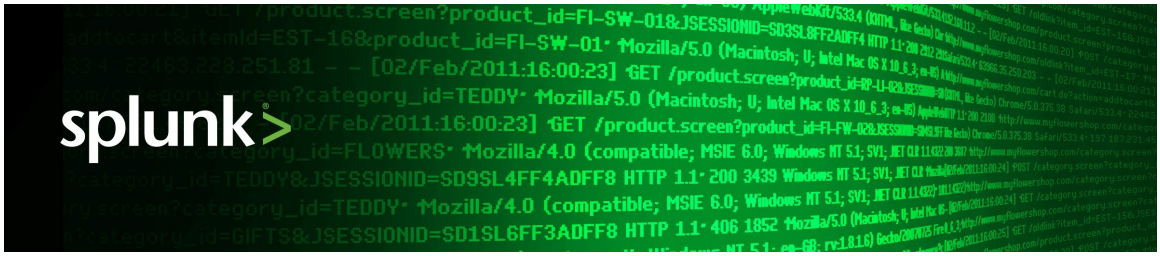

Splunk® Universal Forwarder Forwarder Manual 6.5.0

Protect against loss of in-flight data

Generated: 10/19/2016 9:56 pm

Protect against loss of in-flight data

To guard against loss of data when **forwarding** to an **indexer**, you can use the **indexer acknowledgment** capability. With indexer acknowledgment, the **forwarder** will resend any data not acknowledged as "received" by the indexer.

You enable indexer acknowledgment on the forwarder in `outputs.conf`. By default, the feature is not active.

Indexer acknowledgment and indexer clusters

When you use forwarders to send data to peer nodes in an indexer cluster, you should enable indexer acknowledgment. To learn more about forwarders and clusters, see Use forwarders to get your data in the Splunk Enterprise *Managing Indexers and Clusters of Indexers* manual.

How indexer acknowledgment works in normal operation

The forwarder sends data continuously to the indexer, in blocks of approximately 64kB. The forwarder maintains a copy of each block in memory, in its wait queue, until it gets an acknowledgment from the indexer. While waiting, it continues to send more data blocks.

During normal operation, the indexer:

- Receives a block of data.
- Parses the data.
- Writes the data to the file system as events (raw data and index data).
- Sends an acknowledgment to the forwarder.

The acknowledgment tells the forwarder that the indexer received the data and successfully wrote it to the file system. Upon receiving the acknowledgment, the forwarder releases the block from memory.

If the wait queue is of sufficient size, it does not fill up while waiting for acknowledgments to arrive. It can, however, fill up quickly if network or hardware conditions prevent acknowledgments from getting back to the forwarder. See Indexer acknowledgment and forwarded data throughput for issues and ways to address them.

How indexer acknowledgment works when a failure occurs

When there is a failure in the round-trip process, the forwarder does not receive

an acknowledgment. At this point it attempts to resend the block of data.

Reasons for lack of indexer acknowledgment

These are the reasons that a forwarder might not receive acknowledgment:

- Indexer goes down after receiving the data, possibly due to a hardware failure.
- Indexer is unable to write to the file system, possibly because the disk is full.
- Network goes down while acknowledgment is en route to the forwarder.

How the forwarder handles lack of indexer acknowledgments

After the forwarder sends a data block, it maintains a copy of the data in its wait queue until it receives an acknowledgment. Meanwhile, it continues to send additional blocks. If the forwarder doesn't get acknowledgment for a block within 300 seconds (by default), it closes the connection.

If the forwarder is in an **auto-load-balancing** configuration, it then opens a connection to the next indexer in the group (if one is available) and sends the data to it. If the forwarder is not set up for auto-load balancing, it attempts to open a connection to the same indexer as before and resend the data.

The forwarder maintains the data block in the wait queue until acknowledgment is received. After the wait queue fills up, the forwarder stops sending additional blocks until it receives an acknowledgment for one of the blocks, at which point it can free up space in the queue.

You can change the wait time by setting the `readTimeout` attribute in `outputs.conf`.

Other reasons the forwarder might close a connection to the indexer

There are three conditions that can cause the forwarder to close the network connection:

- *Read timeout.* The forwarder doesn't receive acknowledgment within 300 (default) seconds. This is the condition described above.
- *Write timeout.* The forwarder is not able to finish a network write within 300 (default) seconds. The value is configurable in `outputs.conf` by setting `writeTimeout`.
- *Read/write failure.* This condition usually happens because the indexer has crashed or the network has failed.

In all these cases, the forwarder attempts to open a connection to the next indexer in the load-balanced group, or to the same indexer again if load-balancing is not enabled.

How lack of indexer acknowledgment can cause the duplication of indexed data

It is possible for the indexer to index the same data block twice. This can happen if there is a network problem that prevents an acknowledgment from reaching the forwarder. For instance, assume the indexer receives a data block, parses it, and writes it to the file system. It then generates the acknowledgment. However, on the round-trip to the forwarder, the network goes down, so the forwarder never receives the acknowledgment. When the network comes back up, the forwarder then resends the data block, which the indexer parses and writes as if it were new data.

To deal with such a possibility, every time the forwarder resends a data block, it writes an event to its `splunkd.log` noting that it's a possible duplicate. The admin is responsible for using the log information to track down the duplicate data on the indexer.

Here's an example of a duplicate warning:

```
10-18-2010 17:32:36.941 WARN  TcpOutputProc - Possible duplication of
events with
channel=source::/home/jkerai/splunk/current-install/etc/apps/sample_app
/logs/maillog.1|host::MrT|sendmail|, streamId=5941229245963076846,
offset=131072
subOffset=219 on host=10.1.42.2:9992
```

Enable indexer acknowledgment

Configure indexer acknowledgment on the forwarder.

1. Set the `useACK` attribute to `true` in `outputs.conf`

You can set `useACK` either globally or by target group, at the `[tcpout]` or `[tcpout:<target_group>]` stanza levels. You cannot set it for individual receiving indexers at the `[tcpout-server: ...]` stanza level.

For more information, see the outputs.conf spec file.

```
[tcpout:<target_group>]
server=<server1>, <server2>, ...
```

`useACK=true`

Indexer acknowledgment and forwarded data throughput

The forwarder uses a wait queue to manage the indexer acknowledgment process. This queue has a default maximum size of 21MB, which is generally sufficient. In rare cases, you might need to manually adjust the wait queue size.

If you want more information about the wait queue, read this section. It describes how the wait queue functions as well as how to configure it.

How the wait queue size is configured

You do not set the wait queue size directly. Instead, you set the size of the in-memory output queue, and the wait queue sets to three times the output queue size. To configure the output queue size, use the `maxQueueSize` attribute in `outputs.conf`.

The default for the `maxQueueSize` attribute is `auto`. This is the recommended setting. It optimizes the queue sizes, based on whether indexer acknowledgment is active:

- When `useACK=true`, the output queue size is 7MB and the wait queue size is 21MB.
- When `useACK=false`, the output queue size is 500KB.

You can set `maxQueueSize` to specific values if necessary. See the outputs.conf spec file for further details on `maxQueueSize`.

Note: When you turn on indexer acknowledgment, the increase in queue size takes effect only after you restart the forwarder.

Why the wait queue matters

If you enable indexer acknowledgment, the forwarder uses a wait queue to manage the acknowledgment process. Because the forwarder sends data blocks continuously and does not wait for acknowledgment before sending the next block, its wait queue maintains many blocks, each waiting for its acknowledgment. The forwarder continues to send blocks until its wait queue is full, at which point forwarding stops. The forwarder then waits until it receives an acknowledgment, which lets it release a block from its queue and thus resume forwarding.

A wait queue can fill up when something is wrong with the network or indexer. It

can also fill up even when the indexer functions normally. This is because the indexer only sends the acknowledgment after it has written the data to the file system. Any delay in writing to the file system slows the pace of acknowledgment, which can lead to a full wait queue.

There are a few reasons that a normal functioning indexer might delay writing data to the file system (and thus delay sending acknowledgments):

- The indexer is very busy. For example, at the time the data arrives, the indexer might be dealing with multiple search requests or with data coming from a large number of forwarders.
- The indexer receives too little data.

For efficiency, an indexer only writes to the file system periodically -- either when a write queue fills up or after a timeout of a few seconds. If a write queue is slow to fill up, the indexer waits until the timeout to write. If data comes from only a few forwarders, the indexer can end up in the timeout condition, even if each of those forwarders sends a normal quantity of data. Since write queues exist on a per-hot-bucket basis, the condition occurs when a particular bucket gets only a small amount of data. Usually this means that a particular index gets only a small amount of data.

To ensure that throughput does not degrade because of a stalled forwarder, retain the default setting of `maxQueueSize=auto`. In rare cases, you might need to increase the wait queue size so that the forwarder has sufficient space to maintain all blocks in memory while waiting for acknowledgments to arrive. On the other hand, if you have many forwarders feeding a single indexer and a moderate number of data sources per forwarder, you might be able to conserve a few megabytes of memory by using a smaller size.

When the receiver is a forwarder

You can also use indexer acknowledgment when the data transmission occurs via an intermediate forwarder; that is, where an originating forwarder sends the data to an intermediate forwarder, which then forwards it to the indexer. For this scenario, if you want to use indexer acknowledgment, it is recommended that you enable it along all segments of the data transmission. That way, you can ensure that the data gets delivered along the entire path from originating forwarder to indexer.

Assume you have an originating forwarder that sends data to an intermediate forwarder, which in turn forwards that data to an indexer. To enable indexer acknowledgment along the entire line of transmission, you must enable it twice: first for the segment between originating forwarder and intermediate forwarder,

and again for the segment between intermediate forwarder and indexer.

If you enable both segments of the transmission, the intermediate forwarder waits until it receives acknowledgment from the indexer and then it sends acknowledgment back to the originating forwarder.

However, if you enable just one of the segments, you only get indexer acknowledgment over that part of the transmission. For example, say indexer acknowledgment is enabled for the segment from originating forwarder to intermediate forwarder but not for the segment from intermediate forwarder to indexer. In this case, the intermediate forwarder sends acknowledgment back to the originating forwarder as soon as it sends the data on to the indexer. It then relies on TCP to safely deliver the data to the indexer. Because indexer acknowledgment is not enabled for this second segment, the intermediate forwarder cannot verify delivery of the data to the indexer. This second case has limited value and is not recommended.