Security 3

NiFi Authorization

Date of Publish: 2018-11-15

https://docs.hortonworks.com/

Contents

NiFi Authorization 3

Authorizer Configuration 3

Authorizers.xml Setup 3

Initial Admin Identity (New NiFi Instance) 3

Legacy Authorized Users (NiFi Instance Upgrade) 4

Cluster Node Identities 5

Configuring Users & Access Policies 6

Creating Users and Groups 6

Access Policies 9

Access Policy Configuration Examples 11

NiFi Authorization

After you have configured NiFi to run securely and with an authentication mechanism, you must configure who has access to the system, and the level of their access. You can do this using multi-tenant authorization.

Multi-tenant authorization enables multiple groups of users (tenants) to command, control, and observe different parts of the dataflow, with varying levels of authorization. When an authenticated user attempts to view or modify a NiFi resource, the system checks whether the user has privileges to perform that action. These privileges are defined by policies that you can apply system-wide or to individual components.

Authorizer Configuration

An authorizer grants users the privileges to manage users and policies by creating preliminary authorizations at startup.

Authorizers are configured using two properties in the nifi.properties file:

- The nifi.authorizer.configuration.file property specifies the configuration file where authorizers are defined. By default, the authorizers.xml file located in the root installation conf directory is selected.
- The nifi.security.user.authorizer property indicates which of the configured authorizers in the authorizers.xml file to use.

Authorizers.xml Setup

The authorizers.xml file is used to define and configure available authorizers. The default authorizer is the FileAuthorizer, however, you can develop additional authorizers as extensions. The FileAuthorizer has the following properties:

- Authorizations File - The file where the FileAuthorizer stores policies. By default, the authorizations.xml in the conf directory is chosen.
- Users File - The file where the FileAuthorizer stores users and groups. By default, the users.xml in the conf directory is chosen.
- Initial Admin Identity - The identity of an initial admin user that is granted access to the UI and given the ability to create additional users, groups, and policies. This property is only used when there are no other users, groups, and policies defined.
- Legacy Authorized Users File - The full path to an existing authorized-users.xml that is automatically converted to the multi-tenant authorization model. This property is only used when there are no other users, groups, and policies defined.
- Node Identity - The identity of a NiFi cluster node. When clustered, a property for each node should be defined, so that every node knows about every other node. If not clustered, these properties can be ignored.

Initial Admin Identity (New NiFi Instance)

If you are setting up a secured NiFi instance for the first time, you must manually designate an "Initial Admin Identity" in the authorizers.xml file.

This initial admin user is granted access to the UI and given the ability to create additional users, groups, and policies. The value of this property could be a DN (when using certificates or LDAP) or a Kerberos principal. If you are the NiFi administrator, add yourself as the "Initial Admin Identity".

Here is an example LDAP entry using the name John Smith:

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
```

```
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">cn=John
 Smith,ou=people,dc=example,dc=com</property>
        <property name="Legacy Authorized Users File"></property>
        <!--
        <property name="Node Identity 1"></property>
        <property name="Node Identity 2"></property>
        -->
    </authorizer>
</authorizers>
```

Here is an example Kerberos entry using the name John Smith and realm NIFI.APACHE.ORG:

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">johnsmith@NIFI.APACHE.ORG</
property>
        <property name="Legacy Authorized Users File"></property>
        <!--
        <property name="Node Identity 1"></property>
        <property name="Node Identity 2"></property>
        -->
    </authorizer>
</authorizers>
```

After you have edited and saved the authorizers.xml file, restart NiFi. The "Initial Admin Identity" user and administrative policies are added to the users.xml and authorizations.xml files during restart. Once NiFi starts, the "Initial Admin Identity" user is able to access the UI and begin managing users, groups, and policies.

Note:

For a brand new secure flow, providing the "Initial Admin Identity" gives that user access to get into the UI and to manage users, groups and policies. But if that user wants to start modifying the flow, they need to grant themselves policies for the root process group. The system is unable to do this automatically because in a new flow the UUID of the root process group is not permanent until the flow.xml.gz is generated. If the NiFi instance is an upgrade from an existing flow.xml.gz or a 1.x instance going from unsecure to secure, then the "Initial Admin Identity" user is automatically given the privileges to modify the flow.

Legacy Authorized Users (NiFi Instance Upgrade)

If you are upgrading from a 0.x NiFi instance, you can convert your previously configured users and roles to the multi-tenant authorization model. In the authorizers.xml file, specify the location of your existing authorized-users.xml file in the "Legacy Authorized Users File" property.

Here is an example entry:

```
<authorizers>
    <authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity"></property>
```

```
        <property name="Legacy Authorized Users File">/Users/johnsmith/
config_files/authorized-users.xml</property>
    </authorizer>
</authorizers>
```

After you have edited and saved the authorizers.xml file, restart NiFi. Users and roles from the authorized-users.xml file are converted and added as identities and policies in the users.xml and authorizations.xml files. Once the application starts, users who previously had a legacy Administrator role can access the UI and begin managing users, groups, and policies.

Here is a summary of policies assigned to each legacy role if the NiFi instance has an existing flow.xml.gz:

	Admin	**DFM**	**Monitor**	**Provenance**	**NiFi**	**Proxy**
view the UI	*	*	*			
view the controller	*	*	*		*	
modify the controller		*				
view system diagnostics		*	*			
view the dataflow	*	*	*			
modify the dataflow		*				
view the users/ groups	*					
modify the users/ groups	*					
view policies	*					
modify policies	*					
query provenance				*		
access restricted components		*				
view the data		*		*		*
modify the data		*				*
retrieve site-to-site details					*	
send proxy user requests						*

NiFi fails to restart if values exist for both the "Initial Admin Identity" and "Legacy Authorized Users File" properties. You can specify only one of these values to initialize authorizations.

Do not manually edit the authorizations.xml file. Create authorizations only during initial setup and afterwards using the NiFi UI.

Cluster Node Identities

If you are running NiFi in a clustered environment, you must specify the identities for each node. The authorization policies required for the nodes to communicate are created during startup.

For example, if you are setting up a 2 node cluster with the following DNs for each node:

```
cn=nifi-1,ou=people,dc=example,dc=com
cn=nifi-2,ou=people,dc=example,dc=com
```

```
<authorizer>
        <identifier>file-provider</identifier>
        <class>org.apache.nifi.authorization.FileAuthorizer</class>
        <property name="Authorizations File">./conf/authorizations.xml</
property>
        <property name="Users File">./conf/users.xml</property>
        <property name="Initial Admin Identity">johnsmith@NIFI.APACHE.ORG</
property>
        <property name="Legacy Authorized Users File"></property>
        <property name="Node Identity
 1">cn=nifi-1,ou=people,dc=example,dc=com</property>
        <property name="Node Identity
 2">cn=nifi-2,ou=people,dc=example,dc=com</property>
    </authorizer>
</authorizers>
```

In a cluster, all nodes must have the same authorizations.xml. If a node has a different authorizations.xml, it cannot join the cluster. The only exception is if a node has an empty authorizations.xml. In this scenario, the node inherits the authorizations.xml from the cluster.

Now that initial authorizations have been created, additional users, groups and authorizations can be created and managed in the NiFi UI.

Configuring Users & Access Policies

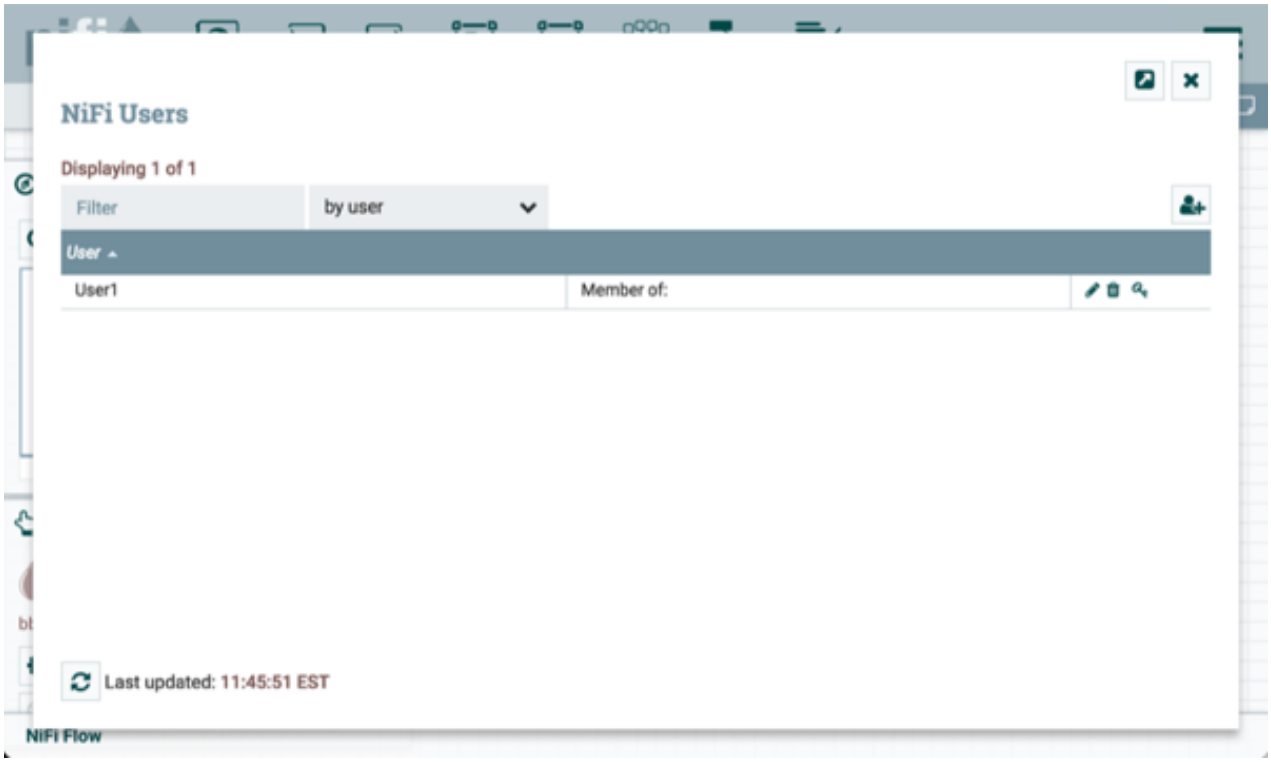
This section describes:

- How to create users and groups
- How access policies are used to define authorizations
- How to configure access policies by walking through specific examples

Instructions requiring interaction with the UI assume the application is being accessed by User1, a user with administrator privileges, such as the "Initial Admin Identity" user or a converted legacy admin user.

Creating Users and Groups

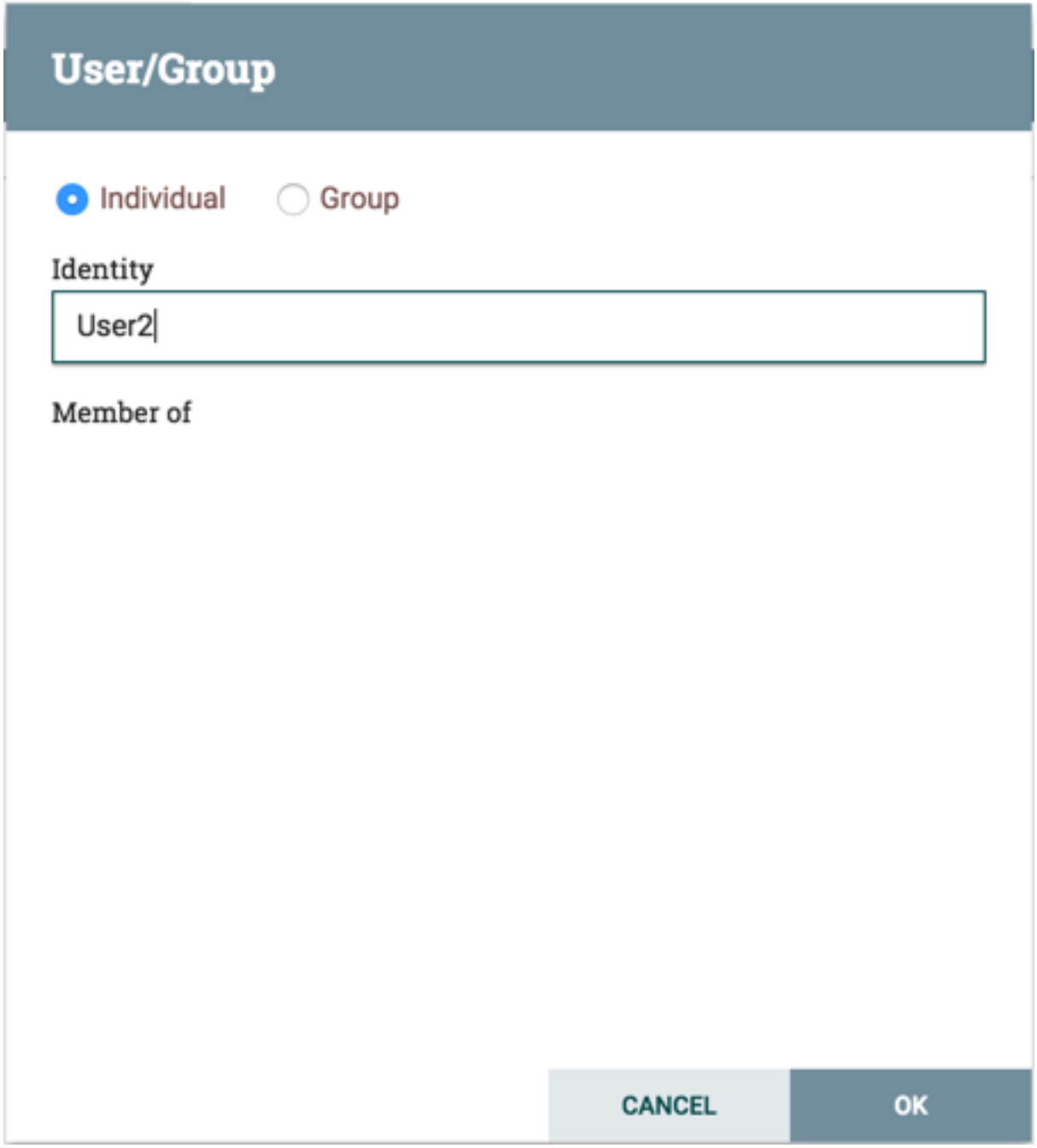
From the UI, select "Users" from the Global Menu. This opens a dialog to create and manage users and groups.



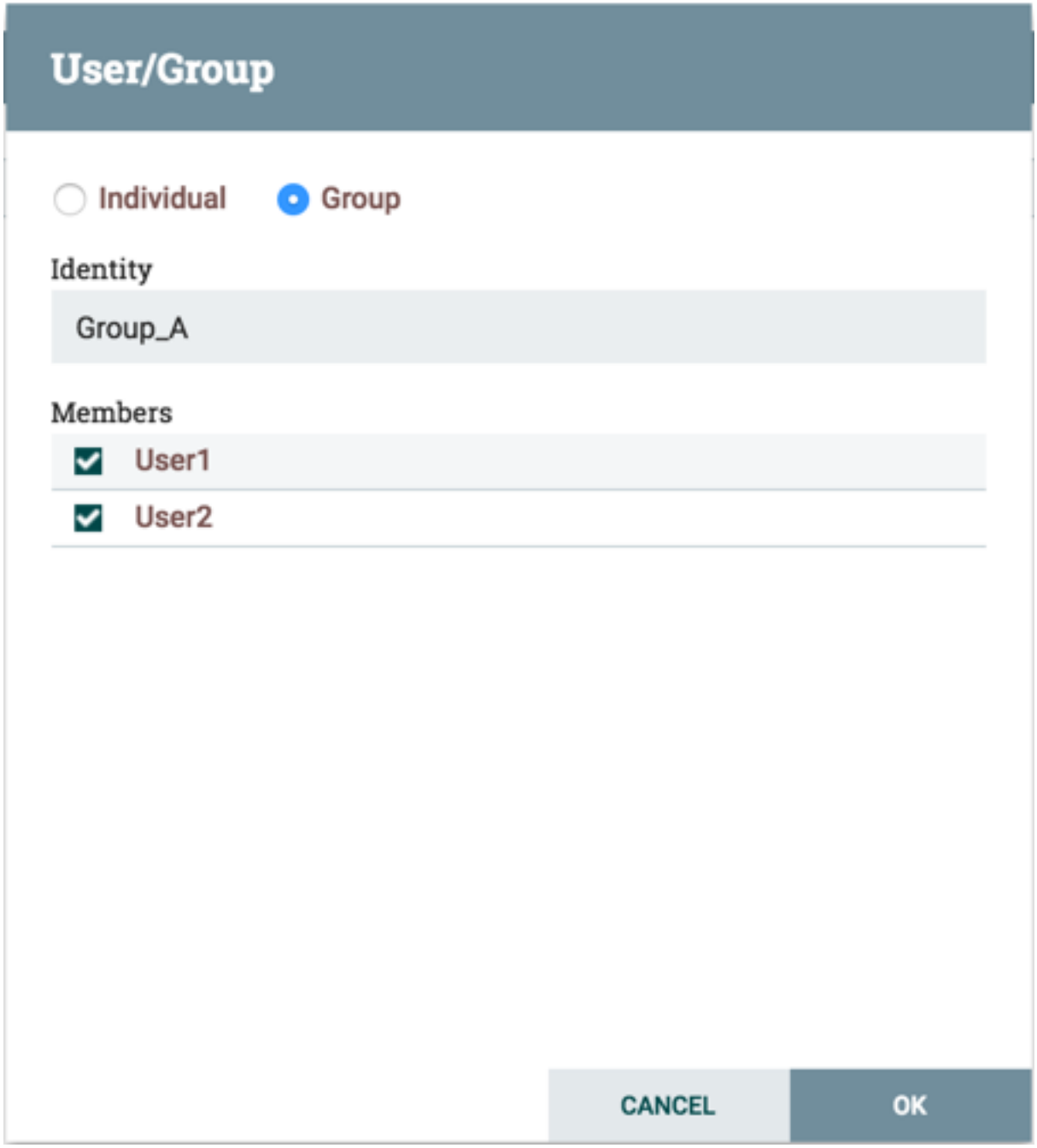
Click the Add icon

().

To create a user, enter the Identity information relevant to the authentication method chosen to secure your NiFi instance. Click OK.



To create a group, select the "Group" radio button, enter the name of the group and select the users to be included in the group. Click OK.



Access Policies

You can manage the ability for users and groups to view or modify NiFi resources using access policies. There are two types of access policies that can be applied to a resource:

- View - If a view policy is created for a resource, only the users or groups that are added to that policy are able to see the details of that resource.
- Modify - If a resource has a modify policy, only the users or groups that are added to that policy can change the configuration of that resource.

You can create and apply access policies on both global and component levels.

Global Access Policies

Global access policies govern the following system level authorizations:

Policy	Privilege	Global Menu Selection
view the UI	Allow users to view the UI	N/A
access the controller	Allows users to view/modify the controller including Reporting Tasks, Controller Services, and Nodes in the Cluster	Controller Settings
query provenance	Allows users to submit a Provenance Search and request Event Lineage	Data Provenance
access restricted components	Allows users to create/modify restricted components assuming otherwise sufficient permissions	N/A
access all policies	Allows users to view/modify the policies for all components	Policies
access users/user groups	Allows users to view/modify the users and user groups	Users
retrieve site-to-site details	Allows other NiFi instances to retrieve Site-To-Site details	N/A
view system diagnostics	Allows users to view System Diagnostics	Summary
proxy user requests	Allows proxy machines to send requests on the behalf of others	N/A
access counters	Allows users to view/modify Counters	Counters

Component Level Access Policies

Component level access policies govern the following component level authorizations:

Policy	Privilege
view the component	Allows users to view component configuration details
modify the component	Allows users to modify component configuration details
view the data	Allows user to view metadata and content for this component through provenance data and flowfile queues in outbound connections
modify the data	Allows user to empty flowfile queues in outbound connections and submit replays
view the policies	Allows users to view the list of users who can view/modify a component
modify the policies	Allows users to modify the list of users who can view/modify a component
retrieve data via site-to-site	Allows a port to receive data from NiFi instances
send data via site-to-site	Allows a port to send data from NiFi instances

You can apply access policies to all component types except connections. Connection authorizations are inferred by the individual access policies on the source and destination components of the connection, as well as the access policy of the process group containing the components. This is discussed in more detail in the creating a connection and editing a connection examples below.

Access Policy Inheritance

An administrator does not need to manually create policies for every component in the dataflow. To reduce the amount of time admins spend on authorization management, policies are inherited from parent resource to child resource.

For example, if a user is given access to view and modify a process group, that user can also view and modify the components in the process group. Policy inheritance enables an administrator to assign policies at one time and have the policies apply throughout the entire dataflow.

You can override an inherited policy (as described in the Moving a Processor example below). Overriding a policy removes the inherited policy, breaking the chain of inheritance from parent to child, and creates a replacement policy to add users as desired. Inherited policies and their users can be restored by deleting the replacement policy.

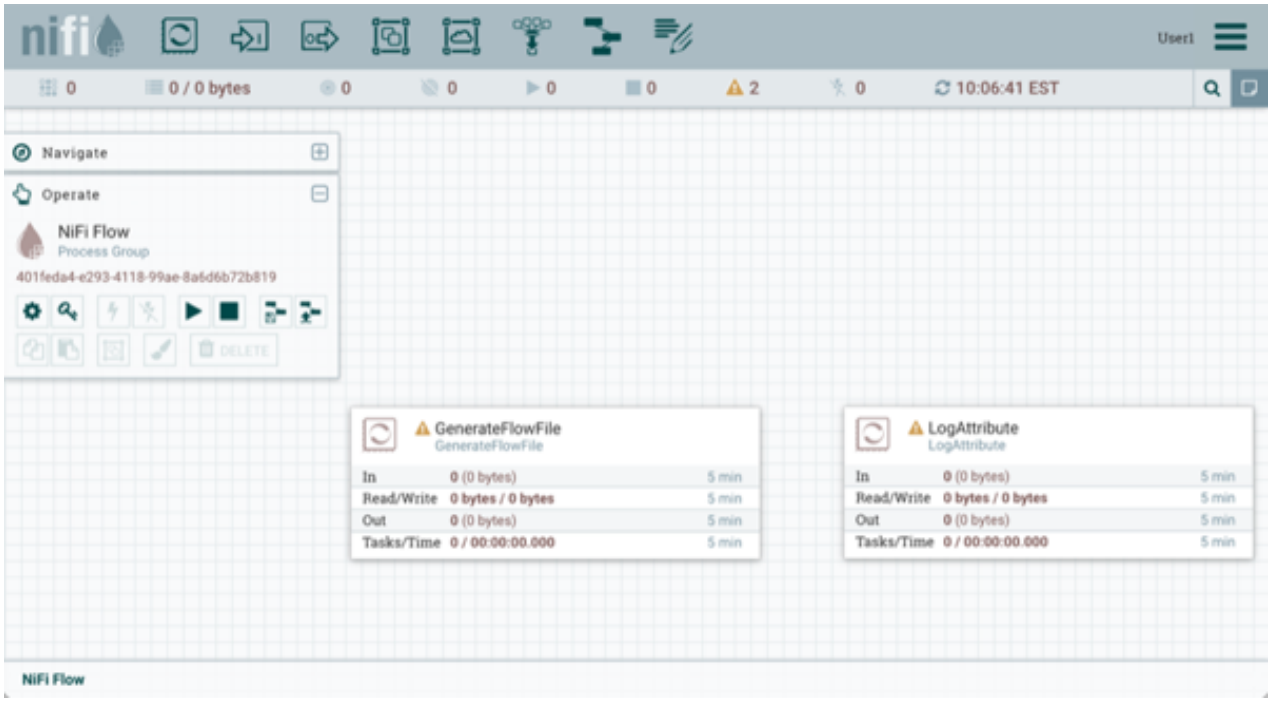
"View the policies" and "modify the policies" component-level access policies are an exception to this inherited behavior. When a user is added to either policy, they are added to the current list of administrators. They do not override higher level administrators. For this reason, only component specific administrators are displayed for the "view the policies" and "modify the policies" access policies.

You cannot modify the users/groups on an inherited policy. Users and groups can only be added or removed from a parent policy or an override policy.

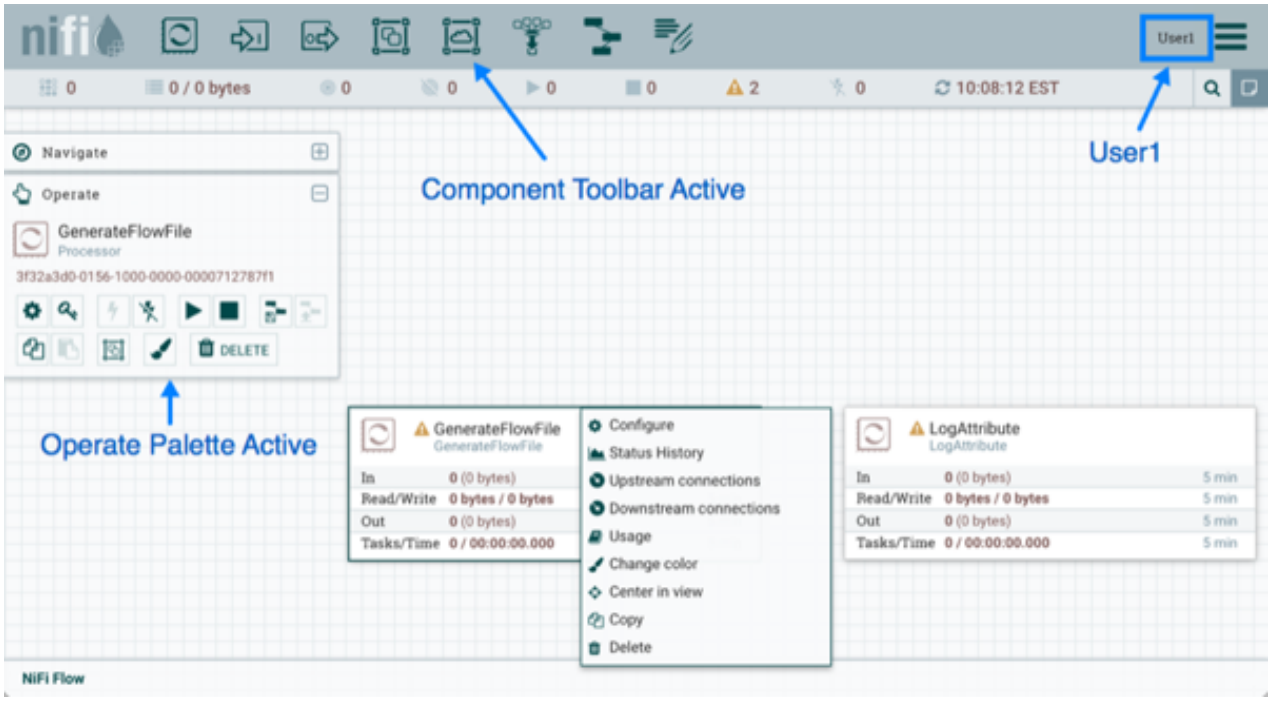
Access Policy Configuration Examples

The most effective way to understand how to create and apply access policies is to walk through some common examples. The following scenarios assume User1 is an administrator and User2 is a newly added user that has only been given access to the UI.

Let's begin with two processors on the canvas as our starting point: GenerateFlowFile and LogAttribute.

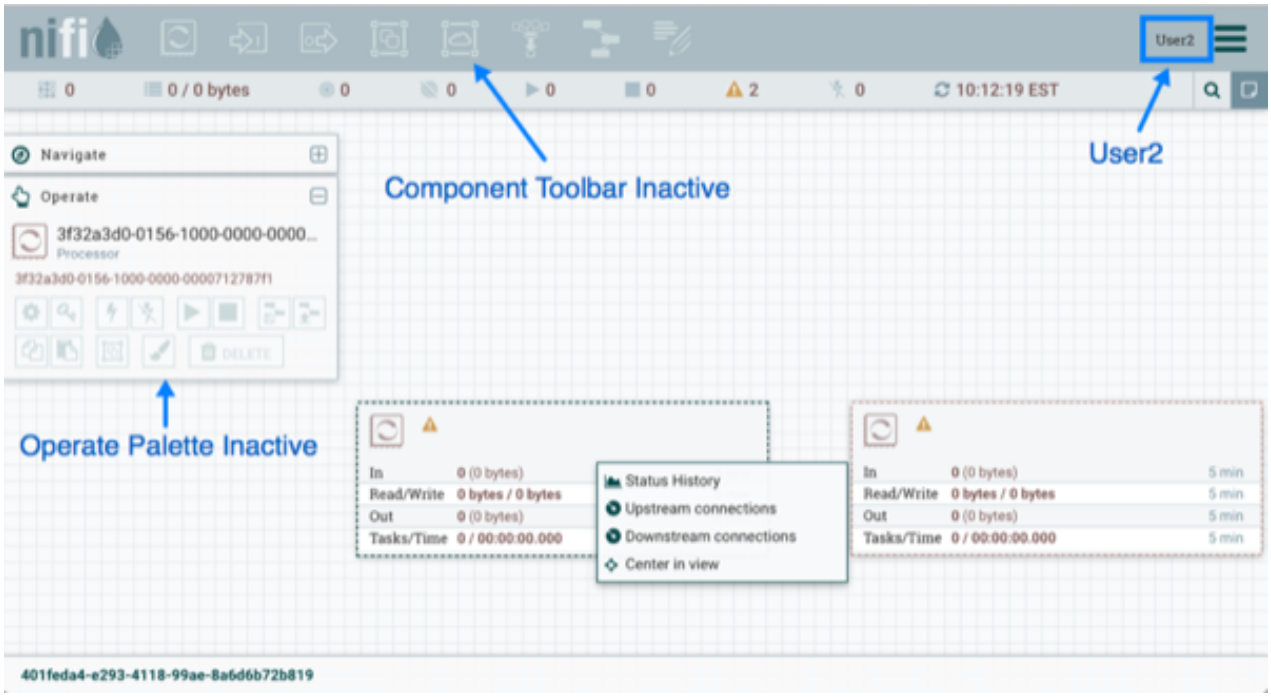


User1 can add components to the dataflow and is able to move, edit and connect all processors. The details and properties of the root process group and processors are visible to User1.



User1 wants to maintain their current privileges to the dataflow and its components.

User2 is unable to add components to the dataflow or move, edit, or connect components. The details and properties of the root process group and processors are hidden from User2.

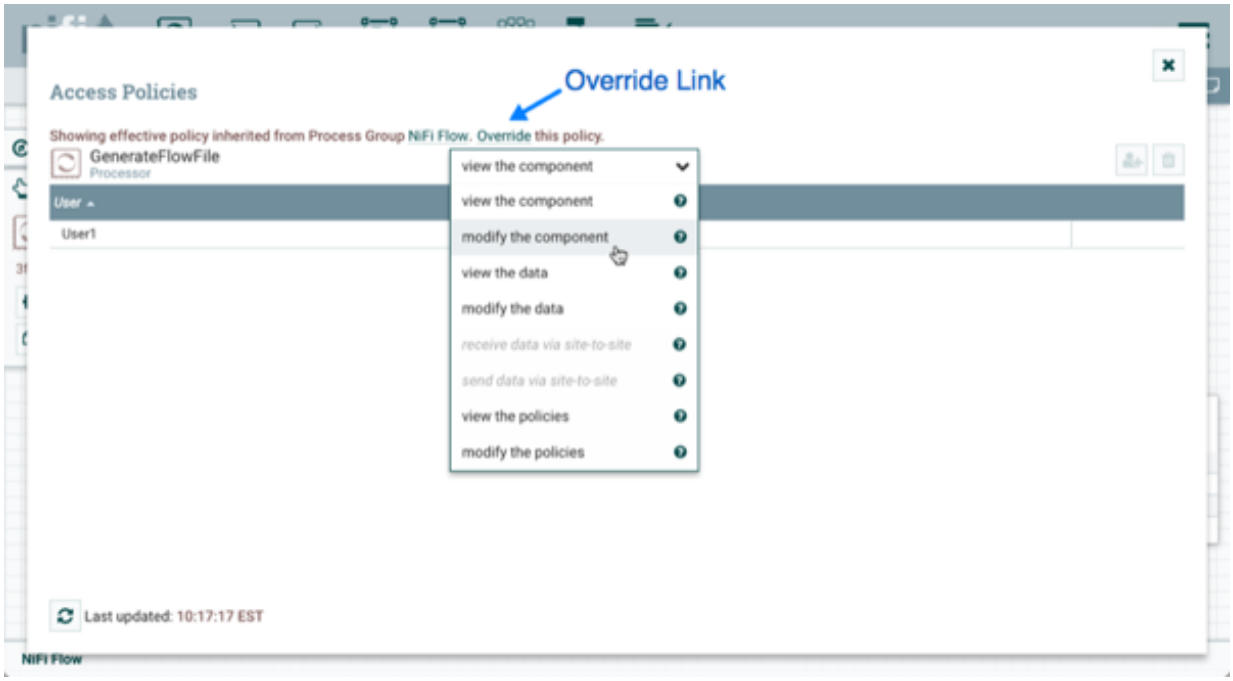


Moving a Processor

To allow User2 to move the GenerateFlowFile processor in the dataflow and only that processor, User1 performs the following steps:

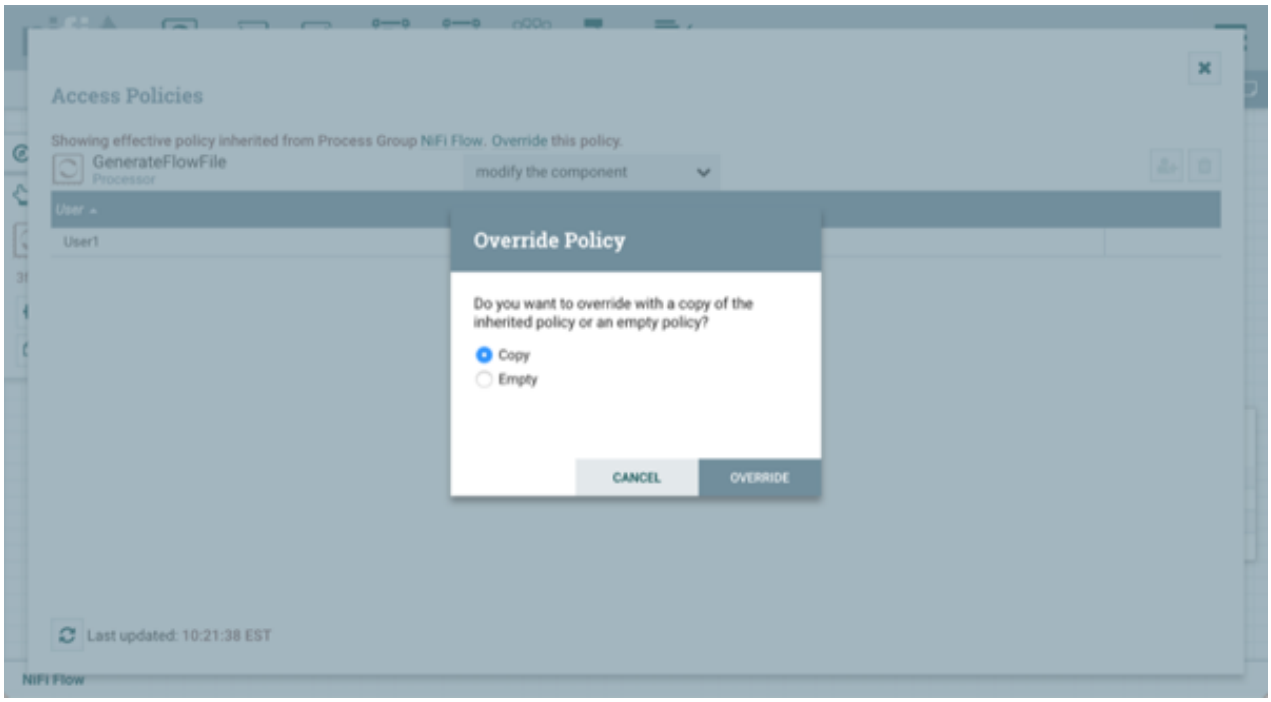
1. Select the GenerateFlowFile processor so that it is highlighted.

2. Select the Access Policies icon
( ) from the Operate palette and the Access Policies dialog opens.
3. Select "modify the component" from the policy drop-down.



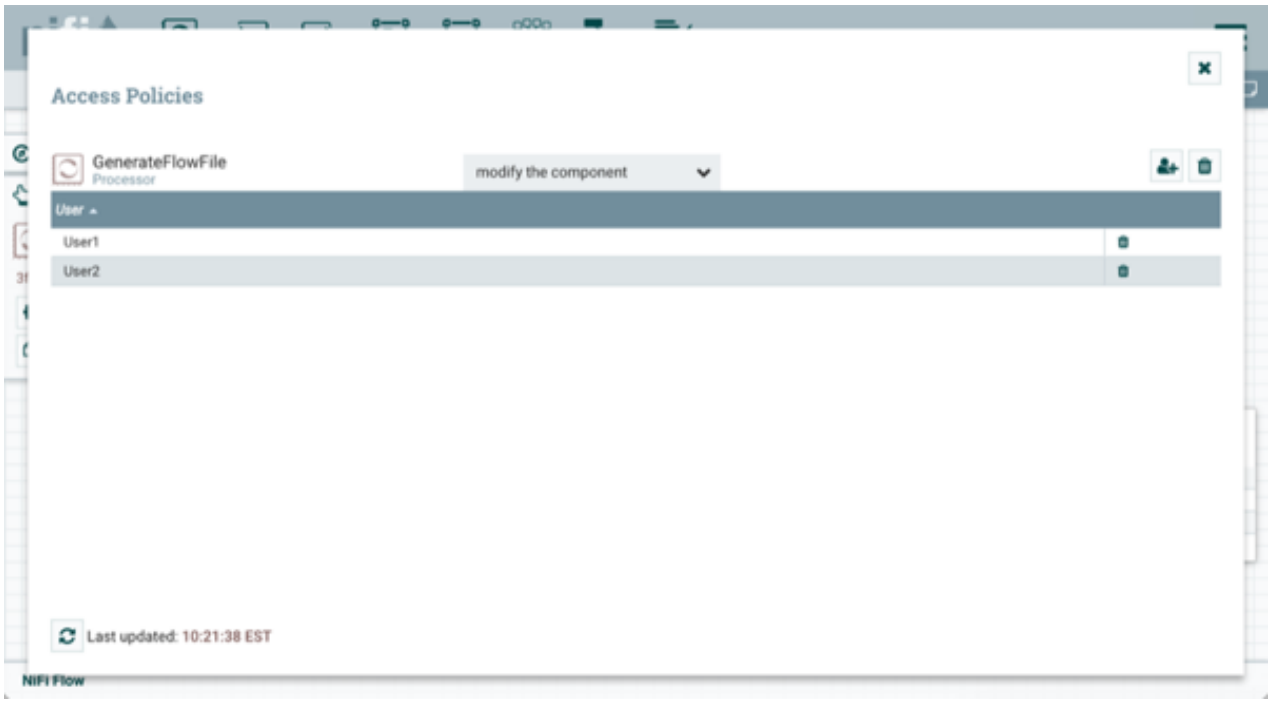
The "modify the component" policy that currently exists on the processor (child) is the "modify the component" policy inherited from the root process group (parent) on which User1 has privileges.

4. Select the Override link in the policy inheritance message. When creating the replacement policy, you are given a choice to override with a copy of the inherited policy or an empty policy.

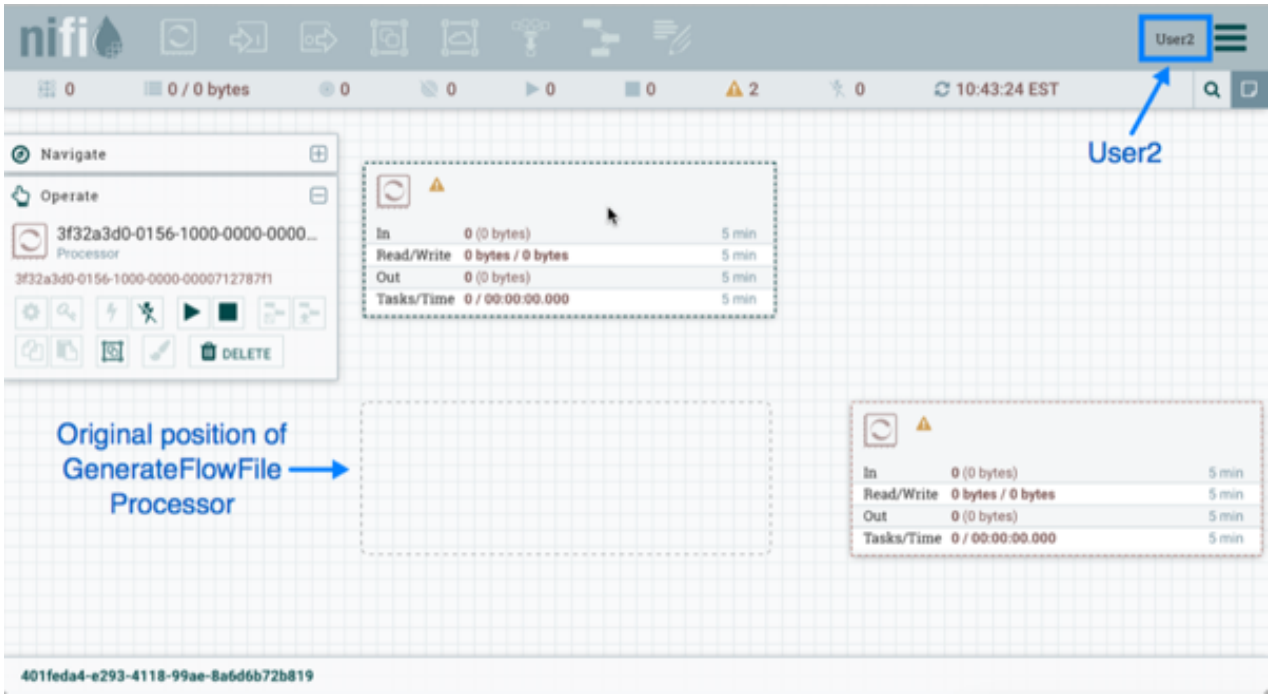


Select the Override button to create a copy.

1. On the replacement policy that is created, select the Add User icon
   ( ).
   Find or enter User2 in the User Identity field and select OK.



With these changes, User1 maintains the ability to move both processors on the canvas. User2 can now move the GenerateFlowFile processor but cannot move the LogAttribute processor.

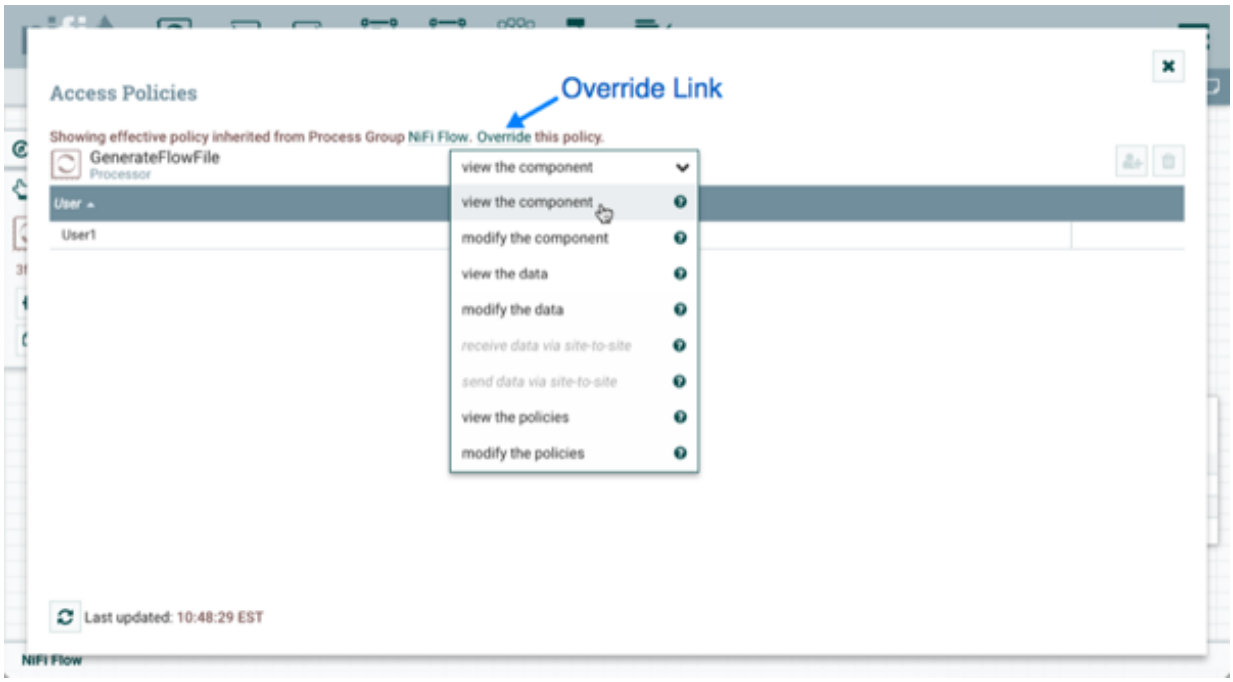


Editing a Processor

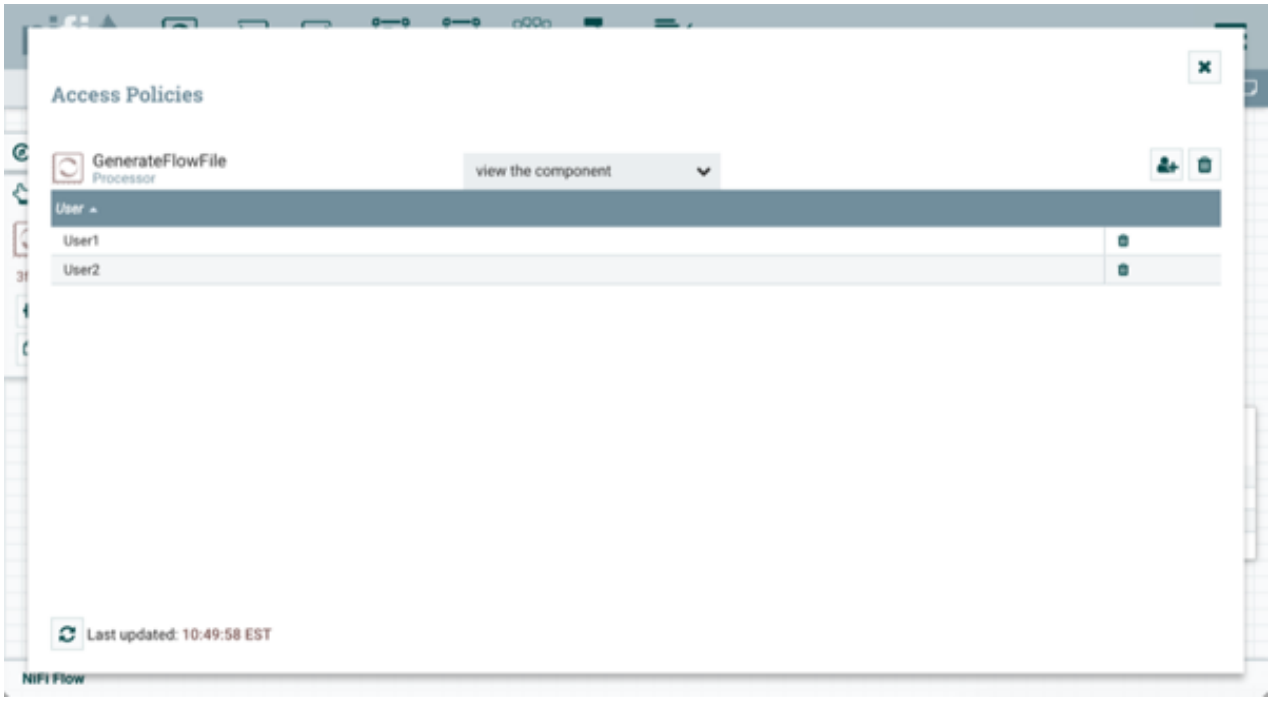
In the "Moving a Processor" example above, User2 was added to the "modify the component" policy for GenerateFlowFile. Without the ability to view the processor properties, User2 is unable to modify the processor's

configuration. In order to edit a component, a user must be on both the "view the component" and "modify the component" policies.

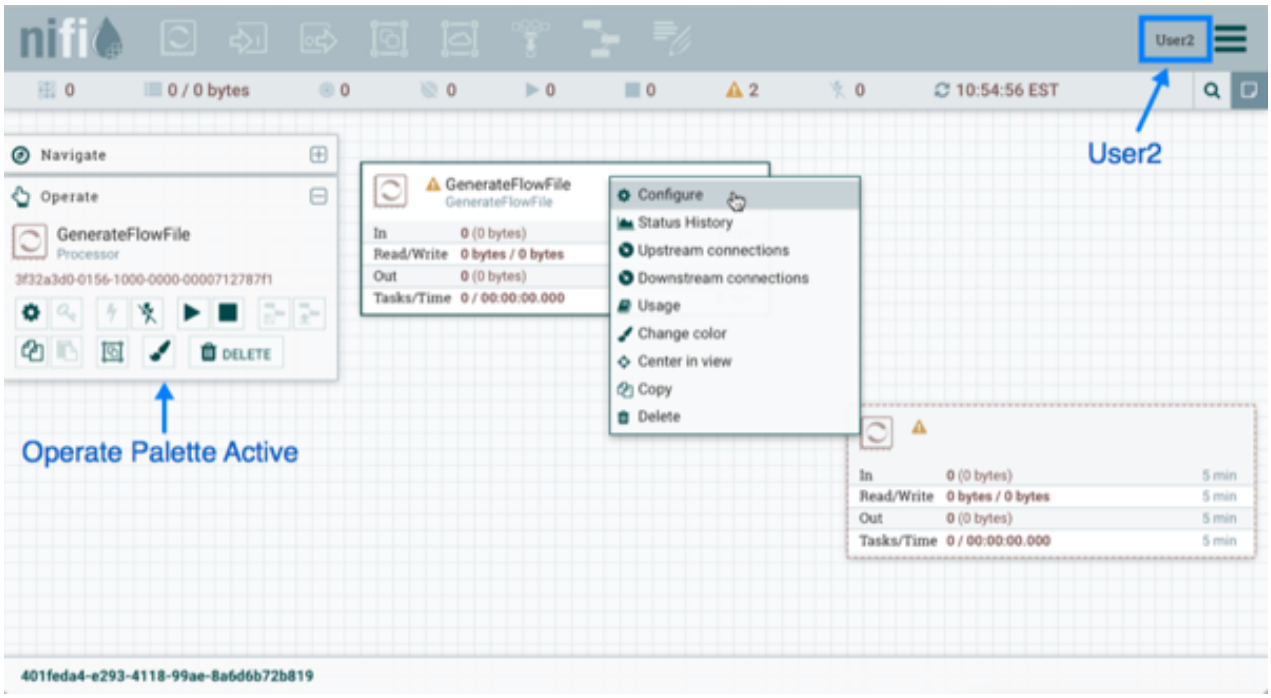
To implement this, User1 performs the following steps:

1. Select the GenerateFlowFile processor.
2. Select the Access Policies icon
   ()
   from the Operate palette and the Access Policies dialog opens.
3. Select "view the component" from the policy drop-down.
   

   The view the component" policy that currently exists on the processor (child) is the "view the component" policy inherited from the root process group (parent) on which User1 has privileges.
4. Select the Override link in the policy inheritance message, keep the default of Copy policy and select the Override button.
5. On the override policy that is created, select the Add User icon
   ().
   Find or enter User2 in the User Identity field and select OK.

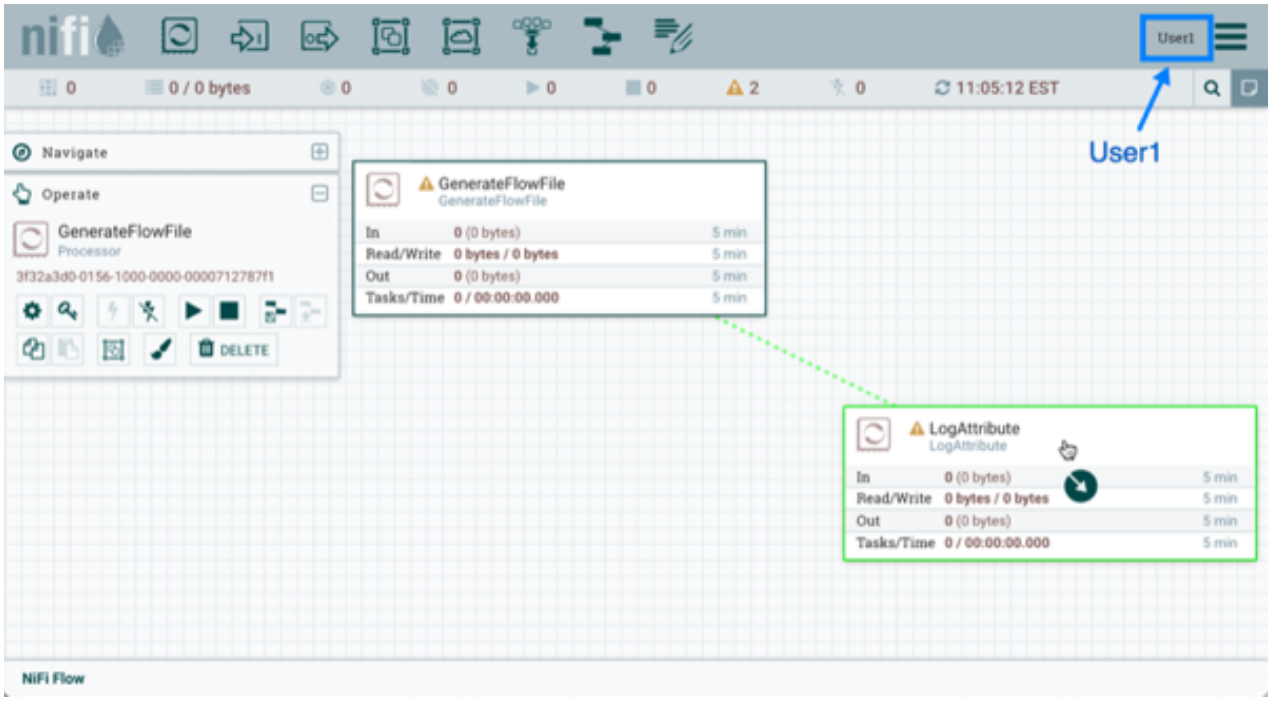


With these changes, User1 maintains the ability to view and edit the processors on the canvas. User2 can now view and edit the GenerateFlowFile processor.

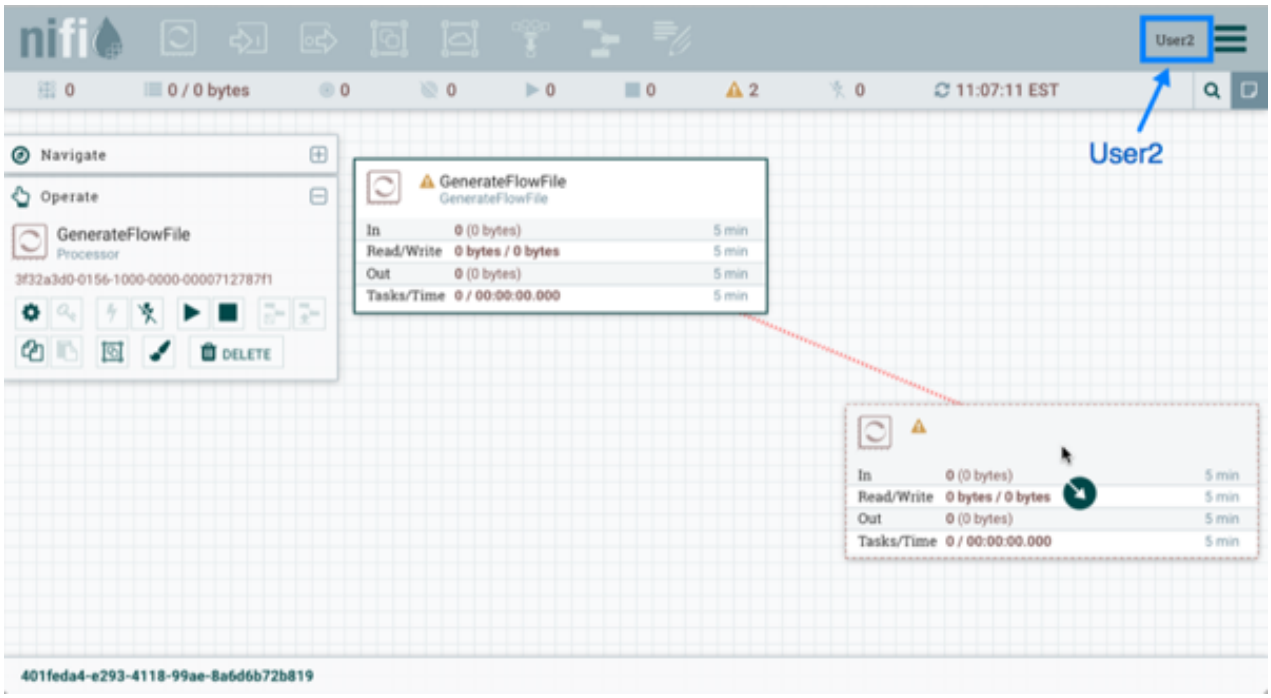


Creating a Connection

With the access policies configured as discussed in the previous two examples, User1 is able to connect GenerateFlowFile to LogAttribute:



User2 cannot make the connection:



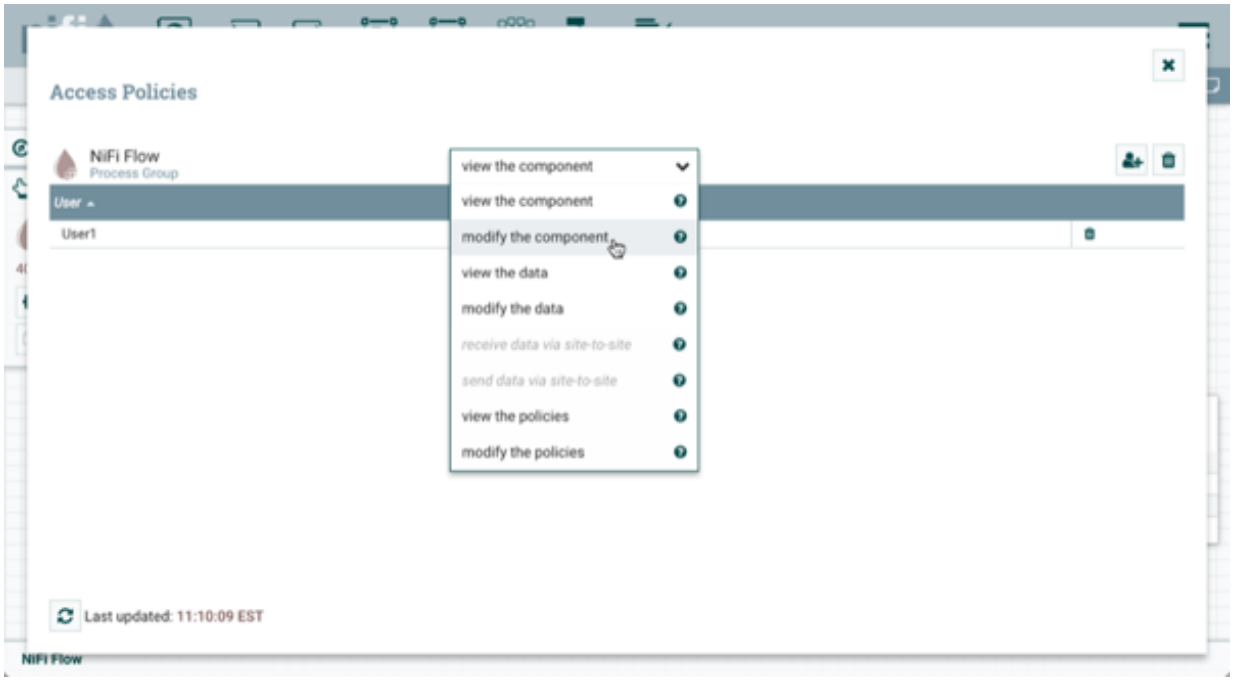
This is because:

- User2 does not have modify access on the process group.
- Even though User2 has view and modify access to the source component (GenerateFlowFile), User2 does not have an access policy on the destination component (LogAttribute).

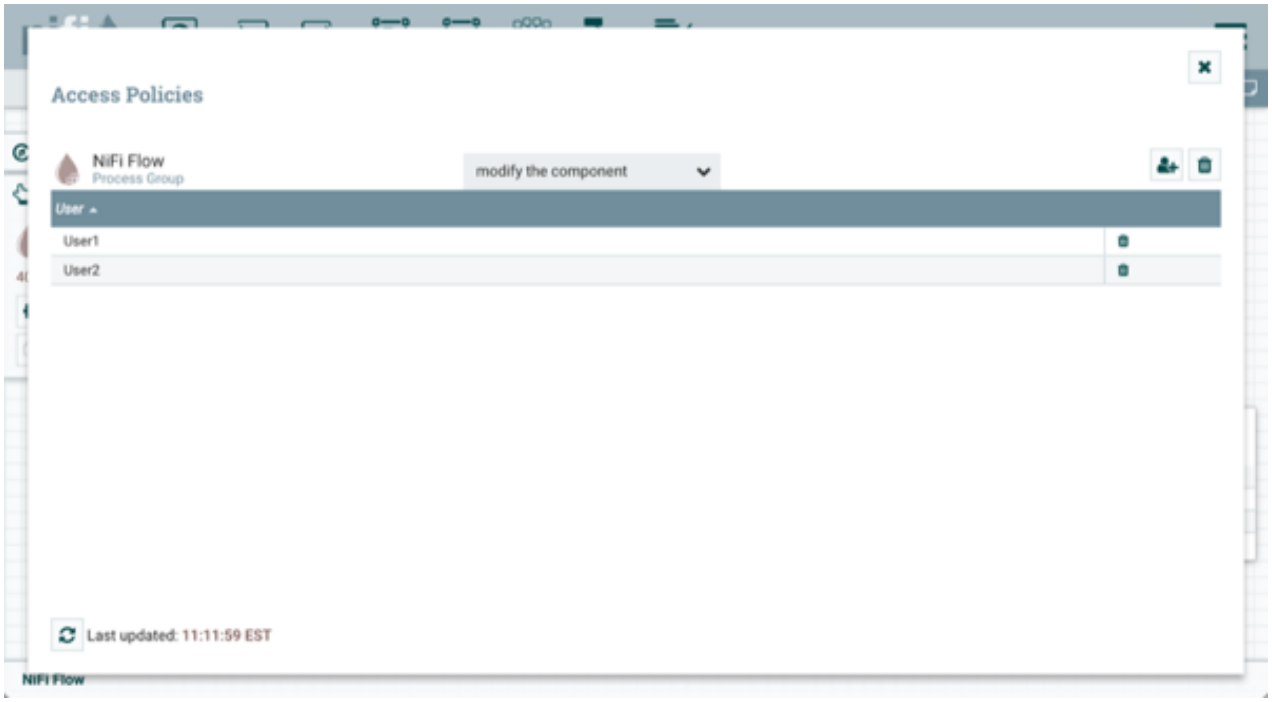
To allow User2 to connect GenerateFlowFile to LogAttribute, as User1:

1. Select the root process group. The Operate palette is updated with details for the root process group.

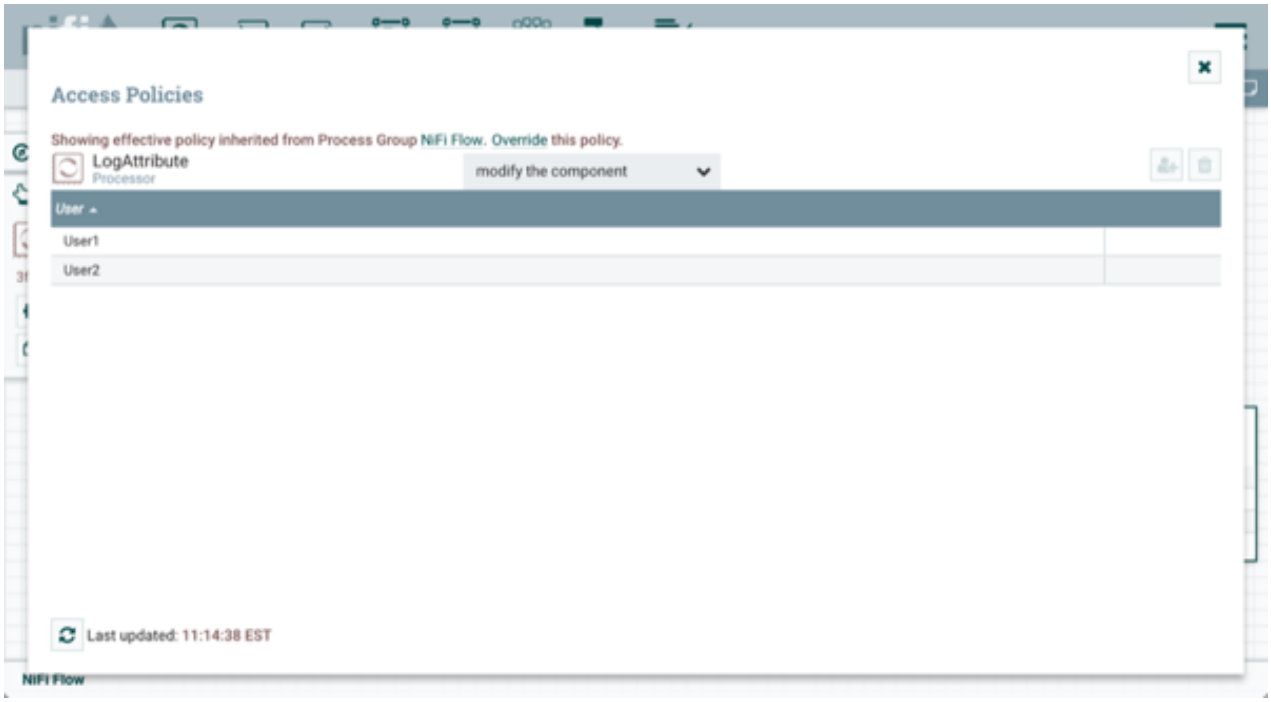
2. Select the Access Policies icon
   ()
   from the Operate palette and the Access Policies dialog opens.
3. Select "modify the component" from the policy drop-down.



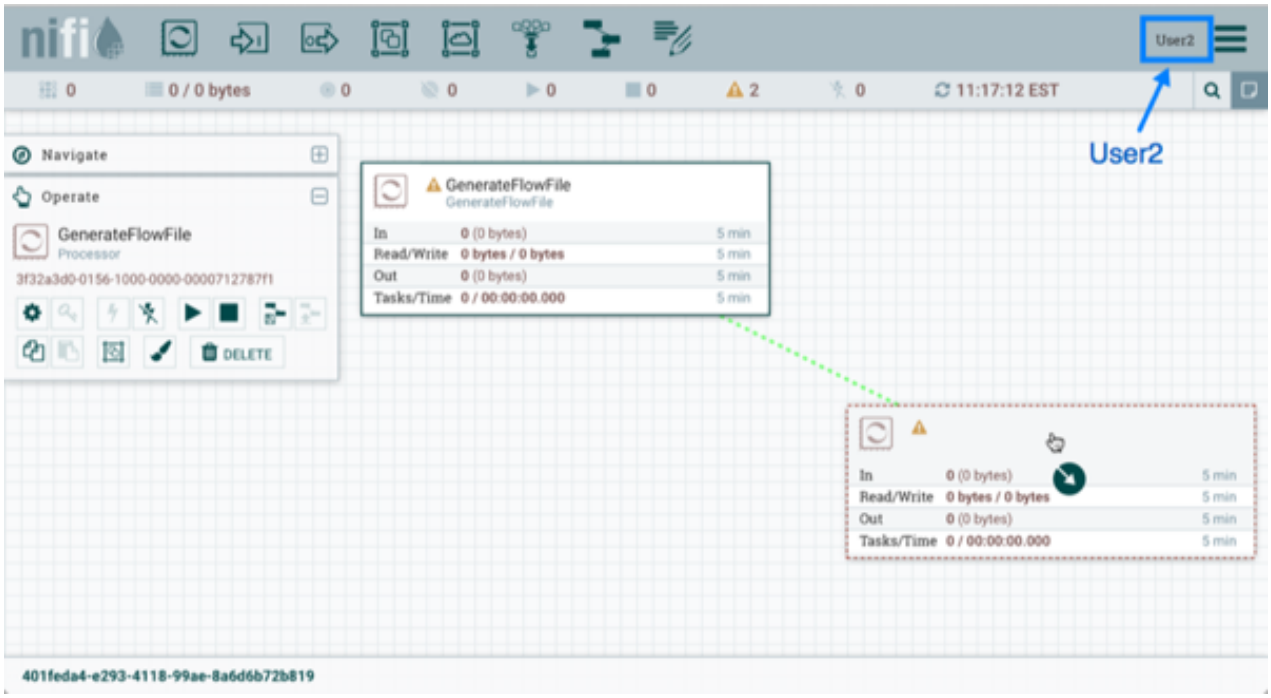
4. Select the Add User icon
   ().
   Find or enter User2 and select OK.

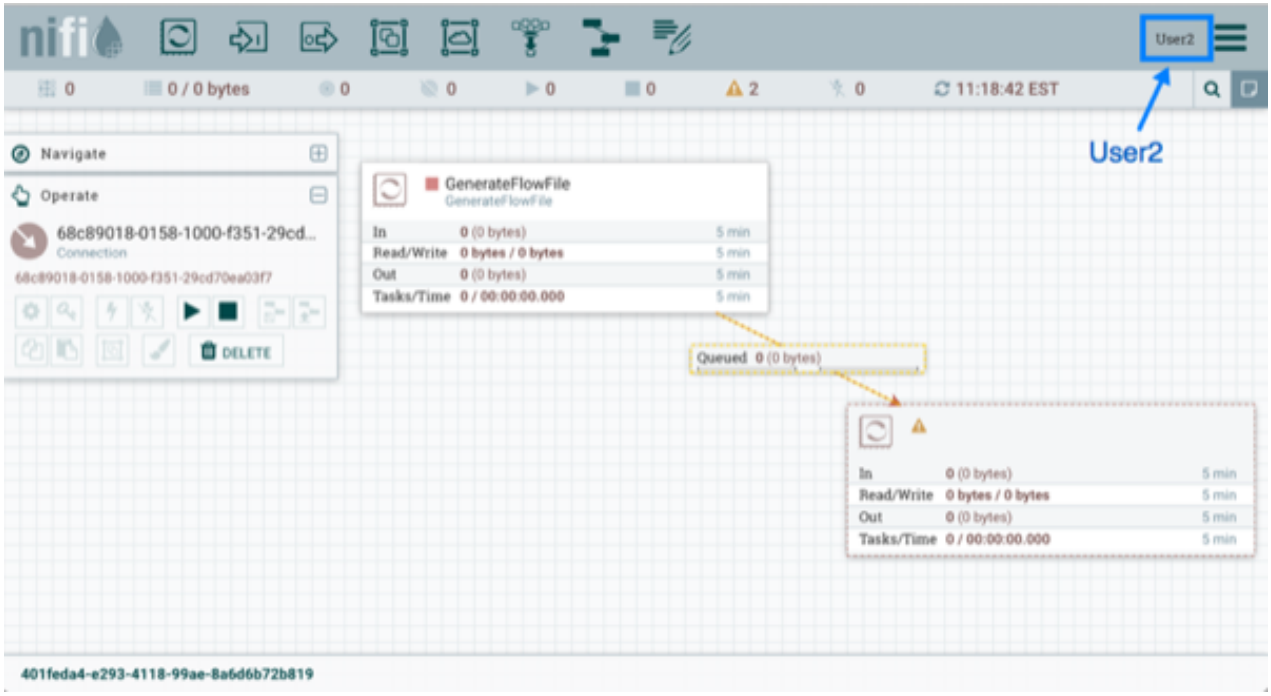


By adding User2 to the "modify the component" policy on the process group, User2 is added to the "modify the component" policy on the LogAttribute processor by policy inheritance. To confirm this, highlight the LogAttribute processor and select the Access Policies icon ( ) from the Operate palette:



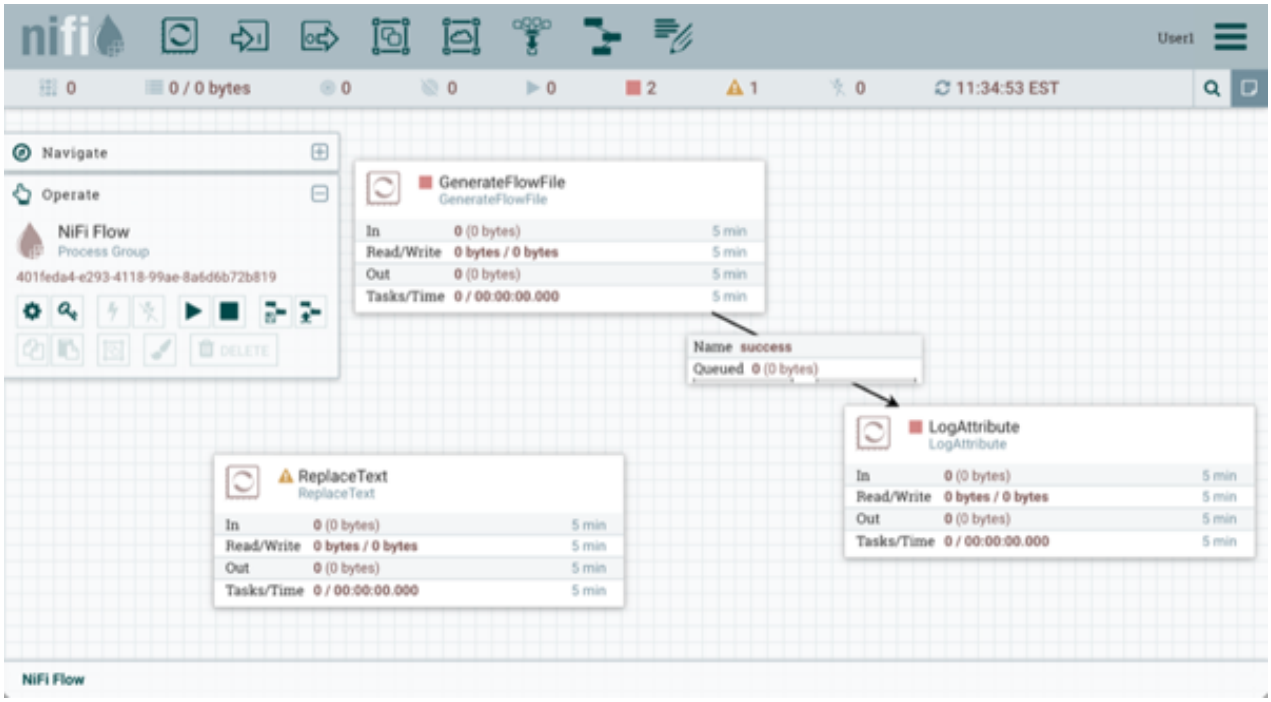
With these changes, User2 can now connect the GenerateFlowFile processor to the LogAttribute processor.



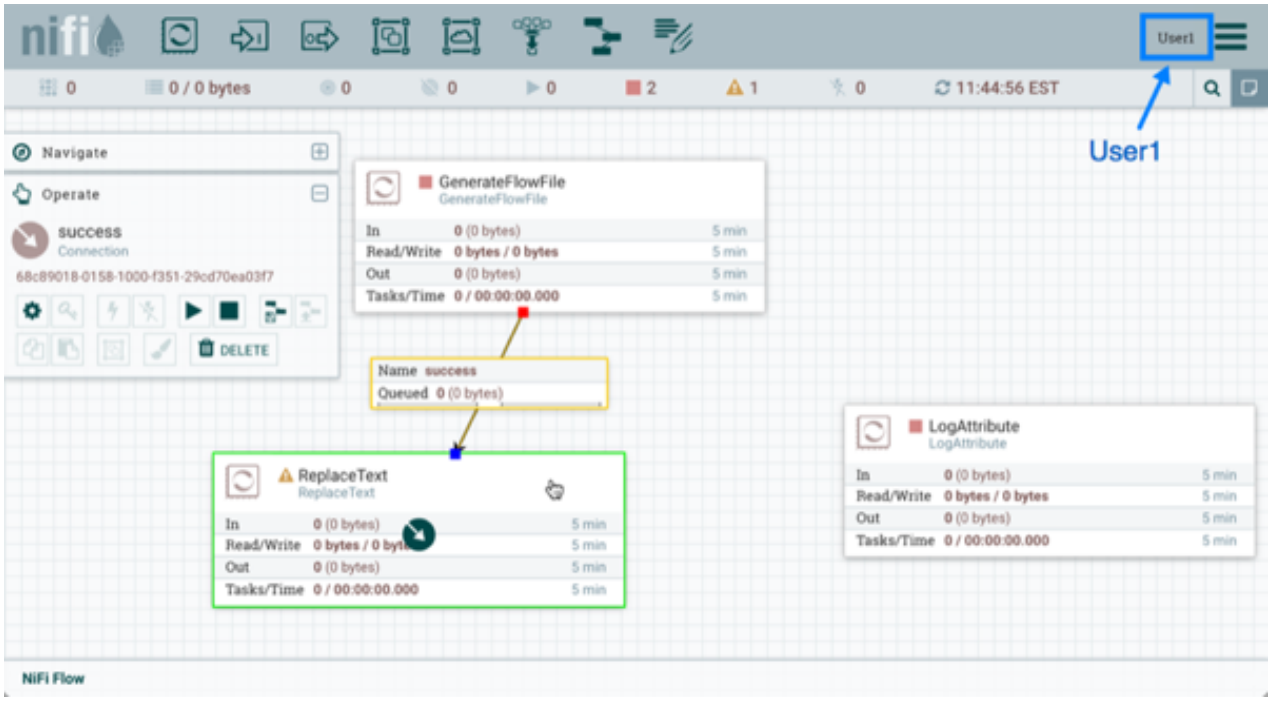


Editing a Connection

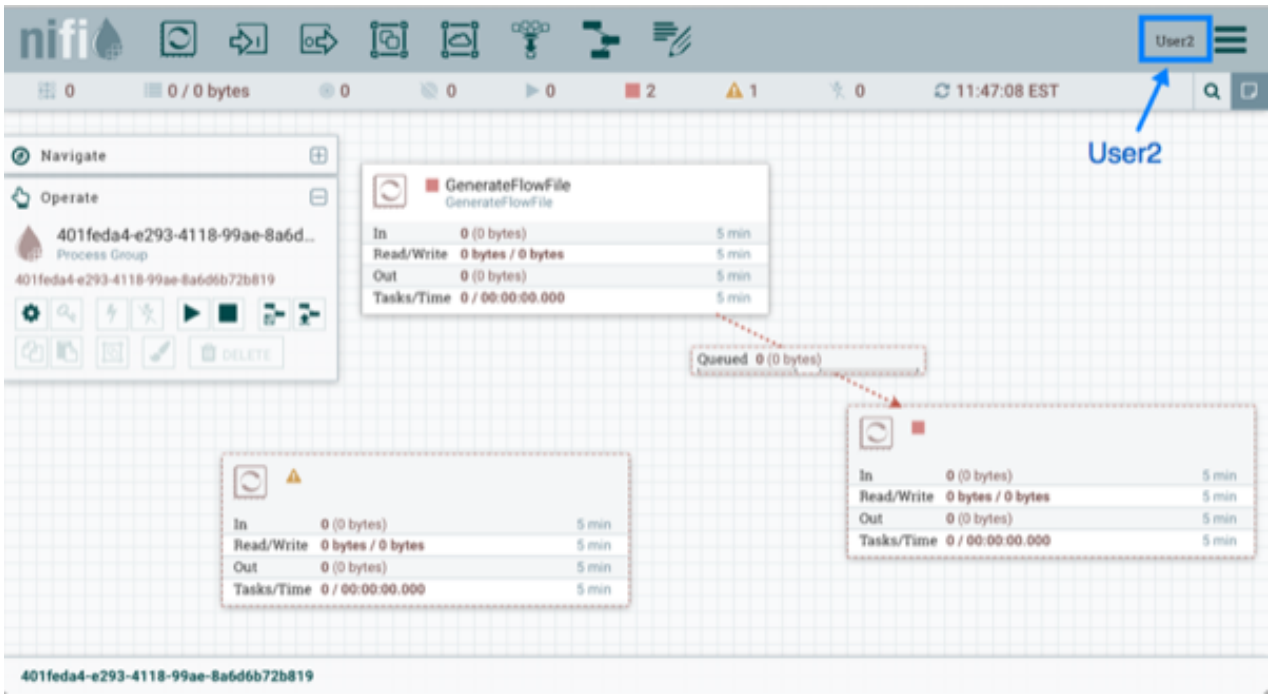
Assume User1 or User2 adds a ReplaceText processor to the root process group:



User1 can select and change the existing connection (between GenerateFlowFile to LogAttribute) to now connect GenerateFlowFile to ReplaceText:



User 2 is unable to perform this action.

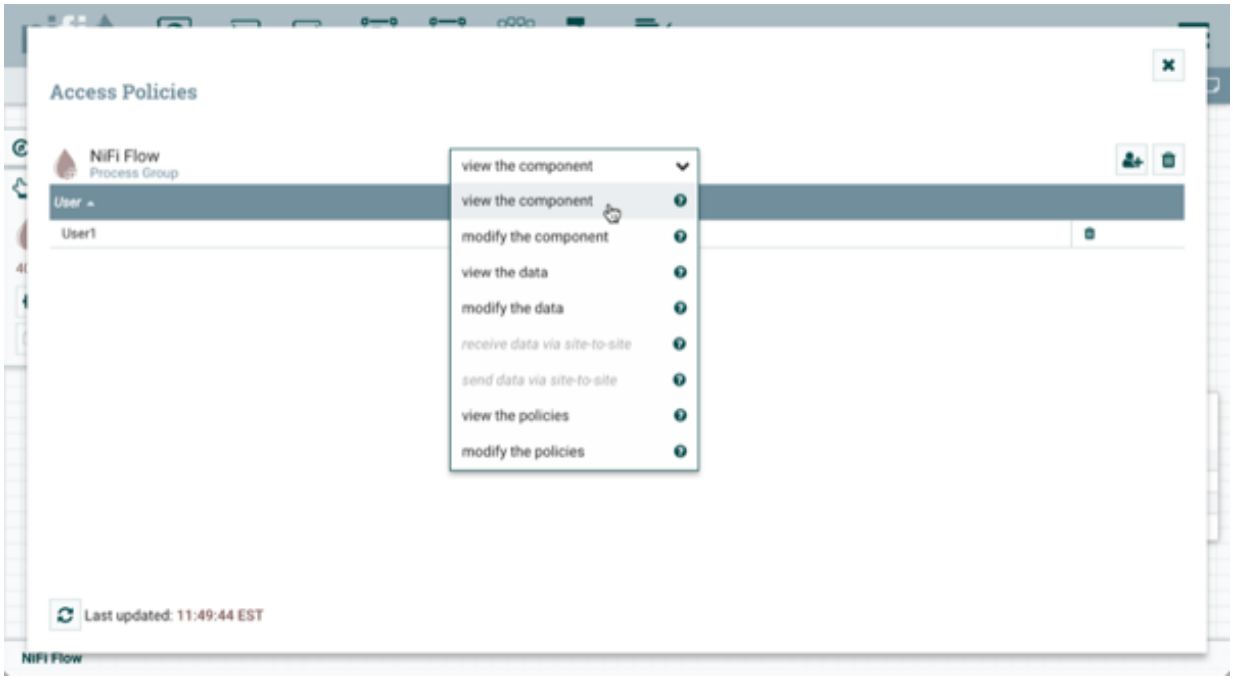


To allow User2 to connect GenerateFlowFile to ReplaceText, as User1:

1. Select the root process group. The Operate palette is updated with details for the root process group.
2. Select the Access Policies icon

( ).

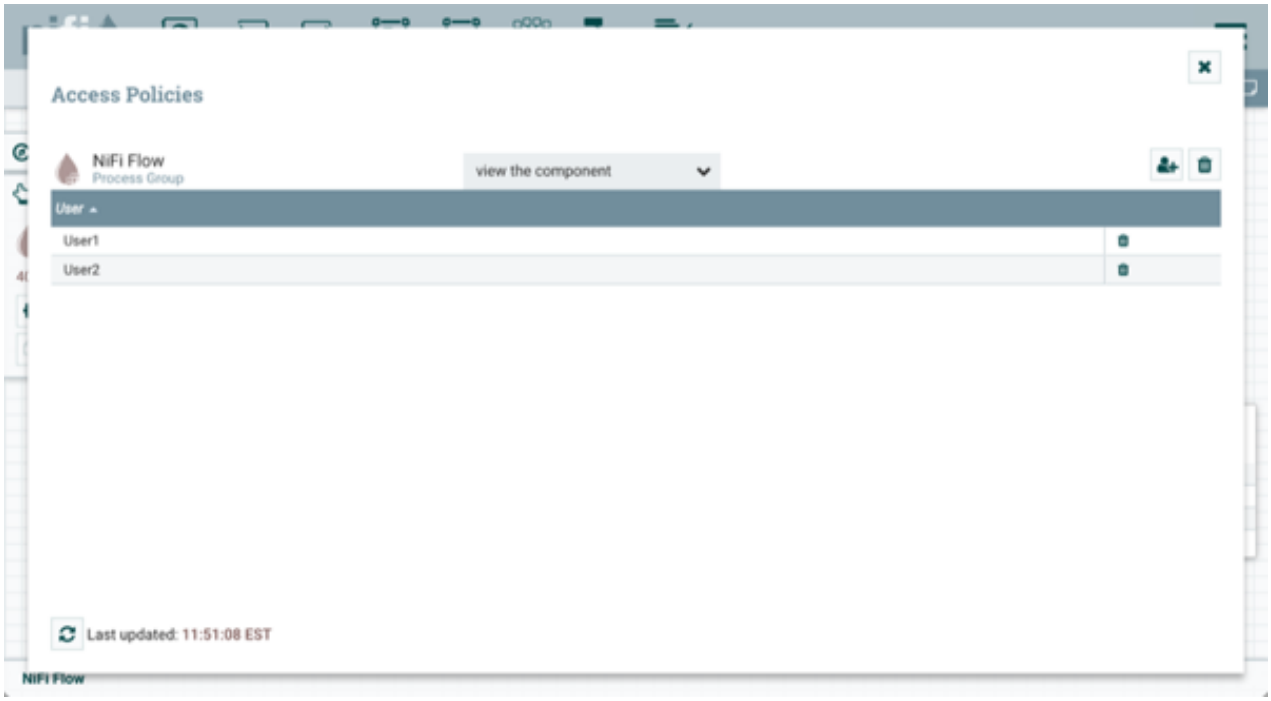
3. Select "view the component" from the policy drop-down.



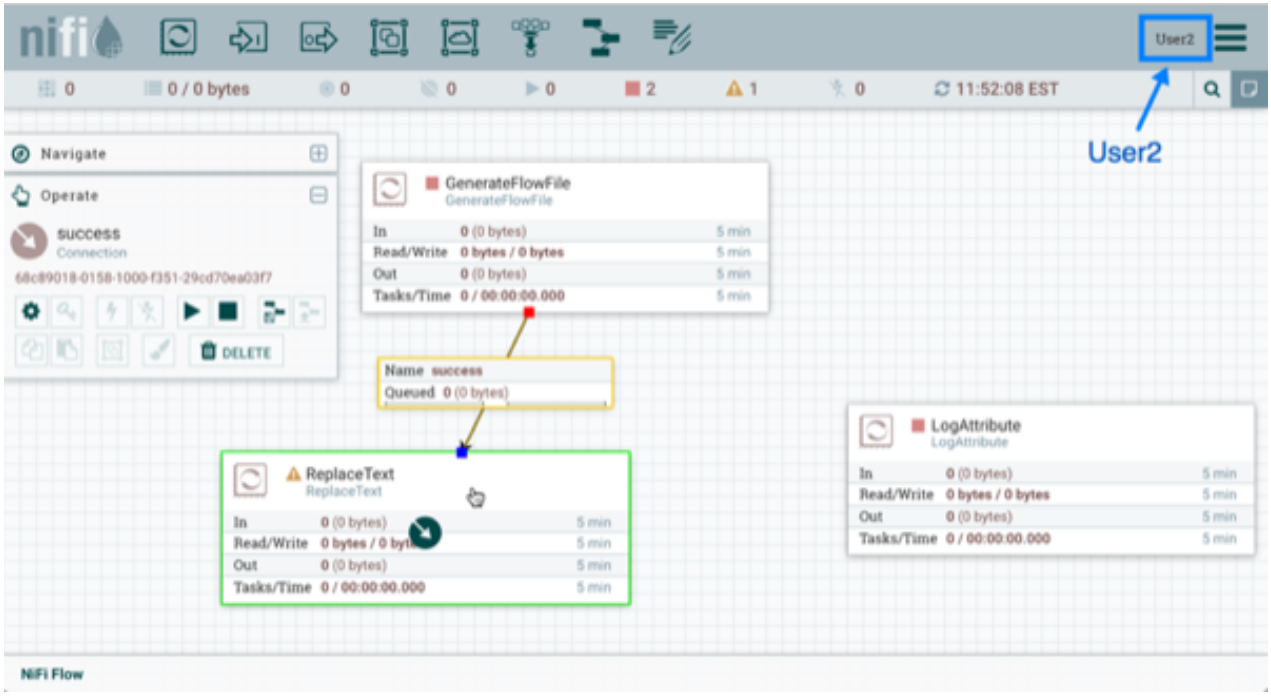
4. Select the Add User icon

( ).

Find or enter User2 and select OK.



Being added to both the view and modify policies for the process group, User2 can now connect the GenerateFlowFile processor to the ReplaceText processor.


User2
User2
Navigate
Operate
success
Connection
68c89018-0158-1000-f351-29cd70ea03f7
DELETE
GenerateFlowFile
GenerateFlowFile
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
Name success
Queued 0 (0 bytes)
ReplaceText
ReplaceText
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
LogAttribute
LogAttribute
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min
NiFi Flow