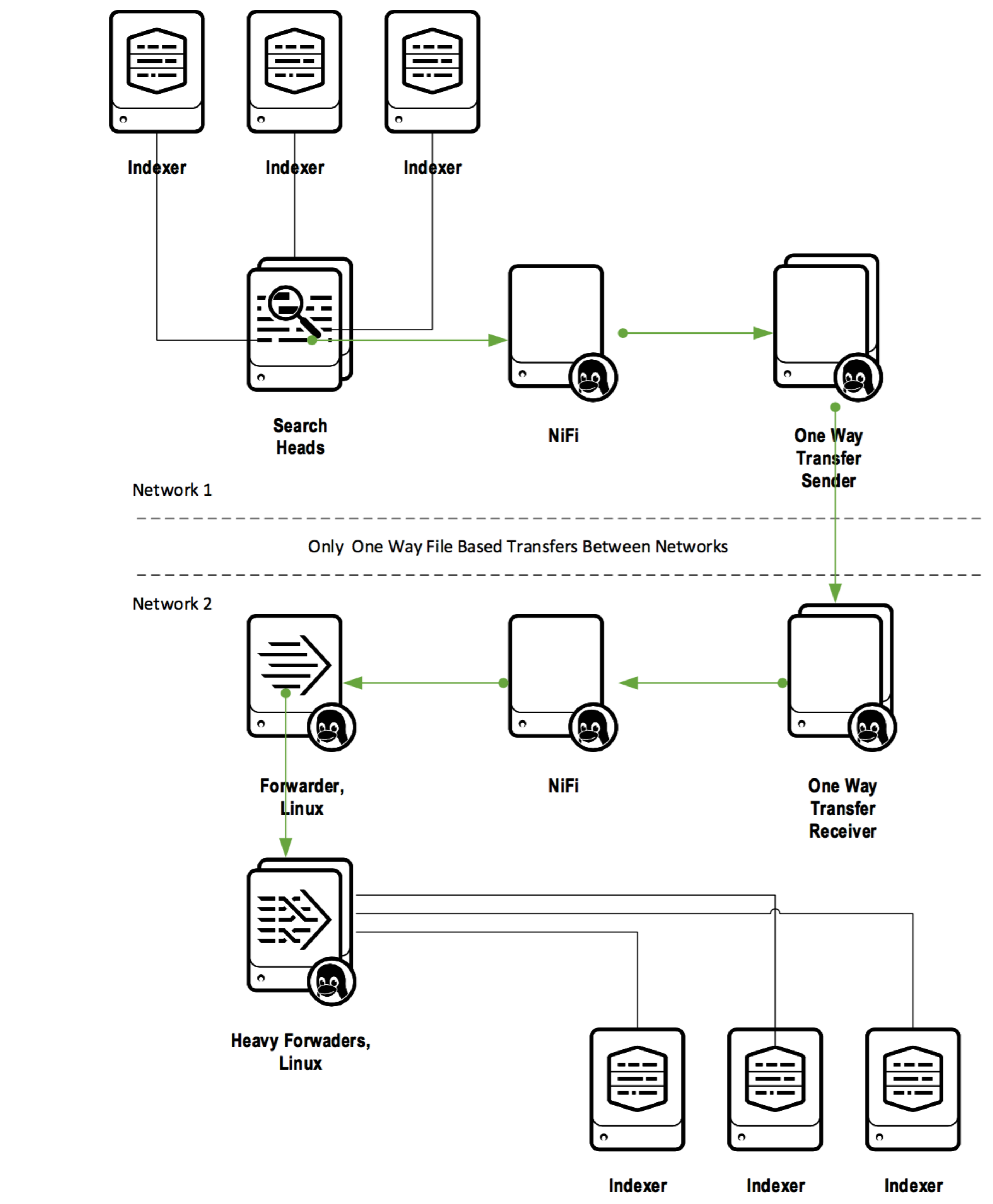
# Topic

Delivering Splunk Indexed data through a One Way File Based Transfer at high volume.

This capability is highly sought after by many Government and Military customers that have to maintain a strong separation of networks due to differing levels of classification. The techniques presented show how to transfer data from one Splunk Infrastructure to another while maintaining the integrity of the source index, host, event time and raw event data. This method offers a streaming solution over a file based one way transfer with only slight delay between origin receipt and destination receipt. The general approach is to use NiFi on the source network to query a search head using a generic table format for output. The tabularized results are converted into files by NiFi and compressed prior to transit through a One Way Transfer. Once the files arrive on the destination One Way Transfer, another instance of NiFi on the destination network retrieves, unpacks the compressed files and deposits them on a universal forwarder. This forwarder is load balanced to a bank of heavy forwarders with a custom developed app that uses the tabularized data to re-assemble the events in the proper index while parsing out the host, source, sourcetype, and \_raw values. This method has been tested with sysmon, wineventlog, and winevenlogsecurity sourced data an shown to preserve the look and feel of the event from the source system. When the TA-Windows was used on the destination search head, the field parsing works identically to the source system. This is huge, since the value of TAs and addons is preserved between networks. We are currently moving about 300 GB/Day between networks with this method and still have capacity left over. Figure 1 on the next page shows the basic data flow and architectural components used in this solution.

Figure 1 Flow and General Architecture



Next, I’m going to provide a walkthrough of the steps necessary to setup a working flow for one sample source.

1. Configure NiFi source collector

First download and install NiFi version 1.1.2

Once NiFi is up, open the URL and configure the following 4 connectors in NiFi (GetSplunk, MergeContent, UpdateAttribute, and PutSFTP)

* Drag and drop a connector onto the panel and select GetSplunk
* Right click and configure as follows:

Time strategy:  use Managed from current

Time Field Strategy: use Index Time

Output Mode: use CSV

Query: search index=main|eval new\_raw="~~~index:".index."~~~host:".host."~~~source:".source."~~~sourcetype:".sourcetype."~~~\_time:".\_time."~~~\_raw:".\_raw|table new\_raw

* Drag and drop a connector onto the panel and select MergeContent
* Right click and configure as follows:

Merge Strategy: Bin-Packing Algorithm

Merge Format: ZIP

Minimum number of entries: 100

Maximum number of entries: 200

minimum group size: 102400 B

* Connect the GetSplunk connector to the MergeContent connector.
* Drag and drop a connector onto the panel and select

UpdateAttribute

* Right click and configure as follows:

filename: main.${filename}.txt

The format for flows is

<index name>.${filename}.txt

substitute the name of the index being sent.

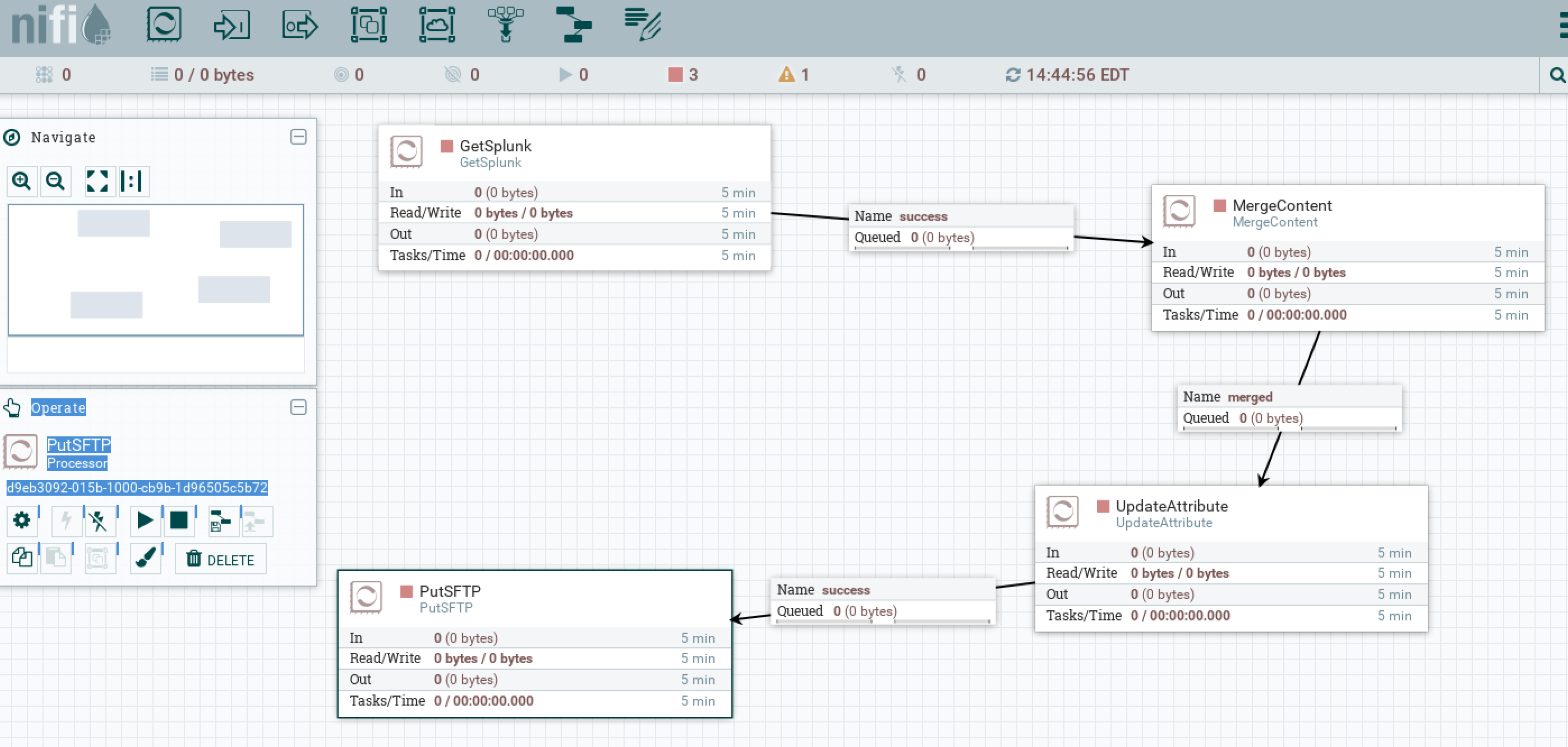
* Connect the MergeContent and UpdateAttribute connector.
* Drag and drop a connector onto the panel and select putSFTP
* Right click and configure the connector as follows:

Hostname: Hostname of destination (One Way Transfer)

Username: UID of user with credentials on destination

Destination: folder: path of folder on the One Way Transfer host to deposit files.

Figure 2 NiFi source collector configuration



1. Configure NiFi destination processor

* First download and install NiFi version 1.1.2
* Open the NiFi URL and configure 3 connectors
* Drag and drop a connector onto the panel and select GetSFTP
* Right click and configure as follows:

Hostname: hostname of destination One Way Transfer

Username: username with credentials to retrieve from One Way Transfer and delete files.

Remote path: file path on One Way Transfer where destination files land.

File filter regex: main.\d+.zip.txt (format is <indexname>.\d+.zip.txt)

* Drag and drop a connector onto the panel and select UnpackContent
* Right click and configure as follows:

Packaging Format: zip

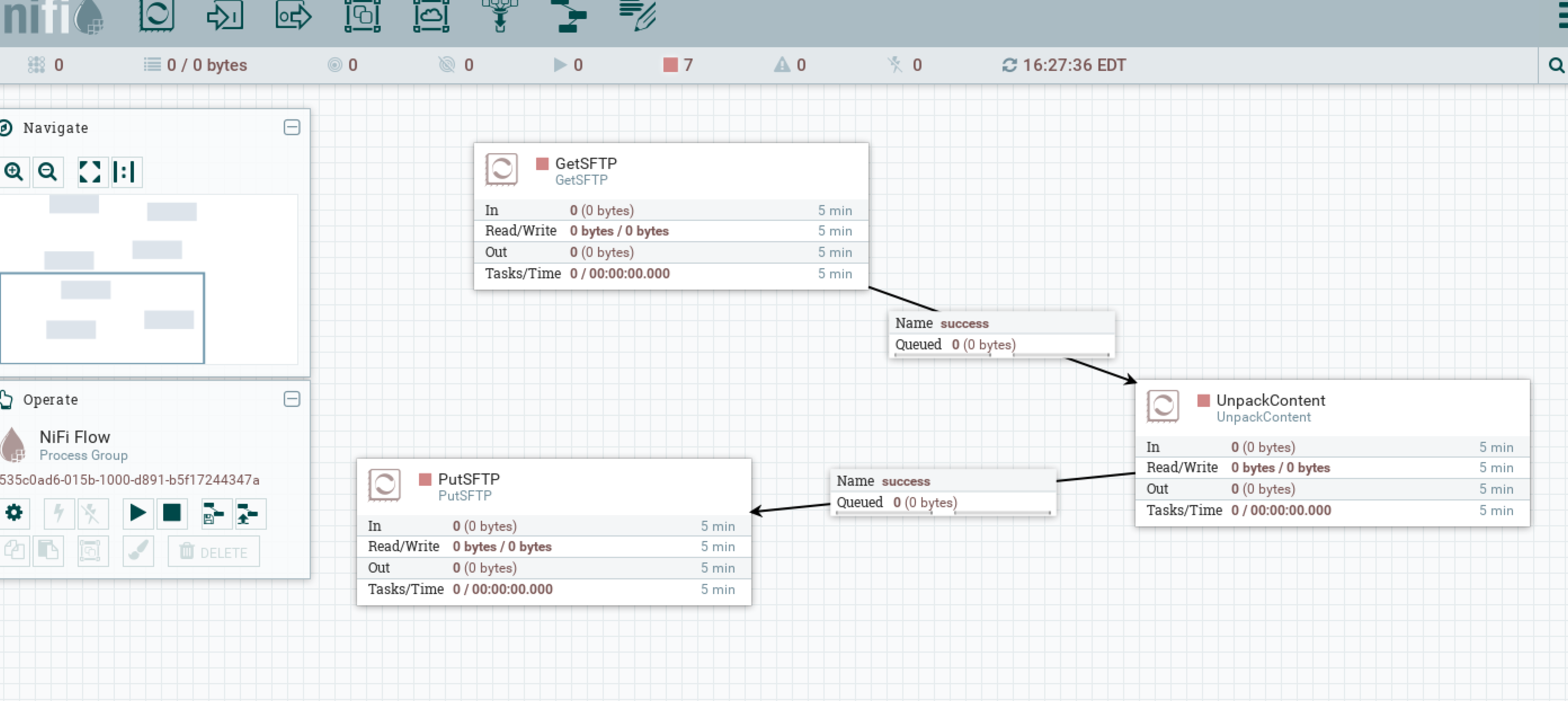
* Drag and drop a connector onto the panel and select putSFTP
* Right click and configure as follows:

Hostname: hostname of the Universal Forwarder

Username: name of user with credentials on Universal Forwarder

Remote path: path on Universal Forwarder where monitor input is setup.

Figure 3. NiFi destination collector configuration



1. Configure Universal Forwarders and Heavy Forwarders on destination environment.

* Install the latest version of the Splunk UF on a linux host.
* Setup the following monitor stanza in inputs.conf on the universal forwarder and restart.

[monitor://data/owt\_main]

crcSalt = <SOURCE>

sourcetype = nifi

Setup outputs.conf to point to a heavy forwarder

* Install the latest version of Splunk on a linux host and setup as a Heavy

Forwarder. Setup outputs to point to an indexer or bank of indexers.

* Deploy the nifi\_parser app to the heavy forwarder and restart.

The app contains a props.conf and transforms.conf which parse the incoming NiFi stream and handle setting source, sourcetype, host, \_time and index. The contents of the app are shown below:

props.conf

[nifi]

LINE\_BREAKER=([\r\n+])\”~~~index:

SHOULD\_LINEMERGE=false

TRUNCATE=0

NO\_BINARY\_CHECK=true

TIME\_PREFIX=~~~\_time:

TRANSFORMS-sethost=a\_sethost\_nifi

TRANSFORMS-setsource=b\_setsource\_nifi

TRANSFORMS-setsourcetype=c\_setsourcetype\_nifi

TRANSFORMS-setindex=d\_setindex\_nifi

TRANSFORMS-setraw=x\_setxraw\_nifi

transforms.conf

[a\_sethost\_nifi]

DEST\_KEY = MetaData:Host

REGEX = ~~~host:([^\~]+)

FORMAT= host::$1

[b\_setsource\_nifi]

DEST\_KEY = MetaData:Source

REGEX = ~~~source:([^\~]+)

FORMAT = source::$1

[c\_setsourcetype\_nifi]

DEST\_KEY = MetaData:Sourcetype

REGEX = ~~~sourcetype:([^\~]+)

FORMAT = sourcetype::$1

[d\_setindex\_nifi]

DEST\_KEY = \_MetaData:Index

REGEX = ~~~index:([^\~]+)

FORMAT = $1

[x\_setraw\_nifi]

DEST\_KEY = \_raw

REGEX = (?s)~~~source:([\~]+)~~~sourcetype:([^\~]+).\*~~~\_raw:(.\*)$

FORMAT = $3~~~source:$1~~~sourcetype:$2~~~

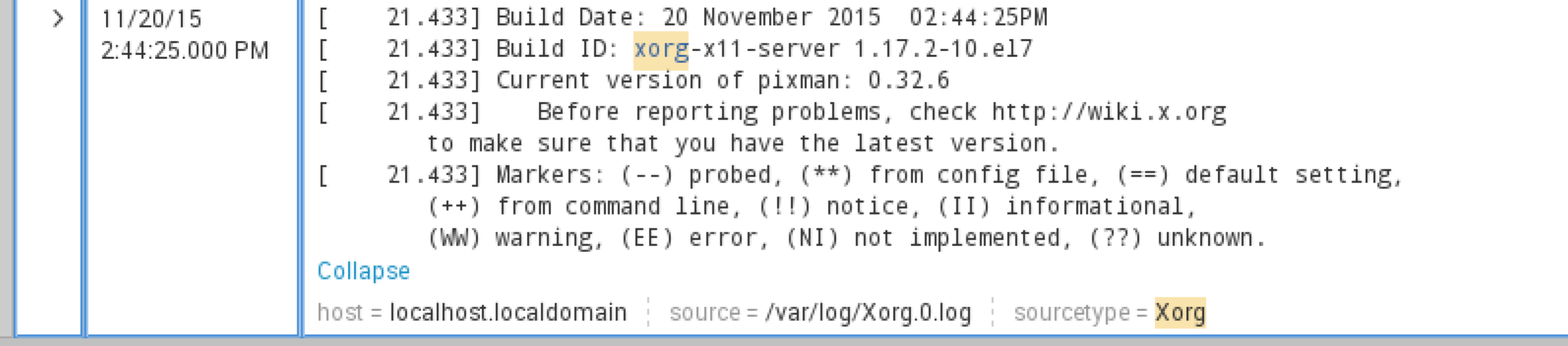
#had add the source and sourcetype data to \_raw. Apparently \_raw is parsed prior to the setting of source and sourcetype regardless of the ascii order of transforms. If not added, source and sourcetype transforms fail. This puts a little extra onto the raw event at the end, but has had not negative impact on field extraction of windows events.

This is a sample event that the base query on the source side will create. This is a TA-nix collected event.

The event will be routed to the OS index, host will be set to localhost.localdomain, source will be set to /var/log/Xorg.0.log, sourcetype will be set to Xorg and \_time will be set from the epoch value. The \_raw field will be rebuilt from the data after \_raw. Carriage returns and linefeeds within \_raw are preserved.

~~~index:os~~~host:localhost.localdomain~~~source:/var/log/Xorg.0.log~~~sourcetype:Xorg~~~\_time:1448048665~~~\_raw:[ 15.088] Build Date: 20 November 2015 02:44:25PM [ 15.088] Build ID: xorg-x11-server 1.17.2-10.el7 [ 15.088] Current version of pixman: 0.32.6 [ 15.088] Before reporting problems, check http://wiki.x.org to make sure that you have the latest version. [ 15.088] Markers: (--) probed, (\*\*) from config file, (==) default setting, (++) from command line, (!!) notice, (II) informational, (WW) warning, (EE) error, (NI) not implemented, (??) unknown.

The source event looks like this…



The destination event looks like this…

