NiFi S2S On Secured Instances

Daniel Aaron Salwerowicz

system developer at National Library of Norway

2024-02-21

Contents

1 S2S Connection Between Two Secured NiFi Instances

When NiFi 1.14.0 released every instance is secure by default. In addition to it in proper production environments NiFi will have certificates, users, and policies to deal with. This is great for security, however it makes tasks like setting up a Site-to-Site (S2S) communication and Reporting Tasks difficult. What makes this even more difficult is lack of proper guidance on how to deal with certificates, LDAP or Kerberos user management, and S2S. As such this guide aims to solve that issue by providing an example of how to set up a "`SiteToSiteProvenanceReportingTask`".

1.1 Motivation

We at National Library of Norway use NiFi to process large number of files and data. NiFi performs varying operations, ranging from simple file transfer to complex systems for creating data packages, validating them, and storing them. As such we end up with flows that might experience some performance issues that are hard to spot or find reason for them. Therefore we wanted to set up provenance reporting task to help us see what files were causing issues and when.

However that requires setting up S2S communication between servers. As security is important to us our NiFi instances need to use both certificates and proper user authorization, as well as use proxies, firewalls, and encryption. This made the task of setting up a Site-to-Site communication hard. After some time finding the way to do it we decided to write this guide to help other NiFi users who might struggle with S2S on secured NiFi instances.

1.2 Prerequisites

This guide requires that you have following:

- Two independent NiFi instances:
  - One "main" instance which will send information to the "reporting" instance
    - These instances can be on separate servers or same server, running in docker or "native"
  - Each instance uses some sort of user management, either LDAP, Kerberos, etc
    - NiFi needs to have access to policy management and ability to add users to the system in order to get most out of this guide
    - Any actual production NiFi instance will likely have a user & policy management system
    - If it's missing this guide should work too, simply skip setting up the user and giving him policies, S2S should work still
  - Both instances use a certificate (either generated by toolkit or from some other source)
    - If the certificates come from other source, they need to "trust each other"
  - If running on separate servers proxy and firewall allow for communication between servers on the ports they use
  - Both instances run on version 1.24.0 (I haven't tested with other versions so I cannot guarantee this will work on them)

1.3 Assumptions

In order to make this guide a bit easier to follow it assumes following:

- Company name: MJ Industries (used in certificate generation)
- Hostname for the **main** NiFi instance: `nifi.mj.xyz`
- Hostname for the **reporting** NiFi instance: `reporting.mj.xyz`
- Main instance uses port `8443` for GUI access
- Reporting instance uses port `9443` for GUI access (just to keep them separate)

1.4 Certificate Tips

When generating a certificate using NiFi Toolkit it requires some arguments, the important arguments are:

Argument	What it does
`-C`	Generates a certificate with the specified DN, use the same value for every certificate
`-K`	Sets key password
`-S`	Sets keystore password
`-P`	Sets truststore password
`-o`	Sets the output directory for the generated cert files
`-d`	Sets certificate validity period in days
`-c`	Sets the certificate issuer, use the same value for every certificate
`-n`	Sets hostname for this certificate, set it to address users will use for accessing NiFi
`-O`	Makes the script override existing host folder if it exists

The most important of these arguments are the `-C` and `-c` arguments. They make sure that the certificates have proper trust chains set up and that the NiFi instances can trust each other.

For the purposes of this guide the `tls-toolkit` ran with following arguments:

```
./nifi-toolkit-1.24.0/bin/tls-toolkit.sh standalone \
  -C "CN=nifi, O=MJ Industries, OU=NIFI, DC=nifi, DC=mj, DC=com" \
  -K "<CERTIFICATE_KEY>" \
  -S "<CERTIFICATE_KEYSTORE_PASSWORD>" \
  -P "<CERTIFICATE_TRUSTSTORE_PASSWORD>" \
  -o ./certificate \
  -d 1825 \
  -c "mj.xyz" \
  -n "<HOSTNAME>" \
  -O
```

This produced a new folder in the "**certificate**" directory for the given host with new `keystore.jks` and `truststore.jks` files. In addition it created a default `nifi.properties` file with properties to use the new certificate and stores that come with it. Make sure to place them in proper locations and make sure to restart NiFi instances (if running) to use the certificates.

An extra tip:
The system administrator should also use the toolkit to encrypt NiFi files to avoid having passwords in plain text on the server.

1.4.1 On TLS Toolkit and NiFi 2.0

With release of NiFi 2.0, NiFi Toolkit will no longer have the `tls-toolkit` and so it requires other ways of generating a valid certificate. NiFi recommends to use other tools to create keystores and truststores, such as cert-mananger for Kubernetes, Let's Encrypt, TinyCert, and others. Read more about it in NiFi 2.0 Documentation under the "**Securing NiFi with mTLS**" section. The same concepts still apply with those certificates, they need to have proper trust chains to enable communication with TLS between instances.

If for some reason none of these options are available or desired, it's possible to generate the needed certificates manually. The NiFi 2.0 Documentation describes how to do it in the "**Manual Keystore Generation**" section. It requires the use of OpenSSL and information such as host's IP address and DNS. This guide won't go into details on manual certificate generation as NiFi docs describe it well and will have more up to date information about that. As before make sure that the certificates have proper trust chains and other values so that instances using them can trust each other.

2 Setting Up the Main Instance

2.1 Set Up an SSL Context Service

First, set up a `StandardRestrictedSSLContextService`, even if both instances run on the same server and use the same certificate. I noticed that this helps with making sure that there are no certificate issues between the instances. This also ensures that if they use different certificates they are able to build a PKIX trust path between them.

To create new service open the hamburger menu in top right corner. Once there open the "**Controller Settings**" menu.

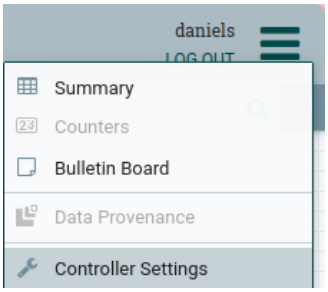

Figure 1. Opening the Controller Settings menu

1. Go to the "**Management Controller Services**" tab, add new `StandardRestrictedSSLContextService`
2. Open configuration menu for `StandardRestrictedSSLContextService` by clicking the "cog" icon
3. **Keystore Filename:** Path to "keystore.jks" file for main instance certificate

4. **Keystore Type:** JKS
5. **Truststore Filename:** Path to “truststore.jks” file for main instance certificate
6. **Truststore Type:** JKS
7. **Key Password:** Certificate Key Password for main instance certificate
8. **Keystore Password:** Keystore Password for main instance certificate
9. **Truststore Password:** Truststore Password for main instance certificate

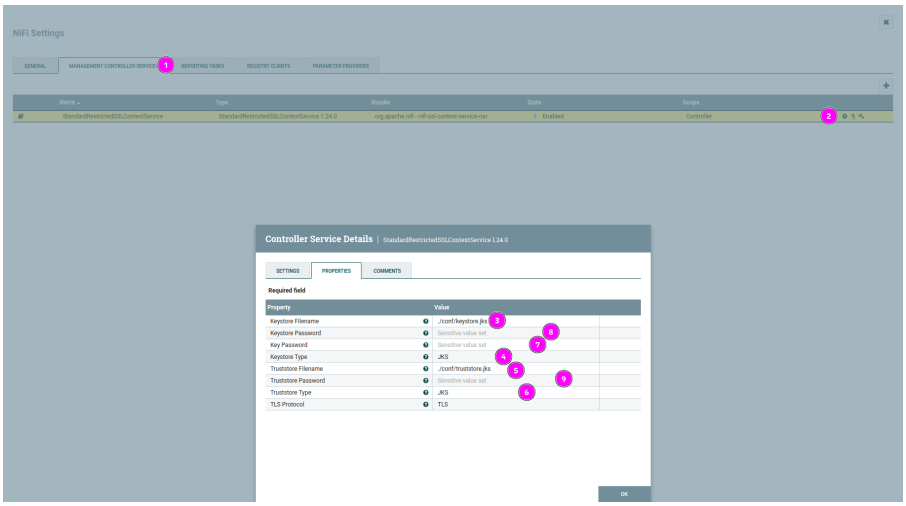

Figure 2. Configuring the SSL Context Service

After closing the configuration menu let the service validate and then enable it by clicking the “lightning” icon.

2.2 Set Up the S2S Reporting Task

With the SSL Service in place, set up the S2S Reporting task. This guide will focus on Provenance Reporting Task, but this should work for other tasks too. Open the hamburger menu as before and go to the “**Controller Settings**” menu.

1. Go to the “**Reporting Tasks**” tab, add new `SiteToSiteProvenanceReportingTask`
2. Open configuration menu for `SiteToSiteProvencanceReportingTask` by clicking the “pencil” icon
3. **Destination URL:** URL to the reporting instance GUI: `https://reporting.mj.xyz:9443/nifi`
4. **Input Port Name:** Name of port on your reporting instance, for example: `provenance-events`
5. **SSL Context Service:** The `StandardRestrictedSSLContextService` created in 2.1
6. **Instance URL:** Change URL so it matches your main instance port and change the “http” to “https”
7. **Batch Size:** Adjust the size as needed by your system, might be smaller or bigger
8. **Transport Protocol:** Set it to HTTP, now RAW
9. **Run Schedule:** Change it to appropriate value, it’s found in the “**Settings**” tab

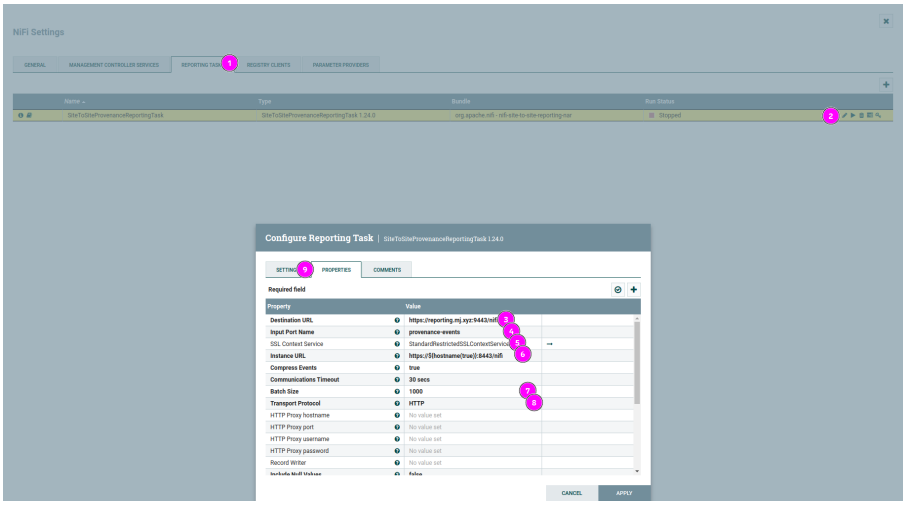

Figure 3. Configuring S2S Reporting Task

In case of Provenance reporting it's good idea to set filter by event type or source. Do not start the Reporting Task until the other instance set up and ready to receive the data over S2S.

2.3 Creating Events

In order to test S2S Provenance Reporting Task it needs some events to send. Simple solution is to just place a `GenerateFlowFile` processor, connect it to a funnel and create some FlowFiles when needed.

3 Setting Up the Reporting Instance

3.1 Create the Input Port

Start by creating a Process Group giving it appropriate name and going into it.

1. Create a Remote Input Port inside the group, to do it drag the "**Input Port**" icon onto the canvas
2. Give it the same name as the "**Input Port Name**" attribute in the Provenance Reporting Task
3. From the drop-down menu "**Receive From**" choose: "**Remote connections (site-to-site)**"

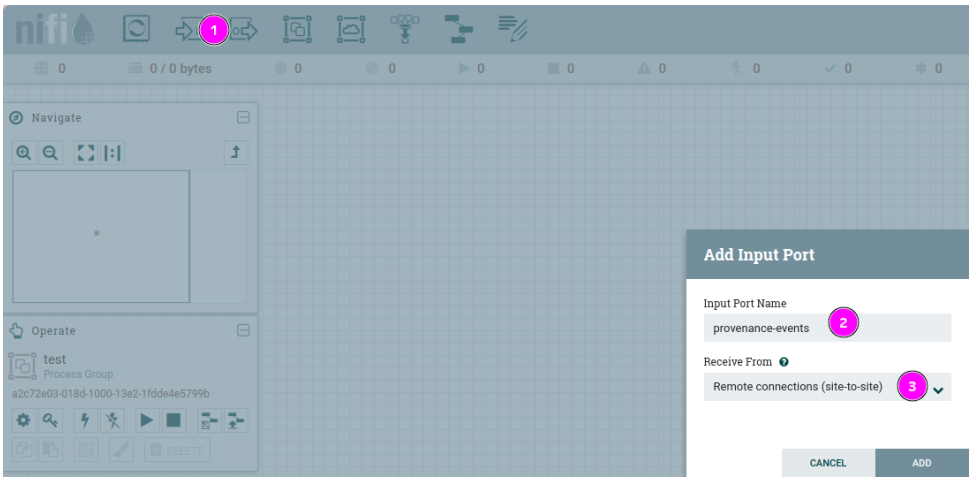

Figure 4. Creating and configuring Remote Input Port

3.2 Create an Instance User

In order to authenticate and authorize a NiFi instance that wants to connect to reporting instance it needs a user. This user will then get appropriate policies to enable it to use S2S communication. By utilizing users several instances can connect to one "hub" instance, but only be able to send data to their own ports or shared

ports. This user's name needs to be the same as the DN from the certificate that the connecting NiFi instance uses. Unless the "**Identity Mapping Properties**" in `conf/nifi.properties` exist, then use the mapped name.

If using certificate as the one generated in 1.4 then the DN is: `CN=nifi.mj.xyz, OU=NIFI`. The DN in certificates that NiFi toolkit generates use the form: `CN=<HOSTNAME>, OU=NIFI`.

To create the user simply open the hamburger menu from the top right corner and open the users menu.

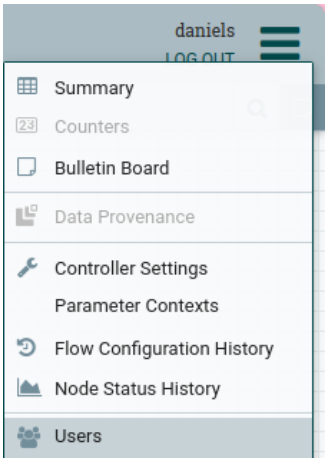

Figure 5. Opening the Users menu

Then just click on the plus icon and add a new user with the DN as username. If there are issues with finding the name or there is uncertainty as to what the certificate's DN is then that info can be easily found in logs. The log file: `logs/nifi-user.log` of the reporting instance will log the names of instances trying to connect with it over S2S. Look for "**Authentication Started**" events for `/site-to-site` endpoint in the logs. Before finding that information the user can use a temporary username and then have it changed to proper name when it's found.

```
Authentication Started 127.0.0.1 [CN=nifi.mj.xyz, OU=NIFI]
  GET https://reporting.mj.xyz:9443/nifi-api/site-to-site/peers
```

3.3 Set Instance User's Policies

After creating the Instance User give it the following Global Policy to "**retrieve site-to-site details**". To do this open the hamburger menu in top right corner, select "**Policies**" menu.

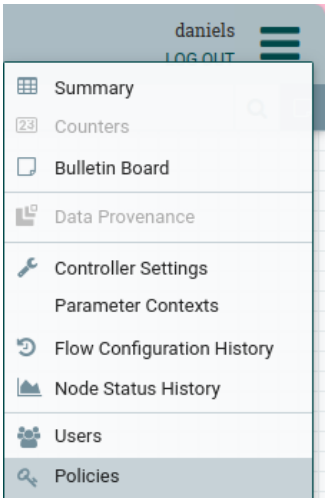

Figure 6. Opening the Policies menu

Find aforementioned policy in the drop-down menu. If it wasn't set before NiFi will ask to create that policy, it will show up above the drop-down menu.

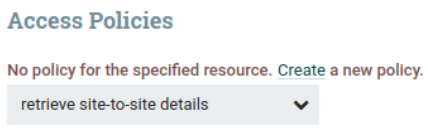

Figure 7. Creating new policy

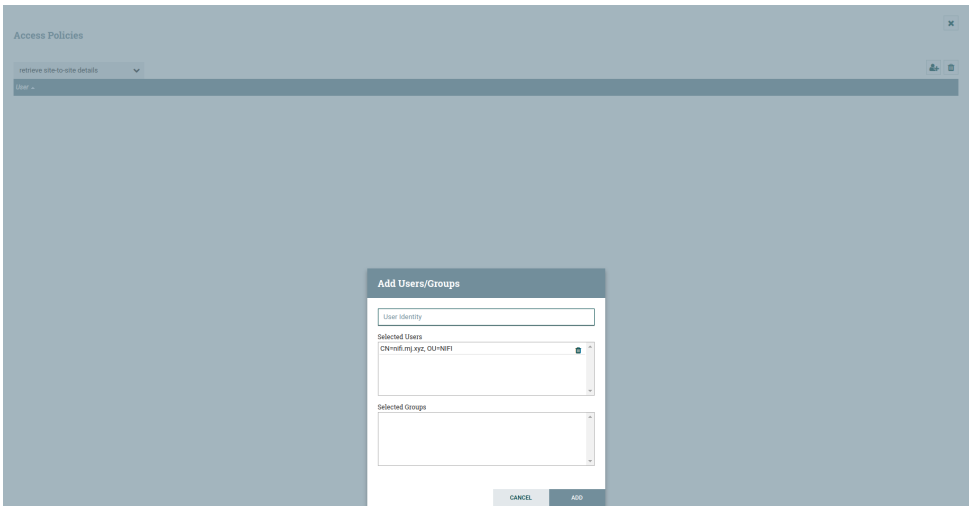

Figure 8. Adding the instance user to "retrieve site-to-site details" policy

Now find the Input Port from 3.1 and right click on it. Select "**Manage access policies**", under "**receive data via site-to-site**" create new policy as before, and add the instance user to it.

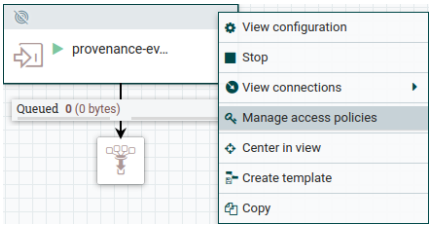

Figure 9. Opening Input Port's policies

3.4 Basic Flow

Add some basic flow connected to the input port, for testing it might even be a simple funnel or a `LogAttribute` processor. Start the Input Port and then go back to the main instance to enable the Reporting Task. This is a good moment to ensure that the Instance User has correct username, check the user log on reporting instance as mentioned in 3.2.

4 Troubleshooting

The main problems I have encountered in setting up the S2S communication between two secured NiFi instances relate to users, configuration, and certificates. Here I will try to showcase how these errors looked like for me so it's easier to identify the error and how to fix it. Some error messages might be confusing or misleading, it's a good idea to have a habit of checking the logs for extra information as they often include stack traces. This additional information helped me fix all the errors I have encountered.

4.1 Certificate Issues

The most tricky issue to figure out the reason for it and fix was the certificate issue. The error message you get is not explaining the issue that well unless you know the topic well. In addition knowing what the issue is doesn't help with fixing it, it took a lot of trial an error to figure out how to set up the certificates. Most important are the certificate issuer and DN, if they are correct then the certificates should work. In addition a SSL Context Service helps with ensuring that the communication proceeds as expected.

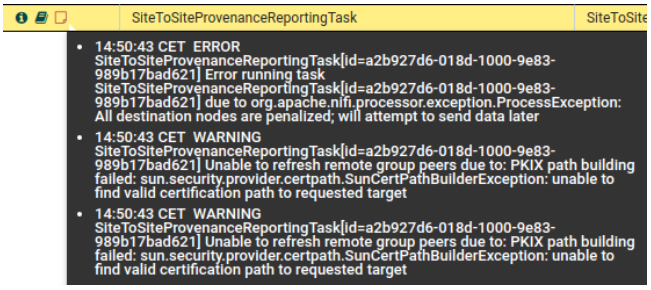

Figure 10. Error messages given by S2S Reporting Task which has certificate issues

If you look in the logs you can find more information about error (shortened here for brevity):

```
o.apache.nifi.remote.client.PeerSelector
  Could not communicate with reporting.mj.xyz:9443 to determine which node(s) exist
  in the remote NiFi instance, due to
    javax.net.ssl.SSLHandshakeException: PKIX path building failed:
      sun.security.provider.certpath.SunCertPathBuilderException:
        unable to find valid certification path to requested target

o.apache.nifi.remote.client.PeerSelector Unable to refresh remote group peers due to:
  Unable to retrieve nodes from remote instance

o.a.n.r.SiteToSiteProvenanceReportingTask SiteToSiteProvenanceReportingTask[id=...]
  Unable to refresh remote group peers due to:
    Unable to retrieve nodes from remote instance

o.a.n.r.util.SiteToSiteRestApiClient Failed to create transaction for
  https://reporting.mj.xyz:9443/nifi-api/data-transfer/input-ports/.../transactions
  javax.net.ssl.SSLHandshakeException:
    PKIX path building failed:
      sun.security.provider.certpath.SunCertPathBuilderException:
        unable to find valid certification path to requested target
```

If you experience this type of error make sure that your Reporting Task uses a `StandardRestrictedSSLContextService` with it's instance certificate. Ensure that the certificates on both instances use the same DN and issuer. For more information see 1.4.

4.2 Missing Instance User

Another issue occurs when the reporting instance doesn't have a user created for the instance that wants to connect through S2S. If user exists but uses other name than the one in certificate it will also give same problem. From what I see the error messages in this case can differ from time to time as some other factor might affect what error message NiFi gives. The error I got most often was similar to this one:

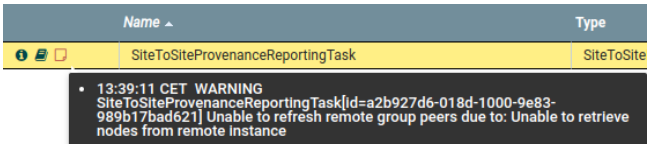

Figure 11. Error messages given by S2S Reporting Task when instance user is missing on reporting instance

Looking at `nifi-user.log` in the reporting instance should yield a message like this one here:

```
Authentication Started 127.0.0.1 [CN=nifi.mj.xyz, OU=NIFI]
  GET https://reporting.mj.xyz:9443/nifi-api/site-to-site/peers
Authentication Success [CN=nifi.mj.xyz, OU=NIFI] 127.0.0.1
  GET https://reporting.mj.xyz:9443/nifi-api/site-to-site/peers
identity[CN=nifi.mj.xyz, OU=NIFI] does not have permission to access the requested resource.
  Unable to view site-to-site details.
    Returning Forbidden response.
```

To fix that find the proper username and add a user with that name to the user list, then give it proper policies as outlined in 3.3. As seen above these messages log what username the connecting instance uses, so make sure to use the same name, in this case: `CN=nifi.mj.xyz, OU=NIFI`. In case it's not clear, here is how to add a new user:

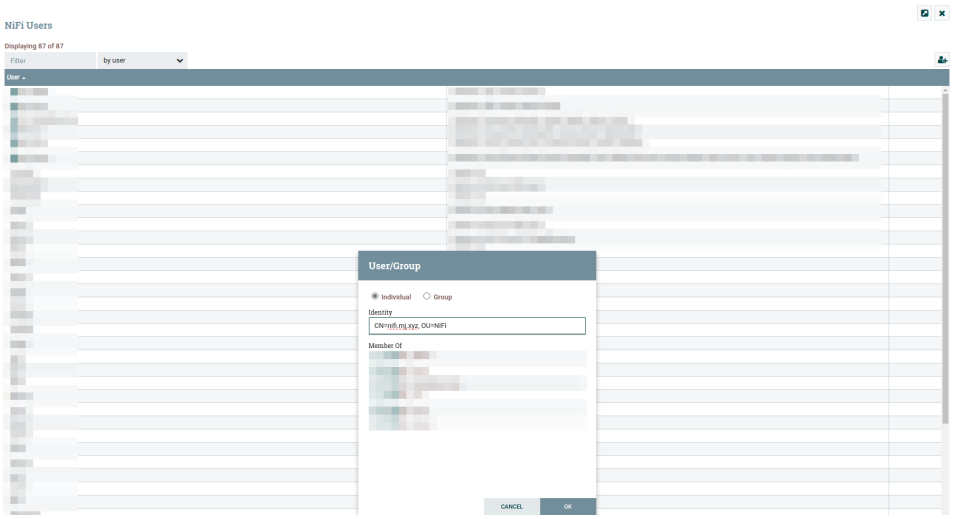

Figure 12. Adding new instance user in the user menu

4.3 Missing Policy Configuration

If the user exists but it does not have the Global Policy to "**retrieve site-to-site details**" NiFi will give one type of error message. On the other hand if the user has the global policy but is missing the "**receive data via site-to-site**" policy on Remote Input Port, another error message should come up.

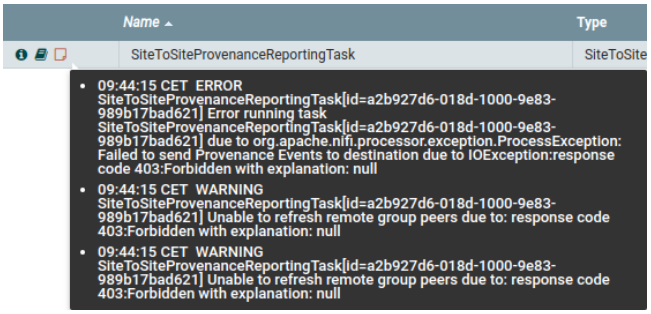

Figure 13. Error messages given by S2S Reporting Task when instance user doesn't have "**retrieve site-to-site details**" policy

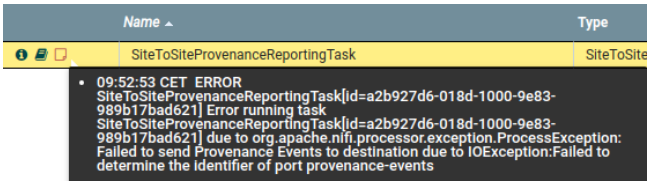

Figure 14. Error messages given by S2S Reporting Task when instance user doesn't have "**receive data via site-to-site**" policy on Remote Input Port

In case of missing global policy the user log on reporting instance will show the same error log as shown in 4.2. In case of missing policy on port following message should show up in `nifi-user.log` instead. (IDs were removed for brevity, message was also re-formatted to fit the document better.)

```
Authentication Started 127.0.0.1 [CN=nifi.mj.xyz, OU=NIFI]
  POST https://reporting.mj.xyz:9443/nifi-api/data-transfer/input-ports/.../transactions
Authentication Success [CN=nifi.mj.xyz, OU=NIFI] 127.0.0.1
  POST https://reporting.mj.xyz:9443/nifi-api/data-transfer/input-ports/.../transactions
identity[CN=nifi.mj.xyz, OU=NIFI] does not have permission to access the requested resource.
```

```
StandardPublicPort[...] authorization failed for user CN=nifi.mj.xyz, OU=NIFI
  because Unable to modify data transfers to Input Port with ID: ...
   Returning Forbidden response.
```

To fix those issues give the instance user proper policies as outlined in 3.3.