные приложения Системы, входящие в состав дистрибутива.

## Требования к подготовке администраторов

Уровень квалификации администратора должен соответствовать следующим требованиям:

* Глубокое понимание Системы на уровне технологий работы;
* Знание реляционных баз данных и SQL-запросов;
* Навыки реализации различных режимов работы операционных систем;
* Понимание основ работы веб-технологий;
* Владение техническим английским языком для работы с технической документацией на английском языке;
* Основы работы в операционной системе на базе ядра Linux – установка серверной операционной системы Linux, установка программного обеспечения, настройка прав доступа, редактирование переменных окружения;
* Понимание основ и правил XML, редактирование XML-файлов;
* Понимание основ и правил YAML, редактирование YML-файлов.

1. ПОДГОТОВКА К РАБОТЕ

Перед установкой сервисов необходимо запустить контейнеры из образов registry.it.vm:6000/cit/nginx:**<tag>** и registry.it.vm:6000/cit/docker-gen:**<tag>**, где **<tag>** в большинстве случаев равен latest, но может иметь и другое значение в зависимости от версии образа.

1. ОПИСАНИЕ ОПЕРАЦИЙ

## Установка сервисов

 *Примечание.* Действия по установке сервисов можно производить с любого компьютера с Ansible. Компьютер должен иметь доступ к серверу, на который будет разворачиваться приложение.

### case\_manager

Чтобы установить сервис case\_manager, необходимо выполнить следующие действия:

1. Склонируйте репозиторий dictionaries-api

git clone http://gitlab.it.vm/rkis/case\_manager.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/case\_manager.git

1. Перейдите в каталог с приложением:

сd case\_manager

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

RCM\_DB\_USER: <user> # настройка подключения к базе PostgreSQL

RCM\_DB\_PASS: <pass> # настройка подключения к базе PostgreSQL

RCM\_DB\_HOST: <host> # настройка подключения к базе PostgreSQL

RCM\_DB\_NAME: <db> # настройка подключения к базе PostgreSQL

RCM\_LOG\_LEVEL: debug # уровень журналирования debug|info|warn|error|fatal

RCM\_BIND\_HOST: 0.0.0.0 # можно не менять

RCM\_PORT: 7070 # можно не менять

RCM\_STOMP\_HOST: 10.33.68.207 # настройка подключения к ActiveMQ

RCM\_STOMP\_PORT: 61613 # настройка подключения к ActiveMQ

RCM\_STOMP\_SUBSCRIBE\_TO: 'smev3.response.queue' # очeредь, на которую подписывается case\_manager

RCM\_STOMP\_SEND\_TO: 'smev3.queue' # очeредь, в которую case\_manager посылает сообщения

VIRTUAL\_HOST: case-manager.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 7070 # должно совпадать с RCM\_PORT

docker\_run\_labels:

app\_name: '{{ app\_name }}'

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

### production\_calendar

Чтобы установить сервис production\_calendar, необходимо выполнить следующие действия:

1. Склонируйте репозиторий dictionaries-api

git clone http://gitlab.it.vm/rkis/production\_calendar.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/production\_calendar.git

1. Перейдите в каталог с приложением:

cd production\_calendar

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

PROD\_CALEND\_DB\_CONNECTION\_URL: 'postgres://<user>:<pass>@<host>:<port>/prod\_calend'

PROD\_CALEND\_PORT: 8080 # можно не менять

VIRTUAL\_HOST: production-calendar.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 8080 # должно совпадать с PROD\_CALEND\_PORT

docker\_run\_labels:

app\_name: '{{ app\_name }}'

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

### template\_service

Чтобы установить сервис template\_service, необходимо выполнить следующие действия:

1. Склонируйте репозиторий dictionaries-api

git clone http://gitlab.it.vm/rkis/template\_service.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/template\_service.git

1. Перейдите в каталог с приложением:

cd template\_service

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_published\_ports: 8090:8010 # первое число это номер порта, на котором будет доступен gui

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

TMPLT\_MONGO\_DB: templates # название базы данных, к которой обращается сервис

TMPLT\_MONGO\_HOST\_URI: mongodb.preprod.rk.local:27017 # адрес базы данных

TMPLT\_LOG\_LEVEL: warn # уровень логирования. Менять не нужно

TMPLT\_PORT: 8010 # можно не менять

VIRTUAL\_HOST: template-service.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 8010 # должно совпадать с TMPLT\_PORT

docker\_run\_labels:

app\_name: '{{ app\_name }}'

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

### frgu-service

Чтобы установить сервис frgu-service, необходимо выполнить следующие действия:

1. Склонируйте репозиторий dictionaries-api

git clone http://gitlab.it.vm/rkis/frgu-service.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/frgu-service.git

1. Перейдите в каталог с приложением:

cd frgu-service

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_published\_ports: 8090:8010 # первое число это номер порта, на котором будет доступен gui

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

SD\_FILESTORAGE\_HOST: '10.33.68.207:8888' # адрес СХД

SD\_MEMCACHE\_HOST: '10.33.68.207:11211' # адрес Memcache

SD\_SPHINX\_HOST: '192.168.252.15' # адрес индексера Sphinx

SD\_SPHINX\_PORT: 9312 # порт, к которому обращается сервис

SD\_DB\_NAME: <db> # название базы данных, к которой обращается сервис

SD\_DB\_HOST: <host> # адрес базы данных

SD\_DB\_USER: <user> # имя пользователя

SD\_DB\_PWD: <password> # пароль

SD\_REGIONAL\_WSDL: 'http://91.227.93.142:8080/RGU\_WAR\_2/ws-services/registryInfoService?wsdl' # менять не нужно

SD\_FEDERAL\_WSDL: 'http://frgu.gosuslugi.ru/ws-services/registryInfoService?wsdl' # менять не нужно

SD\_REDIS\_PORT: 6379 # адрес redis

SD\_REDIS\_HOST: 192.168.252.14 # порт Redis

SD\_REDIS\_DB: 1 # название базы данных Redis

SD\_PORT: 8010 # можно не менять

SD\_CRON\_TIME: '00 00 \* \* \*' # менять не нужно

VIRTUAL\_HOST: frgu.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 8010 # должно совпадать с SD\_PORT

docker\_run\_labels:

app\_name: frgu-service

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

### service\_registry

Чтобы установить сервис service\_registry, необходимо выполнить следующие действия:

1. Склонируйте репозиторий:

git clone http://gitlab.it.vm/microservices/service\_registry.git

или

git clone ssh://git@gitlab.it.vm:2222/microservices/service\_registry.git

1. Перейдите в каталог с приложением:

cd frgu-service

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_published\_ports: 8090:8010 # первое число это номер порта, на котором будет доступен gui

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

SR\_LOG\_LEVEL: debug # уровень логирования. Менять не нужно

SR\_BIND\_HOST: 0.0.0.0 # можно не менять

SR\_PORT: 8016 # менять не нужно

SR\_MONGO\_DB: service\_registry # название базы данных, к которой обращается сервис

SR\_MONGO\_HOST\_URI: mongodb.preprod.rk.local:27017 # адрес базы данных

VIRTUAL\_HOST: service-registry.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 8016 # должно совпадать с SR\_PORT

docker\_run\_labels:

app\_name: frgu-service

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

### smev3\_response\_converter

Чтобы установить сервис smev3\_response\_converter, необходимо выполнить следующие действия:

1. Склонируйте репозиторий:

git clone

http://gitlab.it.vm/microservices/smev3\_response\_converter.git

или

git clone ssh://git@gitlab.it.vm:2222/microservices/smev3\_response\_converter.git

1. Перейдите в каталог с приложением:

cd smev3\_response\_converter

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_published\_ports: 8090:8010 # первое число это номер порта, на котором будет доступен gui

docker\_run\_variables:

RACK\_ENV: production # переменная окружения, используемая Ruby приложением. Менять не нужно

SR\_HOST: 10.33.68.168 # ip-адрес/dns-запись, за которой расположено приложение Service Registry

SR\_PORT: 8016 # порт, на котором расположено приложение Service Registry

VIRTUAL\_HOST: smev3rc.preprod.rk.local # url, по которому приложение будет доступно

VIRTUAL\_PORT: 8016 # должно совпадать с SR\_PORT

docker\_run\_labels:

app\_name: smev3rc.preprod.rk.local

app\_env: '{{ app\_env }}'

app\_url: '{{ docker\_run\_variables["VIRTUAL\_HOST"] }}'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

## Установка клиента СМЭВ 3

Чтобы установить сервис smev-client, необходимо выполнить следующие действия:

1. Склонируйте репозиторий:

git clone http://gitlab.it.vm/rkis/smev-client.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/smev-client.git

1. Перейдите в каталог с приложением:

cd smev-client

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[app\_server:children]

preprod # название окружения

[preprod] # название окружения

10.33.68.12 # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

app\_env: preprod # название окружения

docker\_published\_ports: 8090:8010 # первое число это номер порта, на котором будет доступен gui

docker\_run\_variables:

TEST\_MODE: false # для переключения в тестовый режим. На препроде и проде менять не нужно

SMEV\_FTP\_URL: 85.143.127.136 # настройки подключения к файловому серверу СМЭВ

SMEV\_FTP\_USER: anonymous # настройки подключения к файловому серверу СМЭВ

SMEV\_FTP\_PASSWORD: smev # настройки подключения к файловому серверу СМЭВ

SMEV\_WSDL: 'http://85.143.127.199:7500/ws?wsdl' # различаются на проде и препроде

SMEV\_MONGO\_HOST: '<host>:<port>' # адрес сервера с MongoDB

SMEV\_MONGO\_DB: 'smev3' # название базы

SMEV\_STOMP\_HOST: 10.33.68.207 # настройка подключения к ActiveMQ

SMEV\_STOMP\_PORT: 61613 # настройка подключения к ActiveMQ

SMEV\_STOMP\_DEST: 'dispatch.queue' # менять не нужно

SMEV\_STOMP\_DEST\_V2: 'smev3.response.queue' # менять не нужно

SMEV\_STOMP\_SOURCE: 'smev3.queue' # очередь, которую разбирает smev-client

SMEV\_LOG\_LEVEL: DEBUG # уровень журналирования debug|info|warn|error|fatal

CRYPTO\_DETACHED\_URL: 'http://<host>:<port>/smev-crypto/web/api/v3/detached' # glassfish, на котором развернут smev-crypto

CRYPTO\_PKCS7\_URL: 'http://<host>:<port>/smev-crypto/web/api/v3/pkcs7' # glassfish, на котором развернут smev-crypto

SMEV\_MIN\_UPDATE\_INTERVAL: 60

SMEV\_MAX\_UPDATE\_INTERVAL: 600

SMEV\_NODE\_ID: '{{ app\_env }}'

SMEV\_METADATA\_HOST: 10.33.68.206 # настройка расположения приложения XD

SMEV\_METADATA\_PORT: 4001 # настройка расположения приложения XD

SMEV\_CACHE\_HOST: 10.33.68.207 # настройка расположения приложения Memcached

SMEV\_CACHE\_PORT: 11211 # настройка расположения приложения Memcached

SMEV\_FILESTORAGE\_HOST: 10.33.68.207 # настройка расположения приложения file-storage

SMEV\_FILESTORAGE\_PORT: 8888 # настройка расположения приложения file-storage

docker\_run\_labels:

app\_name: '{{ app\_name }}'

app\_env: '{{ app\_env }}'

app\_url: 'smev-client'

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

## Установка Системы

### Установка базы данных

Чтобы установить базу данных MySQL, необходимо выполнить следующие действия на сервере баз данных:

1. Выполните команду установки MySQL:

sudo apt-get update && sudo apt-get install mysql-server mysql-client libmysqlclient-dev

1. Разрешите подключаться к серверу MySQL с любых серверов. Для этого измените значение параметра bind-address на 0.0.0.0 в конфигурационном файле mysql (/etc/mysql/my.cnf).
2. Перезапустите сервис mysql:

sudo service mysql restart

1. Создайте базу данных MySQL для приложения Системы. Значения, выделенные знаками <…> и полужирным начертанием, необходимо изменить.

mysql -uroot -p'**<password>**' -e 'CREATE DATABASE **`<database\_name>**`'

Замените:

* <password> на пароль пользователя root от сервера MySQL;
* <database\_name> на наименование базы данных. Рекомендуется назвать базу данных «mfc».

1. Создайте учетную запись пользователя с полными правами. Значения, выделенные знаками <…> и полужирным начертанием, необходимо изменить.

mysql -uroot -p'**<password>**' -e 'GRANT ALL PRIVILEGES ON `**<database\_name>**`.\* TO `**<user\_name>**`@`localhost` IDENTIFIED BY "**<user\_password>**" WITH GRANT OPTION'

mysql -uroot -p'**<password>**' -e 'GRANT ALL PRIVILEGES ON `**<database\_name>**`.\* TO `**<user\_name>**`@`%` IDENTIFIED BY "**<user\_password>**" WITH GRANT OPTION'

Замените:

* <password> на пароль пользователя root от сервера MySQL;
* <database\_name> на наименование базы данных, созданной в предыдущем шаге;
* <user\_name> на имя пользователя базы данных. Рекомендуется назвать пользователя user\_mfc;
* <user\_password> на пароль пользователя.

### Установка компонентов Системы

Чтобы установить Систему, необходимо выполнить следующие действия:

1. Склонируйте репозиторий:

git clone http://gitlab.it.vm/rkis/mfc.git

или

git clone ssh://git@gitlab.it.vm:2222/rkis/mfc.git

1. Перейдите в каталог с приложением:

cd mfc

1. Перейдите к нужному тегу, совпадающему с номером версии:

git checkout <версия>

1. Убедитесь в наличии файлов с настройками приложения. Необходимо, чтобы были заполнены следующие файлы и при необходимости зашифрованы.

# для препрод окружения

cm/deploy/inventory/preprod.ini

cm/deploy/group\_vars/preprod/main.yml

cm/deploy/group\_vars/preprod/app\_config.yml

# для продакшн окружения

cm/deploy/inventory/production.ini

cm/deploy/group\_vars/production/main.yml

cm/deploy/group\_vars/production/app\_config.yml

Примеры заполнения файлов:

# cm/deploy/inventory/preprod.ini

[preprod] # название окружения

example.com # адрес docker хоста, на котором будет запущено приложение

# cm/deploy/group\_vars/preprod/main.yml

---

folder\_slave: '<путь до приложения MFC\_slave>'

folder\_master: '<путь до приложения MFC\_master>'

# cm/deploy/group\_vars/preprod/app\_config.yml

---

# Конфигурация подключения к БД

database\_url: example.com #адрес, по которому расположена БД

database\_name: MFC #имя БД

database\_user: user\_MFC #имя пользователя БД

database\_pass: 123456 #пароль пользователя БД

### Конфигурация приложения

# УЭК

uecard\_sdk\_password: mfc1 #пароль sdk из системы УЭК

uecard\_secret\_key: "JAkKLvoLPPOyewHiYglDuYvpIdRFOTciYfOIriDUOTIUiyirtD4eSyPoPKjwtS7&jbIuF&eX70" #ключ из системы УЭК

uecard\_serviceURL: http://91.227.93.54/Autorize #URL системы УЭК

uecard\_localProviderID: 5 #идентификатор аккаунта в системе УЭК

# Сервис метаданных (ЕРРС)

metadata:

host: example.com #имя сервера, на котором запущено приложение ЕРРС

port: 4000 #порт, на котором работает приложение ЕРРС

# Сервис хранения шаблонов, печатных форм и файлов

docview:

host: example.com #имя сервера, на котором запущено приложение docview

port: 8080 #порт, на котором работает приложение docview

# memcached — настройки подключения к кэшу

memcache:

host: example.com #имя сервера, на котором запущен memcache

port: 11211 #порт, на котором работает memcache

# Справочники (green\_proxy)

dictionaries:

host: example.com #имя сервера, на котором запущено приложение green-proxy

port: 9200 #порт, на котором работает приложение green-proxy

# Сервис клиентов (РПП)

cabinet:

host: example.com #имя сервера, на котором запущено приложение РПП

port: 3100 #порт, на котором работает приложение РПП

# Очередь сообщений

mq\_mode: slave #определяет режим работы приложения

mq\_base: mfc-preprod

mq\_smev\_host: example.com #экземпляр брокера ActiveMQ для работы с МИ СМЭВ

mq\_smev\_port: 61613 #порт, на котором брокер ActiveMQ принимает сообщения stomp

mq\_events\_host: example.com #экземпляр брокера ActiveMQ для работы с пользователями

mq\_events\_port: 61712 #порт, на котором брокер ActiveMQ принимает сообщения stomp

# сервис атуентификации РСИА

rsia:

host: example.com

port: 80

recepient: MFC-ARM #уникальный код, который используется для подключения к системам в административном интерфейсе (раздел «Система авторизации», вкладка «Системы»)

redirect: true #перенаправление пользователей на РСИА для авторизации

api\_token: 4b9e737ae2a35852b6c3419c7de59f9795534a37

# сервис подписания

sign:

host: example.com #имя сервера, на котором запущен сервис подписания

port: 8888 #порт, на котором работает сервис подписания

esia\_auth:

sp: #сервис провайдера

issuer: http://example.com:80/ #идентификатор поставщика ЕСИА

# Реестр услуг (ФРГУ)

sd:

host: example.com #имя сервера, на котором запущен ФРГУ

port: 80 #порт, на котором работает ФРГУ

# Реестр шаблонов

template\_service\_host: example.com #имя сервера, на котором запущен Реестр шаблонов

template\_service\_port: 80 #порт, на котором работает Реестр шаблонов

# Менеджер смен

shifts\_manager:

host: example.com #имя сервера, на котором запущен менеджер смен

port: 80 #порт, на котором работает менеджер смен

timeout: 10 #время ожидания соединения

# Сервис организационной структуры

org\_struct\_host: example.com #имя сервера, на котором запущен сервис оргструктуры

org\_struct\_port: 80 #порт, на котором работает сервис оргструктуры

org\_struct\_timeout: 10 #время ожидания соединения

# Сервис Fiasco

fiasco\_host: example.com #имя сервера, на котором запущен сервис Fiasco

fiasco\_port: 80 #порт, на котором работает сервис Fiasco

fiasco\_timeout: 10 #время ожидания соединения

# Менеджер заявок

case\_manager\_host: example.com #имя сервера, на котором запущен сервис case\_manager

case\_manager\_port: 80 #порт, на котором работает сервис case\_manager

# Микросервис prod\_calend

prod\_calend\_host: example.com #имя сервера, на котором запущен сервис prod\_calend

prod\_calend\_port: 80 #порт, на котором работает сервис prod\_calend

organization\_id: mydocuments\_komi #менять не надо

# Клиент СМЭВ 3

smev\_client\_host: example.com #имя сервера, на котором запущен Клиент СМЭВ 3

smev\_client\_port: 8090 #порт, на котором работает Клиент СМЭВ 3

# ИС МДМ

monitoring\_of\_activities:

cron\_schedule: '0 21 \* \* \*' #менять не надо. Расписание сбора и отправки данных в ИС МДМ

resend\_count: 3 #количество повторных отправок в случае ошибки

1. Выполните команду деплоя в зависимости от того, в какую среду производится установка.

# В случае, если есть нет зашифрованных файлов с переменными:

make run-container ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

# В случае, если есть зашифрованные файлы, команда ниже спросит пароль

make run-container-with-vault ENV=<окружение> USER=<под каким пользователем> TAG=<версия приложения>

## Обновление сервисов

Если все сервисы были уже установлены, достаточно выполнить деплой следующей командой. Значения, выделенные знаками <…> и полужирным начертанием, необходимо изменить.

make run-container-with-vault ENV=**<deploy\_environment>** USER=**<deploy\_user>** TAG=**<docker\_tag>**

Замените:

* <deploy\_environment> на наименование окружения. Например, production.
* <deploy\_user> на имя пользователя, под которым осуществляется деплой.
* <docker\_tag> на тэг docker образа.

## Обновление Системы

Обновите Систему до последней версии следующей командой. Замените **<system\_version>** на версию Системы.

make deploy-prod USER=login1 ENV=production VER=**<system\_version>**

## Проверка версии приложения Системы

Для проверки версии приложения Системы необходимо выполнить следующие действия:

1. Перейдите в корневую директорию приложения;
2. Откройте файл /config/build\_info.yml (при его наличии) или файл /lib/mfc-arm/version.rb.
3. ПОДДЕРЖКА РАБОЧЕГО МЕСТА ОПЕРАТОРА СИСТЕМЫ

## Проблема подписания документов ЭП СП

При попытке подписать документы может не сработать ЭП СП. Это связано с устаревшими версиями КриптоПро ЭЦП Browser Plug-in и Java.

Для обновления **КриптоПро ЭЦП Browser Plug-in** выполните следующие действия:

1. Сбросьте кэш браузера.
2. Скачайте актуальную версию плагина для пользователей с официального сайта КриптоПро. Ссылка для скачивания:

<https://www.cryptopro.ru/products/cades/plugin/get_2_0>

1. Запустите файл и установите КриптоПро ЭЦП Browser plug-in.
2. Откройте **Настройки ЭЦП Browser plug-in** через меню Пуск (Рисунок 1).

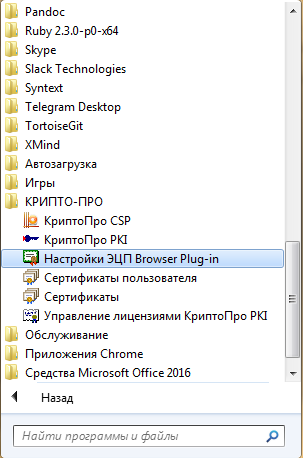


Рисунок 1. Настройки КриптоПро ЭЦП Browser plug-in

1. Отобразится вкладка браузера **Настройки ЭЦП Browser plug-in** (Рисунок 2).

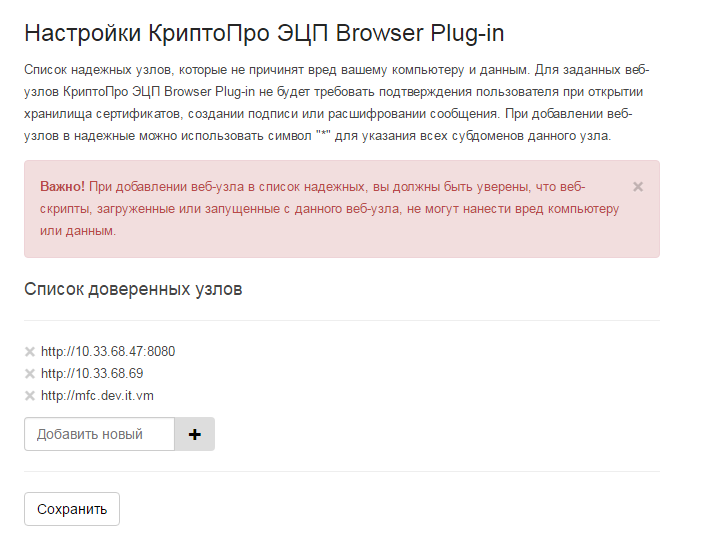


Рисунок 2. Настройки КриптоПро ЭЦП Browser plug-in

1. Добавьте адреса Системы preproduction и production окружения, если еще не добавлены:
   1. Вставьте адрес в строку **Добавить новый**.
   2. Нажмите кнопку .
   3. Нажмите кнопку **Сохранить**.
2. Перезапустите все используемые браузеры.

Для обновления **Java** до последней версии выполните следующие действия:

1. Откройте **Check for Updates** через меню Пуск (Рисунок 3).

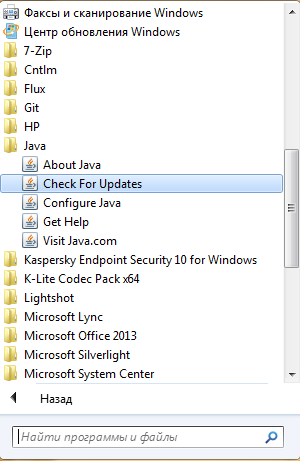


Рисунок 3. Проверка обновлений Java

1. Отобразится окно **Java control panel**, вкладка **Update** (Рисунок 4).

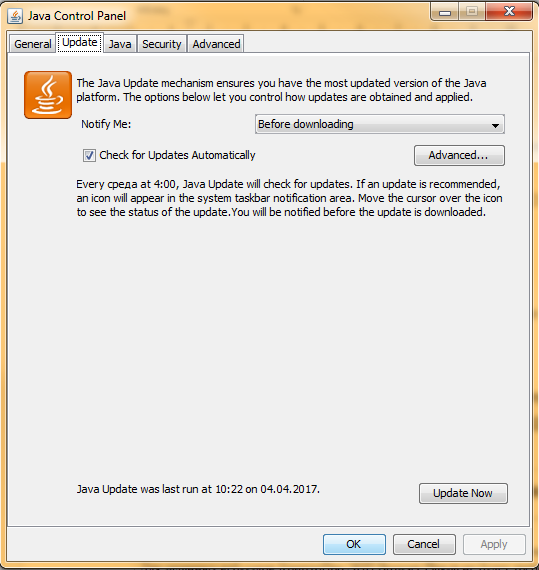


Рисунок 4. Обновление Java

1. Нажмите кнопку **Update Now** внизу окна.
2. Дождитесь окончания обновления.
3. Добавьте адреса Системы preproduction и production окружения, если еще не добавлены:
   1. Перейдите во вкладку **Security**.
   2. Нажмите кнопку **Edit Site List**.
   3. Нажмите кнопку **Add.**
   4. Введите адрес в отобразившуюся строку.
   5. Нажмите кнопку **Add**.
   6. Нажмите кнопку **Ок**.

Если по данному сценарию обновиться невозможно, то скачайте и установите последнюю версию Java с официального сайта <https://java.com/ru/download/> .

## Проблема печати документов в браузере

При попытке распечатать расписку о приеме документов, памятку или заявление в АИС МФЦ 2.0 может не открыться форма печати. Это связано с настройками браузера, потому что браузер Google Chrome автоматически блокирует всплывающие окна. Эту функцию можно отключить.

1. Нажмите на значок Всплывающие окна заблокированы в адресной строке.
2. Выберите опцию **Всегда показывать всплывающие окна с сайта [адрес Системы]**
3. Нажмите кнопку **Готово**.
4. Попробуйте распечатать документ еще раз.

## Памятка по настройке сканера

Чтобы настроить качество скан-копий, выполните следующие действия:

1. Откройте программу сканирования.
2. Автоматически отобразится окно со свойствами сканирования, вкладка **Основные**.
3. Перейдите на вкладку **Изображения** (Рисунок 5).
4. Укажите **Цветное сканирование** → **Полноцветный**.
5. Укажите **Разрешение** → **300dpi**.

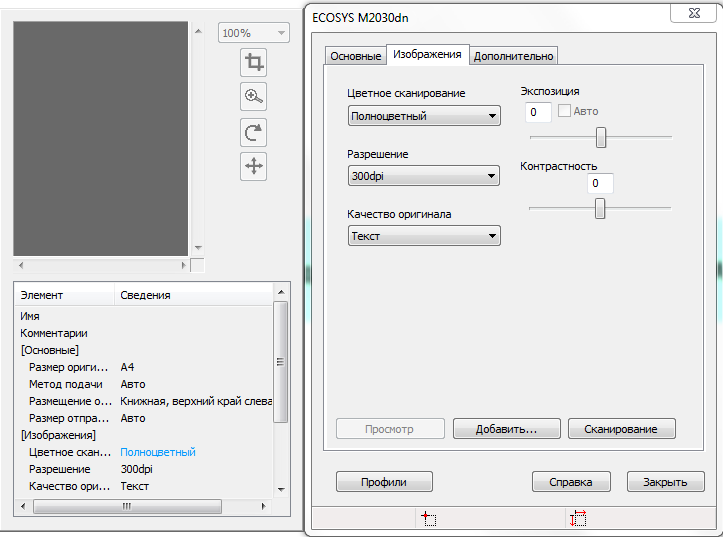


Рисунок 5. Настройки сканера