АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ ПО ТЕХНОЛОГИИ ЕДИНОГО ВХОДА

Руководство по эксплуатации

Версия 10.3.0

nexign

АО «Нэксайн»

Россия, 199155, Санкт-Петербург, ул. Уральская, д.4 лит.Б, помещение 22Н

Тел.: + 7 (812) 326-12-99; факс: + 7 (812) 326-12-98.

office@nexign.com; www.nexign.com

Содержание

1. Общие сведения

Продукт «Аутентификация пользователей по технологии единого входа» (SSO) предназначен для обеспечения унифицированного механизма аутентификации пользователей с поддержкой технологии единого входа и создания центра управления доступом к продуктам АО «Нэксайн».

Продукт выполняет следующие функции:

- автоматическая аутентификация пользователей при входе из сети оператора;
- аутентификация пользователя при входе из сети Wi-Fi, если до этого был выполнен вход из сети оператора;
- поддержка работы с программным обеспечением на мобильных платформах iOS, Android;
- предоставление API для проверки факта аутентификации пользователя;
- администрирование прав доступа через API и через пользовательский интерфейс;
- предоставление перечня прав пользователя на объекты и их атрибуты по запросу от сервера аутентификации SSO или сервисов;
- поддержка актуальности информации о правах пользователей на всех потребителях такой информации;
- взаимодействие с внешними системами в части управления правами пользователей.

В документе используются следующие термины и сокращения:

SSO-CA – центр аутентификации продукта SSO.

Доменный пользователь – пользователь SSO, имеющий учётную запись в Active Directory, логин которой указан в атрибуте аутентификации «Доменный логин».

Доменная роль SSO – роль SSO, соответствующая доменной группе Active Directory, наименование которой указано в атрибуте роли «Доменная группа».

Ограничение (квота) – ограничение доступа к объекту в дополнение к правам доступа.

Набор прав – перечисление объектов прав с указанием разрешенных операций и ограничений, действующих для объектов, на которые пользователю выданы права.

Площадка – технологическая единица развертывания BIS. Идентификатор площадки – числовое значение, идентифицирующее площадку в ЦУД. Используется для различной настройки ролей и политик для различных площадок.

2. Системные требования

В главе приводится перечень требований для функционирования продукта SSO.

2.1. Состав аппаратных средств

Для установки продукта SSO требуется компьютер, обладающий следующим минимальным составом аппаратных средств:

- CPU: 2x4(6) core Intel® Xeon® 5xxx product family;
- RAM: 4GB;
- HDD: 5GB.

2.2. Состав программных средств

Для работы продукта SSO требуется следующий состав программных средств:

- ОС Linux RedHat 8.0, CENTOS 8.1 или РЕД ОС версии 7.2 или выше, с запущенной службой NTP;
- OpenJDK Java;
- СУБД PostgreSQL;
- «Сервер кэшей» (COUCHBASE);
- RabbitMQ – (опционально) для отправки нотификаций;
- «Шлюз доступа к API» (API_GATEWAY) – (опционально) для работы веб-интерфейса.

Время на серверах, где установлены экземпляры компонентов Directory Service, Synchronization Service и СУБД, должно быть синхронизировано. Время и временная зона серверов, на которых работают компоненты SSO, должно соответствовать временной зоне СУБД.

Актуальные версии программного обеспечения см. в файле releasenotes, поставляемом в составе дистрибутива.

2.3. Рекомендации по квалификации персонала

Пользователь продукта должен иметь навыки работы с графическим пользовательским интерфейсом операционной системы.

Администратор продукта должен обладать навыками и знаниями по администрированию операционной системы, базовыми знаниями об администрировании СУБД, Linux.

3. Настройка

В главе приводится информация о порядке настройки продукта.

3.1. Настройка веб-приложений

В соответствии с требованиями стандарта OpenID Connect при интеграции через `API GATEWAY` продукт `SSO` контролирует корректность маршрутизации пользователя в целевое веб-приложение после успешной аутентификации.

Перед переходом на интеграцию через `API GATEWAY` в продукте `SSO` в конфигурации технологического пользователя `api_gateway` необходимо прописать URL возврата во все веб-приложения (`redirect_uris`), аутентификацию в которых через Authorization Code Flow производит продукт `SSO`.

Для удобства конфигурирования в inventory поддерживается параметр `redirect_uris`, позволяющий автоматически зарегистрировать URL возврата в веб-приложения при установке продукта `SSO`.

При регистрации нового веб-приложения в уже работающем экземпляре продукта `SSO`, сконфигурировать URL возврата в новое приложение можно через веб-интерфейс.

4. Журналирование

Параметры журналирования компонентов `Authentication Service`, `Access Control Service`, `Synchronization Service`, `Directory Service`, `Data Streams Processing Service` и `Provisioning Adapter` приведены в файле /schemas/sso-schema.json:

- `level` – общий уровень журналирования:
  - `TRACE` – самый глубокий уровень, трассировочная информация;
  - `DEBUG` – отладочные сообщения;
  - `INFO` – информационные сообщения;
  - `WARN` – предупреждающие сообщения;
  - `ERROR` – сообщения об ошибках, возникающих при выполнении каких-либо задач;
- `max_file_size` – максимальный размер журнального файла; может быть указан в байтах, килобайтах, мегабайтах или гигабайтах;
- `max_history` – максимальное количество архивных файлов для хранения;
- `total_size_cap` – общий размер всех архивных файлов;
- `accessLogEnabled` – флаг наличия файла access.log;
- `accessLogFileDateFormat` – формат даты для файла access.log;
- `accessLogMaxRetainDays` – срок хранения файла access.log.

Значения параметров задаются в конфигурационных файлах logback-spring.xml и application.yml соответствующего компонента.

Для настройки журналирования отредактируйте конфигурационный файл /config/logback-spring.xml: задайте параметры для получателей сообщений.

5. Особенности эксплуатации

В главе приводится описание основных особенностей эксплуатации продукта SSO.

5.1. Способы аутентификации

Продукт поддерживает следующие способы аутентификации:

- получение токена приложением;
- аутентификация веб-приложений;
- базовая аутентификация;
- доверенные каналы;
- digest-аутентификация.

5.1.1. Получение токена приложением

Для получения токена приложением используется REST API-функция «Получение токена приложением» (/ps/auth/api/token) компонента «Сервис аутентификации» (Authentication Service).

5.1.2. Аутентификация веб-приложений

Продукт предоставляет единую точку входа для HTTP(S)-запросов, обращающихся к защищаемым веб-приложениям. Для аутентификации пользователь вводит логин и пароль на веб-форме, либо приложение, которому требуется выполнить аутентификацию пользователя в SSO, выполняет запрос токена. Для проверки существования логина, возможности аутентификации пользователя с указанным логином и соответствия логина и пароля пользователя используются данные, хранимые в SSO. Продукт поддерживает автоматическую аутентификацию пользователей-абонентов при вызове функций HAS или OpenAPI 2.0 из сети оператора и при входе из сети Wi-Fi, если до поступления запроса на выполнение функции HAS или OpenAPI 2.0 выполнен вход пользователя в сеть оператора.

5.1.3. Базовая аутентификация

При использовании базовой аутентификации (аутентификации по протоколу HTTP Basic Authentication) имя и пароль пользователя передаются в заголовке Authorization веб-запроса.

Базовая аутентификация используется:

- пользователями-приложениями, запрашивающими токен в Authentication Service;
- при выполнении служебных запросов сервисами.

  Например, Karaf использует Basic-аутентификацию при получении списка ролей, дающих права на ресуры.

  HAS использует Basic-аутентификацию при получении прав пользователя по логину или aclhash.

  В данном случае решение о проведении Basic-аутентификации принимает компонент Single Entry Point (SEP). По аналогии со сценарием работы доверенного канала SEP запрашивает токен через API компонента Authentication Service, кэширует его и использует для аутентификации последующих запросов с такими же данными в заголовках Basic-аутентификации.

Особенности реализации базовой аутентификации

Базовая аутентификация для всех запросов, для которых включена аутентификация на `API_GATEWAY`, и для запроса получения токена должна раскрываться на `API_GATEWAY`.

Для всех запросов, использующих Basic-аутентификацию, применяется единый алгоритм.

Авторизация запросов на `API_GATEWAY` опциональна и осуществляется только для тех URL, для которых зарегистрированы объекты.

Заголовки аутентификации, используемые сервисами SSO

- `x-ps-login` – логин пользователя (обязательный);
- `x-ps-asur_id` – уникальный идентификатор пользователя (обязательный);
- `x-ps-aclhash` – уникальный идентификатор набора прав (обязательный);
- `x-ps-token` – токен (опциональный);
- `x-ps-auth_type` – тип аутентификации; `password` (обязательный).

5.1.4. Работа по доверенным каналам

Доверенные каналы – интерфейс вызова методов API для систем, не поддерживающих авторизацию с использованием токена доступа.

Позволяет вызывать методы API, передавая в качестве атрибутов аутентификации в строке запроса URL код приложения (параметр `APPL_CODE`), логин пользователя (параметр `LOGIN`) и, опционально, пароль пользователя (параметр `PASSWORD`). Приложение – учётная запись пользователя с типом «Приложение».

Пример:

```
http://sep.domain.ru:47555/openapi/v1/customers/301/calls?LOGIN=9290501
139&APPL_CODE=CAREM
```

В качестве логина и пароля могут быть переданы доменные логин и пароль пользователя (при этом для пользователя должен быть задан атрибут аутентификации «доменный логин» и пользователь должен быть зарегистрирован в Active Directory с этим же атрибутом аутентификации, а в параметрах конфигурации должен быть задан параметр, указывающий шаблон строки для аутентификации пользователей в Active Directory, и настроены параметры подключения к Active Directory).

Дополнительные параметры безопасности при работе по доверенным каналам:

- конфигурационный параметр `forbidden_produce_token_user_types`. Параметр задаётся в секции `ps_auth -> common_settings`. Задаёт список идентификаторов типов учётных записей, для которых запрещено получение токена доступа, как для пользователя;
- использование ролей, выданных пользователю через `SSO`.

`Authentication Service` поддерживает разграничение прав на вызов функции получения токена с использованием разрешений на объекты, выданных через роли с использованием `SSO`.

Режим поддержки ролей, выданных через `SSO`, активизируется параметром `use_acc_rights` в секции `ps_auth -> common_settings`.

Параметр `department_id` задает идентификатор филиала, для которого заданы роли.

Список предустановленных ролей:

- SSO:GetToken:Full – права на получение токена доступа для пользователя с использованием всех возможных способов аутентификации;
- SSO:GetToken:WithPassword – права на получение токена доступа для пользователя с указанием пароля, refresh_token;
- SSO:GetToken:WithControlQuestion – права на получение токена доступа для пользователя с указанием ответа на контрольный вопрос.

Перед переключением `Authentication Service` на режим поддержки ролей, выданных через `SSO`, убедитесь, что всем пользователям и приложениям, которым необходим доступ к методу получения токена доступа, выданы соответствующие роли.

Пользователи и приложения, которым не выданы права на метод получения токена доступа, не смогут явно получить токен доступа.

5.1.5. Digest-аутентификация

`Authentication Service` поддерживает digest-аутентификацию запросов в соответствии со стандартом HTTP Digest Access Authentication.

Данный стандарт предусматривает двухэтапную аутентификацию пользователей с защитой передаваемого пароля пользователя и защитой от подделки запросов.

Поддерживаемые алгоритмы хэширования:

- MD5;
- MD5-sess;
- SHA-256;
- SHA-256-sess;
- SHA-512;
- SHA-512-sess.

Поддерживаемые значения параметра `qop` (quality of operation):

- auth;
- auth-int.

При аутентификации создается сессия приложения с ограниченным временем жизни. Для каждого запроса в рамках сессии (для сохраненного значения `nonce`) выполняется контроль счетчика запросов и при получении запроса с некорректным значением счетчика возвращается ошибка. После окончания времени жизни сессии возвращается ответ с кодом `401 Unauthorized`, заголовком WWW-Authenticate, содержащим значение `stale=true`.

Поддерживается режим однократного использования значения `nonce`. Данный режим рекомендуется использовать для получения служебного токена приложения, который впоследствии будет использоваться для аутентификации конечных пользователей.

Поскольку в `SSO` пароли пользователя хранятся в виде результата хэширования пароля, то пароль пользователя в чистом виде сервису аутентификации неизвестен. В связи с этим, при генерации ответа digest необходимо использовать хэш пароля пользователя. `SSO` поддерживает хэширование паролей пользователей с использованием алгоритмов md5 и SHA-256.

Для однозначного определения используемого алгоритма хэширования пароля необходимо в первоначальном запросе `/token` в теле запроса передать в параметре `client` логин пользователя (приложение, технологический пользователь), для которого требуется выполнить digest-аутентификацию. В ответ на такой запрос в заголовке WWW-Authenticate будет добавлена информация об используемом алгоритме хэширования пароля в параметрах `client-alg` и `client-salt` (используемая соль в base64).

Аутентификация пользователя

При использовании digest-аутентификации аутентификация пользователя выполняется стандартным способом в соответствии с используемым значением `grant_type`. Для защиты пароля пользователя, передаваемого в запросе получения токена, одновременно с digest-аутентификацией реализован `grant_type=digest`. Данный способ получения токена пользователя поддерживается для OAuth 2.0 и OpenId Connect.

Аутентификация пользователя с использованием `grant_type=digest` аналогична `grant_type=password`, но в качестве значения параметра `password` необходимо передать digest пароля пользователя.

Для генерации digest пароля пользователя используйте тот же алгоритм генерации digest response, что используется для аутентификации, и те же самые параметры генерации со следующими отличиями:

- значение параметра `qop` (quality of operation) фиксировано и должно иметь значение "`auth`";
- в качестве `username` и `password` используются логин и пароль аутентифицируемого пользователя;
- если для генерации digest response учитывается тело запроса (`qop=auth-int`), то сначала сформируйте digest пароля пользователя и подставьте его в тело запроса, а затем вычислите digest response.

Поскольку в `SSO` пароли пользователя хранятся в виде результата хэширования, то пароль пользователя в чистом виде сервису аутентификации не известен. Поэтому при генерации ответа digest используйте хэш пароля пользователя. `SSO` поддерживает хэширование паролей пользователей с использованием алгоритмов md5 и SHA-256.

Для однозначного определения используемого алгоритма хэширования пароля необходимо в первоначальном запросе `/token` в теле запроса передать в параметре `user` логин пользователя. В ответ на такой запрос в заголовке WWW-Authenticate будет добавлена информация об используемом алгоритме хэширования пароля в параметрах `user-alg` и `user-salt`.

5.2. Сценарии аутентификации

`SSO` поддерживает следующие сценарии аутентификации пользователей:

- через веб-форму;
- через приложение;
- по логину без пароля (`grant_type=basic`, требуется наличие роли SSO:GetToken:WithoutPassword);
- с использованием ответа на контрольный вопрос;
- с использованием refresh-токена;
- с использованием токена внешней системы (`WebSSO`);
- для запросов, выполняемых по доверенным каналам;
- с использованием доменной учетной записи (LDAP);

Внимание!

В текущей версии аутентификация пользователей выполняется только методом Simple Bind с использованием логина и пароля пользователя. Не поддерживается аутентификация с использованием `cn`. Логин пользователя (используемое значение атрибута аутентификации) должен содержать доменное имя пользователя и домен в формате: `<user-name>@<domain>`.

- с использованием доменной учетной записи (Kerberos);

  Если в `SSO` настроена аутентификация пользователей, выполнивших вход в домен, при получении запроса "/connect/authorize" `Authentication Service` возвращает ответ с кодом `401 Unauthorized` и выставляет заголовок `WWW-Authenticate: Negotiate`. Браузер пользователя в ответ повторяет исходный запрос и добавляет заголовок `Authorization: Negotiate a87421000492aa874209af8bc028`, содержащий Kerberos-тикет пользователя.

  В соответствии со стандартом SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows (https://tools.ietf.org/html/rfc4559) браузер пользователя может не поддерживать Kerberos-аутентификацию или отправить вместо Kerberos-тикета заголовок с другим типом аутентификации. В данном случае `Authentication Service` повторит ответ с кодом `401` и заголовком `WWW-Authenticate`. Количество повторных попыток выполнить Kerberos-аутентификацию пользователя задается в конфигурационном файле `Authentication Service`. После превышения максимального количества попыток пользователь будет перенаправлен на стандартную форму аутентификации с использованием логина и пароля.

  При получении Kerberos-тикета выполняется его валидация и полученный в процессе валидации логин используется как атрибут аутентификации пользователя `SSO`. Если такое значение атрибута аутентификации зарегистрировано у одного из пользователей `SSO`, аутентификация завершается успешно. При отсутствии значения атрибута аутентификации у всех пользователей `SSO` аутентификация завершается ошибкой `credential_not_found` с возвратом на стандартную форму аутентификации.

  

  Внимание!

  При добавлении пользователю атрибута аутентификации, соответствующего его доменному логину, важно указывать полное доменное имя пользователя в том регистре, как оно задано в Active Directory. Поиск пользователей `SSO` по атрибуту аутентификации является регистрочувствительным.

  Если аутентификация пользователя по Kerberos-тикету невозможна и пользователь был перенаправлен на стандартную форму аутентификации, пользователь может ввести логин и пароль `SSO` и будет выполнена стандартная аутентификация по логину и паролю.

  Если сервис аутентификации настроен на связывание доменной учетной записи с существующим профилем пользователя `SSO`, после успешной аутентификации по логину и паролю пользователя `SSO` будет предложено связать доменный логин (определенный по Kerberos-тикету) с учетной записью `SSO`. Пользователь может продолжить аутентификацию без добавления доменного логина или выбрать добавление доменного логина для учетной записи `SSO`.

- через социальные сети;

- с использованием временного пароля.

Поддерживаются следующие способы аутентификации запросов:

- аутентификация на основе заголовков аутентификации API_GATEWAY; для запросов с basic-аутентификацией API_GATEWAY выполняет служебный запрос получения токена с grant_type=basic, используя логин и пароль пользователя из basic-аутентификации; запрос сопровождается цифровой подписью API_GATEWAY; с полученным токеном API_GATEWAY выполняет исходный запрос, добавляя заголовки аутентификации;
- аутентификация на основе цифровой подписи запроса API_GATEWAY;
- basic-аутентификация; не является основной и может использоваться только при необходимости выполнить запрос напрямую на сервисе аутентификации, при этом авторизация запросов должна быть отключена; при выполнении запросов через API_GATEWAY basic-аутентификация завершается на самом API_GATEWAY.

Наличие цифровой подписи API_GATEWAY определяется по наличию заголовков x-ps-request_id и x-ps-signature. В заголовке x-ps-request_id должен передаваться уникальный идентификатор запроса, в заголовке x-ps-signature – подпись запроса.

В процессе аутентификации выполняется несколько проверок:

- проверка наличия прав на выполнение операции с объектом, наличие ограничений;
- проверка на соответствие списочному ограничению для разрешенных/запрещенных провайдеров аутентификации sso.auth.provider.restriction (возможные значения: sso, ldap, kerberos); применяется при аутентификации с использованием следующих значений grant_type: authorization_code, password, login, password_temporary, control_question, digest;
- проверка на соответствие списочному ограничению для разрешенных/запрещенных приложений sso.auth.client.restriction (возможные значения – коды клиентов, см. на форме «Клиенты»);
- проверка соответствия IP-адреса запроса IP-маске пользователя и приложения, если они заданы;
- значение IP-адреса определяется по заголовкам запроса в следующем приоритете: x-ps-ip_addr, x-forwarded-for, IP-адрес клиента или последнего прокси-сервера в цепочке прохождения запроса;
- при выполнении запроса от имени администратора значение IP-адреса может быть переопределено в теле запроса параметром ip_addr;
- проверка блокировки пользователя;
- проверка типа пользователя в соответствии с параметром конфигурации forbiddenUserTypesForTokenRequest;
- проверка периода неактивности в соответствии с параметром password_lock_inactive_period парольной политики пользователя.

5.3. Автоматическое продление токена

SSO (компонент Single Entry Point) поддерживает механизм обновления токена в соответствии со стандартом OAuth2.

Access_token – основной токен, используемый для авторизации запросов.

Refresh_token – токен продления. Выдается в паре с access_token. Как правило, срок действия токена продления значительно больше срока действия основного токена.

В SSO доступны следующие механизмы продления токена:

- автоматическое продление токена веб-приложений, работающих через SSO;

- автоматическое продление токенов при работе через доверенные каналы;
- получение нового access_token'а по refresh_token'у.

Для веб-приложений реализован механизм автоматического продления токена. Токен веб-приложений, работающих через `SSO`, передается через `httpOnly Cookie x-ps-token` и устанавливается после успешной аутентификации. Далее Cookie передаются во всех запросах, отправляемых веб-приложением.

Для всех запросов, проходящих через `SSO`, проверяется срок действия токена:

- если срок действия токена, переданного в Cookie, истек, но для него существует refresh_token, то `SSO` самостоятельно запросит новую пару токенов, выполнив аутентификацию по refresh_token;
- если срок действия refresh_token еще не истек и новая пара токенов успешно получена, то с помощью заголовка `Set-cookie` будет установлен новый accsess_token; веб-приложение продолжит работу с новым токеном;
- если срок действия refresh_token истек, то запрос пользователя завершится ошибкой 401.

Автоматическое продление истекших access_token'ов поддерживается только для веб-приложений (токен передается через заголовок `Cookie: x-ps-token=<token>`).

Клиенты, не являющиеся веб-приложениями, должны передавать токен через заголовок `Authorization: Bearer <token>` и для них автоматическое продление токена не предусмотрено.

Приложения должны самостоятельно обрабатывать 401 ошибку. Возможна реализация упреждающего обмена токена на основе анализа времени жизни токена access_token. Приложение должно продлевать токен в соответствии с протоколом OAuth2, то есть путем обмена refresh_token на новую пару access_token и refresh_token.

Механизм продления токена при работе по доверенным каналам схож с механизмом, используемым для веб-приложений, за исключением того, что токен скрыт от пользователя. По сочетанию `LOGIN`, `APPL_CODE`, `IP` осуществляется поиск токена пользователя и проверяется срок действия. Если срок действия токена истек, то запрашивается новый.

5.4. Подключение WebSockets

Использование технологии WebSockets на основе STOMP-протокола – сигнализирование о произошедших в backend-системах событиях. Данные сигналы информируют компоненты пользовательского интерфейса о том, что необходимо выполнить какой-либо вспомогательный запрос на обновление данных. Сервер DAPI Proxy подписывается на получение сигналов из RabbitMQ, найденных через ZooKeeper `Service_Discovery`, и при получении валидных данных публикует их в персонифицированный канал на основе `userId_ws`, через который проходит информирование компонентов пользовательского интерфейса.

Информирование о наступлении события, пришедшего из какой-либо системы или компонента, (signal-сообщение) говорит о том, что необходимо выполнить какой-либо запрос на получение изменившихся данных. Сигналы должны публиковаться в точке обмена (`exchange`) AMQP с типом `fanout`: `SIGNAL_FANOUT` (при отсутствии будет создан автоматически) в формате JSON (`content_type: application/json`).

5.5. Вычисление прав пользователя с учетом настроенных в Active Directory параметров доступа

`SSO` синхронизирует роли доменных пользователей продукта с назначенными пользователям группами в Active Directory. Синхронизация производится при каждой аутентификации

пользователя:

1. Учетная запись пользователя в SSO связывается с учетной записью пользователя в Active Directory. Для этого задайте в SSO пользователю значение атрибута аутентификации «Доменный логин». Доменные логины пользователей в SSO должны соответствовать значению principal, получаемому после доменной аутентификации пользователя через Kerberos, поскольку поиск пользователя в SSO выполняется по полному соответствию логина. В SSO значение атрибута пользователя «Доменный логин» является регистрозависимым.

Примечание.

При эксплуатации SSO с несколькими доменами Active Directory рекомендуется выполнить настройки так, чтобы principal содержал домен пользователя и в качестве доменного логина пользователя SSO задавать значение в формате UPN (`username@domain`). Это позволит обеспечить однозначную идентификацию пользователей разных доменов.

Внимание!

Логин пользователя (используемое значение атрибута аутентификации) должен содержать доменное имя пользователя и домен в формате: <username>@<domain>.

2. Роли SSO ассоциируются с доменными группами Active Directory. Для этого задайте значение атрибута роли «Доменная группа», в котором укажите полное имя доменной группы, которой соответствует роль. Уникальность значений атрибута в SSO не контролируется, поддерживается возможность настроить одной группе в Active Directory соответствие нескольким ролям в SSO, но не наоборот. В SSO значение атрибута роли «Доменная группа» является регистрозависимым.
3. При аутентификации пользователя в SSO поддерживается возможность синхронизации ролей пользователя в SSO с группами, в которые входит пользователь в Active Directory. В результате синхронизации набор доменных ролей пользователя в SSO приводится в соответствие с группами пользователя в Active Directory:
   - доменные роли пользователя в SSO, которым не нашлось доменных групп, в которые входит пользователь в Active Directory, у пользователя изымаются;
   - доменные роли, которых нет у пользователя в SSO, назначаются без ограничения подразделений, в которых действуют роли;
   - прочие роли, для которых в SSO не настроено соответствие группам Active Directory, и назначенные пользователю в SSO, сохраняются.

Примечание.

Для включения синхронизации ролей пользователя при аутентификации установите для параметра `updateDomainUserRolesOnLogin` конфигурационного файла компонента `Authentication Service` значение `true`.

Если роли пользователя были изменены в Active Directory, но пользователь после этого ни разу не проходил аутентификацию в SSO, то в SSO будут храниться устаревшие данные о его ролях.

Наследование групп в Active Directory не учитывается в SSO; пользователю назначаются только роли, напрямую соответствующие его группам в Active Directory.

Для работы с Active Directory настройте инвентарные файлы sso/sso-cache-sync.yml и sso/sso-authentication.yml , а также установите значение параметра updateDomainUserRolesOnLogin конфигурационного файла компонента Authentication Service в true.

5.6. Ограничение доступа пользователей

Все операции по изменению статуса блокировки пользователя выполняются компонентом Directory Service.

5.6.1. Блокировка пользователя

В процессе аутентификации пользователь может быть заблокирован в результате проверки следующих ограничений:

- количество попыток ввода пароля;
- количество попыток ответа на контрольный вопрос.

Ограничения задаются в параметрах парольной политики пользователя.

В процессе аутентификации пользователь может быть разблокирован в результате проверки условия «окончание срока блокировки».

5.6.2. Ограничение доступа по IP-маске

Если для пользователя установлено ограничение доступа по IP-маске, при аутентификации IP-адрес пользователя проверяется на соответствие маске.

Ограничение доступа задается в атрибуте пользователя IP_MASK.

Маска подсети должна использоваться для ограничения списка IP-адресов, доступ с которых пользователю разрешен. Протокол (если задан) определяет разрешенный протокол выполняемого запроса для заданной маски. Если протокол задан, доступ будет разрешен только при соответствии протокола и IP-адреса заданной маске.

Фактический протокол выполнения запроса определяется по значения заголовка x-forwarded-proto; если заголовок отсутствует, протоколом по умолчанию считается HTTP. Значение протокола в маске регистронезависимое.

Поддерживаются следующие значения IP-маски:

- wild card:
  - 192.168.1.*
  - 192.168.1.10?
- маски подсети:
  - 192.168.2.0/32
- протокол:
  - http:192.168.1.*
  - https:192.168.2.0/32

При проверке анализируется IP-адрес пользователя по порядку в следующих атрибутах запроса (берется первое определенное значение):

- значение параметра ip_addr запроса;
- значение заголовка x-ps-ip_addr;
- значение заголовка x-forwarded-for;

- сетевой адрес клиента.

Проверка IP-адреса распространяется на пользователя, выполняющего запрос, и пользователя, заданного в параметре `client_id` запроса.

5.7. Проверка статуса сервисов

Для проверки статуса сервисов выполните запрос `/check` на стандартный порт `NGINX` (по умолчанию – `8888`).

5.8. Проверка прав пользователя

Набор прав текущего пользователя можно получить, выполнив запрос `GET /users/current/grants`.

Запрос доступен всем пользователям `SSO`, дополнительные права выдавать не нужно.

Запрос поддерживает фильтрацию по имени роли, типу и имени объекта, названию ограничения и возвращает все роли, объекты прав и ограничения.

5.8.1. Использование объектов прав на форме

Пример: форма состоит из двух частей. В первой части – поиск клиентов по параметрам. Во второй части – создание клиента (см. Рис. 1). Для каждого объекта формы, доступ к которому требуется ограничить, регистрируется объект прав типа `GUI_OBJECT`. Также нужно зарегистрировать объект, соответствующий самой форме.

Имена объектов должны быть уникальны в рамках `SSO`, поэтому при регистрации имени объекта рекомендуется формировать имя по шаблону: <продукт>_<имя формы в рамках продукта>_<имя объекта на форме>.

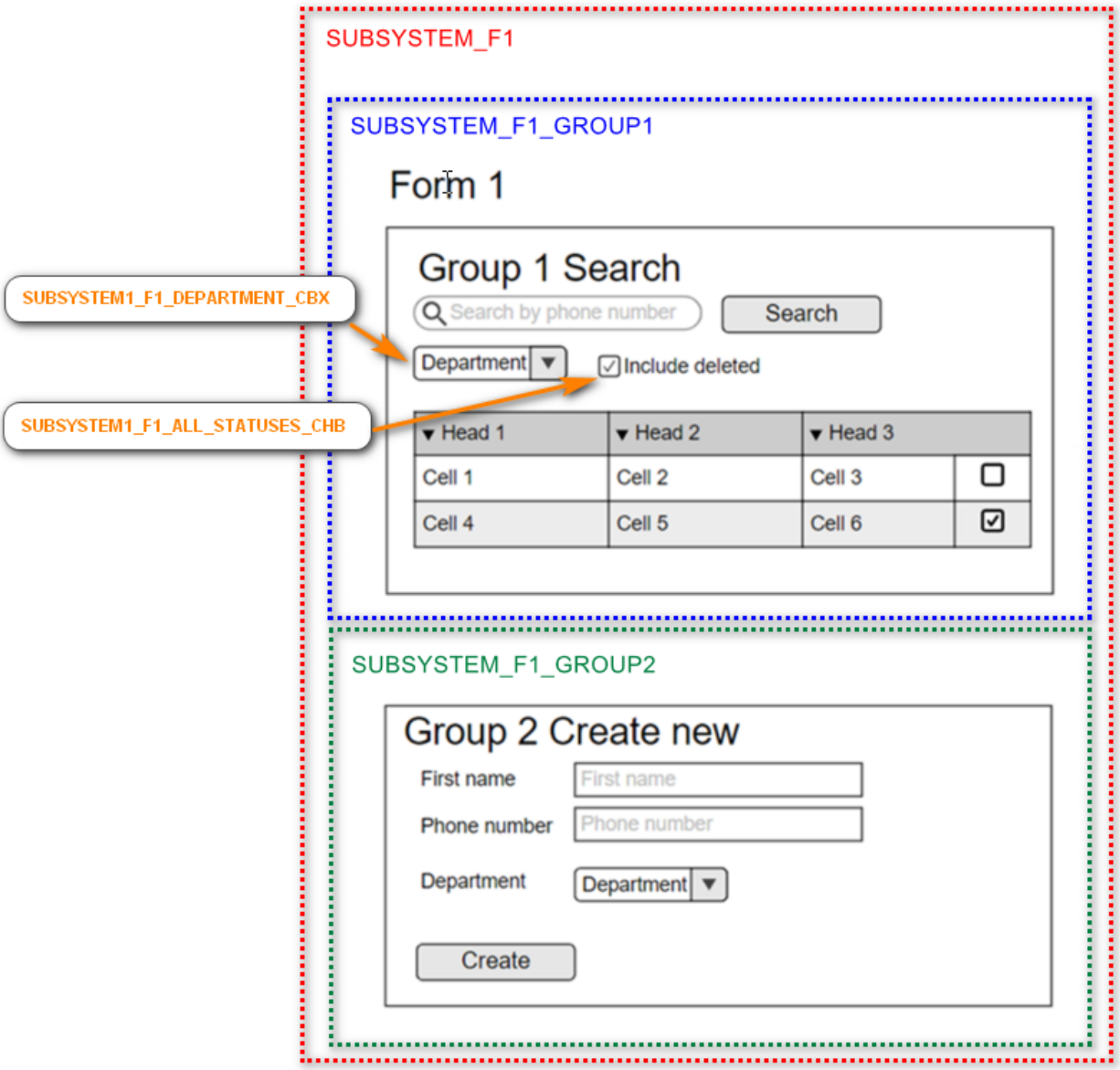

Рис. 1. Пример формы

5.8.2. Проектирование ролей

При проектировании ролей рекомендуется следовать правилу: чем меньше прав, тем лучше. Роль должна содержать минимальный необходимый набор прав. Не следует создавать одну роль, содержащую права на всё, а затем создавать роли, создающие запреты на часть объектов.

Имена ролей, выдающих права на работу с графическим интерфейсом, рекомендуется формировать по правилу: <имя продукта>:GUI:<имя формы>:<имя роли>.

Например, с формой должны работать два вида пользователей:

- оператор (см. Рис. 2): доступен только поиск по номеру телефона, выбор подразделения и поиск по всем статусам недоступны (нет права на чтение);

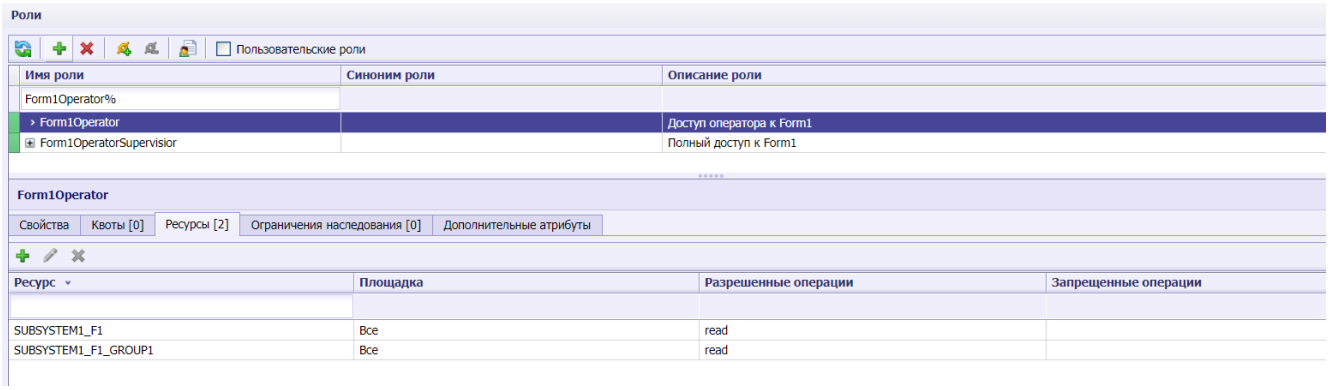

Рис. 2. Права оператора

- главный оператор (см. Рис. 3): все права оператора, поиск по любому подразделению и статусу, создание нового клиента.

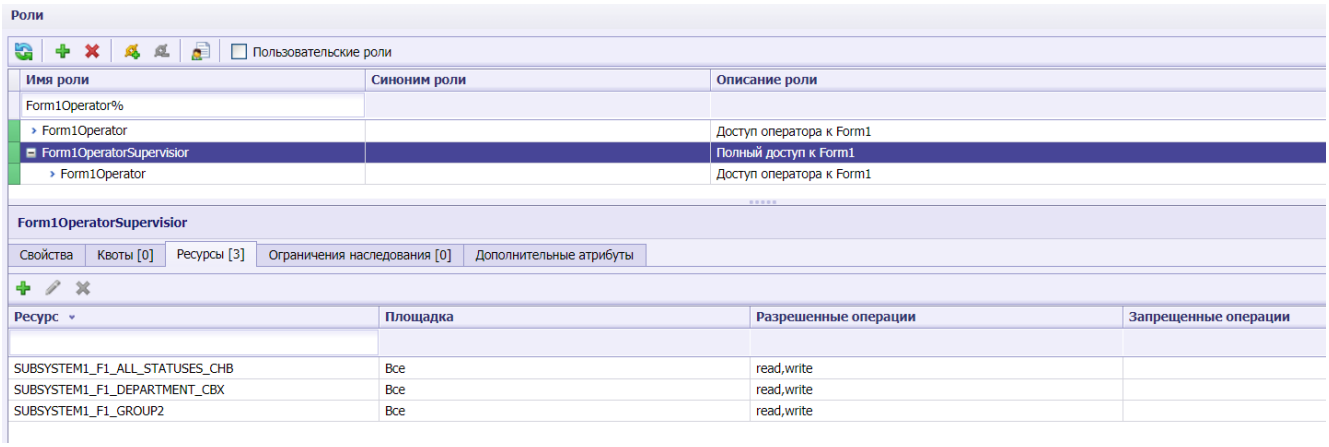

Рис. 3. Права главного оператора

5.8.3. Регистрация объектов прав в SSO

Для того чтобы зарегистрировать созданную модель прав, необходимо сформировать json ролевой модели и выполнить запрос к Provisioning Adapter `SSO`.

У пользователя, выполняющего запрос, должна быть роль Security:Entities:Install.

5.8.4. Набор прав пользователя

Пользователю выдаются созданные роли Form1Operator и Form1OperatorSupervisior. Далее для каждого пользователя запрашиваются токены. При работе через графический интерфейс токен будет передаваться в cookie и права нужно получать с помощью запроса `GET /users/current/grants`.

5.8.5. Использование атрибутов и ограничений объектов

В наборе прав пользователя выполняется поиск объекта и проверяется наличие выполняемого метода в списке разрешенных операций. Не имеет значения, через какую роль выданы права на объект, главное – наличие объекта в наборе прав.

В наборе прав токена, возвращаемого `SSO`, для объектов, кроме разрешенных операций, может быть указан список значений дополнительных атрибутов, связанных с объектом, и список ограничений.

Дополнительные атрибуты объекта прав определяются типом объекта. Атрибуты используются только для объектов типа «Аргумент HAS операции (ARGUMENT)»; определены следующие дополнительные атрибуты:

- `DIRECTION` – направление атрибута (`IN`, `OUT`);
- `MANDATORY` – признак обязательности;
- `MASK` – регулярное выражение для проверки значения;
- `SECURITY_VALUE`, `DEFAULT_VALUE` – значение по умолчанию.

Значения атрибутов не обязательны для заполнения. В составе прав токена возвращаются только атрибуты с заполненными значениями. Справочник дополнительных атрибутов не поддерживает динамическое расширение в процессе работы продукта, заполняется только при установке версии продукта `SSO`.

Ограничения, в отличие от атрибутов объектов, связываются непосредственно с объектом. К объекту может применяться неограниченное количество ограничений. Значения ограничений задаются через роли. Значения ограничений типа «Предикат» объединяются в логическое выражение через операторы AND, NOT, OR. Ограничения остальных типов могут присутствовать только в одном экземпляре для каждого объекта, к которому применимы. При наличии нескольких значений одного и того же ограничения (не предиката) в наборе прав будет присутствовать ближайшее к пользователю (по иерархии ролей).

Например, младший менеджер может просматривать и редактировать только свои заказы. Старший менеджер может просматривать и редактировать все заказы своего филиала. Начальник отдела продаж может просматривать, редактировать и удалять все заказы во всех филиалах.

Для реализации данной модели нужно:

1. Создать объект `Order` типа «HTTP Resource» и разрешить над ним операции GET, POST, PUT, DELETE.
2. Создать ограничение-структуру `Orders control` с атрибутами `onlySelf` типа `Boolean` и `department` типа `Number`.
3. Разрешить применение ограничения `Orders control` к объекту `Order`.
4. Создать роли с правами:
   - менеджер:
     - право на объект `Order` (GET, POST, PUT);
     - значение ограничения `Orders control {onlySelf = true, department = 999}`;
   - старший менеджер:
     - права менеджера;
     - значение ограничения `Orders control {onlySelf = false, department = 999}`;
   - начальник отдела продаж:
     - права старшего менеджера;
     - право на объект `Order` (все операции).

Если не планируется использовать более одного параметра в ограничении, то тип ограничения можно заменить на простое значение или список значений.

5.9. Регистрация ролевой модели в SSO

Внимание!

Все запросы регистрации ролевых моделей должны выполняться от специальных пользователей. Для каждого продукта в `SSO` должен быть предварительно заведён собственный пользователь.

Категорически запрещается регистрировать ролевые модели всех продуктов от имени одного пользователя.

Правила создания пользователей:

- тип пользователя – технологический;
- логин пользователя – PRODUCT_<PRODUCT_NAME>;
- тип пароля – постоянный;
- роли – Security:Entities:Install.

В продукте `COMMON_INSTALLER` реализована специальная роль internal/sso/acl_role для регистрации ролевой модели. Имя пользователя и пароль должны храниться в inventory.

Ролевая модель хранится в формате json в файлах вида `<PRODUCT_NAME>-<XX>-roles.json`, где `XX` – уникальный номер. Рекомендуется использовать один файл.

Внимание!

При регистрации ролевой модели файлы всегда должны быть отсортированы по имени по возрастанию.

Рекомендуется во всех запросах передавать заголовок `pstxid`, сформированный по правилам:

```
<id>-<check|install>-<имя файла ролевой модели>-<номер попытки>
```

где `id` – случайное значение, генерирумое в начале установки ролевой модели и единое для всех запросов в рамках одной установки.

5.9.1. Рекомендации по проектированию ролевой модели

При проектировании ролевой модели важен порядок команд: сначала должны следовать команды создания объектов прав, затем – команды, создающие связи. Например, для того, чтобы создать связь между ролями, сначала нужно создать обе роли и только после этого создать связь между ними.

Все объекты прав (роли, ограничения) должны именоваться по правилу <Уникальный код продукта>:<Группа>:<Уникальный код сущности в рамках продукта>:

- уникальный код продукта необходим для исключения конфликта между продуктами; при нарушении этого правила возможны конфликты в рамках ролевой модели всего решения;
- группа – логическая группа, объединяющая сущности; например, для группировки ролей, регистрирующих доступ к объектам графического интерфейса, рекомендовано использовать группу GUI;
- уникальный код сущности – имя роли/объекта/ограничения, уникальное в рамках продукта.

5.9.2. Применение ролевой модели

Пример запроса применения ролевой модели:

```
curl -X POST \
http://srv.example.ru:47141/security/entities/install \
-H 'authorization: Bearer
LOCAL.AABbCljxKAEOhkErTQxw8aCZJcTn6XXVwQ73DWYwDhsSmXDQ2zGysPQ' \
-H 'cache-control: no-cache' \
-H 'content-type: application/json' \
-H 'postman-token: b049b8e8-f771-97bc-6023-b1b983d40eb9' \
-d '[{
    "method": "PUT",
    "entityType": "ROLE",
    "entityData": {
        "roleCode": "RoleName11",
        "nlsDescription": [{
            "languageCode": "en",
            "description": "RoleName1"
        }, {
            "languageCode": "ru",
            "description": "RoleName1"
        }],
        "public": true
        }
    }]'
```

Коды ответа:

- `204 No Content` – ролевая модель применена;
- `423 Locked` – блокировка, идет параллельная установка ролевой модели; повторите запрос;
- `422 Unprocessable Entity` – произошла ошибка применения ролевой модели; данная ситуация крайне маловероятна при предварительной валидации ролевой модели, но возможна при возникновении ошибок в момент записи в БД;
- `500 Internal Server Error` – произошла внутренняя ошибка сервиса, например, не работает БД `SSO`;
- `503 Service Unavailable` – сервис недоступен; повторите запрос.

6. Мониторинг

Продукт предоставляет наборы метрик для мониторинга (опроса метрик) работы своих компонентов.

Мониторинг сервисов SSO рекомендуется вести на базе продукта «Централизованная платформа мониторинга» (EMON).

В процессе установки SSO с помощью автоинсталлятора регистрируются сервисы в продукте EMON. На серверы, на которых находятся сервисы, инсталлятор EMON устанавливает свои агенты (Telegraf) для сбора метрик процесса мониторинга. EMON ведёт непрерывный мониторинг, анализирует полученные данные и выводит их в графическом интерфейсе и/или при необходимости рассылает уведомления по каналам SMS, E-mail и т.п. Для дополнительной информации см. документацию на продукт EMON.

Метрики мониторинга на базе EMON транслируются по протоколу Prometheus. Метрики расположены в файлах:

<Компонент>/mon/<Продукт>-<Версия>-<Компонент>-mon.zip/prm.<Компонент>.emon.