Arenadata™ Streaming

Версия - v1.4-RUS

Руководство администратора по работе с Nifi

2021 Arenadata™

Оглавление

1 Рекомендации по конфигурации **3**
- 1.1 Максимальное число дескрипторов . . . 3
- 1.2 Maximum Forked Processes . . . 3
- 1.3 Количество доступных портов TCP . . . 3
- 1.4 Статус сокетов TIMED_WAIT . . . 4
- 1.5 Отключение swapping в Linux . . . 4

2 Настройка безопасности **5**
- 2.1 Набор средств генерации TLS . . . 6

3 Аутентификация пользователя **10**
- 3.1 Lightweight Directory Access Protocol (LDAP) . . . 11
- 3.2 Kerberos . . . 13
- 3.3 OpenId Connect . . . 13
- 3.4 Apache Knox . . . 14

4 Настройка пользователей и политик доступа **15**
- 4.1 Создание пользователей и групп . . . 15
- 4.2 Политики доступа . . . 15
- 4.3 Настройка политик доступа на основе конкретных примеров . . . 22

5 Kerberos Service **36**

В руководстве приведены сведения для администраторов системы по работе с платформой ADS в части сервиса Nifi – рекомендации по конфигурации, аутентификация пользователей с настройками их политик и доступа, Kerberos.

Руководство может быть полезно администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих сопровождение платформы.

Important: Контактная информация службы поддержки – e-mail: info@arenadata.io

Глава 1

Рекомендации по конфигурации

При работе в **Linux** необходимо учесть последующие рекомендации, так как типичные значения по умолчанию для **Linux** могут быть не настроены под нужды такого интенсивного приложения с высоким уровнем ввода-вывода, как **NiFi**.

Important: При установке ADS через ADCM рекомендации из данного раздела применяются автоматически

1.1 Максимальное число дескрипторов

Сервис **NiFi** в любой момент может открыть очень большое количество файлов, поэтому необходимо увеличить лимиты, отредактировав файл */etc/security/limits.conf*. Например:

```
*  hard  nofile  50000
*  soft  nofile  50000
```

1.2 Maximum Forked Processes

NiFi может быть настроен для генерации существенного количества потоков. Для увеличения их допустимого числа необходимо отредактировать файл */etc/security/limits.conf*, например:

```
*  hard  nproc  10000
*  soft  nproc  10000
```

При этом дистрибутив может потребовать так же изменить значение в */etc/security/limits.d/90-nproc.conf*, тогда следует добавить в него:

```
*  soft  nproc  10000
```

1.3 Количество доступных портов TCP

Увеличение количества доступных портов TCP особенно важно в случае, если поток устанавливает и срывает большое количество сокетов за короткий промежуток времени:

```
sudo sysctl -w net.ipv4.ip_local_port_range="10000 65000"
```

1.4 Статус сокетов TIMED_WAIT

Для того, чтобы сокеты долго не задерживались, и при необходимости быстрой настройки и отключения новых сокетов следует изменить время нахождения сокетов в статусе *TIMED_WAIT* при их закрытии, например:

```
sudo sysctl -w net.ipv4.netfilter.ip_conntrack_tcp_timeout_time_wait="1"
```

1.5 Отключение swapping в Linux

Для **NiFi** сервисов, всегда находящихся в непрерывной работе, swapping не подходит. Поэтому, чтобы сообщить **Linux** об отключении подкачки, следует отредактировать файл */etc/sysctl.conf*, добавив строку:

```
vm.swappiness = 0
```

При этом для партиций, обрабатывающих различные NiFi-репозитории, необходимо отключить `atime`, что в результате приводит к увеличению производительности. Поэтому рекомендуется отредактировать файл */etc/fstab*, а для конкретных партиций добавить опцию `noatime`.

Глава 2

Настройка безопасности

В целях безопасности сервис **NiFi** предоставляет несколько различных параметров конфигурации. Наиболее важными являются свойства под заголовком *"security properties"* в файле *nifi.properties*.

Для безопасной работы должны быть установлены свойства, приведенные в таблице.

Таблица2.1.: Описание свойств безопасности NiFi

Свойство	Описание
nifi.security.keystore	Имя файла Keystore, содержащего закрытый ключ сервера
nifi.security.keystoreType	Тип Keystore. Должен быть либо PKCS12, либо JKS. JKS является предпочтительным типом, файлы PKCS12 загружаются библиотекой BouncyCastle
nifi.security.keystorePasswd	Пароль Keystore
nifi.security.keyPasswd	Пароль для сертификата в Keystore. Если значение не установлено, используется `nifi.security.keystorePasswd`
nifi.security.truststore	Имя файла Truststore для авторизации при подключении к NiFi. Защищенный инстанс без Truststore отклоняет все входящие подключения
nifi.security.truststoreType	Тип Truststore. Должен быть либо PKCS12, либо JKS. JKS является предпочтительным типом, файлы PKCS12 загружаются библиотекой BouncyCastle
nifi.security.truststorePasswd	Пароль Truststore
nifi.security.needClientAuth	Значение `true` требует пройти аутентификацию при подключении клиентов. Свойство используется протоколом кластера NiFi для подтверждения, что узлы в кластере аутентифицированы и имеют сертификаты, которым доверяют Truststores

После настройки перечисленных свойств можно разрешить доступ к пользовательскому интерфейсу через HTTPS вместо HTTP. Это достигается путем установки свойств *nifi.web.https.host* и *nifi.web.https.port*. Свойство *nifi.web.https.host* указывает, на каком хосте должен работать сервер. При необходимости доступности интерфейса HTTPS со всех сетевых интерфейсов следует использовать значение `0.0.0.0`. Для того, чтобы администраторы могли настраивать приложение для работы только на определенных сетевых интерфейсах, следует указать свойства *nifi.web.http.network.interface* и *nifi.web.https.network.interface*.

Important: При включении HTTPS необходимо исключить свойство *nifi.web.http.port*, так как NiFi поддерживает либо HTTP, либо HTTPS

Так же и с *nifi.security.needClientAuth* – веб-сервер может быть настроен на требование аутентификации

на основе сертификатов у пользователей, обращающихся к интерфейсу. Для этого веб-сервер не должен поддерживать аутентификацию имени пользователя и пароля с помощью протокола **LDAP** или **Kerberos**, так как любой из этих параметров настраивает проверку подлинности клиента на основе сертификатов, а у кого их нет, могут войти в систему под своими учетными данными или получить анонимный доступ. Но если аутентификация по имени пользователя и паролю и анонимный доступ не настроены, то веб-сервер запрашивает аутентификацию клиента на основе сертификата (см. Аутентификация пользователя).

После защиты пользовательского интерфейса следует обеспечить внутренние кластерные коммуникации и связь между сайтами. Это достигается установкой свойств *nifi.remote.input.secure* и *nifi.cluster.protocol.is.secure* в значение `true`.

2.1 Набор средств генерации TLS

Для упрощения установки **NiFi** и автоматического создания необходимых хранилищ ключей, доверительного хранилища и соответствующих файлов конфигурации можно использовать утилиту командной строки **tls-toolkit**, что так же обеспечит безопасность многочисленных узлов **NiFi**.

Wildcard-сертификаты (т. е. два узла *node1.nifi.apache.org* и *node2.nifi.apache.org*, которым назначается тот же сертификат с записью CN или SAN *.nifi.apache.org*) официально не поддерживаются и не рекомендуются. Их использование имеет множество недостатков и приемлемо только, если каждый сертификат поддерживает дополнительную уникальную запись SAN и запись CN.

Потенциальные проблемы использования wildcard-сертификатов:

- Кластерные связи многократно используют идентификаторы сертификатов для определения узла, а если сертификат представляет собой подстановочное DN, то он не даст ответа;
- Администраторам может потребоваться предоставить кастомный идентификатор узла в *authorizers.xml* для *.nifi.apache.org*, поскольку все действия прокси-сервера разрешаются только в сертификате DN (см. Аутентификация пользователя);
- Администраторы не имеют возможности отслеживать, в каком узле выполняется действие, так как все они направляются в один и тот же DN;
- Администраторы, запускающие несколько инстансов на одном компьютере и используя разные порты для их идентификации, могут случайно поместить узел *node1* с портом *node2*, и адрес будет в итоге удален, потому что он использует тот же сертификат, а обработчик узла блокирует его, так как имя узла *node1* не указано в качестве допустимого хоста для инистанса *node2*;
- Если wildcard-сертификат скомпрометирован, все узлы оказываются под угрозой.

Important: Для keystores и truststores в NiFi рекомендуются JKS. Этот инструмент позволяет задавать другие типы хранилищ ключей в командной строке и игнорировать тип PKCS12 для использования в качестве доверительного хранилища, потому что данный формат имеет проблемы совместимости между реализациями BouncyCastle и Oracle

Инструмент командной строки **tls-toolkit** имеет два основных режима работы:

- *Standalone* (автономный) – создает организацию сертификатов, хранилища ключей, доверительные хранилища и файлы *nifi.properties* в одной команде;
- *Client/Server* (Клиент/Сервер) – использует Certificate Authority Server, который принимает запросы на подписание сертификатов от клиентов, подписывает и отправляет обратно. И клиент, и сервер проверяют идентификацию друг друга через общий секрет.

2.1.1 Standalone

Автономный режим вызывается запуском `./bin/tls-toolkit.sh standalone -h` и отображает информацию об использовании с описаниями опций, которые могут быть указаны.

В автономном режиме с **tls-toolkit** можно использовать следующие параметры командной строки:

- `-a`, `--keyAlgorithm <arg>` – алгоритм использования сгенерированных ключей (по умолчанию: *RSA*);
- `-B`, `--clientCertPassword <arg>` – пароль сертификата клиента. Должно быть либо одно значение, либо одно для каждого DN клиента (если не задано, генерируется автоматически);
- `-c`, `--certificateAuthorityHostname <arg>` – имя хоста NiFi Certificate Authority (по умолчанию: *localhost*);
- `-C`, `--clientCertDn <arg>` – создание сертификата клиента, подходящего для использования в браузере, с указанным DN (может быть указан несколько раз);
- `-d`, `--days <arg>` – количество дней, в течение которых выданный сертификат является действительным (по умолчанию: *1095*);
- `-f`, `--nifiPropertiesFile <arg>` – базовый файл *nifi.properties* для обновления (если не указан, используется встроенный файл, идентичный файлу по умолчанию при установке **NiFi**);
- `-g`, `--differentKeyAndKeystorePasswords` – использование другого сгенерированного пароля для ключа и хранилища ключей;
- `-G`, `--globalPortSequence <arg>` – использование последовательных портов, которые вычисляются для всех хостов в соответствии с предоставленными выражениями имен хостов (могут быть указаны несколько раз, но должны быть одинаковыми от запуска до запуска);
- `-h`, `--help` – печать справки и выход;
- `-k`, `--keySize <arg>` – количество бит для генерации ключей (по умолчанию: *2048*);
- `-K`, `--keyPassword <arg>` – пароль ключа. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
- `-n`, `--hostnames <arg>` – список имен хостов через запятую;
- `--nifiDnPrefix <arg>` – строка для добавления имени хоста (в начало) при определении DN (по умолчанию: *CN=*);
- `--nifiDnSuffix <arg>` – строка для добавления имени хоста (в конец) при определении DN (по умолчанию: *OU=NIFI*);
- `-o`, `--outputDirectory <arg>` – каталог для вывода keystore, truststore и config-файлов (по умолчанию: *../bin*);
- `-O`, `--isOverwrite` – перезапись существующего вывода хоста;
- `-P`, `--trustStorePassword <arg>` – пароль truststore. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
- `-s`, `--signingAlgorithm <arg>` – алгоритм подписи сертификатов (по умолчанию: *SHA256WITHRSA*);
- `-S`, `--keyStorePassword <arg>` – пароль keytstore. Либо одно значение, либо одинаковое для каждого хоста (если не задано, генерируется автоматически);
- `--subjectAlternativeNames <arg>` – разделенный запятыми список доменов для использования в качестве альтернативных имен в сертификате;
- `-T`, `--keyStoreType <arg>` – тип создаваемого хранилища ключей (по умолчанию: *jks*).

Шаблоны имен хостов:

- Для указания диапазона имен хостов используются квадратные скобки, например: `[01-20]`;
- Круглые скобки используются для определения, что на хосте (хостах) работает больше, чем один инстанс **NiFi**, например: `(5)`.

Примеры:

- Создать 4 набора хранилищ ключей, truststore, nifi.properties для localhost вместе с сертификатом клиента с предоставленным DN:

```
bin/tls-toolkit.sh standalone -n 'localhost(4)' -C 'CN=username,OU=NIFI'
```

- Создать хранилище ключей, truststore, nifi.properties для 10 имен хостов NiFi в каждом из 4 поддоменов:

```
bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain'
```

- Создать 2 набора хранилищ ключей, truststore, nifi.properties для 10 имен хостов NiFi в каждом из 4 поддоменов вместе с сертификатом клиента с предоставленным DN:

```
bin/tls-toolkit.sh standalone -n 'nifi[01-10].subdomain[1-4].domain(2)' -C 'CN=username,OU=NIFI'
```

2.1.2 Client/Server

Режим Клиент/Сервер опирается на Центр сертификации (Certificate Authority, CA) для выдачи сертификатов. Центр можно остановить, если узлы не подключены к сети.

Server

Сервер CA вызывается запуском `./bin/tls-toolkit.sh -h`, который печатает информацию об использовании с описаниями опций, которые могут быть заданы.

В режиме сервера с **tls-toolkit** можно использовать следующие параметры командной строки:

- `-a`, `--keyAlgorithm <arg>` – алгоритм использования сгенерированных ключей (по умолчанию: *RSA*);
- `--configJsonIn <arg>` – место для чтения информации о конфигурации, подразумевает *useConfigJson*, если установлено (по умолчанию: значение *configJson*);
- `-d`, `--days <arg>` – количество дней, в течение которых выданный сертификат является действительным (по умолчанию: *1095*);
- `-D`, `--dn <arg>` – DN для сертификата CA (по умолчанию: *CN=YOUR_CA_HOSTNAME,OU=NIFI*);
- `-f`, `--configJson <arg>` – место записи информации о конфигурации (по умолчанию: *config.json*);
- `-F`, `--useConfigJson` – флаг, указывающий, что вся конфигурация считывается из *configJson* (для облегчения автоматического использования, иначе в *configJson* производится только запись);
- `-g`, `--differentKeyAndKeystorePasswords` – использование другого сгенерированного пароля для ключа и хранилища ключей;
- `-h`, `--help` – печать справки и выход;
- `-k`, `--keySize <arg>` – количество бит для генерации ключей (по умолчанию: *2048*);
- `-p`, `--PORT <arg>` – порт для прослушивания центром сертификации (по умолчанию: *8443*);
- `-s`, `--signingAlgorithm <arg>` – алгоритм подписи сертификатов (по умолчанию: *SHA256WITHRSA*);
- `-T`, `--keyStoreType <arg>` – тип создаваемого хранилища ключей (по умолчанию: *jks*);
- `-t`, `--token <arg>` – маркер для предотвращения MITM (должен быть таким же, как тот, что используется клиентами).

Client

Клиент может использоваться для запроса новых сертификатов из центра сертификации. Утилита клиента генерирует пару ключей и запрос подписи сертификата (CSR, Certificate Signing Request), после чего отправляет CSR в центр сертификации. Клиент вызывается запуском `./bin/tls-toolkit.sh client -h`, который печатает информацию об использовании с описаниями опций, которые могут быть заданы.

В режиме клиента с **tls-toolkit** можно использовать следующие параметры командной строки:

- `-a`, `--keyAlgorithm <arg>` – алгоритм использования сгенерированных ключей (по умолчанию: *RSA*);
- `-c`, `--certificateAuthorityHostname <arg>` – имя хоста NiFi Certificate Authority (по умолчанию: *localhost*);
- `-C`, `--certificateDirectory <arg>` – каталог записи сертификата CA (по умолчанию: .);
- `--configJsonIn <arg>` – место для чтения информации о конфигурации, подразумевает *useConfigJson*, если установлено (по умолчанию: значение *configJson*);
- `-D`, `--dn <arg>` – DN для сертификата клиента (по умолчанию: *CN=<localhost name>,OU=NIFI*, заполняется автоматически инструментом);
- `-f`, `--configJson <arg>` – место записи информации о конфигурации (по умолчанию: *config.json*);
- `-F`, `--useConfigJson` – флаг, указывающий, что вся конфигурация считывается из *configJson* (для облегчения автоматического использования, иначе в *configJson* производится только запись);
- `-g`, `--differentKeyAndKeystorePasswords` – использование другого сгенерированного пароля для ключа и хранилища ключей;
- `-h`, `--help` – печать справки и выход;
- `-k`, `--keySize <arg>` – количество бит для генерации ключей (по умолчанию: *2048*);
- `-p`, `--PORT <arg>` – порт для прослушивания центром сертификации (по умолчанию: *8443*);
- `--subjectAlternativeNames <arg>` – разделенный запятыми список доменов для использования в качестве альтернативных имен в сертификате;
- `-T`, `--keyStoreType <arg>` – тип создаваемого хранилища ключей (по умолчанию: *jks*);
- `-t`, `--token <arg>` – маркер для предотвращения MITM (должен быть таким же, как тот, что используется клиентами).

В результате запуска клиента предоставляется сертификат CA, keystore, truststore и config.json с информацией о них, а также их пароли.

Сертификат клиента можно легко импортировать в браузер, указав: `-T PKCS12`.

Глава 3

Аутентификация пользователя

NiFi поддерживает аутентификацию пользователей через сертификаты клиента, через имя пользователя и пароль, через **Apache Knox** или через OpenId Connect.

Проверка подлинности имени пользователя и пароля выполняется с помощью “Идентификатора входа в систему” (“Login Identity Provider”) – это подключаемый механизм для аутентификации пользователей через их имя и пароль, настройка которого осуществляется в файле *nifi.properties*. В настоящее время **NiFi** предлагает проверку имени пользователя и пароля с параметрами Provider Identity Provider для **LDAP** и **Kerberos**.

Свойство *nifi.login.identity.provider.configuration.file* указывает файл конфигурации для Идентификатора входа в систему. Свойство *nifi.security.user.login.identity.provider* указывает, какой из настроенных Login Identity Provider должен использоваться. По умолчанию свойство не настроено, что означает, что username/password должно быть явно включено.

При аутентификации через **OpenId Connect** сервер **NiFi** перенаправляет пользователей для проверки подлинности в Провайдер, а затем **NiFi** вызывает Провайдер для получения идентификации пользователя.

При аутентификации через **Apache Knox** сервер **NiFi** перенаправляет пользователей для проверки подлинности в **Apache Knox**, а затем **NiFi** во время аутентификации пользователя проверяет токен **Apache Knox**.

Important: Аутентификация пользователя в NiFi может быть настроена только по username/password, OpenId Connect или Apache Knox. Сервер не поддерживает одновременный запуск каждого из них. При этом NiFi потребуются сертификаты клиентов для аутентификации пользователей через HTTPS, если ничто иное не настроено

К защищенному инстансу **NiFi** нельзя получить доступ анонимно, если в **LDAP** или **Kerberos** не настроен Login Identity Provider, который, в свою очередь, должен быть настроен на явное разрешение анонимного доступа. При этом анонимный доступ в настоящее время невозможен по умолчанию в *FileAuthorizer* (NIFI-2730).

Important: NiFi не выполняет аутентификацию пользователя через HTTP (используя HTTP, всем пользователям предоставляются все роли)

3.1 Lightweight Directory Access Protocol (LDAP)

Далее приведен пример с описанием настроек Login Identity Provider, который интегрируется с Directory Server для аутентификации пользователей.

```
<provider>
    <identifier>ldap-provider</identifier>
    <class>org.apache.nifi.ldap.LdapProvider</class>
    <property name="Authentication Strategy">START_TLS</property>

    <property name="Manager DN"></property>
    <property name="Manager Password"></property>

    <property name="TLS - Keystore"></property>
    <property name="TLS - Keystore Password"></property>
    <property name="TLS - Keystore Type"></property>
    <property name="TLS - Truststore"></property>
    <property name="TLS - Truststore Password"></property>
    <property name="TLS - Truststore Type"></property>
    <property name="TLS - Client Auth"></property>
    <property name="TLS - Protocol"></property>
    <property name="TLS - Shutdown Gracefully"></property>

    <property name="Referral Strategy">FOLLOW</property>
    <property name="Connect Timeout">10 secs</property>
    <property name="Read Timeout">10 secs</property>

    <property name="Url"></property>
    <property name="User Search Base"></property>
    <property name="User Search Filter"></property>

    <property name="Identity Strategy">USE_DN</property>
    <property name="Authentication Expiration">12 hours</property>
</provider>
```

С помощью данной конфигурации аутентификация имени пользователя и пароля может быть активирована путем ссылки на провайдер в *nifi.properties*:

```
nifi.security.user.login.identity.provider=ldap-provider
```

Таблица3.1.: Описание настроек Login Identity Provider для LDAP

Свойство	Описание
Authentication Strategy	Аутентификация подключения к LDAP-серверу. Возможные значения: ANONYMOUS, SIMPLE, LDAPS или START_TLS
Manager DN	DN менеджера, который используется для привязки к LDAP-серверу для поиска пользователей
Manager Password	Пароль менеджера, который используется для привязки к LDAP-серверу для поиска пользователей
TLS - Keystore	Путь к Keystore при подключении к LDAP с использованием LDAPS или START_TLS
TLS - Keystore Password	Пароль для Keystore при подключении к LDAP с использованием LDAPS или START_TLS
TLS - Keystore Type	Тип Keystore при подключении к LDAP с использованием LDAPS или START_TLS (то есть JKS или PKCS12)
TLS - Truststore	Путь к Truststore при подключении к LDAP с использованием LDAPS или START_TLS
TLS - Truststore Password	Пароль для Truststore при подключении к LDAP с использованием LDAPS или START_TLS
TLS - Truststore Type	Тип Truststore при подключении к LDAP с использованием LDAPS или START_TLS (то есть JKS или PKCS12)
TLS - Client Auth	Политика аутентификации клиента при подключении к LDAP с использованием LDAPS или START_TLS. Возможные значения: REQUIRED, WANT, NONE
TLS - Protocol	Протокол при подключении к LDAP с использованием LDAPS или START_TLS (TLS, TLSv1.1, TLSv1.2 и т.д.)
TLS - Shutdown Gracefully	Указывает, следует ли корректно завершать работу TLS перед закрытием целевого контекста. По умолчанию: false
Referral Strategy	Стратегия обработки рефералов. Возможные значения: FOLLOW, IGNORE, THROW
Connect Timeout	Время ожидания соединения (10 секунд)
Read Timeout	Время ожидания чтения (10 секунд)
Url	Разделенный пробелами список URL-адресов серверов LDAP (например, ldap://<hostname>:<port>)
User Search Base	Базовый DN для поиска пользователей (например, CN=Users,DC=example,DC=com)
User Search Filter	Фильтр для поиска пользователей в User Search Base (sAMAccountName={0}). Указанное пользователем имя вставляется в *{0}*
Identity Strategy	Стратегия идентификации пользователей. Возможные значения: USE_DN и USE_USERNAME. По умолчанию: - USE_DN (для сохранения обратной совместимости). USE_DN использует полный DN пользовательской записи (рекомендуется). USE_USERNAME использует имя пользователя, под которым он вошел в систему
Authentication Expiration	Продолжительность действия проверки подлинности пользователя. Если пользователь никогда не выходит
	из системы, он должен будет снова войти в систему в течение указанного времени

3.2 Kerberos

Далее приведен пример с описанием настроек Login Identity Provider, который интегрируется с Kerberos Key Distribution Center (KDC) для аутентификации пользователей.

```
<provider>
    <identifier>kerberos-provider</identifier>
    <class>org.apache.nifi.kerberos.KerberosProvider</class>
    <property name="Default Realm">NIFI.APACHE.ORG</property>
    <property name="Kerberos Config File">/etc/krb5.conf</property>
    <property name="Authentication Expiration">12 hours</property>
</provider>
```

С помощью данной конфигурации аутентификация имени пользователя и пароля может быть активирована путем ссылки на провайдер в *nifi.properties*:

```
nifi.security.user.login.identity.provider=kerberos-provider
```

Таблица3.2.: Описание настроек Login Identity Provider для Kerberos

Свойство	Описание
Default Realm	Область по умолчанию для предоставления пользователю в случае, если пользователь вводит неполный пользовательский принципал (например, NIFI.APACHE.ORG)
Kerberos Config File	Абсолютный путь к файлу конфигурации клиента Kerberos
Authentication Expiration	Продолжительность действия проверки подлинности пользователя. Если пользователь никогда не выходит из системы, он должен будет снова войти в систему в течение указанного времени

Описание настройки допуска по единому входу через клиентские тикеты Kerberos приведено в Kerberos Service.

3.3 OpenId Connect

Для включения аутентификации через OpenId Connect необходимо настроить свойства в *nifi.properties*, представленные далее в таблице.

Таблица3.3.: Описание настроек для аутентификации через OpenId Connect

Свойство	Описание
nifi.security.user.oidc.discovery.url	URL-адрес обнаружения необходимого OpenId Connect Provider (http://openid.net/specs/openid-connect-discovery-1_0.html)
nifi.security.user.oidc.connect.timeout	Время ожидания соединения при обмене данными с OpenId Connect Provider
nifi.security.user.oidc.read.timeout	Время ожидания чтения при обмене данными с OpenId Connect Provider
nifi.security.user.oidc.client.id	Идентификатор клиента для NiFi после регистрации в OpenId Connect Provider
nifi.security.user.oidc.client.secret	Секрет клиента для NiFi после регистрации в OpenId Connect Provider
nifi.security.user.oidc.preferred.jwsalgorithm	Предпочтительный алгоритм проверки токенов идентификации. Если значение свойства пустое, по умолчанию используется *RS 256*, поддерживаемое OpenId Connect Provider в соответствии со спецификацией. Если значение равно *HS256*, *HS384* или *HS512*, NiFi пытается проверить защищенные токены HMAC, используя указанный секретный ключ. Если значение свойства равно *none*, NiFi пытается проверить незащищенные/простые токены. Иные значения для алгоритма анализируются как алгоритм RSA или EC, который используется совместно с JSON Web Key (JWK), предоставленным через jwks_uri в метаданных URL-обнаружения

3.4 Apache Knox

Для включения аутентификации через **Apache Knox** необходимо настроить свойства в *nifi.properties*, представленные далее в таблице.

Таблица3.4.: Описание настроек для аутентификации через Apache Knox

Свойство	Описание
nifi.security.user.knox.url	URL-адрес страницы входа в Apache Knox
nifi.security.user.knox.publicKey	Путь к открытому ключу Apache Knox для проверки подписей токенов аутентификации в HTTP Cookie
nifi.security.user.knox.cookieName	Имя файла HTTP Cookie, которое Apache Knox создает после успешного входа в систему
nifi.security.user.knox.audiences	(Опционально) Разделенный запятыми список разрешенных подключений. Если значение задано, подключение должно присутствовать в списке. Разрешенные подключения, заполненные токеном, могут быть настроены в Knox

Глава 4

Настройка пользователей и политик доступа

В зависимости от характеристик настроенных параметров *UserGroupProvider* и *AccessPolicyProvider* пользователи, группы и политики могут конфигурироваться в пользовательском интерфейсе. Если расширения не настроены, то в пользовательском интерфейсе пользователи, группы и политики доступны только для чтения. Если сконфигурированный авторизатор не использует *UserGroupProvider* и *AccessPolicyProvider*, пользователи и политики могут быть или не быть видимыми и настраиваемыми в пользовательском интерфейсе на основе базовой реализации.

Далее в главе предполагается, что пользователи, группы и политики настраиваются в пользовательском интерфейсе, и описывается:

- *Создание пользователей и групп*
- *Политики доступа*
- *Настройка политик доступа на основе конкретных примеров*

4.1 Создание пользователей и групп

В пользовательском интерфейсе необходимо выбрать “Users” в глобальном меню, при этом открывыается диалоговое окно для создания пользователей и групп и управления ими (Рис.4.1.). Для создания пользователей и групп используется кнопка “Add User”.

Для создания пользователя в новом открывшемся диалоговом окне необходимо выбрать “Individual” и ввести информацию “Identity”, соответствующую методу аутентификации защиты инстанса **NiFi**. После чего нажать “OK” (Рис.4.2.).

Для создания группы в диалоговом окне следует выбрать “Group”, ввести имя группы в поле “Identity” и отметить пользователей в “Members”, которые необходимо включить в группу. После чего нажать “OK” (Рис.4.3.).

4.2 Политики доступа

Управление возможностями пользователей и групп **NiFi** осуществляется с помощью политик доступа. Существует два типа политик доступа, которые могут быть применены к ресурсу:

- View (просмотр) – если ресурсу назначается политика просмотра, то добавленные в эту политику пользователи и группы могут только видеть детали данного ресурса;

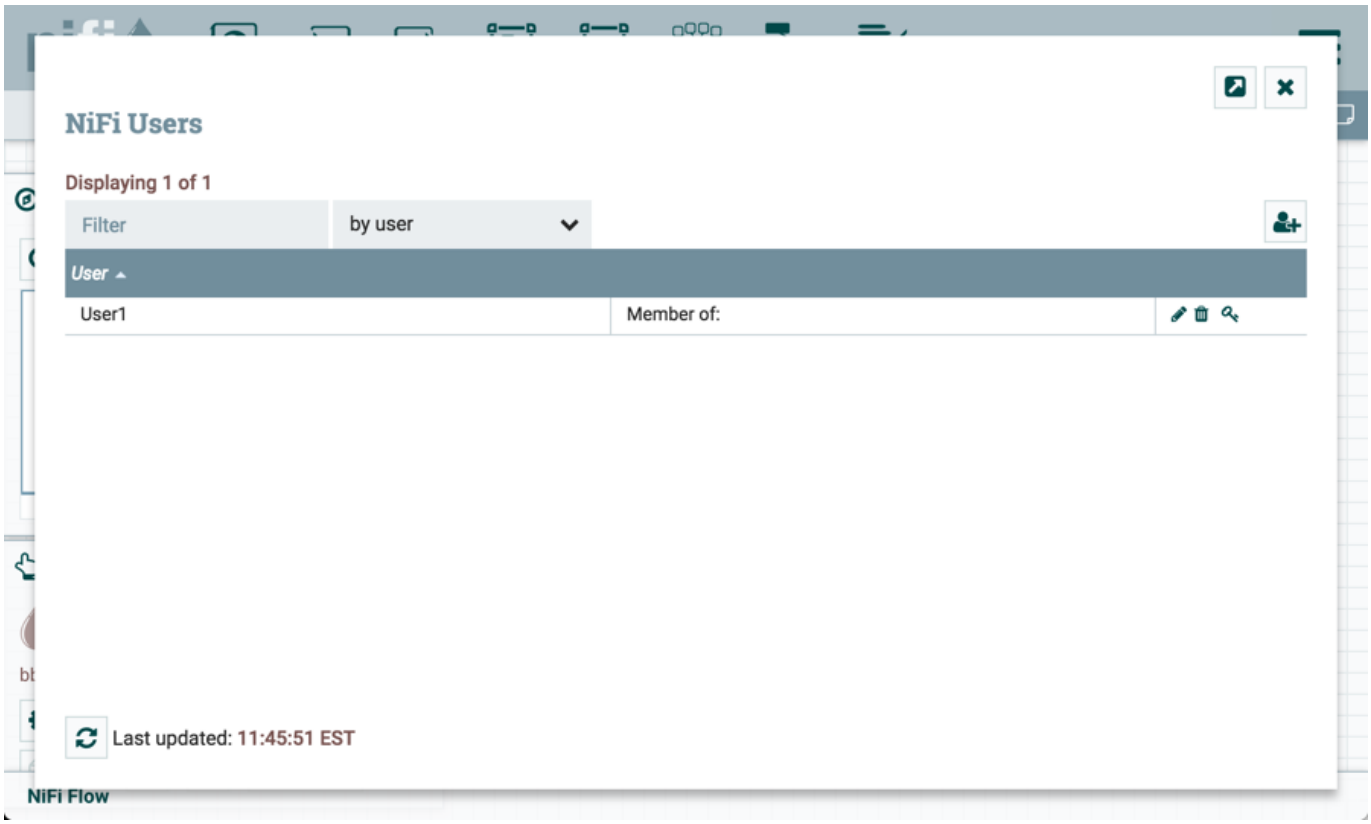


Рис.4.1.: Nifi Users

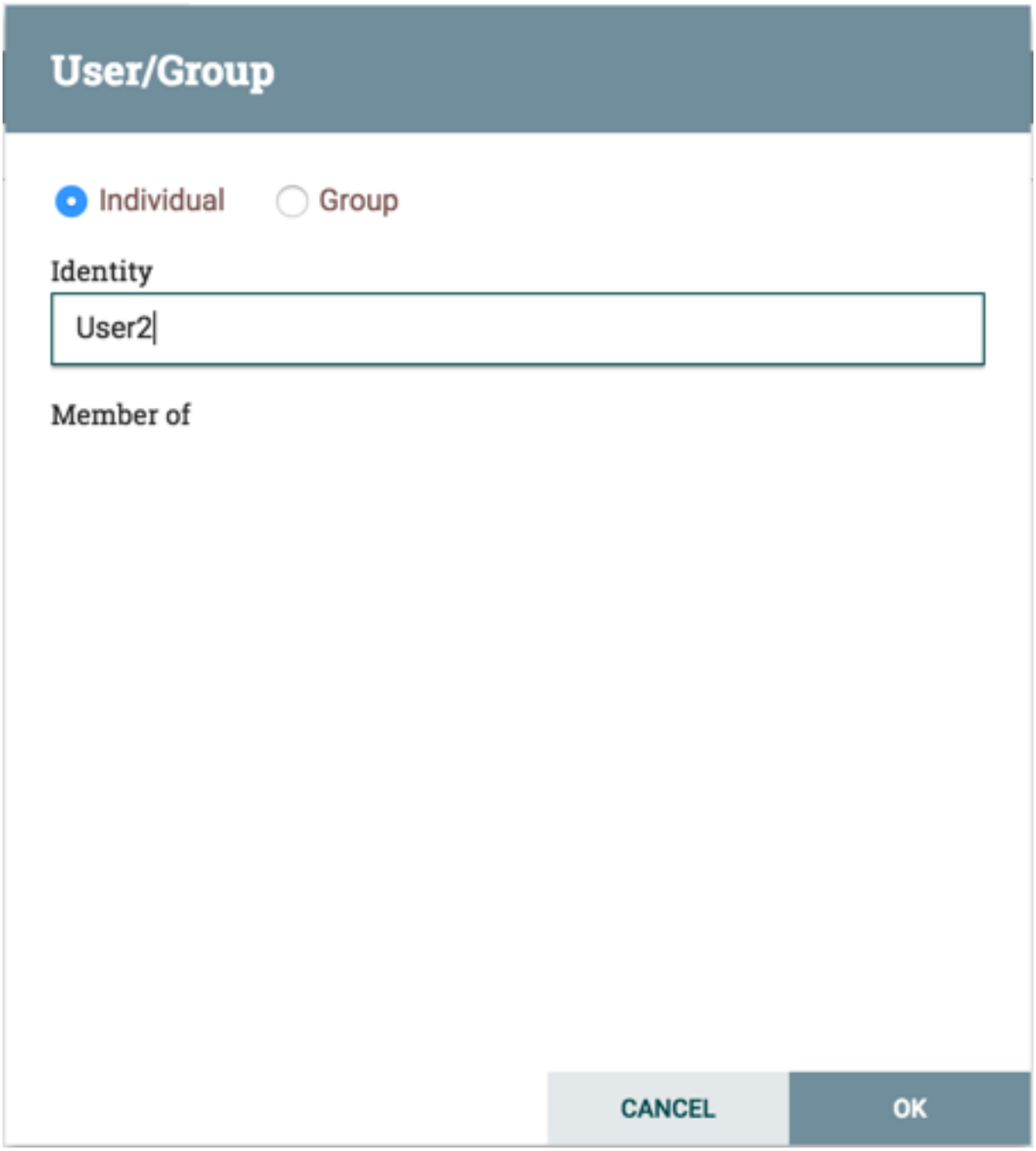


Рис.4.2.: Создание пользователя

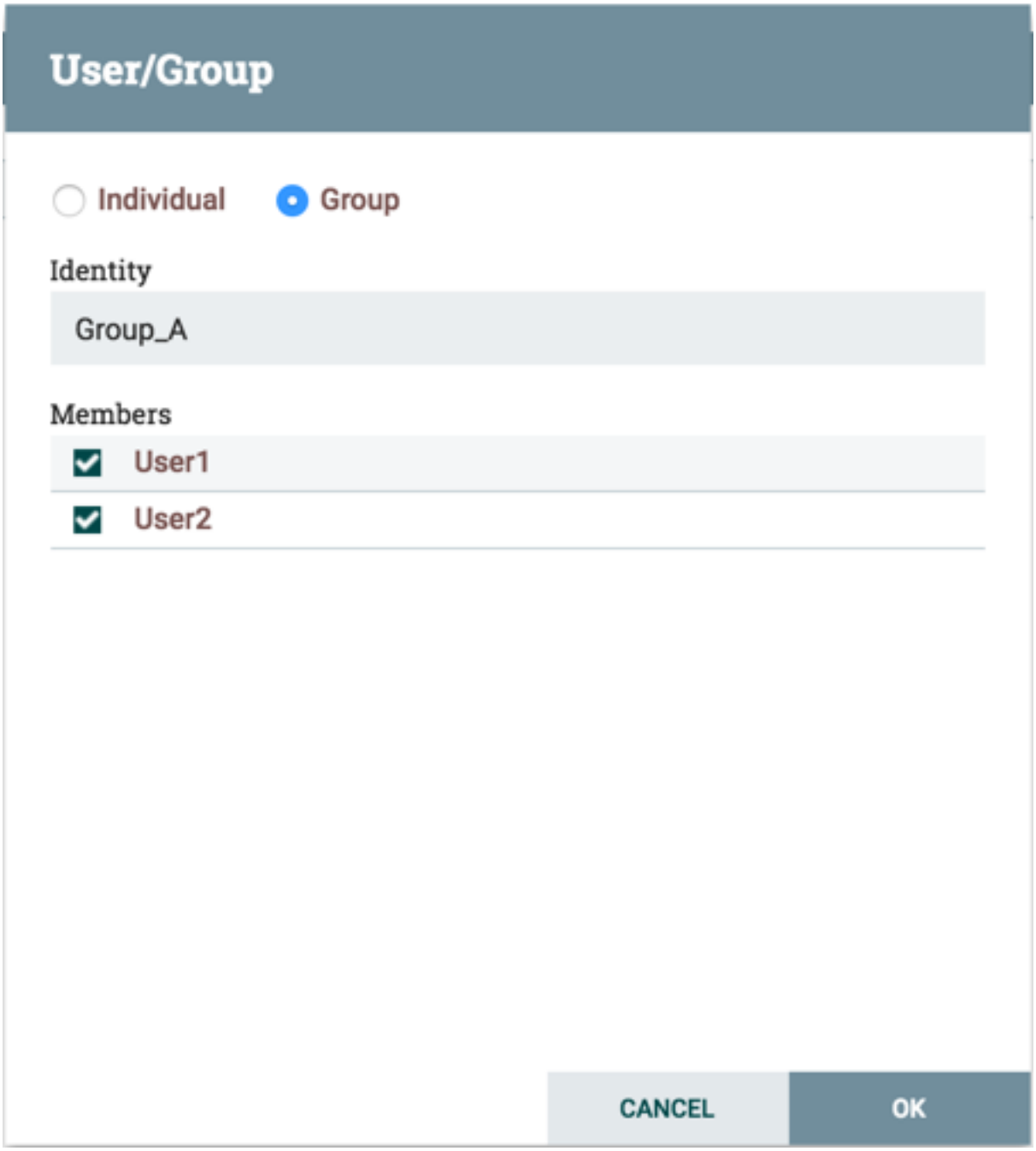


Рис.4.3.: Создание группы

- Modify (изменение) – если ресурсу назначается политика изменения, то добавленные в эту политику пользователи и группы могут изменить конфигурацию данного ресурса.

Создавать и применять политики доступа можно как на глобальном уровне (*Global Access Policies*), так и на уровне компонентов (*Component Level Access Policies*).

4.2.1 Global Access Policies

Политики глобального доступа управляют следующими полномочиями на уровне системы:

Таблица4.1.: Global Access Policies

Policy	Privilege	Global Menu Selection	Resource Descriptor
view the UI	Разрешение пользователям просматривать UI	N/A	/flow
access the controller	Позволяет пользователям просматривать/изменять контроллер, включая задачи отчетности, службы контроллеров и узлы в кластере	Controller Settings	/controller
query provenance	Позволяет пользователям отправлять Provenance Search и запрашивать Event Lineage	Data Provenance	/provenance
access restricted components	Позволяет пользователям создавать/изменять ограниченные компоненты при условии наличия других разрешений. Ограниченные компоненты могут указывать, какие конкретные разрешения требуются. Разрешения могут предоставляться для определенных ограничений или независимо от них. Если разрешение предоставляется независимо от ограничений, пользователь может создавать/изменять все ограниченные компоненты	N/A	/restricted-components
access all policies	Позволяет пользователям просматривать/изменять политики для всех компонентов	Policies	/policies
access users/user groups	Позволяет пользователям просматривать/изменять пользователей и группы пользователей	Users	/tenants
retrieve site-to-site details	Позволяет другим инстансам NiFi извлекать информацию site-to-site	N/A	/site-to-site
view system diagnostics	Позволяет пользователям просматривать системную диагностику	Summary	/system

4.2.2 Component Level Access Policies

Политики доступа на уровне компонентов управляют следующими полномочиями на уровне компонентов:

Таблица4.2.: Component Level Access Policies

Policy	Privilege	Resource Descriptor & Action
view the component	Позволяет пользователям просматривать детали конфигурации компонентов	resource=”/<component-type>/<component-UUID>” action=”R”
modify the component	Позволяет пользователям изменять детали конфигурации компонентов	resource=”/<component-type>/<component-UUID>” action=”W”
view provenance	Позволяет пользователям просматривать события происхождения, созданные компонентом	resource=”/provenance-data/<component-type>/<comp action=”R”
view the data	Позволяет пользователям просматривать метаданные и содержимое компонента в очередях потока в исходящих соединениях и через события происхождения	resource=”/data/<component-type>/<component-UUID action=”R”
modify the data	Позволяет пользователям очищать очереди потоков в исходящих соединениях и повторно отправлять через события происхождения	resource=”/data/<component-type>/<component-UUID action=”W”
view the policies	Позволяет пользователям просматривать список пользователей, которые могут просматривать/изменять компонент	resource=”/policies/<component-type>/<component-U action=”R”
modify the policies	Позволяет пользователям изменять список пользователей, которые могут просматривать/изменять компонент	resource=”/policies/<component-type>/<component-U action=”W”
receive data via site-to-site	Позволяет порту получать данные из инстансов NiFi	resource=”/data-transfer/input-ports/<port-UUID>” action=”W”
send data via site-to-site	Позволяет порту отправлять данные из инстансов NiFi	resource=”/data-transfer/output-ports/<port-UUID>” action=”W”

Important: Политики доступа можно применять ко всем типам компонентов, кроме соединений. Разрешения на соединения определяются по индивидуальным политикам доступа к исходному и целевому компонентам соединения, а так же по политике доступа группы процессов, содержащей компоненты. Более подробно это описано далее в примерах

Important: Для доступа к List Queue и Delete Queue для соединения пользователю требуются политики “view the data” и “modify the data” на компоненте. Так же все узлы в кластерной среде должны быть добавлены к этим политикам, так как запрос пользователя может быть реплицирован через любой узел в кластере

4.3 Настройка политик доступа на основе конкретных примеров

Самый эффективный способ понять, как создавать и применять политики доступа, – это пройтись по некоторым распространенным примерам. В приведенных далее сценариях *User1* является администратором, а *User2* – недавно добавленным пользователем, которому предоставлен доступ только к пользовательскому интерфейсу. На рисунке в качестве отправных точек показаны два процессора в рабочей области: GenerateFlowFile и LogAttribute (Рис.4.4.).

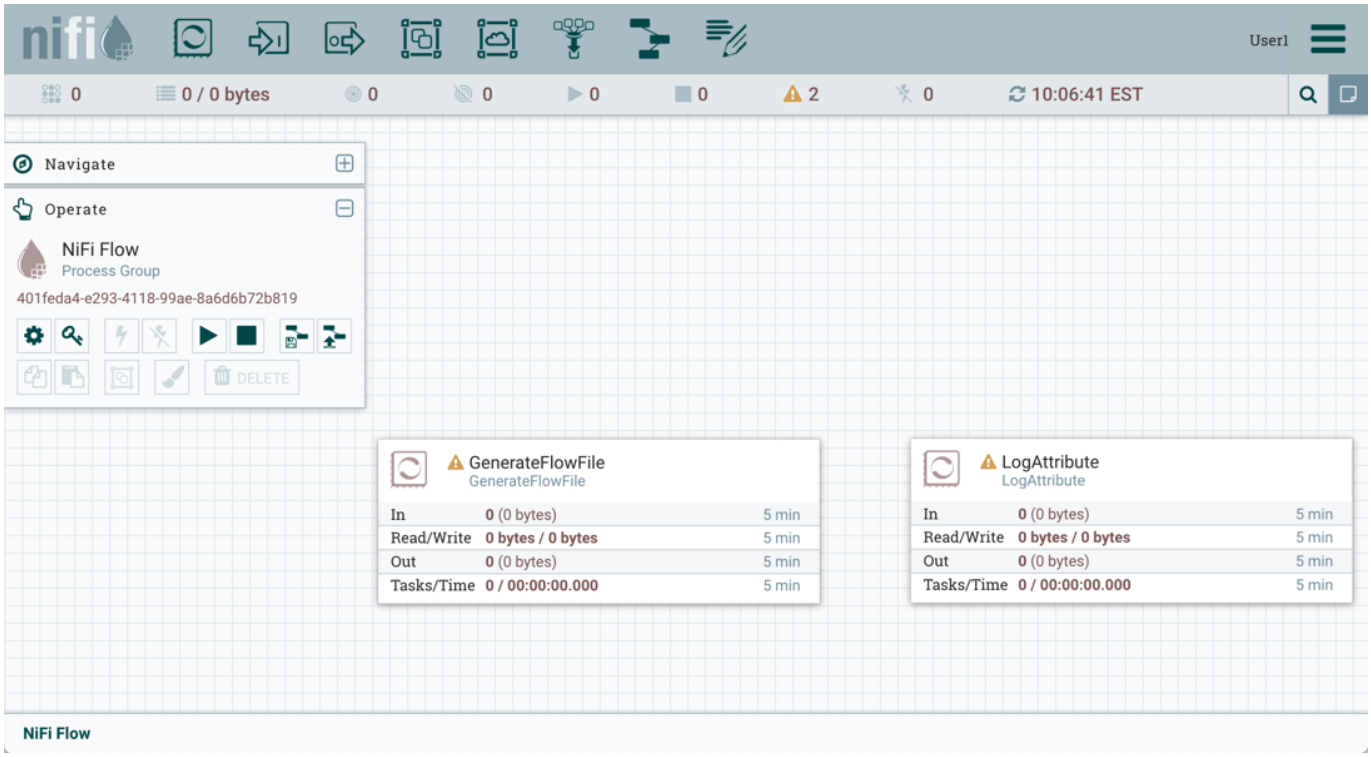


Рис.4.4.: GenerateFlowFile и LogAttribute

User1 может добавлять компоненты в поток данных, а так же перемещать, редактировать и подключать все процессоры. Детали и свойства процессоров и групп процессов root видны для *User1* (Рис.4.5.).

User2 не может добавлять компоненты в поток данных, а так же перемещать, редактировать и подключать компоненты. Детали и свойства процессоров и групп процессов root скрыты от *User2* (Рис.4.6.).

4.3.1 Перемещение процессора

User1 необходимо выполнить следующие шаги для выдачи разрешения пользователю *User2* на перемещение процессора GenerateFlowFile в потоке данных с сохранением привилегий у *User1*:

1. Выбрать процессор GenerateFlowFile.
2. Нажать значок “Access Policies” на панели управления “Operate”. При этом открывается диалоговое окно “Access Policies”.
3. Выбрать “modify the component” в раскрывающемся списке политики. Политика “modify the component”, которая в настоящее время существует на процессоре (дочернем), является унаследованной от группы процессов root (родительской), на которой *User1* имеет привилегии (Рис.4.7.).
4. Нажать ссылку “Override”. При замещении политики необходимо выбрать ее переопределение либо на копию унаследованной политики, либо на пустую политику. Для создания копии следует в диалоговом

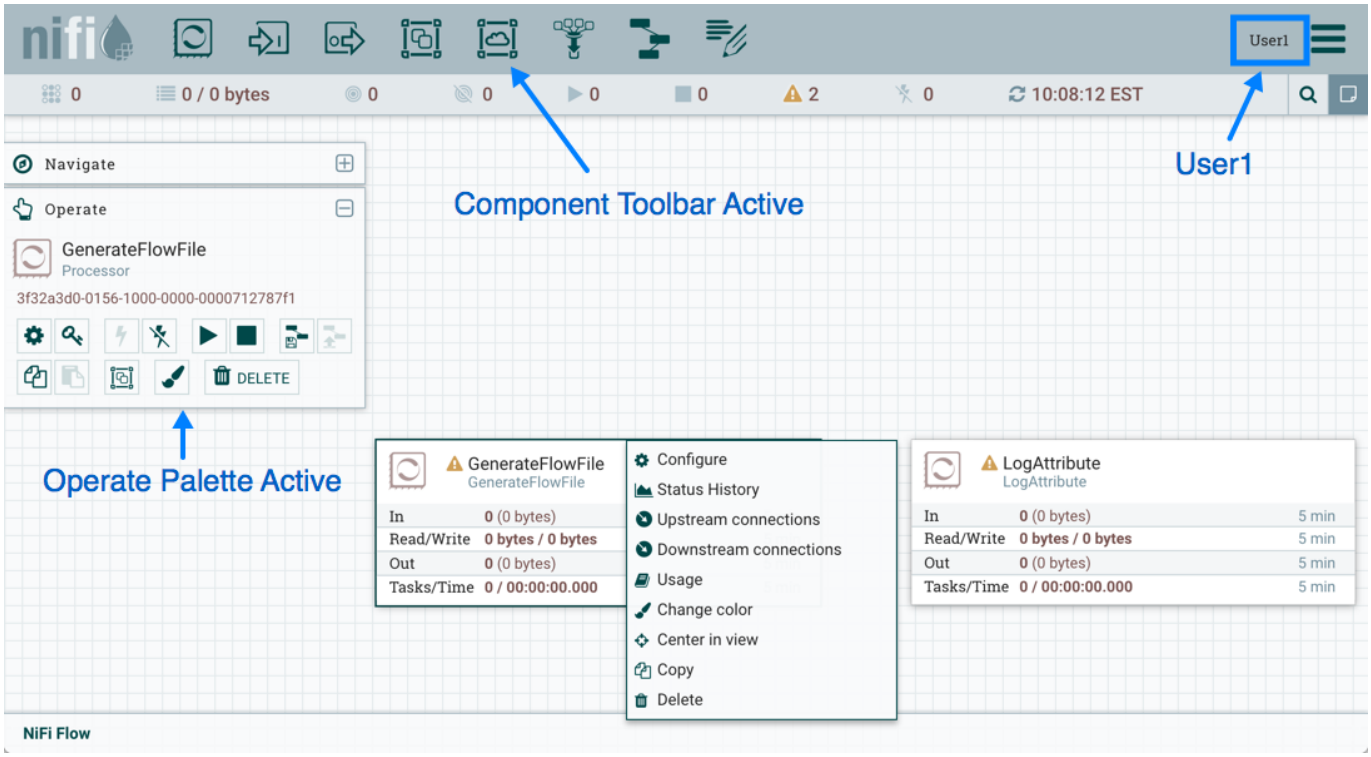


Рис.4.5.: User1 (администратор)

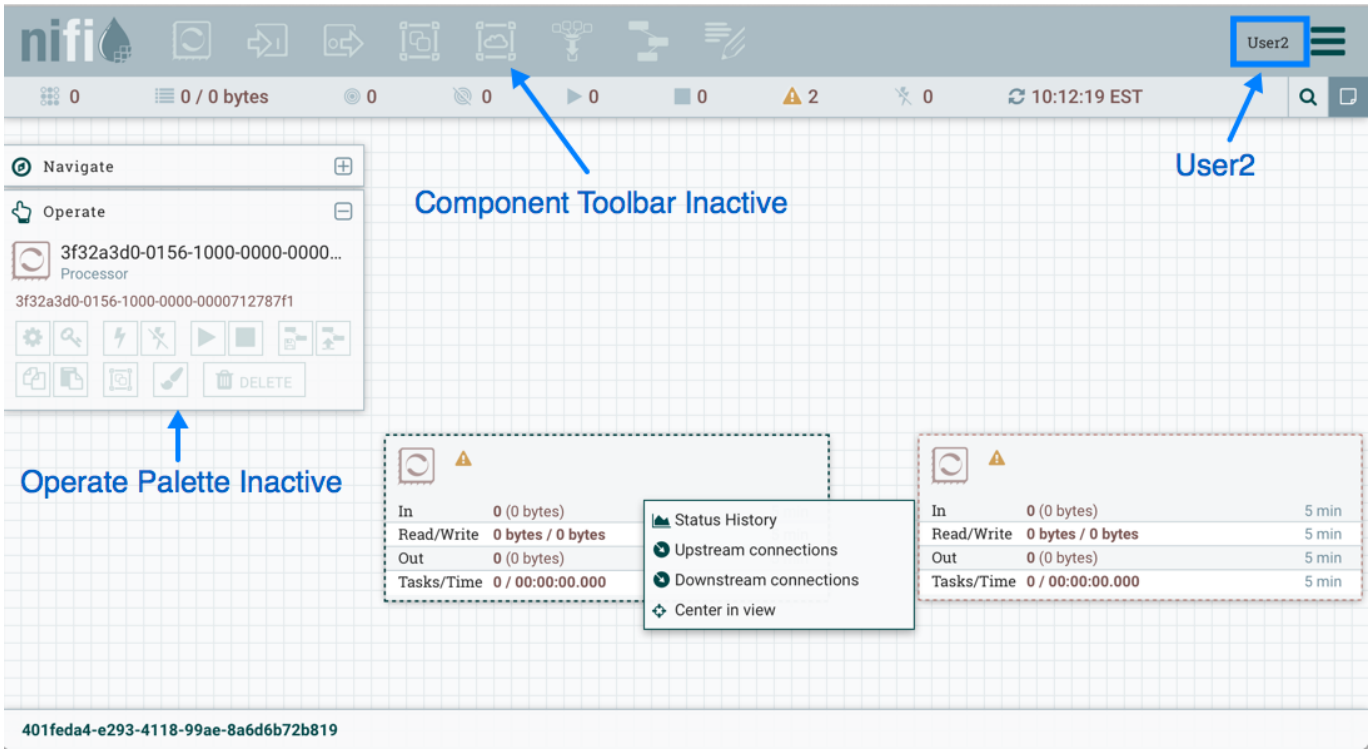


Рис.4.6.: User2 (недавно добавленный пользователь)

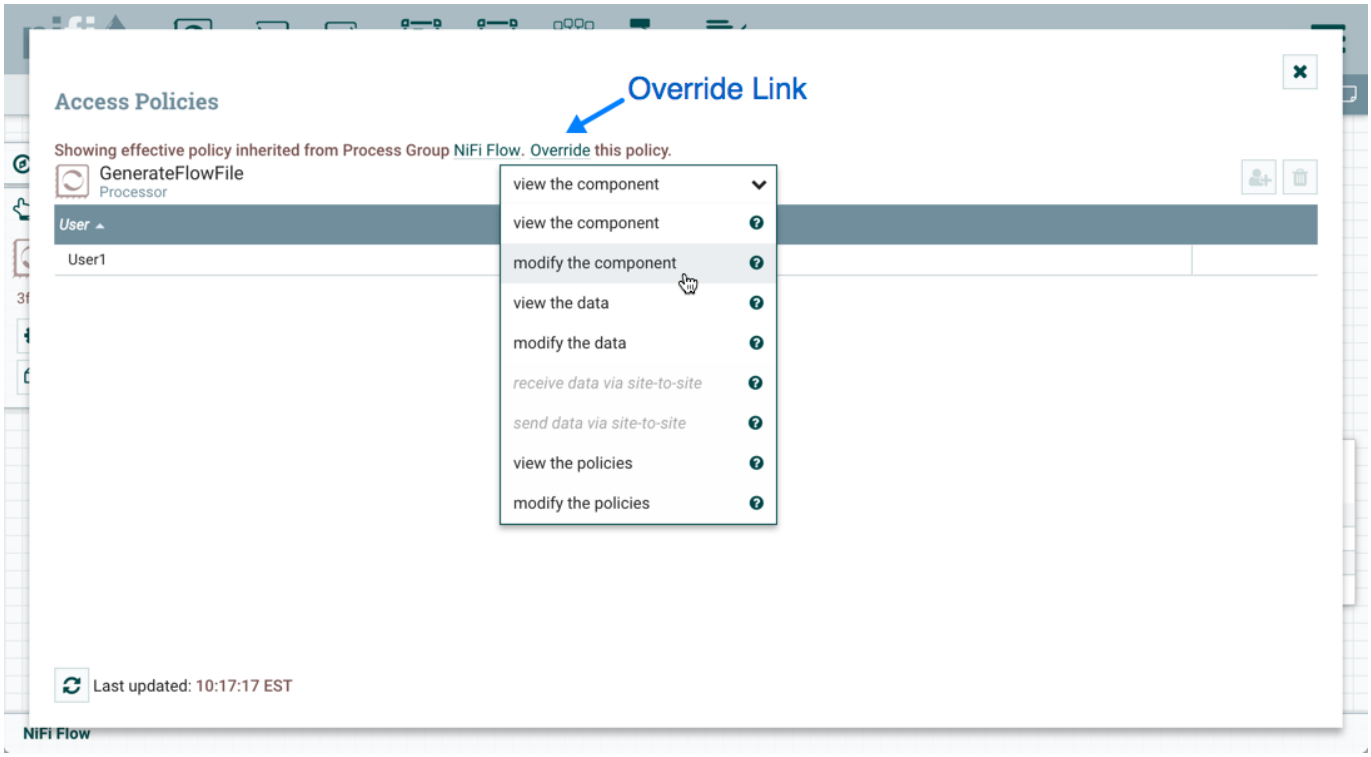


Рис.4.7.: Modify the component

окне “Override Policy” выбрать “Copy” и нажать кнопку “Override” (Рис.4.8.).

5. В созданной политике выбрать значок “Add User”. В поле “User Identity” ввести вручную или найти в списке *User2* и нажать “OK” (Рис.4.9.).

С такими изменениями *User1* сохраняет возможность перемещения обоих процессоров в рабочей области. А *User2* теперь может перемещать процессор GenerateFlowFile (Рис.4.10.).

4.3.2 Изменение процессора

В приведенном примере “Перемещение процессора” *User2* добавлен в политику “modify the component” для процессора GenerateFlowFile. Но без возможности просмотра свойств процессора *User2* не может изменять его конфигурацию – чтобы отредактировать компонент, пользователь должен быть также включен в политику “view the component”.

User1 необходимо выполнить следующие шаги для реализации возможности изменения конфигурации процессора пользователю *User2*:

1. Выбрать процессор GenerateFlowFile.
2. Нажать значок “Access Policies” на панели управления “Operate”. При этом открывается диалоговое окно “Access Policies”.
3. Выбрать “view the component” в раскрывающемся списке политики. Политика “view the component”, которая в настоящее время существует на процессоре (дочернем), является унаследованной от группы процессов root (родительской), на которой *User1* имеет привилегии (Рис.4.11.).
4. Нажать ссылку “Override” и в открывшемся диалоговом окне, сохранив политику копирования по умолчанию, нажать кнопку “Override”.

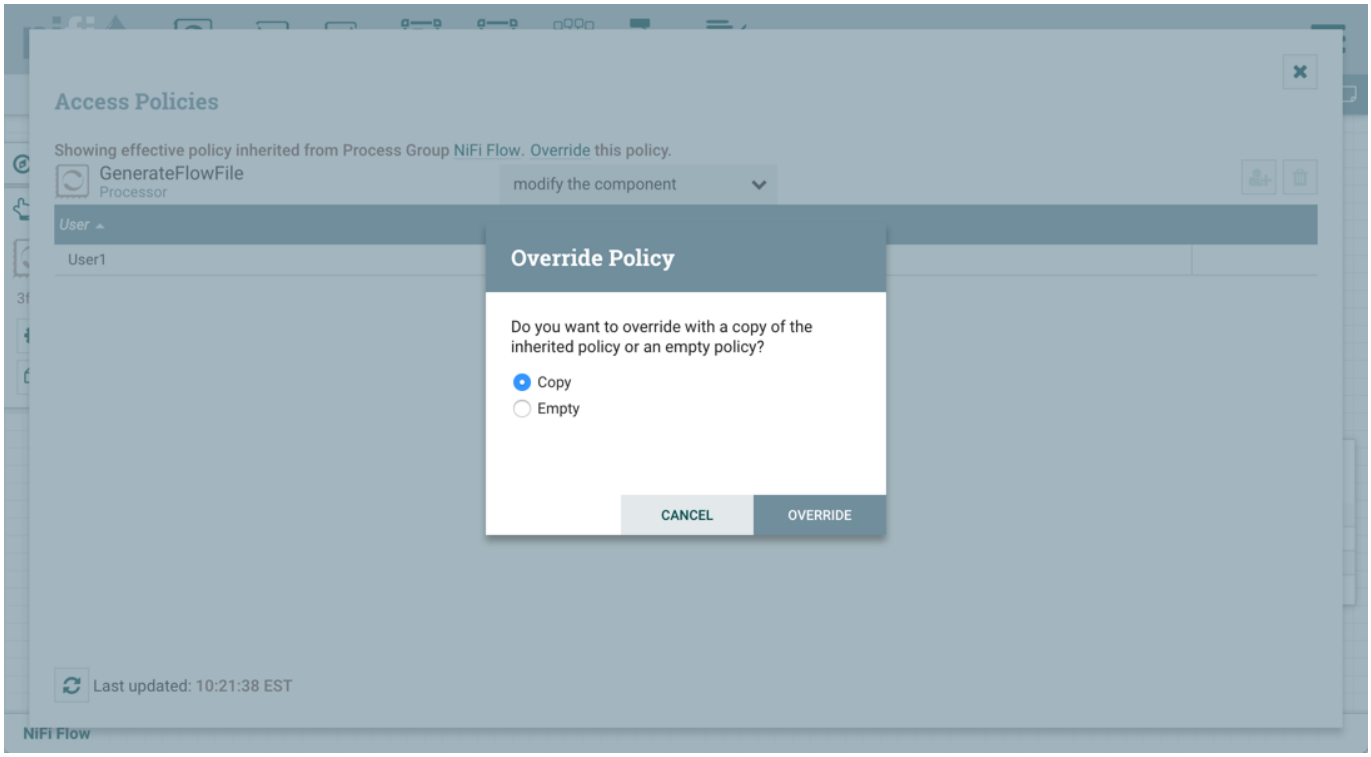


Рис.4.8.: Override Policy

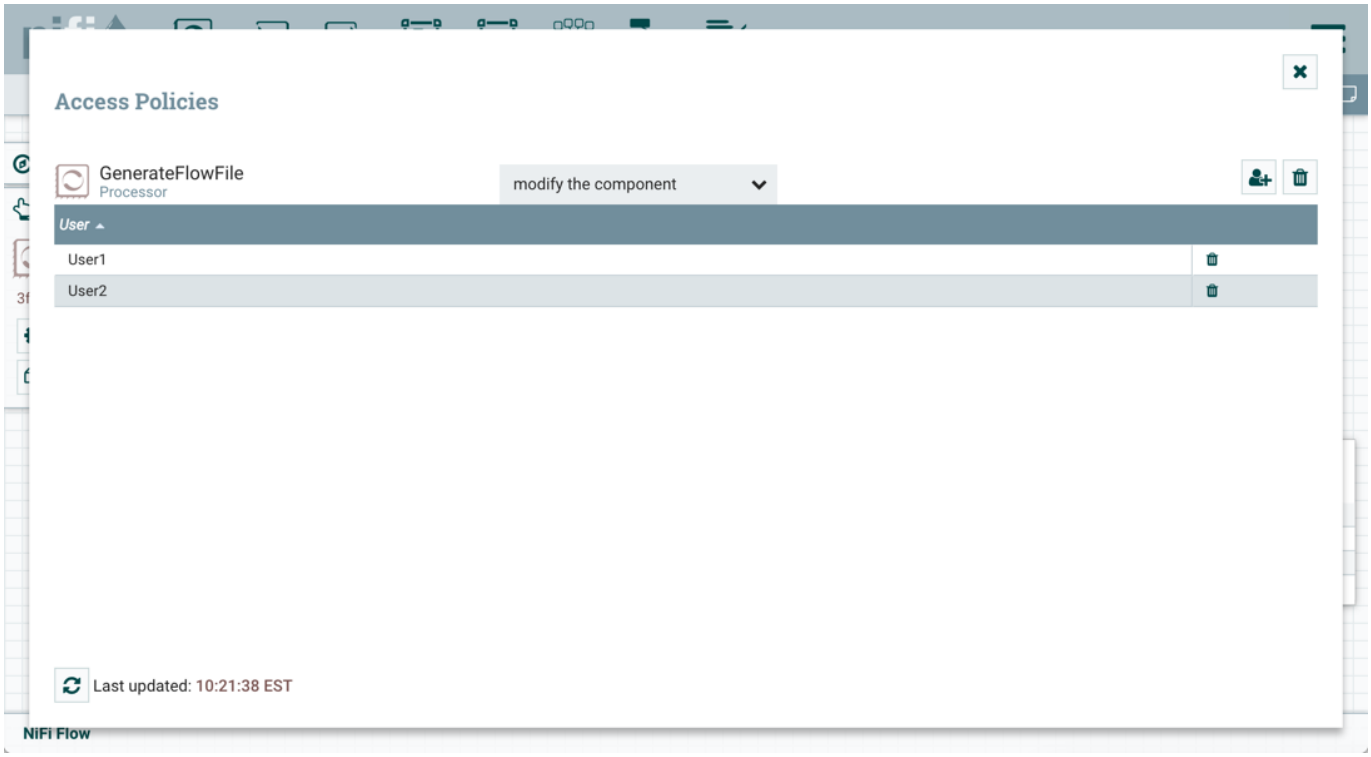


Рис.4.9.: Добавление User2 в политику

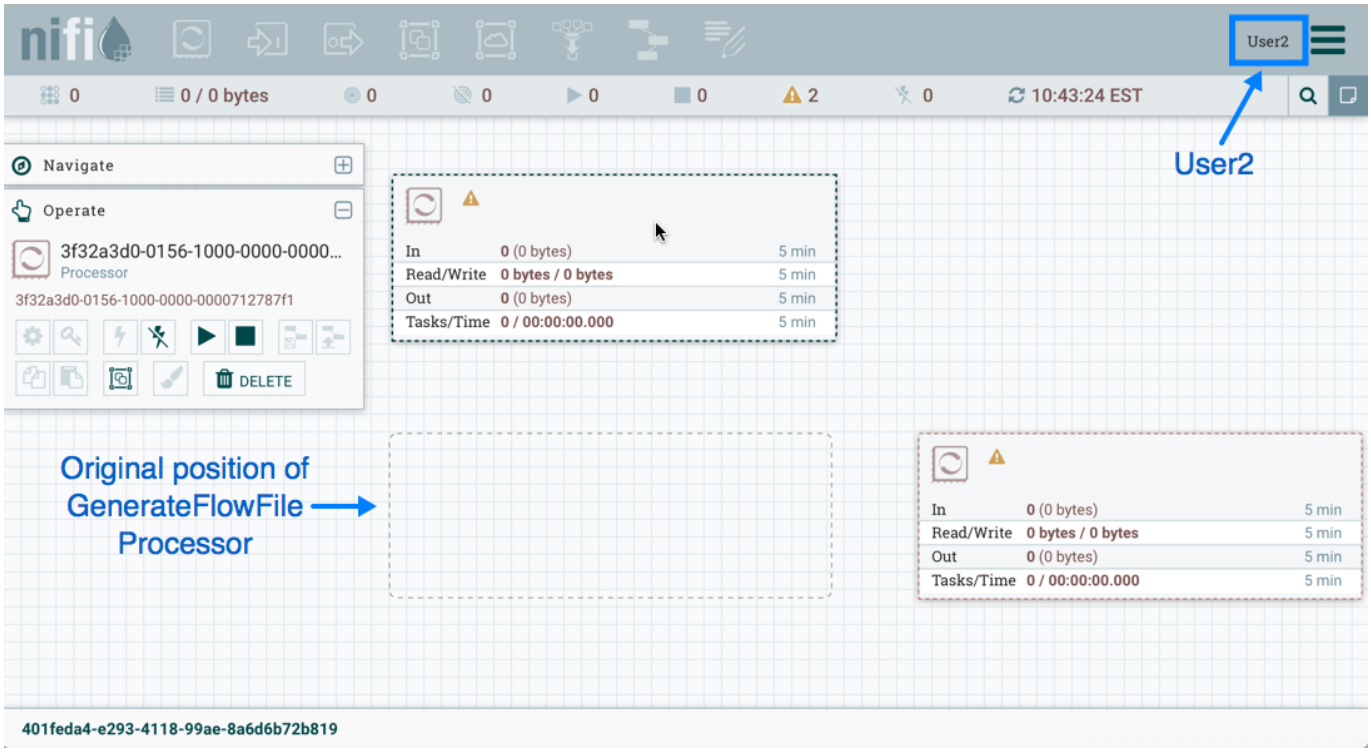


Рис.4.10.: Результат действий

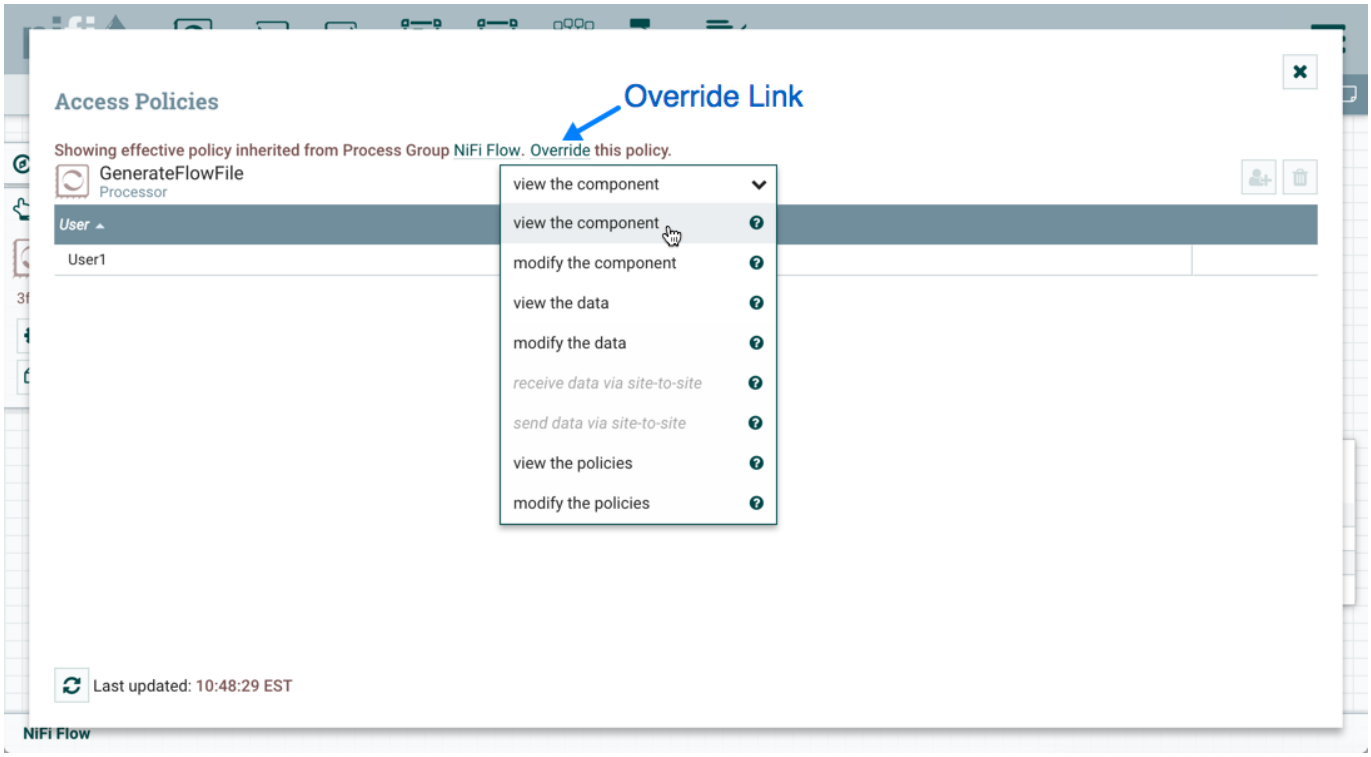


Рис.4.11.: View the component

5. В созданной политике выбрать значок “Add User”. В поле “User Identity” ввести вручную или найти в списке *User2* и нажать “OK” (Рис.4.12.).

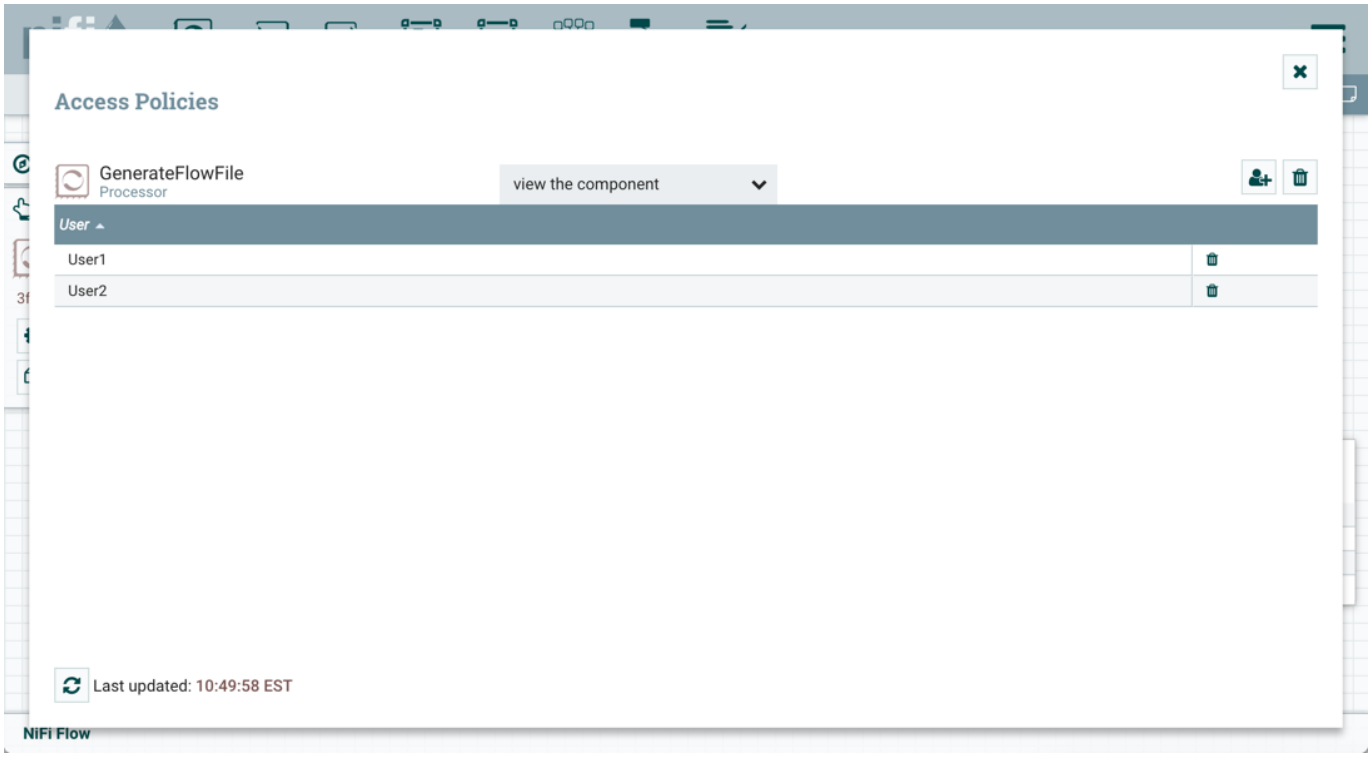


Рис.4.12.: Добавление User2 в политику

С такими изменениями *User1* сохраняет возможность просмотра и редактирования процессоров в рабочей области. А *User2* теперь может просматривать и редактировать процессор GenerateFlowFile (Рис.4.13.).

4.3.3 Создание подключения

При настройке политик так, как описано в предыдущих двух примерах, *User1* может подключить GenerateFlowFile к LogAttribute (Рис.4.14.).

При этом *User2* не имеет права доступа на установку соединения процессоров (Рис.4.15.).

Это объясняется тем, что:

- *User2* не имеет доступа к изменениям в группе процессов;
- Несмотря на то, что *User2* имеет право на просмотр и изменение исходного компонента (GenerateFlowFile), *User2* не имеет политики доступа к целевому компоненту (LogAttribute).

User1 необходимо выполнить следующие шаги для реализации возможности подключения GenerateFlowFile к LogAttribute пользователю *User2*:

1. Выбрать группу процессов root, при этом панель управления “Operate” обновляется с подробными сведениями.
2. Выбрать значок “Access Policies” на панели управления “Operate”. При этом открывается диалоговое окно “Access Policies”.
3. В диалоговом окне в раскрывающемся списке политики выбрать “modify the component” (Рис.4.16.).

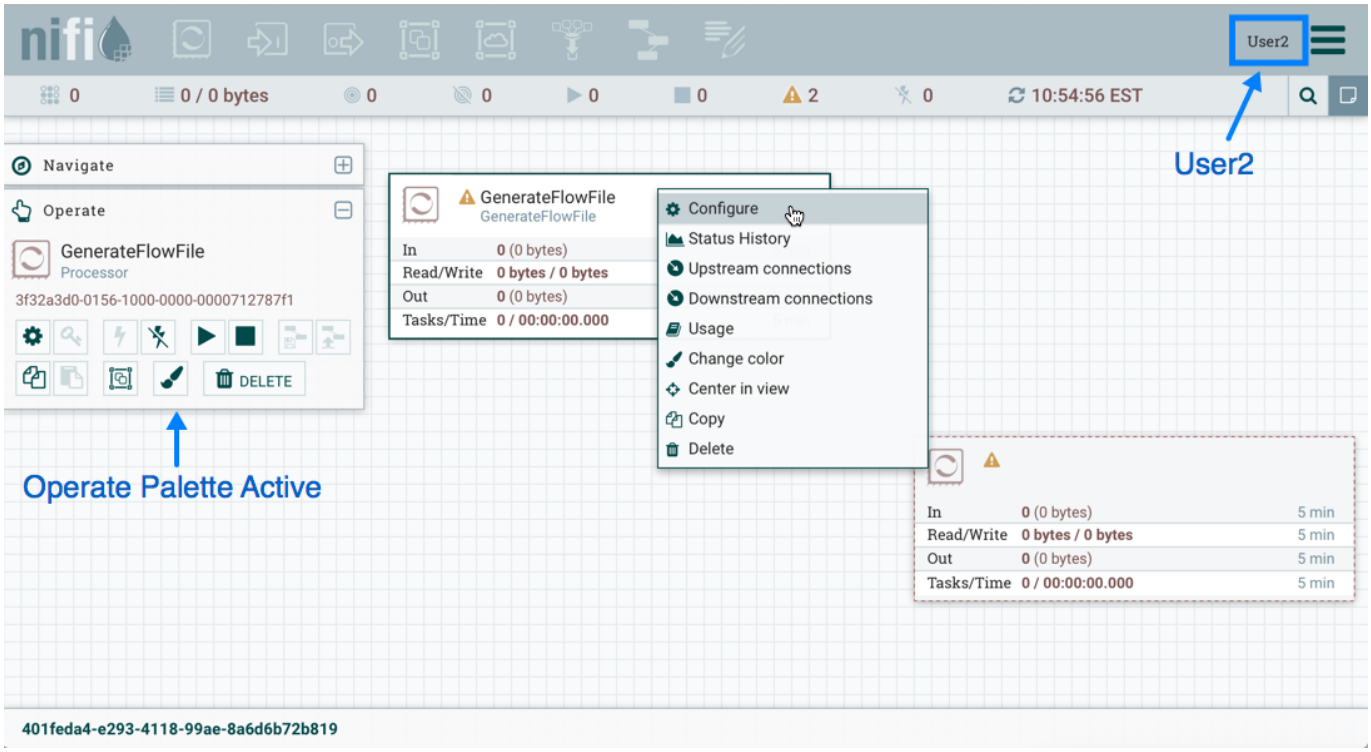


Рис.4.13.: Результат действий

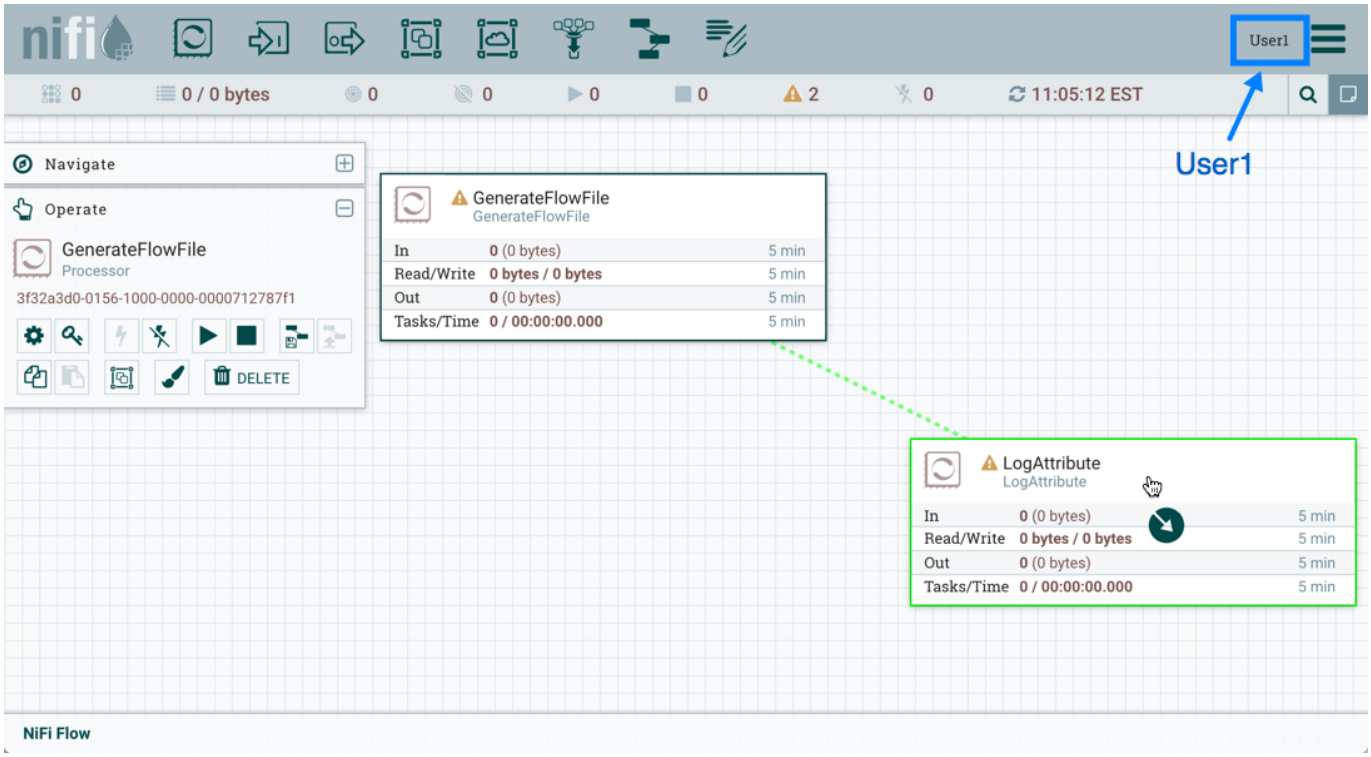


Рис.4.14.: User1 – подключение процессоров

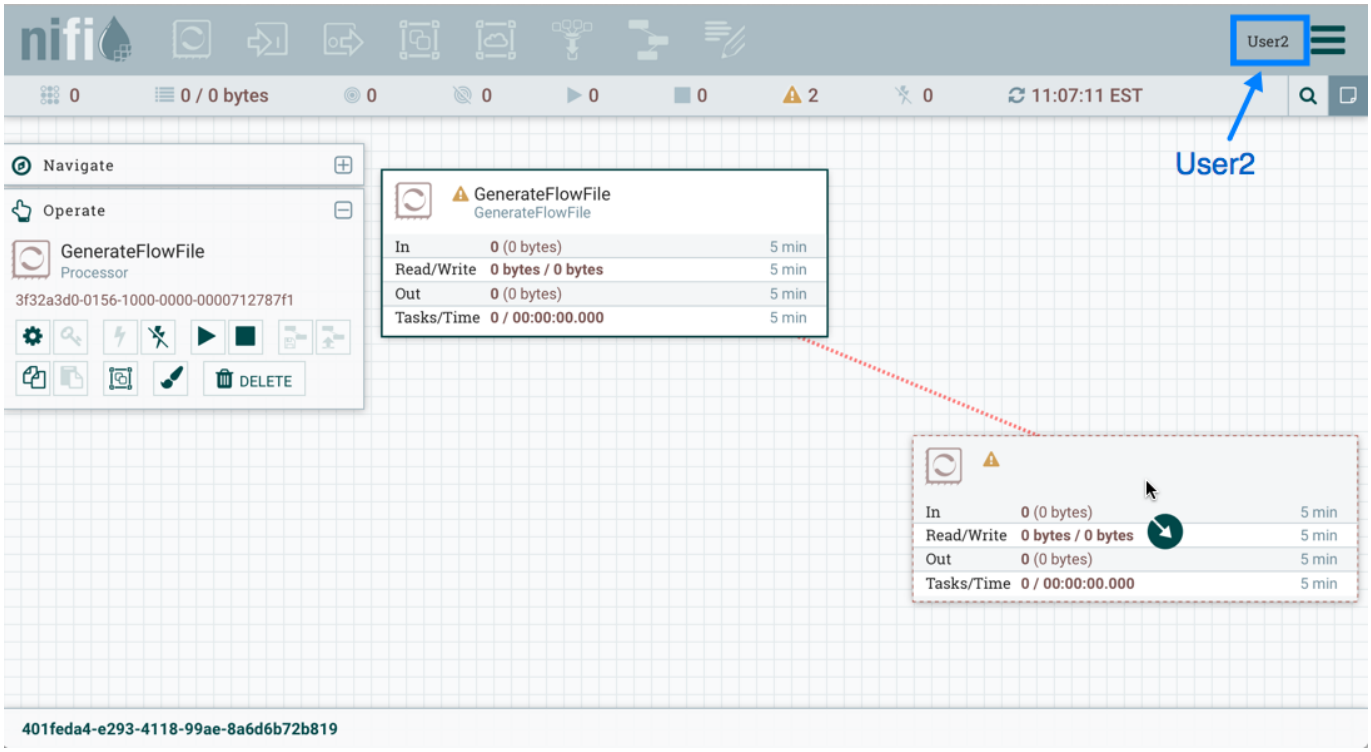


Рис.4.15.: User2 – невозможность подключения процессоров

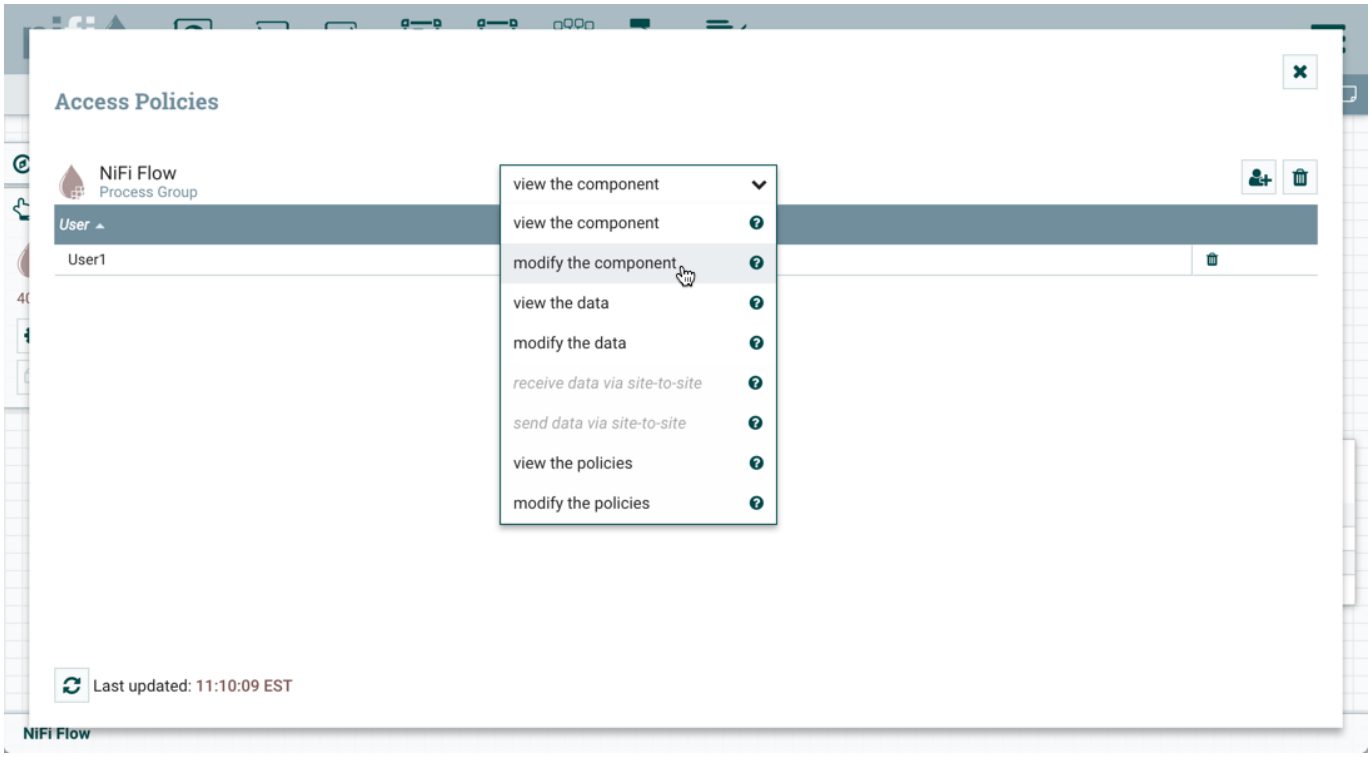


Рис.4.16.: Modify the component

4. Выбрать значок “Add User”. В поле “User Identity” ввести вручную или найти в списке *User2* и нажать “OK” (Рис.4.17.).

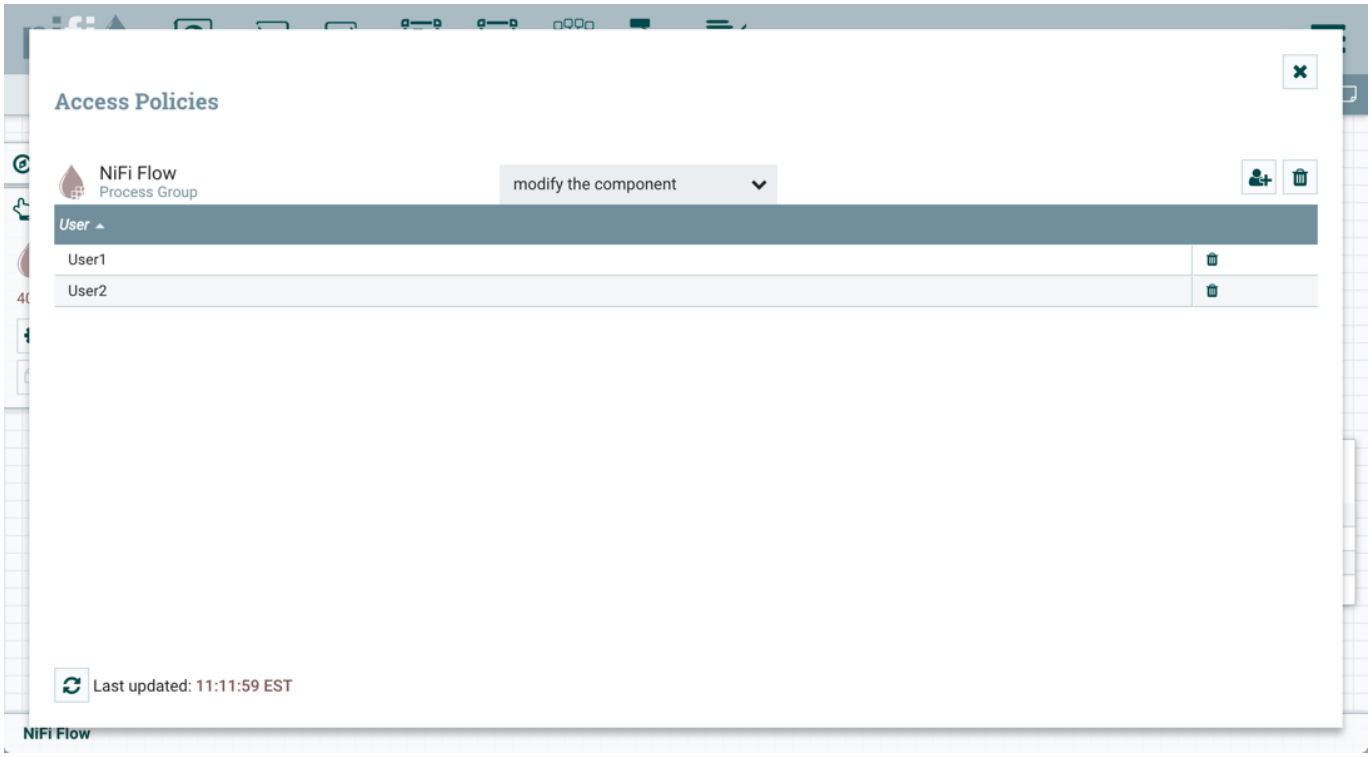


Рис.4.17.: Добавление User2 в политику группы

Добавляя *User2* в политику “modify the component” группы процессов, *User2* так же добавляется к политике “modify the component” в процессоре LogAttribute путем наследования. Чтобы проверить это, необходимо в рабочей области выделить процессор LogAttribute и выбрать значок “Access Policies” на панели управления “Operate”. При этом открывается диалоговое окно политик доступа процессора LogAttribute с наличием пользователя *User2* в политике “modify the component” (Рис.4.18.).

С такими изменениями *User2* теперь может подключать процессор GenerateFlowFile к процессору LogAttribute (Рис.4.19., Рис.4.20.).

4.3.4 Изменение соединения

В следующем сценарии *User1* и *User2* добавляют процессор ReplaceText в группу процессов root (Рис.4.21.).

User1 может выбрать и изменить существующее соединение между GenerateFlowFile и LogAttribute, чтобы подключить GenerateFlowFile к ReplaceText (Рис.4.22.).

При этом *User2* не имеет возможности выполнить такое действие (Рис.4.23.).

User1 необходимо выполнить следующие шаги для реализации возможности подключения GenerateFlowFile к ReplaceText пользователю *User2*:

1. Выбрать группу процессов root, при этом панель управления “Operate” обновляется с подробными сведениями.
2. Выбрать значок “Access Policies” на панели управления “Operate”. При этом открывается диалоговое окно “Access Policies”.

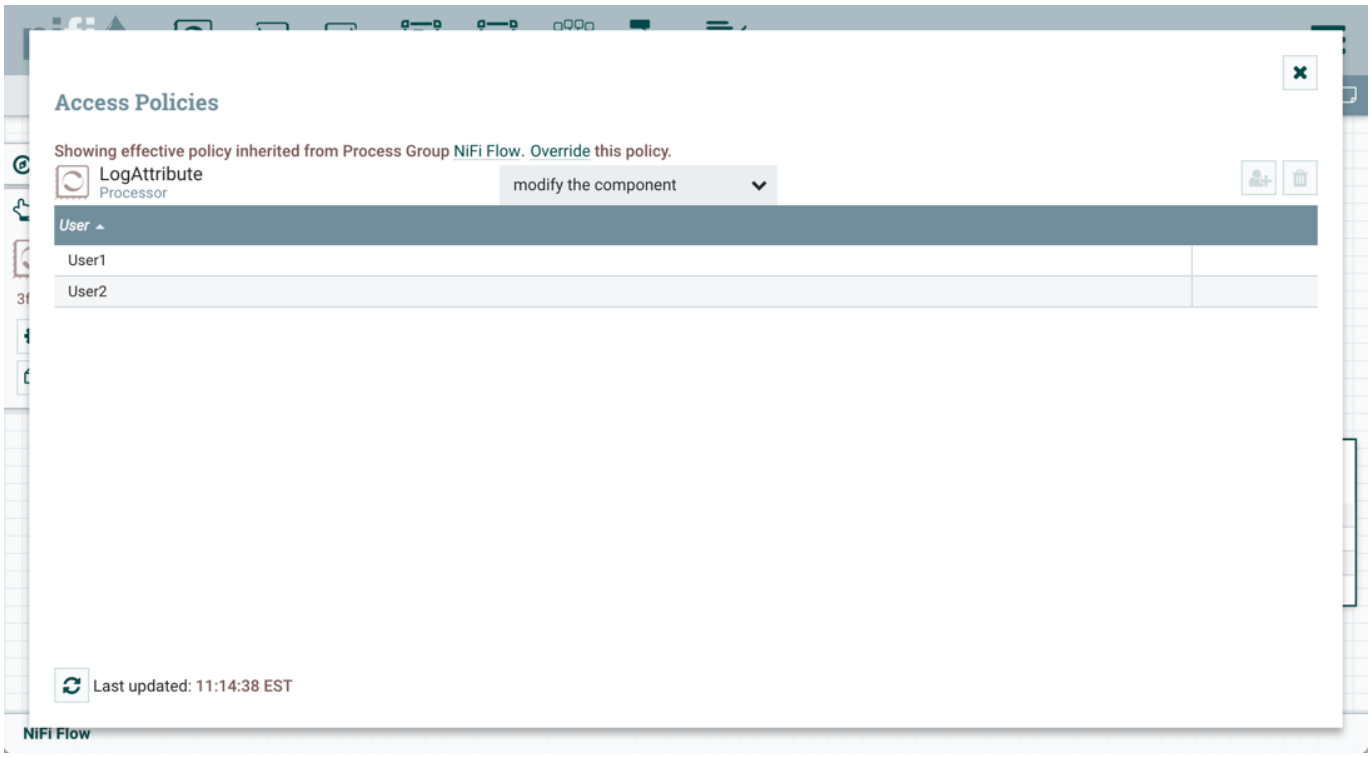


Рис.4.18.: Проверка наличия политики User2

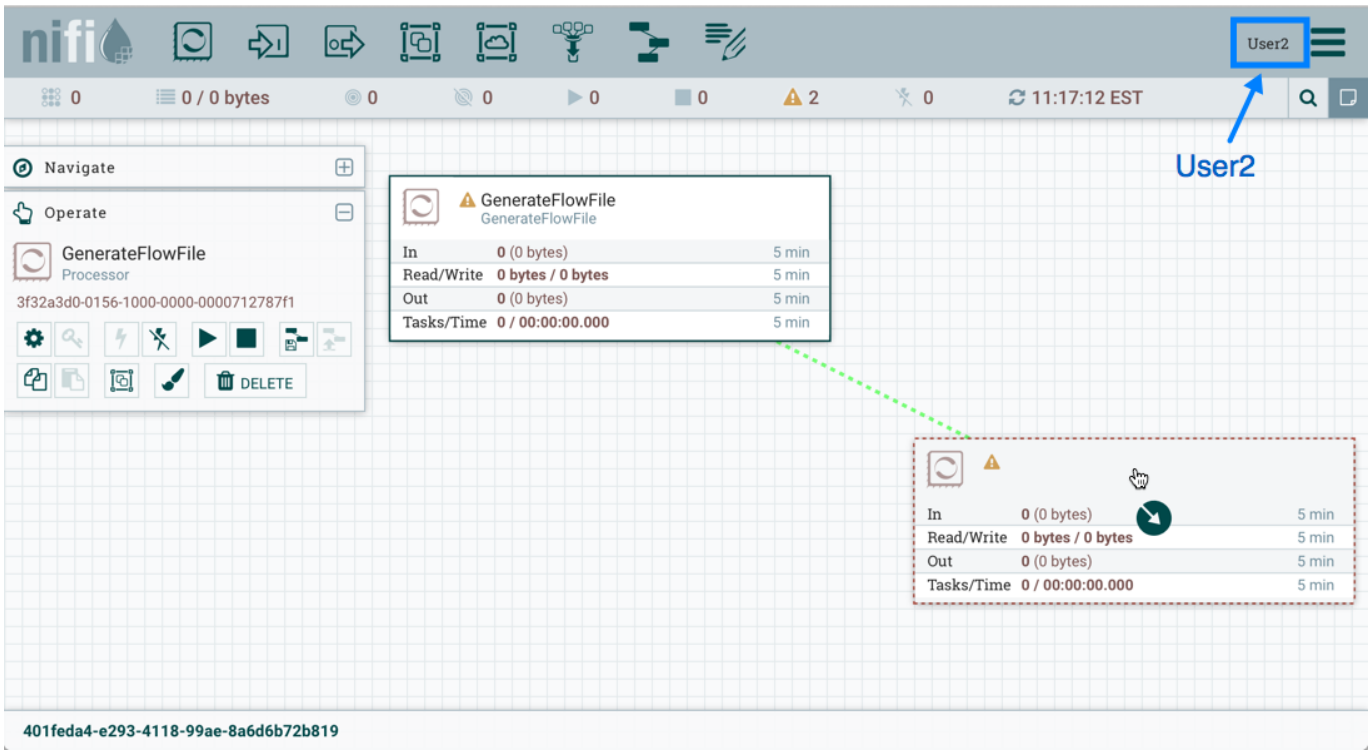


Рис.4.19.: User2 – подключение процессоров

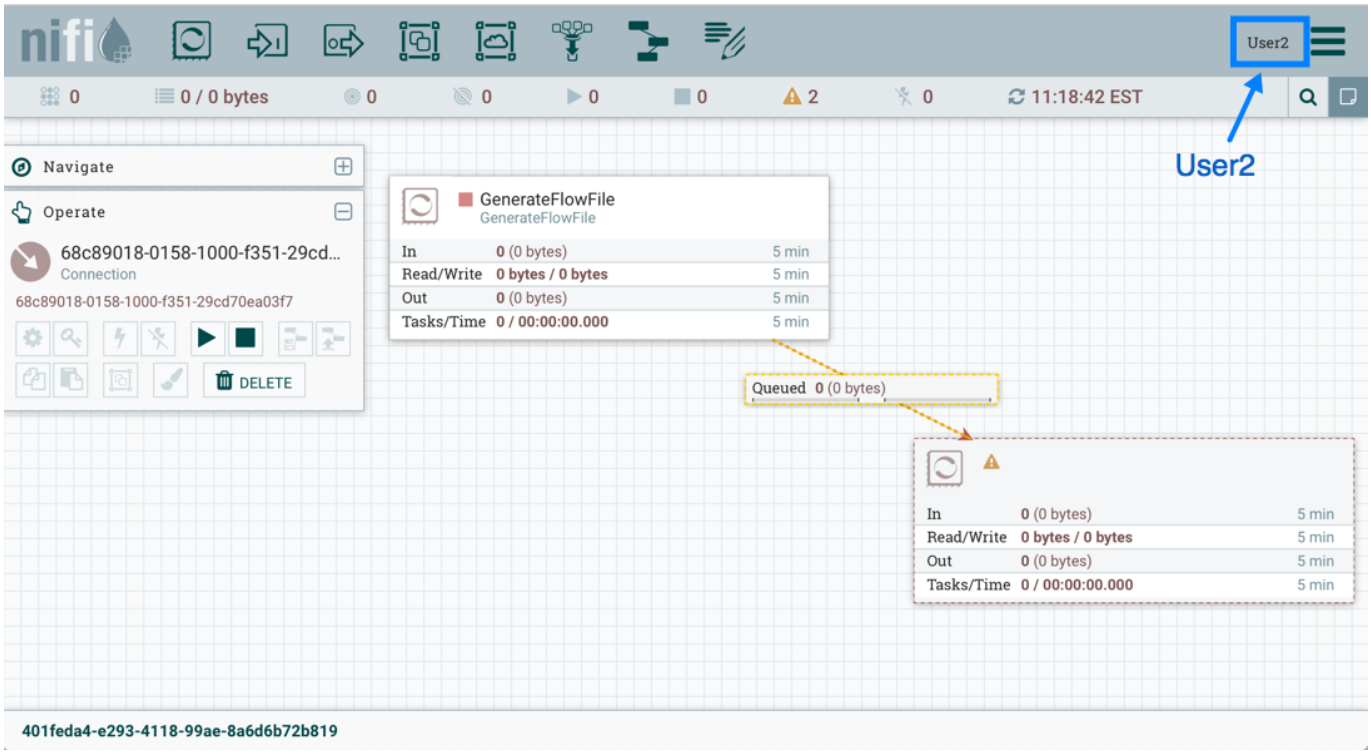


Рис.4.20.: User2 – подключение процессоров

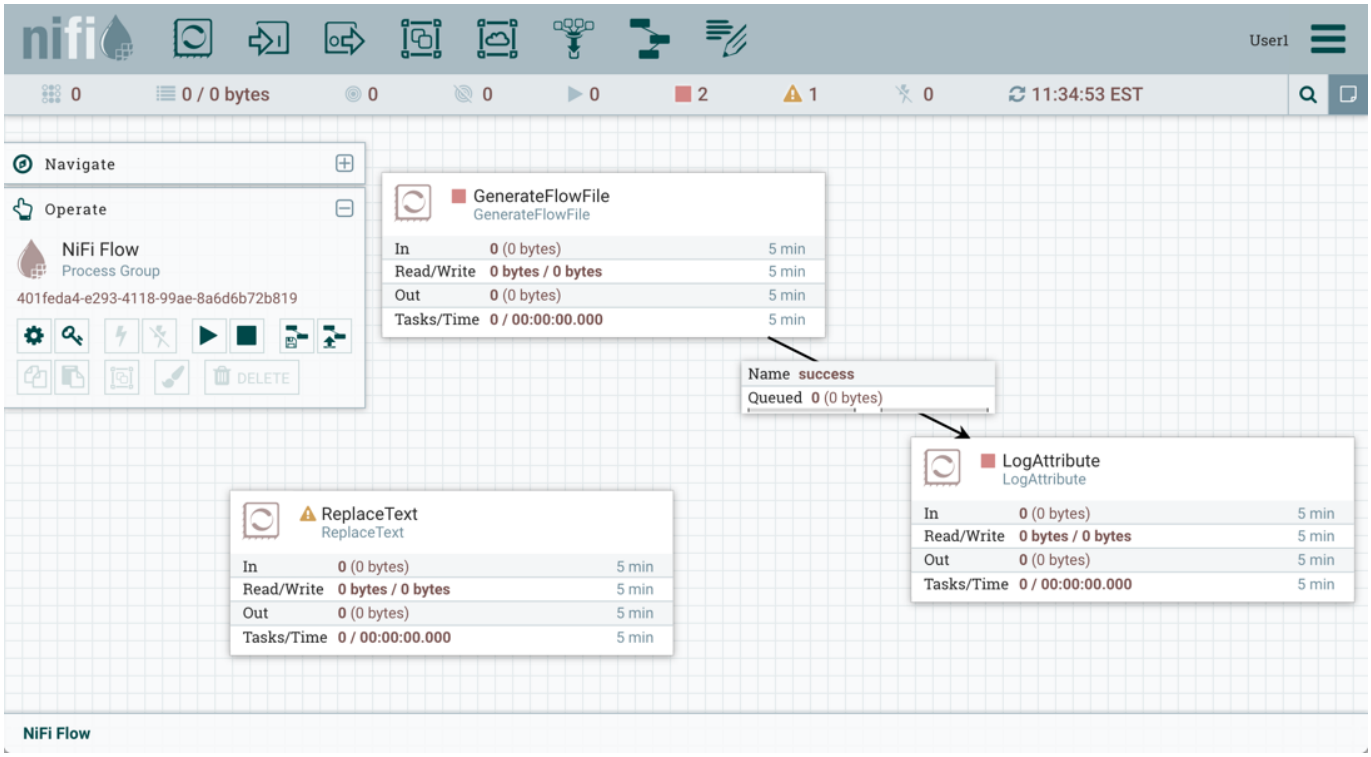


Рис.4.21.: Добавление процессора ReplaceText

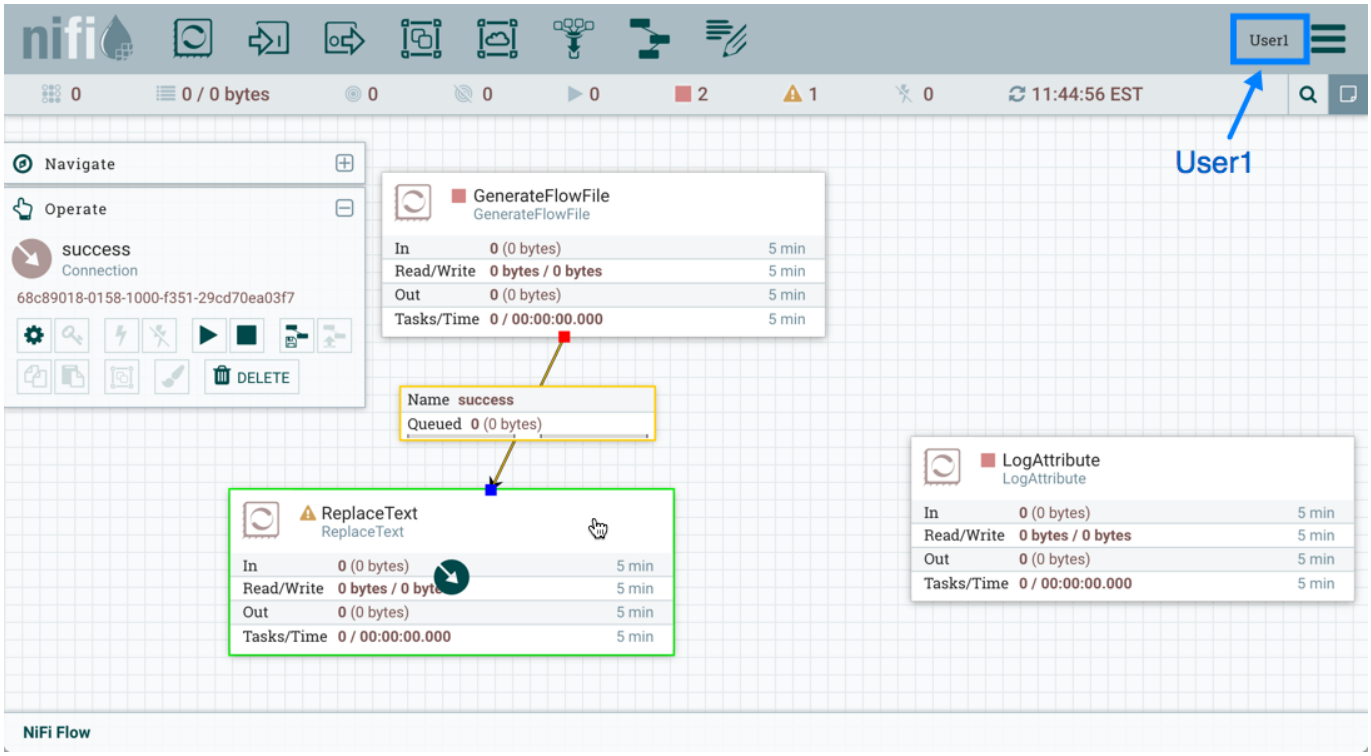


Рис.4.22.: User1 – изменение соединения

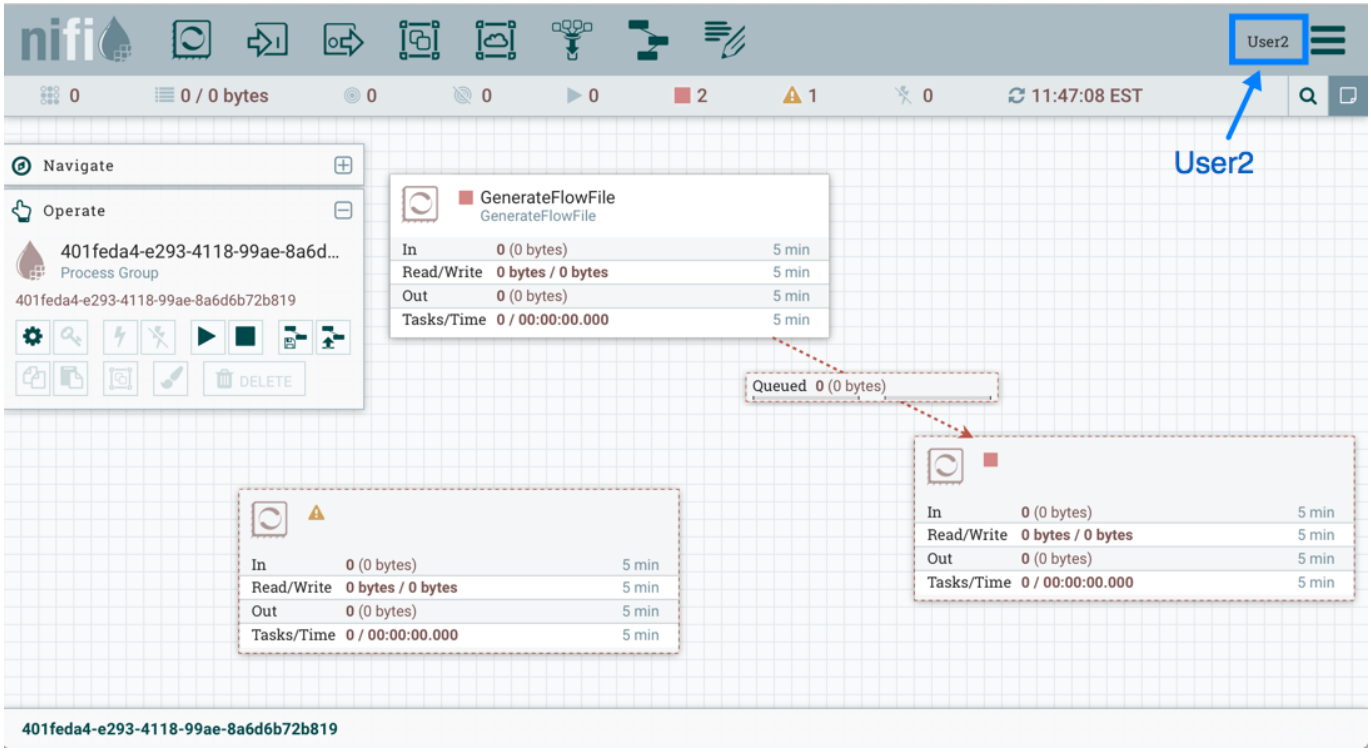


Рис.4.23.: User2 недоступно изменение соединения

3. В диалоговом окне в раскрывающемся списке политики выбрать “view the component” (Рис.4.24.).

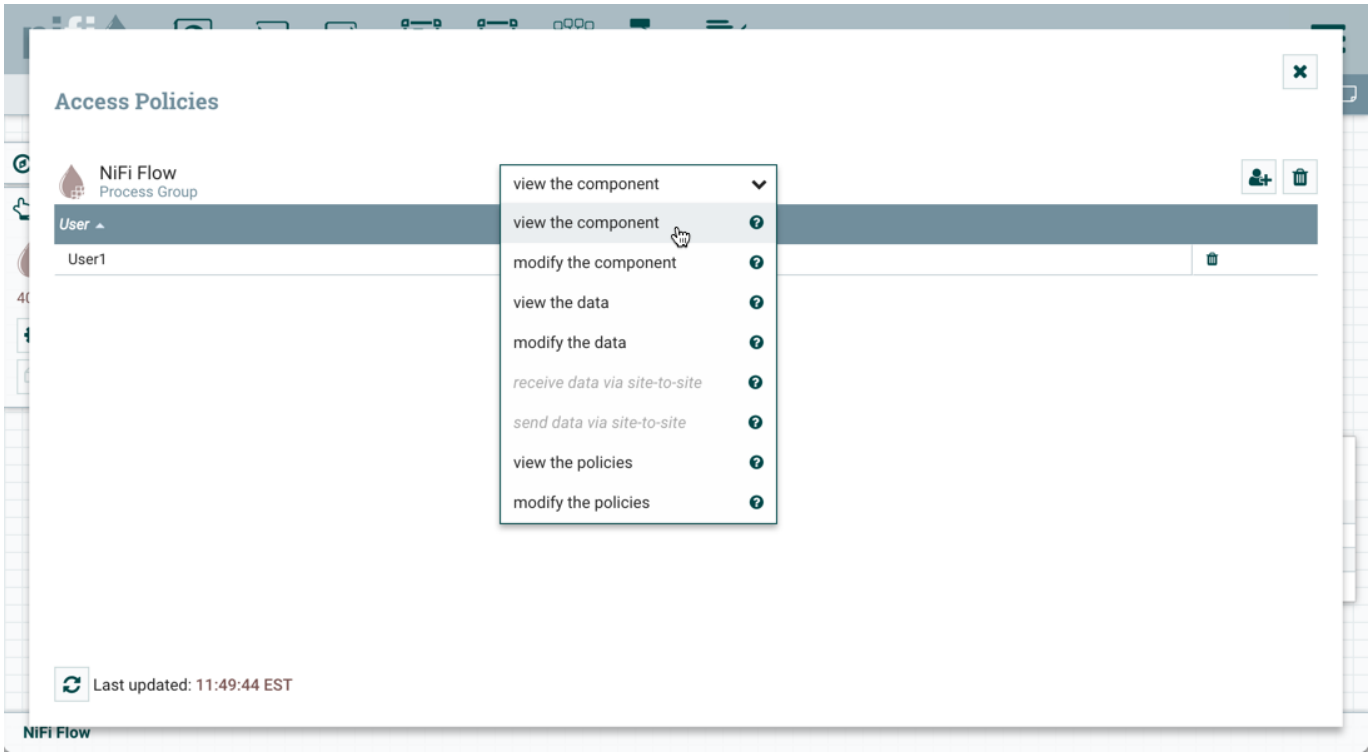


Рис.4.24.: View the component

4. Выбрать значок “Add User”. В поле “User Identity” ввести вручную или найти в списке *User2* и нажать “OK” (Рис.4.25.).

Будучи добавленным к политикам просмотра и изменения для группы процессов *User2* теперь может подключать процессор GenerateFlowFile к процессору ReplaceText (Рис.4.26.).

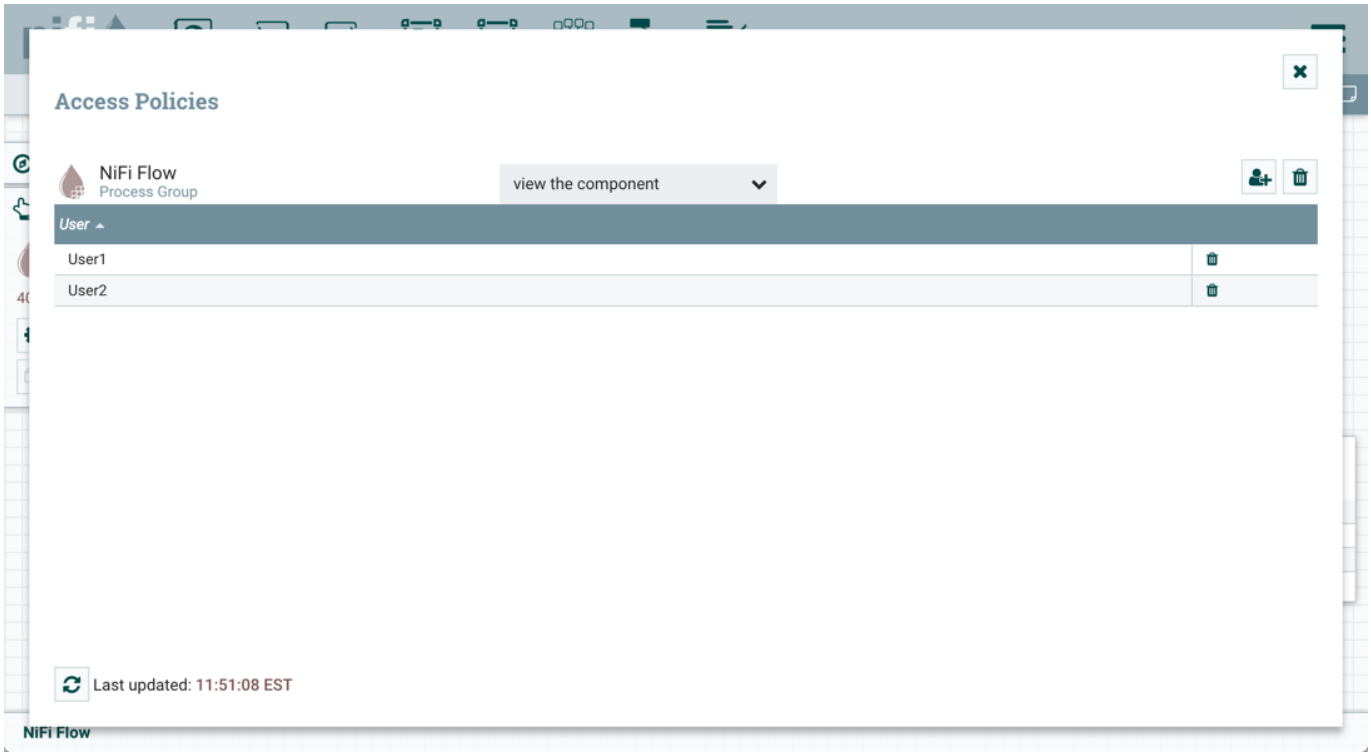


Рис.4.25.: Добавление User2 в политику группы

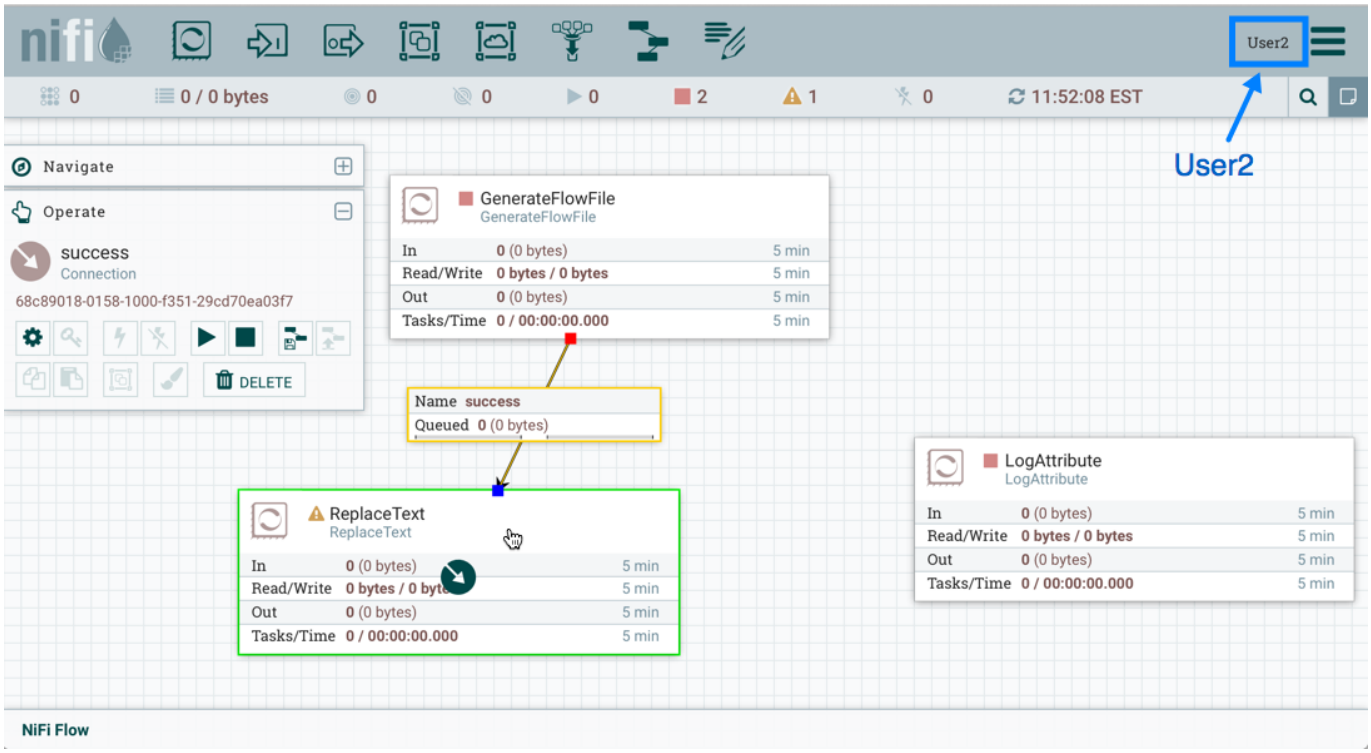


Рис.4.26.: User2 – изменение соединения

Глава 5

Kerberos Service

NiFi может быть настроен для использования Kerberos SPNEGO (или “Kerberos Service”) для аутентификации. В таком случае пользователи попадают в конечную точку REST */access/kerberos*, и сервер отвечает кодом состояния *401* с заголовком задачи *WWW-Authenticate: Negotiate*. Далее сервер связывается с браузером для использования GSS-API и загрузки тикета пользователя Kerberos с указанием его в качестве заголовка *Base64* в последующем запросе. Он принимает форму *Authorization: Negotiate YII...*, и **NiFi** пробует подтвердить этот билет с помощью KDC. В случае успеха принципал пользователя возвращается как подлинный, и поток следует аутентификации login/credential, в результате которой в ответ выдается JWT для предотвращения ненужных издержек аутентификации Kerberos при каждом последующем запросе. В случае если билет пользователя не подтверждается, то он возвращается с соответствующим кодом ошибки. После чего пользователь может предоставить свои учетные данные для формы регистрации Kerberos при условии настроенного *KerberosLoginIdentityProvider*. Дополнительную информация приведена в главе Kerberos.

NiFi отвечает на запросы Kerberos SPNEGO только по соединению HTTPS, поскольку незащищенные запросы никогда не проходят проверку подлинности.

Для включения аутентификации службы Kerberos в *nifi.properties* должны быть настроены следующие свойства:

Таблица5.1.: Свойства для включения аутентификации Kerberos Service

Свойство	Значаение	Описание
Service Principal	true	Принципал сервиса, используемый NiFi для связи с KDC
Keytab Location	true	Путь к файлу keytab, содержащему принципал сервиса

Примечания:

- Kerberos чувствителен к регистру во многих местах, и сообщения об ошибках (или их отсутствие) могут быть недостаточно понятны. Рекомендуется проверить у службы чувствительность к регистру в конфигурационных файлах. Конвенция – *HTTP/fully.qualified.domain@REALM*;
- Браузеры имеют разные уровни ограничений при работе с SPNEGO. Некоторые из них предоставляют локальный тикет Kerberos в любой запрашиваемый домен, в то время как другие выдают пустой список доверенных доменов. Справочная информация для общих браузеров приведена по ссылке;
- Некоторые браузеры (например, устаревший IE) не поддерживают последние алгоритмы шифрования, такие как AES, и ограничены устаревшими алгоритмами (например, DES). Это следует учитывать при создании keytabs;

- Должен быть настроен KDC, определен принципал сервиса для NiFi и экспортирован keytab. Подробные инструкции по настройке и администрированию Kerberos Service выходят за рамки данного документа (MIT Kerberos Admin Guide), но далее приведен пример.

  Пример добавления принципала для сервера на *nifi.nifi.apache.org* и экспорта ключа из KDC:

```
root@kdc:/etc/krb5kdc# kadmin.local
Authenticating as principal admin/admin@NIFI.APACHE.ORG with password.
kadmin.local:  listprincs
K/M@NIFI.APACHE.ORG
admin/admin@NIFI.APACHE.ORG
...
kadmin.local:  addprinc -randkey HTTP/nifi.nifi.apache.org
WARNING: no policy specified for HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG; defaulting to no␣
↪policy
Principal "HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG" created.
Principal "HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG" created.
kadmin.local:  ktadd -k /http-nifi.keytab HTTP/nifi.nifi.apache.org
Entry for principal HTTP/nifi.nifi.apache.org with kvno 2, encryption type des3-cbc-sha1 added to␣
↪keytab WRFILE:/http-nifi.keytab.
Entry for principal HTTP/nifi.nifi.apache.org with kvno 2, encryption type des-cbc-crc added to␣
↪keytab WRFILE:/http-nifi.keytab.
kadmin.local:  listprincs
HTTP/nifi.nifi.apache.org@NIFI.APACHE.ORG
K/M@NIFI.APACHE.ORG
admin/admin@NIFI.APACHE.ORG
...
kadmin.local: q
root@kdc:~# ll /http*
-rw------- 1 root root 162 Mar 14 21:43 /http-nifi.keytab
root@kdc:~#
```