Инструкция по интеграции Комплекса программных средств информационной системы (КПСИС) в прикладную систему (ПС) для обеспечения взаимодействия с Единой системой идентификации физических и юридических лиц с целью получения сервисов единой аутентификации

1. Состав КПСИС

КПСИС состоит из следующих компонентов:

– модуль поддержки OpenID Connect (OIDC);

– программный TLS-сервер;

– модуль терминала;

– сервисы выработки и проверки ЭЦП;

– сервис предварительного шифрования;

– сервис генерации псевдослучайной числовой последовательности (ПСЧП);

– сервис контроля целостности;

– сервис генерации личного ключа прикладной системы и выпуска запроса на сертификат открытого ключа (СОК).

Модуль терминала, сервисы выработки и проверки ЭЦП, сервис предварительного шифрования, сервис генерации ПСЧП, сервис контроля целостности, сервис генерации личного ключа прикладной системы и выпуска запроса на СОК содержатся в Docker-контейнерах.

КПСИС функционирует под управлением ОС Centos 7.

2. Подготовительные действия

Для начала работы КПСИС и использования сервисов аутентификации ЕС ИФЮЛ необходимо:

выполнить установку КПСИС;

сгенерировать личный ключ ПС;

издать запрос на выпуск СОК ПС;

направить запрос в ГосСУОК;

получить СОК ПС;

направить заявку на регистрацию ПС оператору ЕС ИФЮЛ (к заявке необходимо приложить СОК ПС);

зарегистрировать терминал в Центре управления терминалов инфраструктуры открытых ключей облегченных сертификатов, получить enrollpwd;

получить от оператора ЕС ИФЮЛ: clientID, client\_secret, СОК ЕС ИФЮЛ;

выполнить настройку компонентов КПСИС (поместить личный ключ и СОК в каталог с сервисами предварительного шифрования и выработки ЭЦП);

выполнить интеграцию модуля поддержки OpenID Connect в ПС.

3. Установка КПСИС

3.1. Установка модуля терминала; сервисов выработки и проверки ЭЦП; сервиса предварительного шифрования; сервиса генерации псевдослучайной числовой последовательности; сервиса контроля целостности; сервиса генерации личного ключа прикладной системы и выпуска запроса на СОК.

3.1.1. Установить на сервер следующие программы:

– «Docker» (<https://docs.docker.com/install/>);

– «Docker-compose» (<https://docs.docker.com/compose/install/>).

– «Node.js » (<https://node.js.org/en/>

Для установки на сервер программы «Docker», ввести в терминале следующие команды:

sudo yum install -y yum-utils \

device-mapper-persistent-data \

lvm2

sudo yum-config-manager \

--add-repo \

<https://download.docker.com/linux/centos/docker-ce.repo>

sudo yum install docker-ce docker-ce-cli containerd.io.

systemctl enable docker

systemctl start docker

Для установки на сервер программы «Docker-compose», ввести в терминале следующие команды:

sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose.

Для установки на сервер программы «Node.js», ввести в терминале следующие команды:

curl -sL https://rpm.nodesource.com/setup\_10.x | sudo bash –

sudo yum install nodejs

3.1.2. Перейти в каталог с проектом КПСИС, введя в терминале команду:

cd esiful-kpsis , где esiful-kpsis – наименование каталога с проектом КПСИС. Наименование может быть другим.

3.1.3. Изменить файл client/config/production.json. В значении ключа "СОК" указать СОК ПС, а в значение "nest\_port" – порт, на котором будет запущен КПСИС.

Пример:

"nest\_port": "3000",

"COK":"<значение СОК в формате BASE64>"

3.1.4. Ввести в терминале команды:

sudo chmod +x install\_docker\_images.sh

sudo ./install\_docker\_images.sh

3.3.4. Для запуска Docker-контейнеров ввести в терминале команду:

docker-compose up

3.3.5. Для остановки Docker-контейнеров нажать сочетание клавиш: Ctrl + C

3.3.6. Для запуска Docker-контейнеров в фоновом режиме ввести в терминале команду:

docker-compose start

3.3.7. Для остановки Docker-контейнеров фонового режима ввести в терминале команду:

docker-compose stop

3.2. Установка TLS-сервера

Запустить файл «kpsistls\_server\_install-1.0-1.x86\_64.rpm» на сервере, функционирующем под управлением ОС семейства Linux, введя в терминале команду:

sudo rpm -ivh kpsistls\_server\_install-1.0-1.x86\_64.rpm.

4. Генерация личного ключа и издание запроса на выпуск СОК ПС

Для генерации личного ключа необходимо направить http-запрос на сервис генерации личного ключа и выпуска запроса на СОК на адрес сервиса вида «http://host:port/api/v1/gencertreq». Адрес размещения сервиса генерации личного ключа ПС и выпуска запроса на СОК указывается при настройке КПСИС.

4.1. Запрос:

Endpoint: http://host:port/api/v1/gencertreq.

Method: POST.

Content-Type: application/json Parameters:

| № пп | Параметр | Тип | Значение |
| --- | --- | --- | --- |
| 1. | key\_param | JSON Object | содержит параметры для генерации ключа, состоит из следующих объектов: |
| 2. | key\_param.security\_level | int | уровень стойкости (128, 192, 256) |
| 3. | key\_param.iteration\_count | int | количество итераций для алгоритма построения ключа по паролю PBKDF2 (не меньше 10000) |
| 4. | key\_param.users\_num | int | количество участников для разделения секрета (не больше 16) |
| 5. | key\_param.treshold\_num | int | минимальное число частичных секретов, необходимых для восстановления секрета (не меньше 2) |
| 6. | key\_param.passwords | JSON Array | массив паролей, на которых будут зашифрованы частичные секреты, число паролей должно совпадать с числом участников |
| 7. | cert\_req\_param | JSON Object | содержит параметры для запроса на СОК |
| 8. | cert\_req\_param.type | String | тип СОКа, может быть:  entity - сертификат юридического лица  service - сертификат сервиса |
| 9. | cert\_req\_param.cert\_info | JSON Object | информация о субъекте сертификата, состав зависит от типа |

Состав cert\_req\_param.cert\_info для сервиса:

| № пп | Параметр | Тип | Значение |
| --- | --- | --- | --- |
| 1. | cert\_req\_param.cert\_info.organization | String | название организации |
| 2. | cert\_req\_param.cert\_info.common\_name | String | DNS-имя, ID-сервера и т.д. |
| 3. | cert\_req\_param.cert\_info.description | String | Общее наименование сервера, устройства, процесса |
| 4. | cert\_req\_param.cert\_info.state | String | область |
| 5. | cert\_req\_param.cert\_info.locality | String | населённый пункт |
| 6. | cert\_req\_param.cert\_info.street | String | юридический адрес |
| 7. | cert\_req\_param.cert\_info.gis\_oid | String | OID для поля Идентификатор ГИС |
| 8. | cert\_req\_param.cert\_info.gis\_id | String | Идентификаторы ГИС, зарегистрированных в Государственном регистре информационных систем |
| 9. | cert\_req\_param.cert\_info.email | String | адрес электронной почты |

4.2. Сервис возвращает следующий ответ:

Content-Type: application/json Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| № пп | Параметр | Тип | Значение |
|  | error | String | код ошибки |
|  | key | String | зашифрованный личный ключ в PEM формате |
|  | secret\_shards | JSON Array of String | массив частичных секретов в PEM формате |
|  | cert\_request | String | запрос на СОК в Base64 кодировке |

4.3. Пример запроса

{

"key\_param": {

"security\_level": 128,

"iteration\_count": 10000,

"users\_num": 5,

"treshold\_num": 3,

"passwords": [

"privet",

"poka",

"privet",

"poka",

"privet" ] },

"cert\_req\_param": {

"type": "service",

"cert\_info": {

"common\_name": "id сервиса",

"description": "Какой-то сервис",

"organization": "ООО Организация",

"state": "Минский район",

"locality": "Минск",

"street": "ул. Улица, 322, 212212",

"email": "org@example.com",

"gis\_oid": "1.2.112.1.2.1.1",

"gis\_id": "1.1111.11.1111" } }}

4.4. Пример ответа

{ "cert\_request": "<запрос на СОК в формате BASE64>",

"error": "0",

"key": "<личный ключ в ключевом контейнере в формате BASE64>",

"secret\_shards": [

"-----BEGIN ENCRYPTED PRIVATE KEY-----

<значение частичного секрета 1 в формате BASE64>

-----END ENCRYPTED PRIVATE KEY-----",

"-----BEGIN ENCRYPTED PRIVATE KEY-----

<значение частичного секрета 2 в формате BASE64>

-----END ENCRYPTED PRIVATE KEY-----",

"-----BEGIN ENCRYPTED PRIVATE KEY-----

<значение частичного секрета 3 в формате BASE64>

-----END ENCRYPTED PRIVATE KEY-----",

"-----BEGIN ENCRYPTED PRIVATE KEY-----

<значение частичного секрета 4 в формате BASE64>

-----END ENCRYPTED PRIVATE KEY-----",

"-----BEGIN ENCRYPTED PRIVATE KEY-----

<значение частичного секрета 1 в формате BASE64>

-----END ENCRYPTED PRIVATE KEY-----" ] }

После получения запроса на выпуск СОК от сервиса генерации личного ключа и выпуска запроса на СОК изданный запрос необходимо направить в ГосСУОК для получения СОК.

Полученный СОК, личный ключ в ключевом контейнере, зашифрованные частичные секреты пароли от них необходимо использовать для запуска сервисов предварительного шифрования, проверки ЭЦП.

5. Регистрация ПС в ЕС ИФЮЛ, получение client\_id.

Для регистрации ПС в ЕСИФЮЛ необходимо заполнить регистрационную форму в соответствии с приложением 1. Направить регистрационную форму оператору ЕС ИФЮЛ любым удобным способом. К регистрационной форме необходимо приложить СОК регистрируемой ПС.

После завершения процедуры регистрации, ПС должна получить от оператора ЕС ИФЮЛ client\_id, СОК ЕС ИФЮЛ.

6. Настройка компонентов КПСИС

6.1. Настройка ТЛС Сервера

Для настройки TLS-сервера, функционирующего под управлением ОС семейства Linux, необходимо скорректировать конфигурационный файл «kpsistls.conf», который находится в каталоге «/opt/kpsistls/kpsistls.conf», следующим образом:

– в директиве «ssl\_certificate» указать путь к СОК сервера;

– в директиве «ssl\_certificate\_key» оставить значение «engine:ntctwe:config»;

– в директиве «ssl\_crl» указать путь к актуальным спискам отзыва СОК в формте PEM;

– в директиве «ssl\_client\_certificate» указать путь к доверенным корневым СОК;

– в поддирективе «listen» директивы «server» указать порт, который будет прослушивать TLS-сервер;

– в поддирективе «proxy\_pass» директивы «server» указать IP-адрес т порт конечного сервиса.

Также необходимо скорректировать конфигурационный файл «ntctwe.conf», который находится в каталоге «/opt/kpsistls/sbin/ntctwe.conf», следующим образом:

– в поддирективе «id» директивы «key» указать путь к файлу ключа сервера;

– в поддирективе «treshold» директивы «secret» указать пороговое количество частичных секретов, необходимых для восстановления секрета;

– в поддирективе «shard» директивы «secret» указать путь к файлу частичного секрета. Количество поддиректив «shard» должно совпадать с количеством файлов секретов.

6.2. Настройка сервиса предварительного шифрования и сервиса выработки ЭЦП

Поместить личный ключ и СОК ПС, полученные в ГосСУОК, в каталог /keys  
докер-контейнера. СОК должен иметь кодировку base64 и имя: cert.pem.

Поместить личный ключ и частичные секреты в каталог /keys. Ключ должен иметь название key.pem, частичные секреты должны иметь названия: shard0.pem, shard1.pem, shard2.pem, shard3.pem, shard4.pem.

Указать в конфигурационном файле docker-compose.yml пути к расположению частичных секретов, пароли от контейнеров с частичными секретами.

6.3. Настройка сервиса проверки ЭЦП

Поместить СОС в каталог /data/COK/CRL докер-контейнера.

Поместить корневые СОК в каталог /data/COK/СА докер-контейнера.

7. Настройка модуля поддержки OpenID connect.

7.1. Входные и выходные данные для модуля поддержки OpenID connect и КП:

| HTTP-метод | Метод веб-сервиса | Описание метода веб-сервиса |
| --- | --- | --- |
| POST | /api/v2/login | Формирование ссылки для аутентификации в ЕСИФЮЛ |
| POST | /select\_auth | Первый шаг для начала протокола аутентификации |
| POST | /api/v2/login\_callback | Получение билета аутентификации, билета доступа, билета обновления и профайла пользователя |
| POST | /api/v2/logout | Выход из ЕСИФЮЛ |
| POST | /tls\_init | Установка защищенного tls-соединения |

7.2. Входные параметры запроса для метода /api/v2/login приведены в таблице 2.

Таблица 2

| Наименование | Описание | Тип параметра | Тип данных |
| --- | --- | --- | --- |
| settings.authority | URL-адрес сервера идентификации ЕС ИФЮЛ | JSON | string |
| settings.client\_id | Уникальный идентификатор клиентского приложения, зарегистрированного в сервере идентификации ЕС ИФЮЛ | JSON | string |
| settings.grant\_type | Тип потока авторизации. Значение по умолчанию "authorization\_code" | JSON | string |
| settings.response\_type | Тип ответа, требуемый от сервера идентификации ЕС ИФЮЛ. Значение по умолчанию "code" | JSON | string |
| settings.scope | Список данных, которые запрашиваются у сервера идентификации ЕСИФЮЛ. Значение по умолчанию "openid" | JSON | string |
| settings.redirect\_uri | URI перенаправления клиентского приложения для получения ответа от провайдера ЕС ИФЮЛ. | JSON | string |
| settings.post\_logout\_redirect\_uri | URI перенаправления клиентского приложения после выхода из сервера идентификации ЕС ИФЮЛ. | JSON | string |
| settings.prompt | Обязательный параметр для получения билета обновления. Значение по умолчанию "consent" | JSON | string |
| settings.state | Значение, используемое для поддержания состояния между запросом и обратным вызовом, будет возвращено после успешной аутентификации. | JSON | string |

Пример запроса:

POST /api/v1/login

HTTP/1.1

Accept: application/json

Content-Type: application/json

{"settings": {

"authority": "https://esiful.nces.by:8081",

"client\_id": "5c0SbSd9IiDIlhmF7q\_V0pq-uGp8z6y8dzWHQIisFIM",

"grant\_type": "authorization\_code",

"response\_type": "code",

"scope" : "openid profile offline\_access",

"redirect\_uri": "http://localhost:8080",

"post\_logout\_redirect\_uri": "http://localhost:8080",

"prompt": "consent",

"state": "niitzi.by"}}

Выходные параметры ответа для метода /api/v2/login приведены в таблице 3.

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип данных | Тип параметра |
| enveloped\_and\_signed\_auth\_url | Ccылка для перехода в сервер идентификации ЕС ИФЮЛ | string | JSON |
| code\_verifier | Случайно сгенерированная последовательность, которую необходимо сохранить и после успешной аутентификации предъявить в /api/v1/login\_callback | string | JSON |
| signed\_data\_to\_check\_in\_cp | Параметр, необходимый для передачи в КП для протоколов BAuth и SAuth | string | JSON |

Пример ответа:

HTTP/1.1

Content-Type: application/json

{

enveloped\_and\_signed\_auth\_url: 'https://esiful.nces.by:8081/authz?settings=&client\_id=NWMwU2JTZDlJaURJbGhtRjdxX1YwcHEtdUdwOHo2eThkeldIUUlpc0ZJTQ==',

code\_verifier:'Hduyqezr3v99sVuW5GDTn96ExMlHzKT1Y8oWyGPchZ~Yu3URIUR',

signed\_data\_to\_check\_in\_cp: ''

}

7.3 Параметры метода [/](https://apimgw.core.oais.by:9447/store/apis/info?name=Minjust-30104&version=v1&provider=CORE.OAIS.BY%2Fminjust#/operations/default/get_egr_short_num)select\_auth

Вызов метода

http://127.0.0.1:8084[/](https://apimgw.core.oais.by:9447/store/apis/info?name=Minjust-30104&version=v1&provider=CORE.OAIS.BY%2Fminjust#/operations/default/get_egr_short_num)select\_auth

Входные параметры:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Описание | Тип данных | Обязатель-ный | Примечание |
| data | данные для ЭЦП в формате Base64 | Base64 | да | Signed\_ПС (Scope\_ПС, Enveloped\_СИ (Req\_ПС)) |

Выходные параметры:

|  |  |  |  |
| --- | --- | --- | --- |
| Элемент | Описание | Тип данных | Примечание |
| step 0 | параметр, что проверка эцп, сохранение Scope\_Пс, выделения и сохранения базовой подписи прошло успешно | string |  |

GET /select\_auth HTTP/1.1

Host: 127.0.0.1:8084

Content-Type: application/json

{

"data":"<Signed\_ПС(Scope\_ПС, Enveloped\_СИ(Req\_ПС))>"

}

Ответ успешный:

Код HTTP 200 OK

{

"step 0": "OK"

}

7.4 Входные параметры запроса для метода /api/v2/login\_callback приведены в таблице 4.

Таблица 4

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип параметра | Тип данных |
| params | Query параметр data, который был получен в URL-строке после успешной аутентификации | JSON | string |
| code\_verifier | Описание параметра в таблица 3 | JSON | string |
| settings | Описание настроек в таблице 2 | JSON | object |

POST /api/v2/login\_callback HTTP/1.1

Host: 192.168.0.165:3000

Content-Type: application/json

{"params": "<парметры авторизации>",

"code\_verifier": "drVg7tl0WinWXSZAtfpaBzymMCeCTT7Nn67rAXiwsqSpI6ZG0wA",

"settings": {

"authority": "https://esiful.nces.by:8081",

"client\_id": "5c0SbSd9IiDIlhmF7q\_V0pq-uGp8z6y8dzWHQIisFIM",

"grant\_type": "authorization\_code",

"response\_type": "code",

"scope": "openid profile offline\_access",

"redirect\_uri": "http://localhost:8080",

"post\_logout\_redirect\_uri": "http://localhost:8080",

"prompt": "consent",

"state": "niitzi.by" }}

Выходные параметры ответа для метода /api/v2/callback приведены в таблице 5.

Таблица 5

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип данных | Тип параметра |
| userinfo | Профайл пользователя | string | JSON |
| id\_token | Билет аутентификации | string | JSON |
| access\_token | Билет доступа | string | JSON |
| refresh\_token | Билет обновления | string | JSON |

Пример ответа:

HTTP/1.1

Content-Type: application/json

{id\_token: '<значение билета аутентификации в формате BASE64',

userinfo: {

sub: 'MTAxMDEwMU0xMDFQQjE=',

name: 'Валерий Иванович',

surname: 'Иванов',

serialNumber: '1010101M101PB1' },

access\_token: '<значение билета доступа в формате BASE64>',

refresh\_token: '<значение билета обновления в формате BASE64' }

7.5 Входные параметры запроса для метода /api/v2/logout приведены в таблице 6.

Таблица 6

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип  параметра | Тип  данных |
| id\_token | Билет аутентификации | JSON | string |
| authority | URL-адрес сервера идентификации ЕС ИФЮЛ | JSON | string |
| post\_logout\_redirect\_uri | URI перенаправления клиентского приложения после выхода из сервера идентификации ЕС ИФЮЛ. | JSON | string |

Пример запроса /api/v2/logout:

HTTP/1.1

Content-Type: application/json

{"id\_token": "<значение билета аутентификации в формате BASE64",

"authority": "https://esiful.nces.by:8081",

"post\_logout\_redirect\_uri": "http://localhost:8080" }

Выходные параметры ответа для метода /api/v2/logout приведены в таблице 7.

Таблица 7

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип данных | Тип параметра |
| logout\_url | Ccылка, которая позволяет завершить сеанс в сервера идентификации ЕС ИФЮЛ | string | JSON |

Пример ответа:

HTTP/1.1

Content-Type: application/json

{"logout\_url": "https://esiful.nces.by:8081/session/end?id\_token\_hint= <значение билета аутентификации в кодировке BASE64> &post\_logout\_redirect\_uri=http://localhost:8080"}

7.6 Входные параметры запроса для метода /tls-init приведены в таблице 8.

Таблица 8

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип параметра | Тип данных |
| Ip\_port | Ip-адрес и порт TLS-сервера | JSON | string |

Пример запроса:

POST /tls\_init HTTP/1.1

Host: 127.0.0.1:8084

Content-Type: application/json

{ "ip\_port": "192.168.0.165:8443"}

Выходные параметры ответа для метода /api/v2/logout приведены в таблице 9.

Таблица 9

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип данных | Тип параметра |
| error | Сообщение об ошибке. По умолчанию 0, если ошибок нет | string | JSON |

8. Выполнение аутентификации с помощью модуля поддержки OpenID Connect

Для прохождения аутентификации необходимо выполнить следующие последовательные шаги:

а) создание файла с настройками авторизации;

б) формирование и отправка запроса авторизации на КП;

в) формирование и отправка запроса авторизации, перенаправление на ЕС ИФЮЛ;

г) формирование и отправка запроса для обмена code на БА, БД, БО, обмен БД на Userinfo;

д) создание защищенного ТЛС-соединения.

е) выход из ЕСИФЮЛ (завершение сессии).

8.1. Создание файла с настройками аутентификации

Файл в формате JSON, содержащий настройки аутентификации должен иметь следующий вид:

{ "settings": {

"authority": "https://esiful.nces.by:8081",

"client\_id": "5c0SbSd9IiDIlhmF7q\_V0pq-uGp8z6y8dzWHQIisFIM",

"grant\_type": "authorization\_code",

"response\_type": "code",

"scope" : "openid profile offline\_access",

"post\_logout\_redirect\_uri": "http://localhost:8080",

"redirect\_uri": "http://localhost:8080",

"prompt": "consent",

"state": "niitzi.by" },

"kpsis": "http://192.168.0.165:3000"},

где

* authority (тип данных: строка): URL-адрес провайдера ЕС ИФЮЛ.
* client\_id (тип данных: строка): идентификатор вашего клиентского приложения, зарегистрированный у провайдера ЕС ИФЮЛ.
* grant\_type (Тип данных: строка, по умолчанию authorization\_code): Тип потока
* response\_type (тип данных: строка, по умолчанию ‘code’:) - тип ответа, требуемый от провайдера ЕС ИФЮЛ.
* scope (тип данных: строка, по умолчанию 'openid') : Список данных, которые запрашиваются у провайдера ЕС ИФЮЛ.
* redirect\_uri (тип данных: строка): URI перенаправления вашего клиентского приложения для получения ответа от провайдера ЕС ИФЮЛ.
* post\_logout\_redirect\_uri (тип данных: строка): URI перенаправления вашего клиентского приложения после выхода из провайдера ЕС ИФЮЛ.
* prompt: Обязательный параметр для получения билета обновления. Значение по умолчанию "consent"
* state: Значение, используемое для поддержания состояния между запросом и обратным вызовом, будет возвращено после успешной аутентификации.
* kpsis : ip-адрес КПСИС

8.2.  После нажатия на кнопку "Войти в ЕСИФЮЛ" с Front-end ПС необходимо отправить http-запрос на Backend часть ПС для начала аутентификации.

Пример запроса на JavaScript:

const { data: { signed\_data\_to\_check\_in\_cp } } = await axios.get("/log-in");

8.3. На backend части отправить http-запрос на модуль поддержки OpenID connect КПСИС для формирования ссылки аутентификации. Формат запроса в пункте 7.3.

Пример запроса на JavaScript:

сonst { data: { enveloped\_and\_signed\_auth\_url, code\_verifier, signed\_data\_to\_check\_in\_cp } } = await axios.post(`${config.get('kpsis')}/api/v2/login`, { settings: config.get('settings'), })

После получения параметров ответа от КПСИС, параметр code\_verifier необходимо сохранить в cookie, а параметр enveloped\_and\_signed\_auth\_url в кэш или базу данных.

Пример на JavaScript сохранения code\_verifier:

res.cookie('code\_verifier', code\_verifier, { httpOnly: true });

Пример на JavaScript сохранения code\_verifier в коллекции по типу ключ-значение, используя объект Map:

const sessionStore = new Map()

sessionStore.set(code\_verifier, enveloped\_and\_signed\_auth\_url) // code\_verifier служит ключом в хранилище

Далее необходимо отправить ответ на Front-end-часть.

Пример ответа:

res.status(200).json({signed\_data\_to\_check\_in\_cp })

8.4 На Front-end части необходимо отправить запрос на КП

Пример запроса:

const {data } = await axios.post(`http://127.0.0.1:8084/select\_auth`, {

data: signed\_data\_to\_check\_in\_cp

});

После успешного ответа от КП перенаправить пользователя на бэкенд-часть для перенаправления в ЕСИФЮЛ

Пример перенаправления:

const form = document.createElement("form");

form.action = "/redirect-to-esiful";

form.method = "GET";

form.style.display = "none";

document.body.append(form);

form.submit();

Пример обработчика запроса на бэкенд части и перенаправления в ЕСИФЮЛ

const code\_verifier = req.cookies.code\_verifier

const signed\_data\_to\_check\_in\_cp = sessionStore.get(code\_verifier) // пример получения ссылки на аутентификацию из объекта Map. Создание объекта Map в пункте 8.3

sessionStore.delete(code\_verifier) //очистка объекта Map

res.writeHead(302, { 'Location': signed\_data\_to\_check\_in\_cp });

return res.end(); // перенаправление в ЕСИФЮЛ

8.5. После успешной аутентификации в ЕСИФЮЛ пользователь будет перенаправлен на callback-страницу с query-параметром data. Необходимо выделить параметр data и отправить его на Backend часть ПС.

Пример http-запроса на JavaScript:

const {

data: { id\_token, userinfo, access\_token, refresh\_token },

} = await axios.get("/login-callback", {

headers: {

params: location.search.split("?data=")[1],

}, });

8.6 После получения запроса на Backend части ПС, необходимо отправить запрос на модуль поддержки OpenID connect для получения билета аутентификации, билета доступа, билета обновления и профайла из пункта 7.4.

Пример запроса на JavaScript:

const {params} = req.headers

const { cookies: { code\_verifier } } = req;

const { data: {userinfo, id\_token, access\_token, refresh\_token} } = await axios.post(`${config.get ('kpsis')}/api/v2/login\_callback`, { params, code\_verifier, settings: config.get('settings') })

Если в дальнейшем планируется использовать токены для получения информации от ЕС ИФЮЛ, необходимо их сохранить.

8.7. Для того, чтобы завершить сессию в ЕСИФЮЛ, необходимо отправить http-запрос в модуль поддержки OpenID connect из пункта 7.5.

Пример запроса на JavaScript:

const { data: {logout\_url} } = await axios.post(`${config.get('kpsis')}/api/v2/logout`, {

id\_token, authority: config.get('settings').authority,

post\_logout\_redirect\_uri: config.get('settings').post\_logout\_redirect\_uri })

Далее необходимо отправить ответ на Front-end часть ПС со статусом 302 и параметром ‘Location’: logout\_url для перенаправления на сервер идентификации ЕСИФЮЛ.

Пример ответа:

res.writeHead(302, { 'Location': logout\_url });

res.end();

8.9 Для установки защищенного tls-соединения необходимо отправить http-запрос из пункта 7.6. Пример запроса:

const {data: {error}} = await axios.post('http://127.0.0.1:8084/tls\_init', {

"ip\_port": "192.168.0.165:8443" })

Если в ответе от КП параметр error будет равен “0”, то необходимо перенаправить пользователя по адресу <http://127.0.0.1:8085>.

Пример:

if (error === "0") { location.href = http://127.0.0.1:8085 }

Приложение 1

Заявка

на регистрацию информационного ресурса в качестве клиента

Единой системы идентификации физических и юридических лиц

1. Прошу зарегистрировать \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ в качестве

(наименование информационного ресурса)

клиента Единой системы идентификации физических и юридических лиц.

1. Регистрационные данные:

|  |  |
| --- | --- |
| Наименование информационного ресурса |  |
| Номер свидетельства о гос. регистрации |  |
| Дата регистрации |  |
| Перечень атрибутов конечного пользователя, которые необходимо предоставить при успешной аутентификации |  |
| URL-адрес перенаправления при авторизации |  |
| URL-адрес перенаправления при выходе |  |
| URL-адрес логотипа |  |
| URL-адрес политики безопасности |  |
| URL-адрес клиента |  |

Руководитель организации-оператора

информационного ресурса Фамилия, инициалы

Приложение 2

Перечень доступных к запросу

атрибутов конечного пользователя

1. openid – обязательный параметр для идентификации протокола OpenID Connect
2. profile – данные, составляющие профиль конечного пользователя. Для физического лица profile содержит: идентификационный номер, фамилию, имя, отчество.
3. offline\_access – параметр для получение билета обновления (Refresh Token). Билет обновления необходим для получения нового билета доступа (когда срок его действия истечет). Билет доступа (Access Token) необходим для получения данных аутентифицированного пользователя.