# Viz One and SwiftStack
# How-To Guide

**Viz One + SwiftStack**
How-To Guide

---

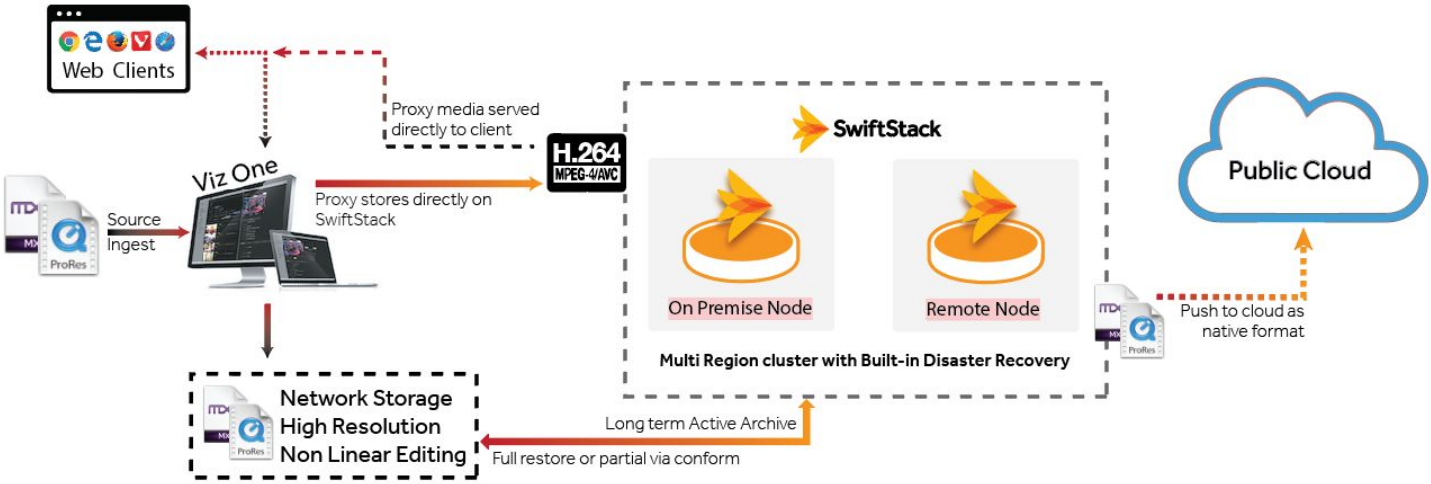
# Introduction and Summary

From small one-man edit shops all the way to enterprise level media giants, the benefits of a Media Asset Management (MAM) solution is undeniable. Viz One, one of the leading MAM solutions from Vizrt, is expanding these benefits by integrating with SwiftStack, a high availability and highly scalable storage solution with multi-region functionality and built-in disaster recovery.

When high-resolution media is ingested into Viz One, it creates web playable proxy media and stores it into SwiftStack which serves any web video play request by any web client across the globe. Viz One can also move its high resolution media from its finite local file system or SAN storage, complete with its custom metadata mapped to the object metadata header. Any data in SwiftStack can then be automatically replicated, based on policy, across multiple geographic regions or the public cloud

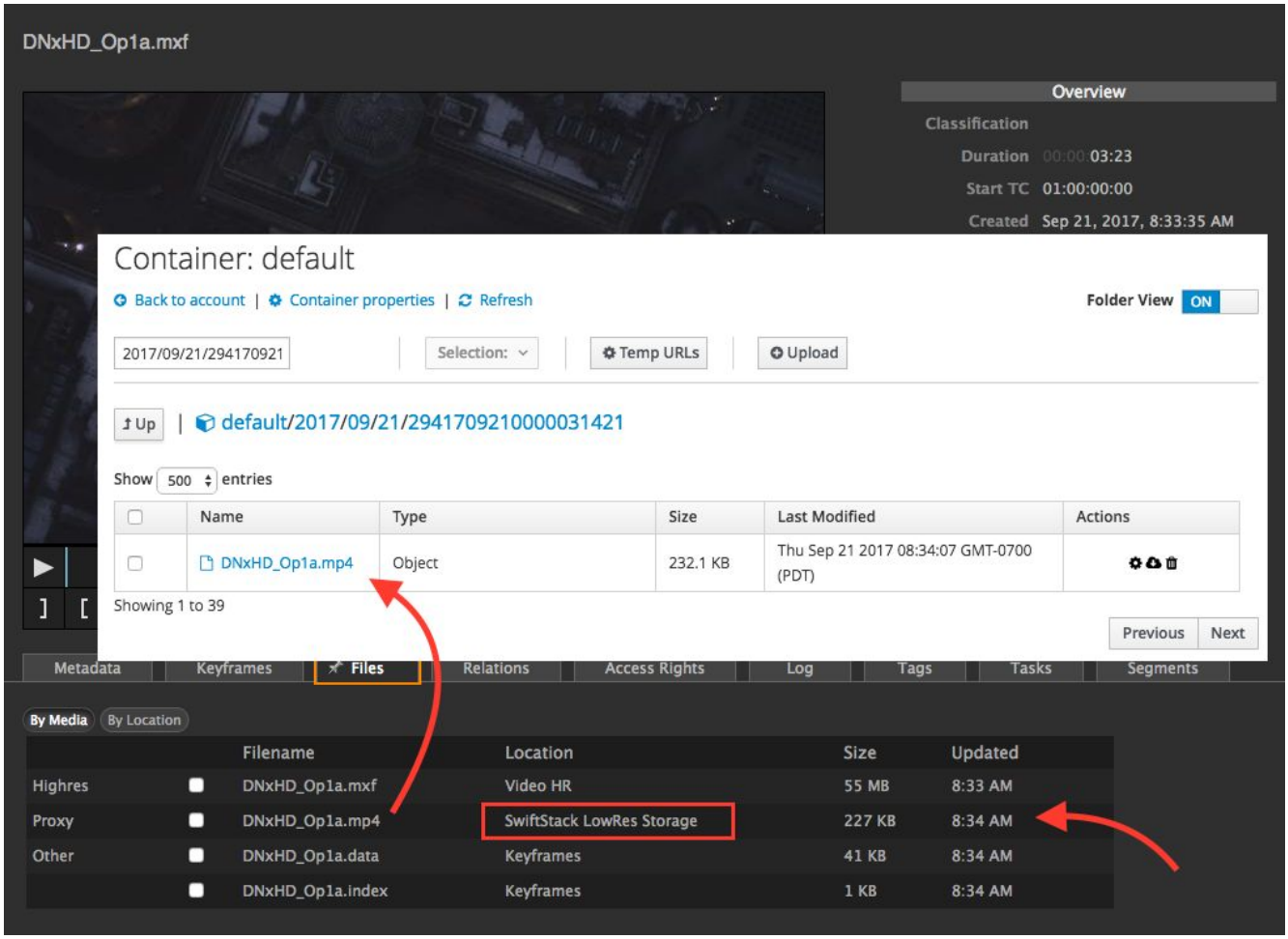
High-resolution media archived on SwiftStack via Viz One can easily be restored into your bandwidth optimized storage for NLE usage. Using only a few clicks within Viz One, you can restore media in its entirety, or selected sections of the large assets via conform to efficiently use NLE storage space.

# Current Integration Capabilities



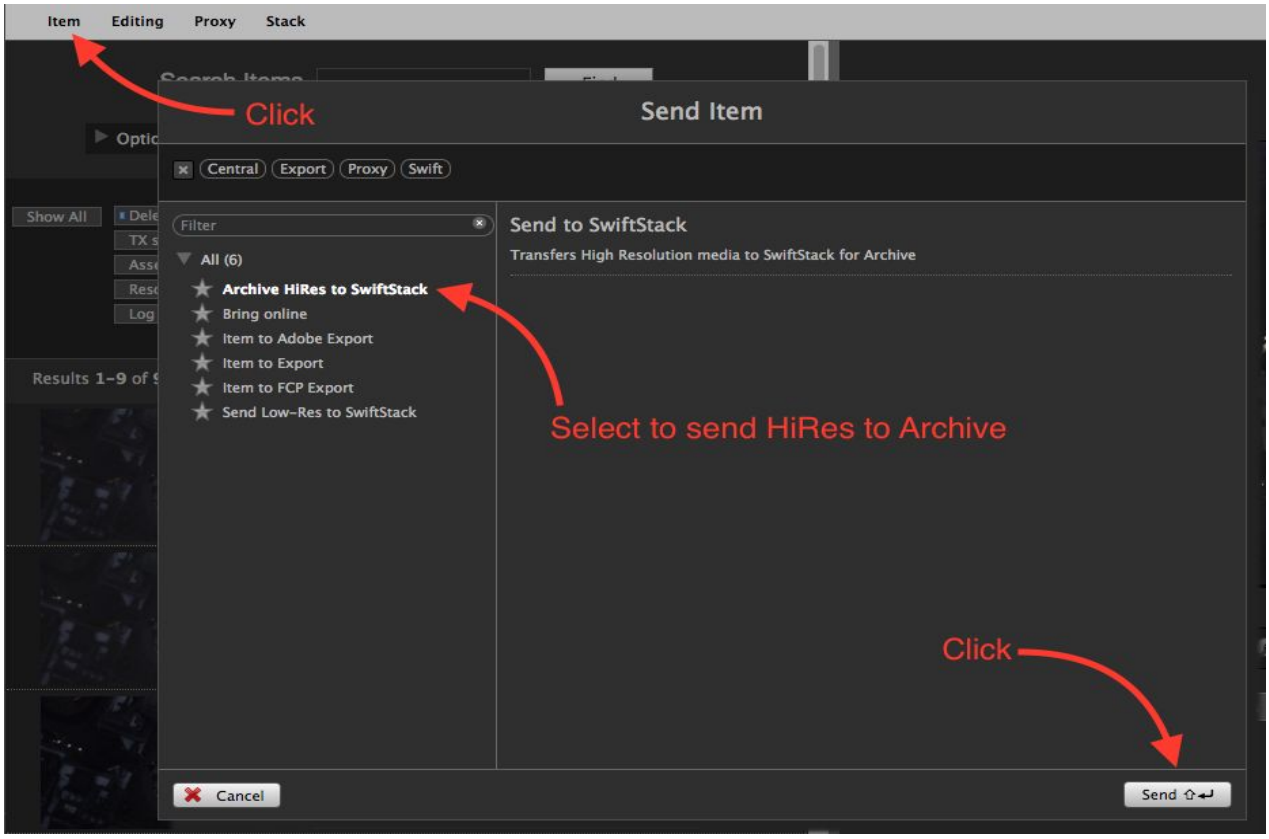
Viz One has four specific integration points with SwiftStack as of release version 5.14.0(48).

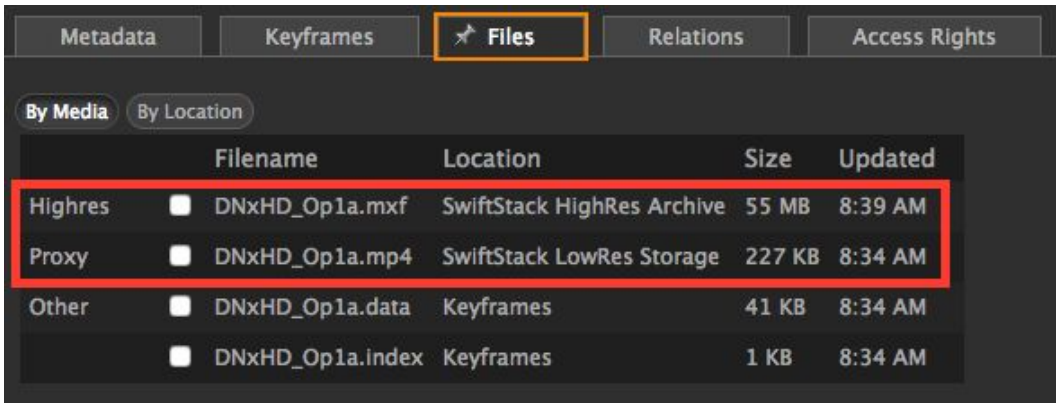
1. It stores all generated proxy files directly in SwiftStack as shown here.



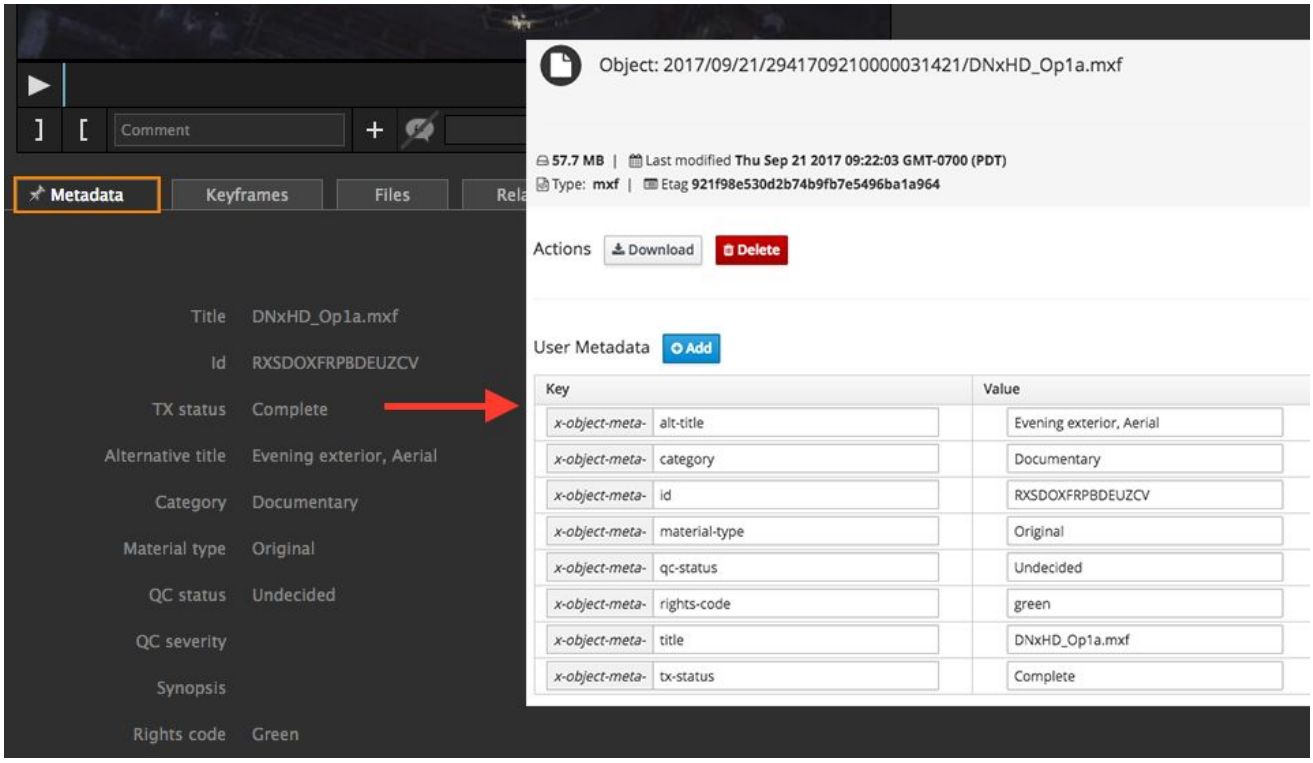

   These files are then served to web clients via reverse proxy from SwiftStack using the Swift API.

2. Ingested high-resolution media can be sent to SwiftStack for long-term active archive. When needed, this media can then be restored back from SwiftStack either via full restore or partial restore using the conform workflow (details are dependent on the codec used).

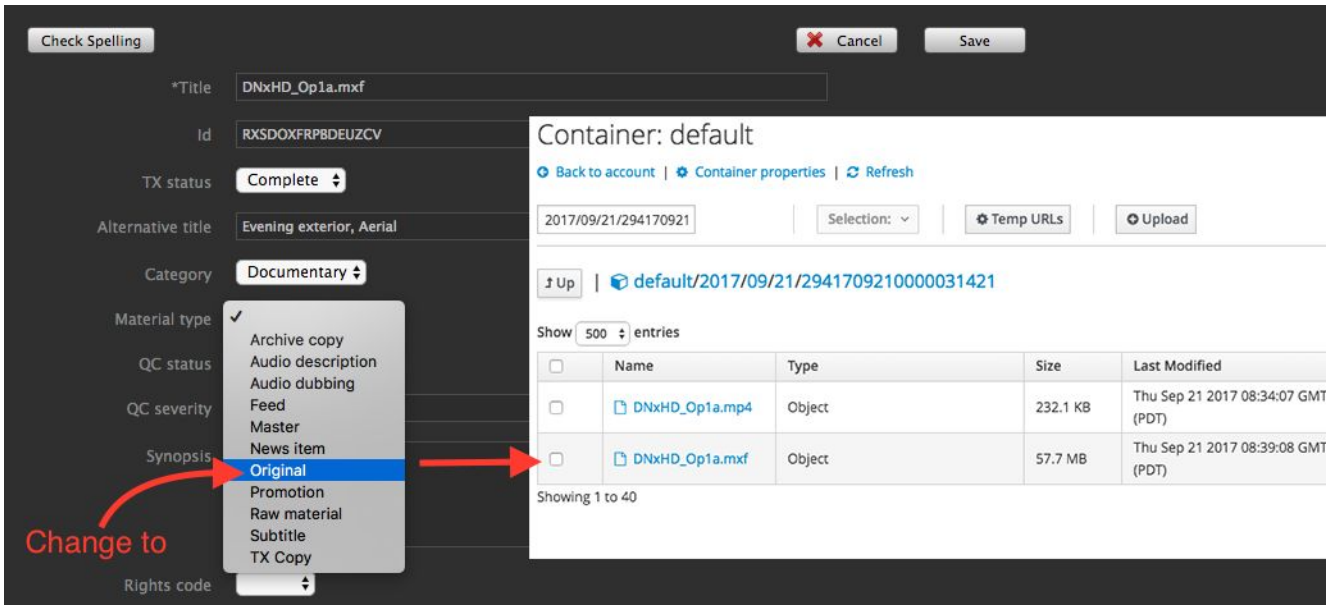



3. Viz One's "hubitem" metadata fields are mapped to SwiftStack's "X-Object" metadata. Metadata changes in Viz One are automatically reflected in SwiftStack object metadata. NOTE: This is only available for high-resolution archive material; "hubitem" metadata in Viz One for proxy media does not currently get mapped to X-Object metadata in SwiftStack.




4. When an asset's "Material Type" value is set to "Original" in Viz One, this will automatically trigger a high-res archive, which moves the original high-res media to SwiftStack.



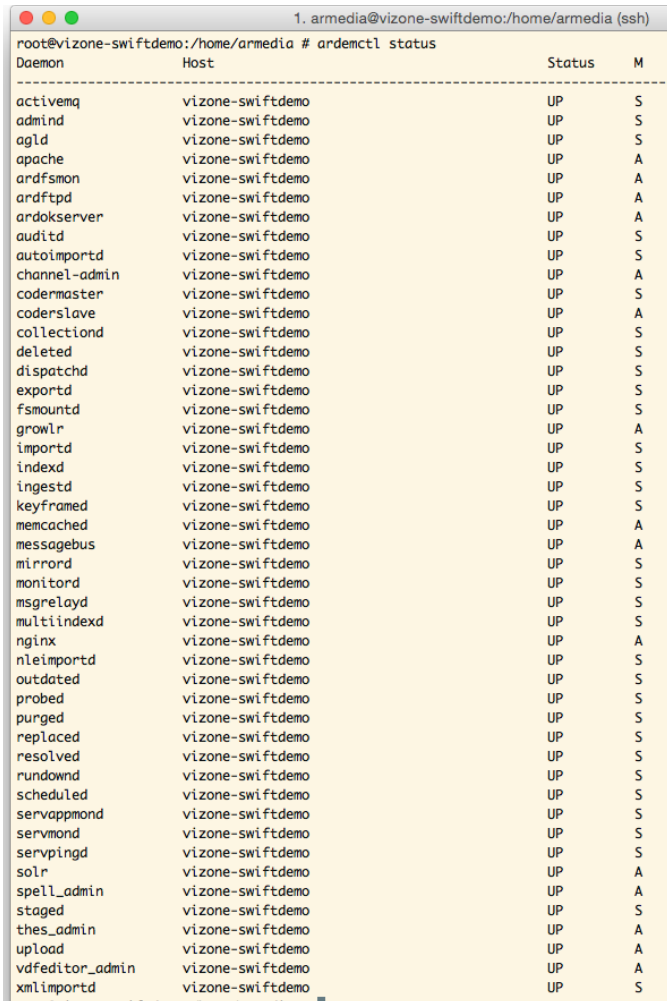

# Configuring Viz One to Use SwiftStack Storage

There are several configuration parameters that need to be set correctly to configure Viz One to use SwiftStack as a storage location for proxy media and high-resolution archives. Details will be different for each

environment, but the following example from a SwiftStack lab installation provides an example of the parameters that may be considered in a production deployment. Please contact Vizrt and SwiftStack for assistance in planning for a production deployment in your environment.

Below is a sample shell script to automate a reconfiguration of Viz One's SwiftStack credentials. This example script assumes that Viz One already has a working SwiftStack storage configuration, and it is tailored to a single Viz One node. Before executing a configuration script like this, make sure all Viz One daemons are reporting with an "UP" status ( "UP Respawn" does not count) by running the following command as root:

```
# ardemctl status
```




------ start of update_swift_config.sh ----------

```
#!/bin/bash
set -e
################### Variables Section ###############################
# NEW CONFIG: Add the new variables that you want configured.
swiftauthuser="vizone"
swiftauthkey="swiftstack"
swifturl="https://tellus.swiftstack.com/auth/v1.0/"      # trailing slash is required
swifthost="tellus.swiftstack.com"
swiftproxyhost="tellus.swiftstack.com:443"
```

```
#### Match the following. Original variables should reflect the existing
# OLD CONFIG. Matching is used to change them.
orig_swiftauthuser="oldUser"
orig_swiftauthkey="oldPassword"
orig_swifturl="https://swiftnode.vizrt.internal/auth/v1.0/"
orig_swiftproxyhost="swiftnode.vizrt.internal:443"
#
backupdir="/root/update_swift_backup"
# Keep backups of stuff we change
mkdir -p ${backupdir}

# There is typically one swift accessmethod but let's check.
echo "... Update the swift accessmethod ... "
for x in `storagemgr list acc|grep -w "swift "|awk '{print $1}'`; do
   storagemgr edit acc $x swift -- auth_url=${swifturl} username=${swiftauthuser} \
   password=${swiftauthkey}
done
# Search for all swift_clients and update
echo "... Update swift_client accessmethod ... "
for x in `storagemgr list acc|grep swift_client|awk '{print $1}'`; do
   storagemgr edit acc $x swift_client -- username=${swiftauthuser} \
   password=${swiftauthkey}
done
# Update swiftnode endpoint
echo "... Update swift endpoint ... "
storagemgr --no-pager edit server swiftnode host=${swifthost} state=A

## Swift LowRes (Proxy) ##
# 5.13+: use secure_link_swift.nginx
# Update nginx secure_link config
echo "... Updating secure_link_swift.nginx ..."
cp -L --backup=numbered /var/ardendo/conf/nginx/common/secure_link_swift.nginx \
${backupdir}/
sed -i -e "s#${orig_swifturl}#${swifturl}#g" \
/var/ardendo/conf/nginx/common/secure_link_swift.nginx
sed -i -e "/X-Auth-Key / s#${orig_swiftauthkey}#${swiftauthkey}#g" \
/var/ardendo/conf/nginx/common/secure_link_swift.nginx
sed -i -e "/X-Auth-User / s#${orig_swiftauthuser}#${swiftauthuser}#g" \
/var/ardendo/conf/nginx/common/secure_link_swift.nginx

# 5.14+: adds swift_upstream.nginx for lowres proxy
echo "... Updating swift_upstream.nginx ..."
cp -L --backup=numbered /var/ardendo/conf/nginx/main/swift_upstream.nginx ${backupdir}/
sed -i -e "s#${orig_swiftproxyhost}#${swiftproxyhost}#g" \
/var/ardendo/conf/nginx/main/swift_upstream.nginx

# Restart nginx
echo "... Restarting nginx and xfer daemons ... "
/opt/ardome/bin/ardemctl restart xfer,nginx
echo "... nginx start .. just in case ..."
/opt/ardome/bin/ardemctl start nginx

echo "All done!"
#[eof]
```

------ end of update_swift_config.sh ----------

# Additional config changes

As of the writing of this document using Viz One release v5.14.0(48), it is necessary to change the following configuration items. Future releases of Viz One may not require these additional changes.

To enable proxy playback, you need to change a variable within this nginx configuration file:
`/var/ardendo/conf/nginx/common/secure_link_swift.nginx`

Locate the variable “proxy_pass,” and and update its value so it reflects the new user.

`https://swift/v1/AUTH_oldUser/$1/$2` to `https://swift/v1/AUTH_vizone/$1/$2;`

To enable XDCAM transcoding, run this as root:

```
# confmgr set transcode.target.REMXF_XDCAMHD 0
```

# Adobe Assistant Panel

Viz One offers Editors on Adobe Premiere as a way to integrate directly into the NLE via Adobe Panel. This plugin is available directly from Viz One.

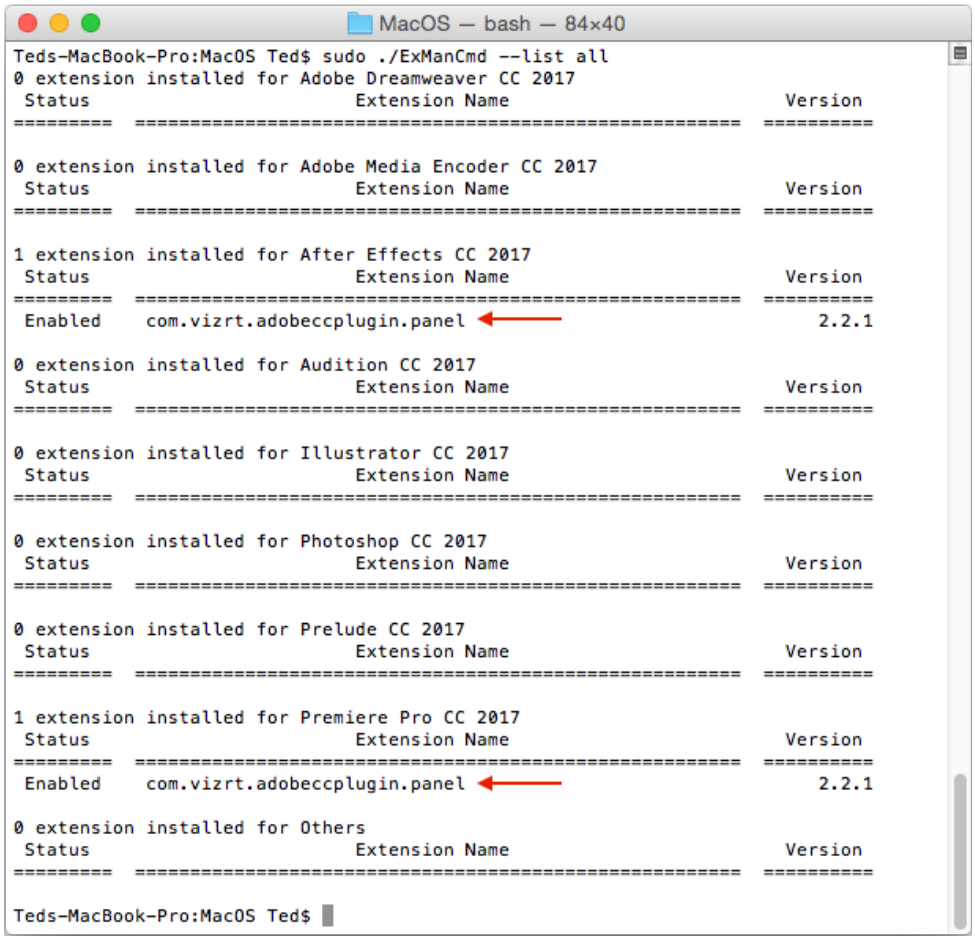
To install this Adobe Premiere add-on, you need to download the Adobe Extension Tool specifically for the host machine you are on. This is straight forward, but make sure you run the tool “ExManCmd” from within the extracted container, as it links to a few libraries here ( ../Frameworks/* ).

**Specific to macOS**

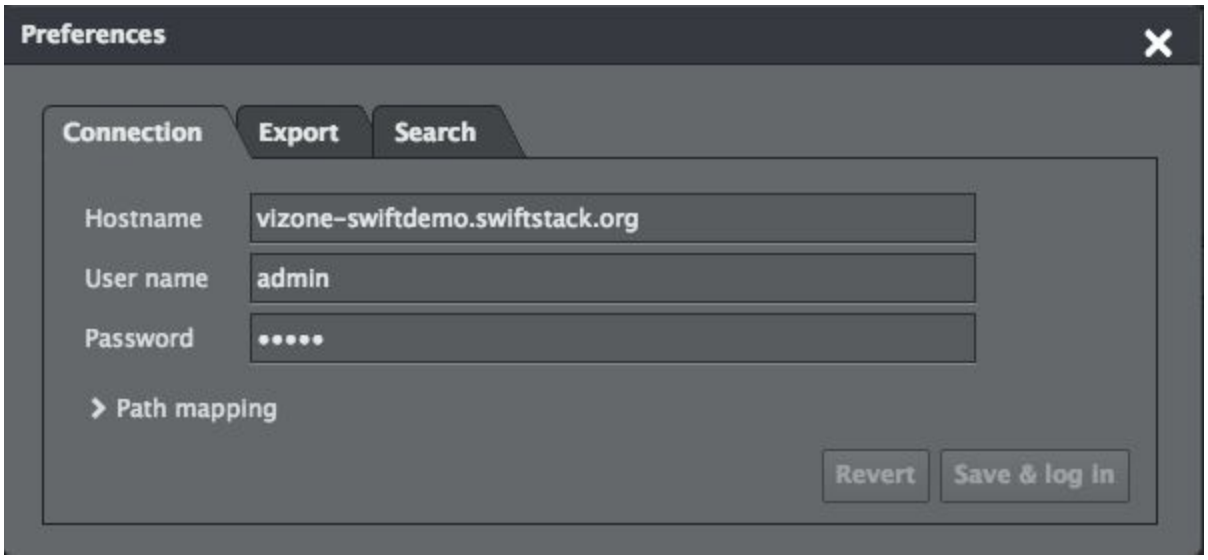
```
$ sudo ExManCmd --install AdobeAssistant.zxp
```

To confirm that the plugin has been successfully installed, run the following command, and confirm installation as in the image below.

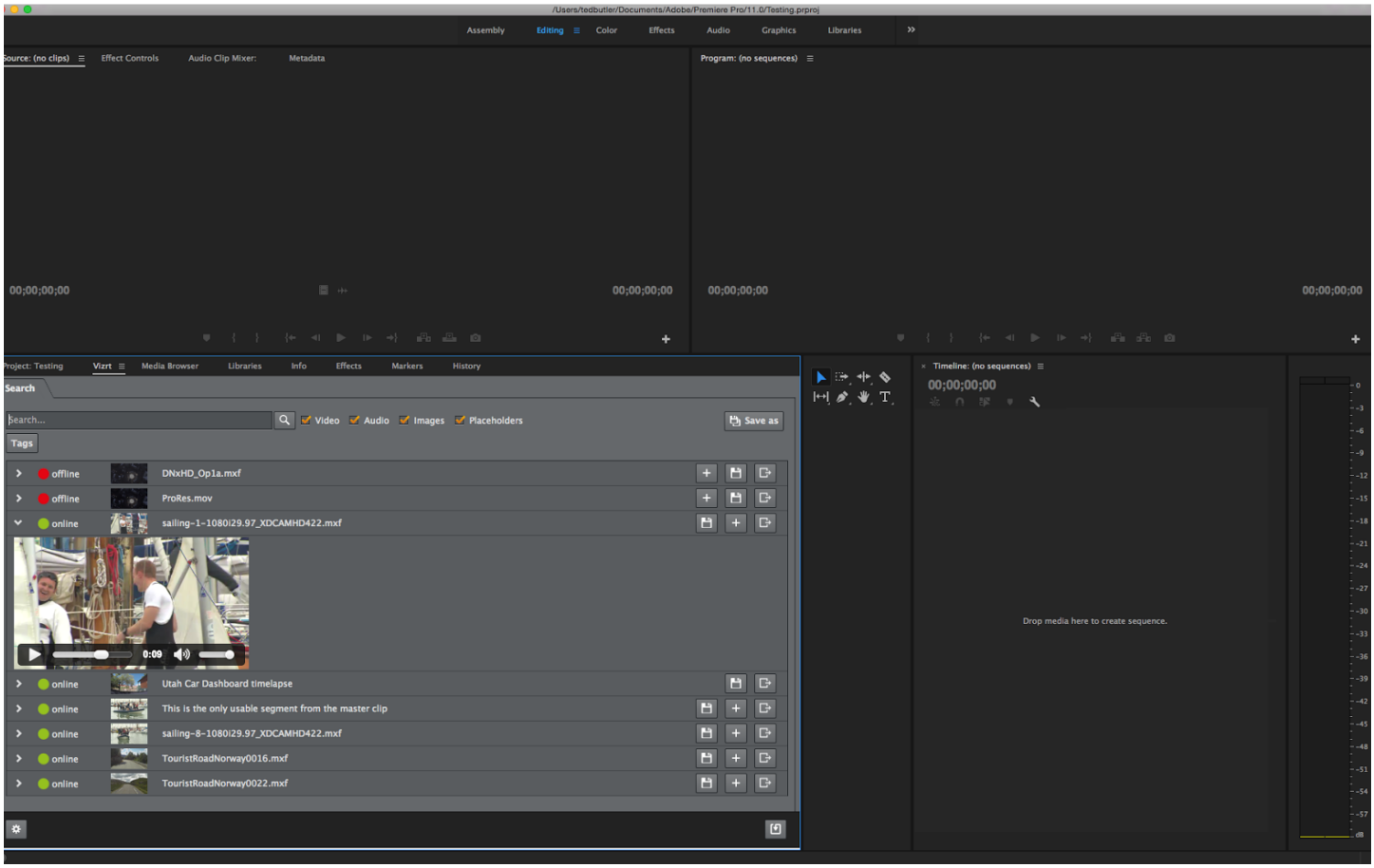
```
$ sudo ./ExManCmd --list all
```




Launch Adobe Premiere, and set up Viz One connections via Windows>Extensions>Vizrt. Select the gear icon on the lower left-hand corner, and enter corresponding values.



Save, and login. If the connection is successful, Adobe Premiere will have access to media on Viz One (according to user credentials), as seen below.




# Generating Sample Material for Ingest

To test your integration setup, use ffmpeg to generate media in a variety of video codecs for testing.

# DNxHD
ffmpeg -i <input file> -codec dnxhd -r 24000/1001 -threads 0 -vf scale=1920x1080 -b:v <output>.mxf

# ProRes
ffmpeg -i <input file> -vcodec prores -profile 2 -threads 0 -acodec copy <output>.mov

# XDCAM HD422
ffmpeg -i <input file> -vcodec mpeg2video -b:v 50000k -minrate 50000k -maxrate 50000k -bufsize 36408333 -f mxf <output>.mxf

# Known Issues and Workarounds

1. Viz One can sometimes raise an error (e.g., file is locked, permission error) when ingesting large files via macOS copy-and-paste. This is likely a timing issue due to the local filesystem event notification firing on Viz One's server before the file gets copied completely.  A workaround is to ingest via scp into a watch folder Viz One uses to trigger ingest workflows:

   ```
   scp <file> armedia@host:/VizOneHost:/ardome/media/smbimp/smbimp1/
   ```

# Conclusion

The benefits of SwiftStack and Viz One is extremely valuable to any decent size media house all the way to enterprise scale media and entertainment organizations. This is a no argument proposition - the current pain points felt by the media and entertainment industry is not only solved but improved with great efficiencies.