Genesys Interaction Recording Solution Guide

Deploying Recording Muxer Script

4/3/2026

Deploying Recording Muxer Script

Contents

Genesys Interaction Recording (GIR) needs the Recording Muxer Script to combine the call and screen recordings for a seamless playback.

Recording Muxer Script (Python 3)

Prerequisites

Before installing and configuring the Recording Muxer Script, you must have the following prerequisites:

- An Interaction Recording Web Services 8.5.205.32 (or higher) instance where the call recording and screen recording metadata is stored.
- A Recording Crypto Server 8.5.095.16 (or higher) instance to decrypt the encrypted recordings.
- Network access to the WebDAV storage or S3 Premise where the recordings are stored.
- For Recording Muxer Script 8.5.500.10 (or higher), Recording Processor Script must be upgraded to 8.5.500.13 (or higher) if using Recording Processor Script.

Installing Recording Muxer Script

Installing on Windows

1. Install 64-bit Python 3.11.5 from the Python website. To make Python 3 to work with OpenSSL 3.0.13, follow the below steps:
   - Download `libcrypto-3.dll` and `libssl-3.dll` from the Python Binary repository.
   - In `[python-source-folder]\DLLs`, replace with the above downloaded DLL files.
2. Install **Recording Muxer Script IP** with the installer.

   Note: Install the following third-party libraries in the order they appear and untar the files in `Administrator` mode.

3. Untar the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1.tar.gz` file.
4. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1` directory.
5. Untar the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16.tar.gz` file.
6. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16` directory.
7. Untar the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0.tar.gz` file.

8. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0` directory.
9. Untar the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2.tar.gz` file.
10. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2` directory.
11. Untar the `<Recording Muxer Install Directory>/thirdparty/idna-3.4.tar.gz` file.
12. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/idna-3.4` directory.
13. Untar the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22.tar.gz` file.
14. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22` directory.
15. Untar the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0.tar.gz` file.
16. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0` directory.
17. Untar the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0.tar.gz` file.
18. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0` directory.
19. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0.tar.gz` file.
20. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0` directory.
21. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0.tar.gz` file.
22. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0` directory.
23. Untar the `<Recording Muxer Install Directory>/thirdparty/botocore-1.31.36.tar.gz` file.
24. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/botocore-1.31.36` directory.
25. Untar the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2.tar.gz` file.
26. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2` directory.
27. Untar the `<Recording Muxer Install Directory>/thirdparty/boto3-1.28.36.tar.gz` file.
28. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/boto3-1.28.36` directory.
29. Untar the `<Recording Muxer Install Directory>/thirdparty/h11-0.14.0.tar.gz` file.
30. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/h11-0.14.0` directory.
31. Untar the `<Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0.tar.gz` file.
32. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0` directory.
33. Untar the `<Recording Muxer Install Directory>/thirdparty/anyio-4.0.0.tar.gz` file.

34. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/anyio-4.0.0` directory.
35. Untar the `<Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0.tar.gz` file.
36. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0` directory.
37. Untar the `<Recording Muxer Install Directory>/thirdparty/httpx-0.25.0.tar.gz` file.
38. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/httpx-0.25.0` directory.
39. Untar the `<Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8.tar.gz` file.
40. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8` directory.
41. Unzip the `<Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-4.4-win64-static-gpl3.0.zip`.
42. Unzip the `<Recording Muxer Install Directory>/thirdparty/openssl/openssl-3.0.13-win64.zip`. This OpenSSL library is used to encrypt the resulting muxed recording file when required.

Installing on Linux (RHEL)

1. Install `zlib-devel (yum install zlib-devel)`.
2. Install `sqlite devel (yum install sqlite-devel.x86_64)`.
3. Install `libffi devel (yum install libffi-devel)`.
4. Install OpenSSL.
   - For 8.5.500.09 or lower versions, install OpenSSL version 1.1.1.
   - For 8.5.500.10 or higher versions, install OpenSSL 3.0.13. Download OpenSSL 3.0.13 from OpenSSL website and compile it. Example config command - `./config --prefix=/usr/home/openssl-3.0.13 --openssldir=/usr/home/openssl-3.0.13 --libdir=lib no-shared`
5. Install 64 bit Python 3.11.5.
   - For 8.5.500.09 or lower versions, compile with OpenSSL 1.1.1 from the Python website. While compiling Cpython 3.11.5 with custom openssl, use `--with-openssl` flag while compilation. Example config command - `./configure --with-openssl=/usr/home/openssl-1.1.1 --enable-optimizations`
   - For 8.5.500.10 or higher versions, compile with OpenSSL 3.0.13 from the Python website.While compiling Cpython 3.11.5 with custom openssl, use `--with-openssl` flag while compilation. Example config command - `./configure --with-openssl=/usr/home/openssl-3.0.13 --enable-optimizations`
6. Install the **Recording Muxer Script IP** with the installer provided.

   Note: Install the following third-party libraries in the order they appear.
7. Untar the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1.tar.gz` file.
8. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1` directory.

9. Untar the <Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16.tar.gz file.
10. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16 directory.
11. Untar the <Recording Muxer Install Directory>/thirdparty/six-1.16.0.tar.gz file.
12. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/six-1.16.0 directory.
13. Untar the <Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2.tar.gz file.
14. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2 directory.
15. Untar the <Recording Muxer Install Directory>/thirdparty/idna-3.4.tar.gz file.
16. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/idna-3.4 directory.
17. Untar the <Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22.tar.gz file.
18. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22 directory.
19. Untar the <Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0.tar.gz file.
20. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0 directory.
21. Untar the <Recording Muxer Install Directory>/thirdparty/requests-2.31.0.tar.gz file.
22. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/requests-2.31.0 directory.
23. Untar the <Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0.tar.gz file.
24. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0 directory.
25. Untar the <Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0.tar.gz file.
26. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0 directory.
27. Untar the <Recording Muxer Install Directory>/thirdparty/botocore-1.31.36.tar.gz file.
28. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/botocore-1.31.36 directory.
29. Untar the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2.tar.gz file.
30. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2 directory.
31. Untar the <Recording Muxer Install Directory>/thirdparty/boto3-1.28.36.tar.gz file.
32. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/boto3-1.28.36 directory.
33. Untar the <Recording Muxer Install Directory>/thirdparty/h11-0.14.0.tar.gz file.
34. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/h11-0.14.0 directory.

35. Untar the <Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0.tar.gz file.
36. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0 directory.
37. Untar the <Recording Muxer Install Directory>/thirdparty/anyio-4.0.0.tar.gz file.
38. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/anyio-4.0.0 directory.
39. Untar the <Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0.tar.gz file.
40. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0 directory.
41. Untar the <Recording Muxer Install Directory>/thirdparty/httpx-0.25.0.tar.gz file.
42. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/httpx-0.25.0 directory.
43. Untar the <Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8.tar.gz file.
44. Run `python3 -m pip install .` from the <Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8 directory.
45. Perform one of the following steps depending on the Muxer version.
    - For 8.5.500.03, untar the <Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-4.4-centos7-x86_64-static-gpl3.0.tar.bz2.
    - For 8.5.500.09 or higher versions, untar the <Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-4.4-rhel8-x86_64-static-gpl3.0.tar.bz2.
46. Execute `chmod a+x ffmpeg` and `chmod a+x ffprobe`.
47. Perform one of the following steps depending on the Muxer version. The OpenSSL library is used to encrypt the resulting muxed recording file when required.
    - For 8.5.500.03, untar the <Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.1.1l-linux-x86_64.tar.bz2.
    - For 8.5.500.09, untar the <Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.1.1l-rhel8-x86_64.tar.bz2.
    - For 8.5.500.10 or higher, untar the <Recording Muxer Install Directory>/thirdparty/openssl/openssl-3.0.13-rhel8-x86_64.tar.bz2.
48. Execute `chmod a+x openssl`.

Important

- GIR does not support direct upgrade of Muxer from Python 2 to Python 3.
- Do not use the `setup.py install` command for installing libraries, instead use `pip install` command as mentioned above.
- Run `sudo dnf install libnsl` if you encounter the following error while executing Muxer installation script (`install.sh`):
  ./Perl: error while loading shared libraries: libnsl.so.1: cannot open shared

object file: No such file or directory.

Configuring Recording Muxer Script

This section describes how to configure the Recording Muxer Script for your environment.

Configure Passwords (Optional)

Important

In a Linux or Windows environment, Muxer supports the use of environment variables instead of parameters in the configuration file for certain parameters. When both are available, the environment variable take precedence.

The following definitions describe the mapping of the environment variables to the corresponding configuration parameter:

- **HTCC_PASSWORD**—maps to the existing configuration parameter under the `htcc` section, `password` value.
- **RCS_PASSWORD**— maps to the existing configuration parameter under the `rcs` section, `password` value.
- **WEBDAV_PASSWORD**—maps to the existing configuration parameter under the `webdav` section, `password` value.

In a Windows only environment, Recording Muxer Script supports storing all passwords in a secure keystore instead of storing in plain-text in the **muxer.cfg** file.

1. From the **muxer** directory folder in the Recording Muxer installation folder (for example, **<Recording Muxer Install Directory>\muxer**), execute the following command:
   `py encryptPassword.py`
   The command will prompt for the appropriate values to be entered for the password/key configuration parameters. See the Genesys Interaction Recording Options Reference for the descriptions of the parameters.
2. Configure the **muxer.cfg** file leaving the following parameter values empty:

```
[webdav]
password =

[htcc]
password=

[rcs]
password =
```

Configuring the Connection to Interaction Recording Web Services (Web Services)

To configure the Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) connection, set the following parameters in the **[htcc]** section of the Recording Muxer **muxer.cfg** configuration file:

Parameter Name	Default Value	Description
base_uri		Specifies the host and port of the Interaction Recording Web Services (Web Services) server—for example, `https://<web services host>:<web services port>/`.
contact_center_id		Specifies the unique identifier of the contact center.
username	`ops`	Specifies the username used to access the Interaction Recording Web Services (Web Services) account.
password	`ops`	Specifies the password used to access the Interaction Recording Web Services (Web Services) account. **Note:** • If the "Configuring the Secure Password Storage" step was performed, leave this value empty. • The password can be overridden by the `HTCC_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Interaction Recording Web Services (RWS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Interaction Recording Web Services on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Parameter Name	Default Value	Description
rws_timeout	30	Specifies the timeout duration, in seconds, for Recording Muxer Script while sending a request to Interaction Recording Web Services. **Note:** The timeout value must be greater than or equal to 30.

Configuring the Connection to Recording Crypto Server

To configure the connection to the Recording Crypto Server, set the following parameters in the **[rcs]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
base_uri	Empty	Specifies the host and port of the Recording Crypto Server: `https://<Recording Crypto Server host>:<Recording Crypto Server port>`
username	Empty	Specifies the contact center admin username used to access the Recording Crypto Server account belonging to the contact center specified by the **contact_center_id** option in the **[htcc]** section. **Note:** The user must have the media decrypt permission.
password	Empty	Specifies the contact center admin password used to access the Recording Crypto Server account belonging to the contact center specified by the the **contact_center_id** option in the **[htcc]** section. **Note:** • If the Configuring the Secure Password Storage step was performed, leave this value empty. • The password can be overridden by the `RCS_PASSWORD` environment variable.

Parameter Name	Default Value	Description
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Recording Crypto Server (RCS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Recording Crypto Server on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Processing Commands

1. The Recording Muxer uses libraries for analyzing and handling multimedia data. To configure these commands, set the following parameters in the **muxer.cfg** file, the **[processing]** section:

   - **ffmpeg** = The path to the ffmpeg executable file.

   > **Important**
   > The ffmpeg executable is located under the directory where the thirdparty ffmpeg package was unzipped/ untarred.

   - **ffprobe** = The path to the ffprobe executable file.

   > **Important**
   > The ffprobe executable is located under the directory where the thirdparty ffmpeg package was unzipped/ untarred.

2. To enable Muxer to read multiple screen recordings metadata with one request, configure the following parameters using the **muxer.cfg** configuration file (optional):

   - **batch_read_screen_recording_metadata**: Determines how screen recording metadata is received. The new algorithm reads multiple screen recordings metadata in one request. The previous algorithm reads one request at a time.
     Valid Values: Using Bulk API = 1 / Using previous algorithm the integer <>1
     Default Value: 1
   - **query_slice_size**: Defines the maximum number of call recording records whose screen recordings should be queried.

Valid Values: all integers > 0
Default Value: 100

3. Configure the **openssl** parameter to set the path to the openssl executable.

> **Important**
>
> - The openssl executable is located under the directory where the thirdparty openssl package was unzipped/untarred.
> - On Linux, specifying the absolute path to the openssl executable path is recommended to ensure that the default installed openssl (for example, `/usr/bin/openssl`) is not executed instead.

4. Configure the **window_past** and **window_past_older_than** parameters to set the time in the past to search for the call recordings to multiplex with the screen recordings. See the "Configure HA" section for the recommended values for these parameters.

5. Configure the **clean_temp_folder_timeout** parameter in the **[processing]** section to determine how often the recording files are cleaned up in the **temp folder**. **clean_temp_folder** should only be configured when **auto_clean_temp_folder** is set to 1. By default the **clean_temp_folder** value is 43200 (that is, cleanup occurs every 12 hours). If this value is set to `-1`, Muxer will attempt to perform a cleanup when it is idle.

For more information about the **[processing]** section parameters, see the Genesys Interaction Recording Options Reference.

Configuring Sharding (Optional)

Sharding can be used to increase the capacity of the Recording Muxer Script solution. When configured, the muxing workload is divided among multiple active instances. By default, Sharding is disabled and `muxer_id = -1`.

When Sharding is in use, a Muxer instance can be configured to run in primary or in backup mode:

- In primary mode, the Muxer should be configured to query for call records from the last `n` minutes (`window_past_older_than=0`, `window_past=n minutes`), based on configuration in the `muxer.cfg` file for that instance.
- In backup mode, the Muxer should be configured to query for call records that are older than the last `n` minutes but newer than `m` minutes (`window_past_older_than= n`, `window_past= m` minutes), based on configuration in the `muxer.cfg` file for that instance.

Sharding is configured based on the following command line or configuration file parameters within the `[processing]` section:

- **muxer_id:** A unique Muxer ID.
  Valid values: A non-negative integer starting with 0 (the Muxer ID should be incremented by 1 for each additional instance).
  If you are not using Sharding, the value should be empty or `-1`.
- **total_muxers:** The total number of primary Muxer instances deployed (excluding the backup).
  Valid Values: `max(muxer_id) + 1`
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.
- **muxer_type:** indicates if the Muxer is operating in primary mode or backup mode.
  Valid Values: primary, backup
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.

To specify Sharding parameters using the command line, the following arguments are used:

- muxer-type
- muxer-id
- total-muxers

Note: The Sharding parameter values passed in the command line overrides the corresponding values specified within the configuration file. The following is the supported command line:
`python.exe muxer_process.py --config-file=CONFIG_FILE --muxer-type=MUXER_TYPE --muxer-id=MUXER_ID --total-muxers=TOTAL_MUXERS`

For example: When using the following values, the system will have two instances of Muxer running:

- muxer_type=primary
- muxer_id=0 (for the first instance)
- muxer_id=1 (for the second instance)
- total_muxers=2

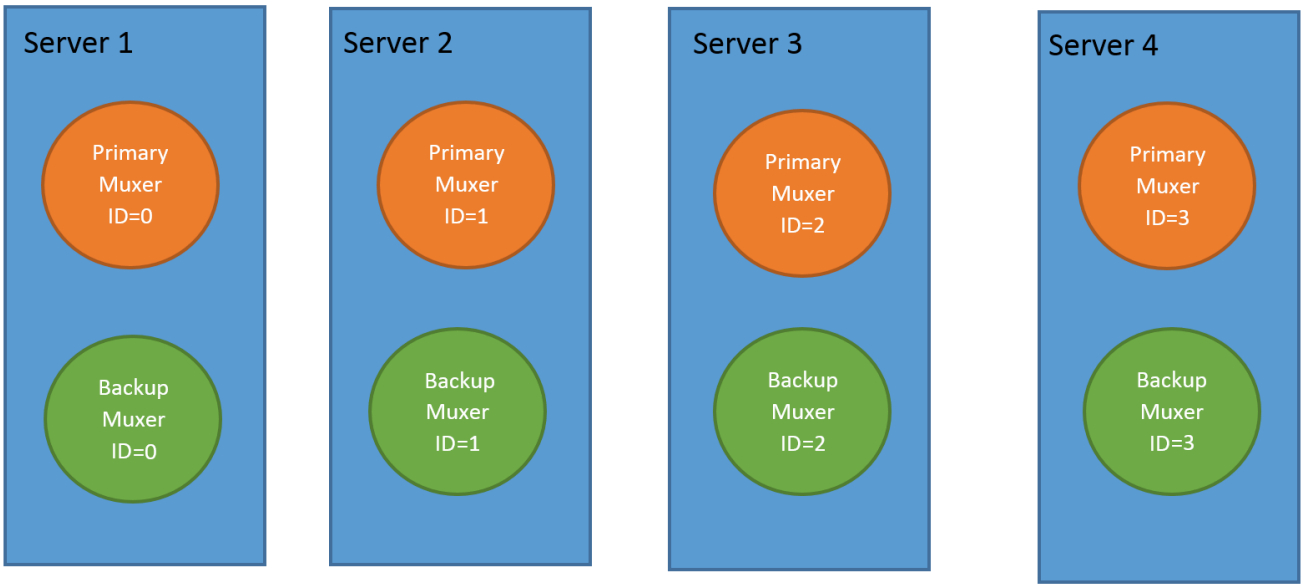
The following is the command line example for running the first instance: `python.exe ../muxer/muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=0 --total-muxers=2`

The following is the command line example for running the second instance: `python.exe ../muxer/muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=1 --total-muxers=2`

Note: When there are multiple instances of Muxers deployed on the same machine, then, a different **temp_dir** value for each instance of the Muxer must be configured in the `[processing]` section of the `muxer.cfg` file, and each Muxer instance must use a separate Muxer configuration file. This avoids the issues of one Muxer deleting the temporary files for the other instances.

> Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that each active primary instance be run on a separate machine. For a high availability deployment, a primary instance and a backup instance can be run on the same machine; however, in this case the instances should be configured so that the node IDs overlap (so that a single machine does not provide primary and backup coverage for the same `muxer_id`).



When running in backup mode, the Muxer will automatically calculate the `muxer_id` to be used to support this deployment mechanism, based on the specified `muxer_id`. The configured `muxer_id` used for the backup instance should match the `muxer_id` that is configured for the primary instance on the same machine, if both primary mode and backup mode instances are deployed together. For example, if `muxer_id=2` and `total_muxers=4` in the Muxer configuration file:

- When `muxer_type=primary`, the `muxer_id` used will be 2.
- When `muxer_type=backup`, the `muxer_id` used will be 3.

Important

If a Muxer instance is added or removed:

- The `total_muxers` value must be changed for each existing Muxer instance.
- All muxer instances must be restarted.
- Before starting the Muxer application, create and configure the **temp_dir** and **logfile_path** folders for both the Primary Muxer instance and the Backup Muxer instances running on the same machine.

Configuring High Availability (HA)

Important

The content in the Configure HA tab only applies if the Sharding configuration is not in use (see: Configure Sharding (Optional) tab). If Sharding is in use, refer to the high availability configuration described in the Configure Sharding (Optional) tab.

Recording Muxer Cluster

The Recording Muxer Script provides High Availability support using multiple instances of the Recording Muxer Script (all active). HA supports:

- Active/active pairs with the aim to load balance equally between the Recording Muxer nodes by splitting and configuring the time window on each node, so that it is close to equal the number of recordings found on each time window.
- When one of the node dies, recordings are still multiplexed.

Limitations:

- If the node with time window, now - N/2, dies, multiplexing will still occur, but a slower rate since the second node's time window is from N/2 to N.
- If the node with time window, N/2 - N, dies, screen recordings that are uploaded with the delay more than N/2 might not be multiplexed.
- Nodes should be configured so that the time windows are exclusive of each other, otherwise it may result in two multiplexed files being uploaded.

To configure HA:

1. In each Recording Muxer's **muxer.cfg** configuration file, in the **[processing]** section, set the following values for each node. For example,
   - On first node:
     - **window_past**= `720`
     - **window_past_older_than** = `5`

2. On second node:
   - **window_past** = `1440`
   - **window_past_older_than** = `725`

   The above will multiplex all recordings that were recorded within the last 1 day.
3. As a general rule, if the screen recording upload occurs with a delay of *N*, the configuration on each node can be set to:
   - On first node:
     - **window_past** = *N* / 2
     - **window_past_older_than** =
   - **min-poll_interval** = `N/200`
4. On second node:
   - **window_past**= *N*
   - **window_past_older_than** = *N* / 2
   - **min-poll_interval** = `N/200`

   Ensure that all Recording Muxer instances have the same configuration other than the above.

Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that multiple Recording Muxer instances be deployed on different hosts to provide better HA and also not to have machine resource contentions.
- If the recording upload is delayed by more than the time window configured for the Recording Muxer Script, it is possible that the recording will be outside of the processing window and not be multiplexed. For such cases, the Recording Muxer Script can be run as a migration tool to batch process the records matching any desired criteria. For more information see the **call_recording_query_string** parameter under **Configuring the Advanced Options** in the **Advanced Configuration** tab.
- If the screen recording upload is delayed longer than 24 hours, configure a separate Muxer instance or Muxer sharding group for every 12 hours. When the Screen Recording Service is provisioned to upload files during non-business hours, the actual delay can be a couple of days if the agent workstation is shut down when the agent signs off from the Agent Desktop.

Configuring the Connection to WebDAV

To configure the connection to WebDAV, set the following parameters in the **[webdav]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
username	Empty	Specifies the username to allow read/write access to the WebDAV storage server.
password	Empty	Specifies the password to allow read/write access to the WebDAV storage server. **Note:** • If multiple WebDAV storage are used for same contact center region, make sure to use the same username and password. • If the "Configuring the Secure Password Storage" step was performed, leave the password value empty. • A password can be overridden by the `WEBDAV_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to WebDAV. Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to WebDAV on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Advanced Options

The following advanced options can be configured in the **[advanced]** section of the **muxer.cfg** file:

- **worker_threads** = The number of parallel processing threads.
- **pagination** = The maximum number of records returned with each Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) query.

- **max_overlap_allowed** = The overlap time before truncating.
- **video_padding_slice_length_ms** = If the video starts later or ends earlier than the audio, set the duration needed to prepend or append a padded video slice. Genesys recommends to set it to `5000`.
- **mark_screen_recording_label** = Whether to apply the label "screenRecording" to the associated call recording metadata after muxing. This configuration is optional. The default value is 1.
- **call_recording_extra_query_string** = Used to specify parameter value pairs other than `startTime`, `endTime`, and `limit`.
  If left empty, the **call_recording_extra_query_string** value will be defaulted internally to `userData=SRSScreenRecordingStateStarted>anAndScroll=true`, if the RWS version is >= 8.5.201.14, otherwise, it remains an "" (empty string).
  Specify "disable" (without quotes) to force it to be an empty string without checking the RWS version.
  When the final value of this configuration is not empty, the Recording Muxer Script will continually poll for records that match the searching criteria according to the final value of the configuration that should be processed.
  Genesys recommends that this parameter be left empty. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`.
  The following table describes values (query parameters) that are available (except `startTime` and `endTime`).
- **call_recording_query_string** = When not empty, `[call_recording_query_string]` queries Interaction Recording Web Services (Web Services) with the given string for records to process. Instead of continually polling for records to process, the Recording Muxer script will exit once the returned records are processed. Genesys recommends that this parameter be left empty unless the Muxer script is to be used for batch migrating the old recordings. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`. The following table describes values (query parameters) that are available:

Parameter Name	Description
callerPhoneNumber	Retrieves all recordings which apply to any call containing the specified ANI attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard which can substitute any number of any symbols in the request. Search is case-sensitive.
dialedPhoneNumber	Retrieves all recordings which apply to any call containing the specified DNIS attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard - which can substitute any number of any symbols in request. Search is case-sensitive.
startTime	Retrieves all recordings that started >= the specified time.
endTime	Retrieves all recordings that ended <= the specified time.
userName	Retrieves all recordings in eventHistory->contacts of which the passed userName/ firstName/Lastname is present. User can use wildcards to specify only part of the username/ firstname/lastname. If more than 1 word is used (divided by spaces) -the records containing any of provided terms as username, firstname or lastname will be included. If user wants to

Parameter Name	Description
	retrieve records containing ALL terms - the AND keyword should be used. Sample: ?userName= Alice AND Amber - will seek for recording with events->contact-> username/firstName/ lastName containing Alice and Amber (possible - in different users). Search is case-insensitive.
userData	Retrieves all recordings in eventHistory->data of which the passed userData is present as value of HashMap. These matches are supported: • Exact match - match the entire value (for example, "tom" will find "tom"). • Wildcarded value (for example, "tom*" will find a record with "tomas"). • Combination of matches - If the query terms are separated by spaces (for example, "tom jerry" will look for recordings that contain "tom" or "jerry").

Configuring the Recording Muxer Using Genesys Administrator Extension (Optional)

The Recording Muxer uses a configuration file instead of a specific application object in Configuration Server. However, it is possible to configure the Recording Muxer as a "third-party server" application enabling Genesys Administrator Extension to monitor, start, and stop the process.

The following steps describe how to setup Recording Muxer as a "third party server" application in Genesys Administrator Extension. For more information, see the *Using the Management Layer* section of the Framework 8.5.1 Management Layer User's Guide

Configuring Recording Muxer Script to Start/Stop via LCA using Genesys Administrator Extension:

1. Install and deploy the latest Recording Muxer script.
2. Make sure that the Local Control Agent (LCA) is running.
3. Create a new application template in Genesys Administrator Extension called `Recording Muxer script` of type `Third Party Server`.
4. Create a new application (for example, `myRecordingMuxer`) in Genesys Administrator Extension using this new application template.
5. On Windows:
   a. Set the `Command Line` parameter to the python executable (for example, `C:\Python311\python.exe`).
   b. Set the `Host` parameter in the application's server info to the correct Host object.
   c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>\muxer` directory. For example, `C:\Program Files\GCTI\Recording Muxer Script\muxer`.
   d. Set the `Command Line Arguments` parameter to the python arguments: `muxer_process.py --config-file=muxer.cfg`.
6. On Linux:

a. Set the `Command Line` parameter to `env`.

b. Set the `Host` parameter in the application's server info to the correct Host object.

c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>/muxer` directory. For example, `/opt/genesys/Recording_Muxer_Script_8.5/muxer/`.

d. Set the `Command Line Arguments` parameter. The `LD_LIBRARY_PATH` must be set to include the openssl binary directory before muxer script execution. For example, `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<untarred openssl directory> /opt/python311/python muxer_process.py –-config-file=muxer.cfg`.

> **Important**
>
> The Recording Muxer does not support configuration through Genesys Administrator Extension. Configuration is acquired using a local configuration file.

Configuring Transport Layer Security (TLS) Connections (Optional)

Python provides the OpenSSL library that is used to establish TLS connections. The OpenSSL library that Python uses is not related to the OpenSSL library installed during installation of third-party libraries, which are used to encrypt muxed recording files.

Configuring TLS connection to Interaction Recording Web Services

1. Set up TLS on Interaction Recording Web Services (RWS). For more information, see Configuring TLS on the Server-Side for Interaction Recording Web Services section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**htcc**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use `https`.
3. In the [**htcc**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, then set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject

alternative name.

Configuring TLS connection to Recording Crypto Server

1. Set up TLS on Recording Crypto Server. For more information, see Configuring an HTTP Port section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**rcs**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use the secure port.
3. In the [**rcs**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

Configuring TLS connection to WebDAV

1. Set up TLS on WebDAV. For more information, see Configuring TLS for the WebDAV Server section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**webdav**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the

last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

For more information about the Recording Muxer Script parameters, see the Genesys Interaction Recording Options Reference.

Starting the Recording Muxer Script

Important

For **muxer.cfg**, if **temp_dir** is configured, verify that the path exists and is writable by the muxer process.

To launch the Recording Muxer script, run the following command from the <Recording Muxer Install Directory> (where x = 6):

On Windows:

```
<python3.11.5 executable> muxer_process.py --config-file=muxer.cfg
```

On Linux:

```
env LD_LIBRARY_PATH=<untarred openssl directory>:$LD_LIBRARY_PATH <python3.11.5
 executable> muxer_process.py --config-file=muxer.cfg
```

By default the Recording Muxer's log file is stored in the working directory. This can be changed by specifying a preexisting folder in the **logfile_path** parameter in the **[logfile]** section of the configuration file. For example, in Windows:

logfile_path = `C:\logs\recordingMuxer`

Recording Muxer Script (Python 3) RHEL 7

Prerequisites

Before installing and configuring the Recording Muxer Script, you must have the following prerequisites:

- An Interaction Recording Web Services 8.5.205.32 (or higher) instance where the call recording and screen recording metadata is stored.

- A Recording Crypto Server 8.5.095.16 (or higher) instance to decrypt the encrypted recordings.
- Network access to the WebDAV storage or S3 Premise where the recordings are stored.

Installing Recording Muxer Script

Installing on Windows

1. Install 64-bit Python 3.11.5 from the Python website.
2. Install **Recording Muxer Script IP** with the installer.

   Note: Install the following third-party libraries in the order they appear and untar the files in `Administrator` mode.

3. Untar the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1.tar.gz` file.
4. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1` directory.
5. Untar the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16.tar.gz` file.
6. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16` directory.
7. Untar the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0.tar.gz` file.
8. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0` directory.
9. Untar the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2.tar.gz` file.
10. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2` directory.
11. Untar the `<Recording Muxer Install Directory>/thirdparty/idna-3.4.tar.gz` file.
12. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/idna-3.4` directory.
13. Untar the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22.tar.gz` file.
14. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22` directory.
15. Untar the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0.tar.gz` file.
16. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0` directory.
17. Untar the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0.tar.gz` file.
18. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0` directory.
19. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0.tar.gz` file.
20. Run `py -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/`

pyasn1-0.5.0 directory.

21. Untar the <Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0.tar.gz file.
22. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0 directory.
23. Untar the <Recording Muxer Install Directory>/thirdparty/botocore-1.31.36.tar.gz file.
24. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/botocore-1.31.36 directory.
25. Untar the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2.tar.gz file.
26. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2 directory.
27. Untar the <Recording Muxer Install Directory>/thirdparty/boto3-1.28.36.tar.gz file.
28. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/boto3-1.28.36 directory.
29. Untar the <Recording Muxer Install Directory>/thirdparty/h11-0.14.0.tar.gz file.
30. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/h11-0.14.0 directory.
31. Untar the <Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0.tar.gz file.
32. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0 directory.
33. Untar the <Recording Muxer Install Directory>/thirdparty/anyio-4.0.0.tar.gz file.
34. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/anyio-4.0.0 directory.
35. Untar the <Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0.tar.gz file.
36. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0 directory.
37. Untar the <Recording Muxer Install Directory>/thirdparty/httpx-0.25.0.tar.gz file.
38. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/httpx-0.25.0 directory.
39. Untar the <Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8.tar.gz file.
40. Run py -m pip install . from the <Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8 directory.
41. Unzip the <Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-4.4-win64-static-gpl3.0.zip.
42. Unzip the <Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.1.1l-win64.zip. This OpenSSL library is used to encrypt the resulting muxed recording file when required.

Installing on Linux (RHEL)

1. Install zlib-devel (yum install zlib-devel).
2. Install sqlite devel (yum install sqlite-devel.x86_64).

3. Install `libffi devel (yum install libffi-devel)`.
4. Install OpenSSL 1.1.1.
   - For RHEL 7:
     1. Download OpenSSL 1.1.1 from OpenSSL website and compile it. Example config command - `./config --prefix=/usr/home/openssl-1.1.1 --openssldir=/usr/home/openssl-1.1.1`
     2. Add OpenSSL lib path in LD_LIBRARY_PATH. Example command - `export LD_LIBRARY_PATH=/usr/home/openssl-1.1.1/lib:$LD_LIBRARY_PATH`
5. Install 64 bit Python 3.11.5 compiled with OpenSSL 1.1.1 from the Python website.
   - While compiling Cpython 3.11.5 with custom openssl, use `--with-openssl` flag while compilation. Example config command - `./configure --with-openssl=/usr/home/openssl-1.1.1 --enable-optimizations`
6. Install the **Recording Muxer Script IP** with the installer provided.

   Note: Install the following third-party libraries in the order they appear.

7. Untar the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1.tar.gz` file.
8. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/jmespath-1.0.1` directory.
9. Untar the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16.tar.gz` file.
10. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/urllib3-1.26.16` directory.
11. Untar the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0.tar.gz` file.
12. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/six-1.16.0` directory.
13. Untar the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2.tar.gz` file.
14. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.8.2` directory.
15. Untar the `<Recording Muxer Install Directory>/thirdparty/idna-3.4.tar.gz` file.
16. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/idna-3.4` directory.
17. Untar the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22.tar.gz` file.
18. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/certifi-2023.7.22` directory.
19. Untar the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0.tar.gz` file.
20. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/charset-normalizer-3.2.0` directory.
21. Untar the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0.tar.gz` file.
22. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/requests-2.31.0` directory.
23. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0.tar.gz` file.

24. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.5.0` directory.
25. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0.tar.gz` file.
26. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/pyasn1_modules-0.3.0` directory.
27. Untar the `<Recording Muxer Install Directory>/thirdparty/botocore-1.31.36.tar.gz` file.
28. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/botocore-1.31.36` directory.
29. Untar the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2.tar.gz` file.
30. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.6.2` directory.
31. Untar the `<Recording Muxer Install Directory>/thirdparty/boto3-1.28.36.tar.gz` file.
32. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/boto3-1.28.36` directory.
33. Untar the `<Recording Muxer Install Directory>/thirdparty/h11-0.14.0.tar.gz` file.
34. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/h11-0.14.0` directory.
35. Untar the `<Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0.tar.gz` file.
36. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/sniffio-1.3.0` directory.
37. Untar the `<Recording Muxer Install Directory>/thirdparty/anyio-4.0.0.tar.gz` file.
38. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/anyio-4.0.0` directory.
39. Untar the `<Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0.tar.gz` file.
40. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/httpcore-0.18.0` directory.
41. Untar the `<Recording Muxer Install Directory>/thirdparty/httpx-0.25.0.tar.gz` file.
42. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/httpx-0.25.0` directory.
43. Untar the `<Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8.tar.gz` file.
44. Run `python3 -m pip install .` from the `<Recording Muxer Install Directory>/thirdparty/webdav4-0.9.8` directory.
45. Untar the `<Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-4.4-rhel7-x86_64-static-gpl3.0.tar.bz2`.
46. Execute `chmod a+x ffmpeg` and `chmod a+x ffprobe`.
47. Untar the `<Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.1.1l-rhel7-x86_64.tar.bz2`.
48. Execute `chmod a+x openssl`.

Important

- GIR does not support direct upgrade of Muxer from Python 2 to Python 3.
- Do not use the `setup.py install` command for installing libraries, instead use `pip install` command as mentioned above.
- Run `sudo dnf install libnsl` if you encounter the following error while executing Muxer installation script (`install.sh`):
  ./Perl: error while loading shared libraries: libnsl.so.1: cannot open shared object file: No such file or directory.

Configuring Recording Muxer Script

This section describes how to configure the Recording Muxer Script for your environment.

Configure Passwords (Optional)

Important

In a Linux or Windows environment, Muxer supports the use of environment variables instead of parameters in the configuration file for certain parameters. When both are available, the environment variable take precedence.

The following definitions describe the mapping of the environment variables to the corresponding configuration parameter:

- **HTCC_PASSWORD**—maps to the existing configuration parameter under the `htcc` section, `password` value.
- **RCS_PASSWORD**— maps to the existing configuration parameter under the `rcs` section, `password` value.
- **WEBDAV_PASSWORD**—maps to the existing configuration parameter under the `webdav` section, `password` value.

In a Windows only environment, Recording Muxer Script supports storing all passwords in a secure keystore instead of storing in plain-text in the **muxer.cfg** file.

1. From the **muxer** directory folder in the Recording Muxer installation folder (for example, **<Recording Muxer Install Directory>\muxer**), execute the following command:
   `py encryptPassword.py`
   The command will prompt for the appropriate values to be entered for the password/key configuration parameters. See the Genesys Interaction Recording Options Reference for the descriptions of the parameters.

2. Configure the **muxer.cfg** file leaving the following parameter values empty:

```
[webdav]
password =

[htcc]
password=

[rcs]
password =
```

Configuring the Connection to Interaction Recording Web Services (Web Services)

To configure the Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) connection, set the following parameters in the **[htcc]** section of the Recording Muxer **muxer.cfg** configuration file:

Parameter Name	Default Value	Description
base_uri		Specifies the host and port of the Interaction Recording Web Services (Web Services) server—for example, `https://<web services host>:<web services port>/`.
contact_center_id		Specifies the unique identifier of the contact center.
username	`ops`	Specifies the username used to access the Interaction Recording Web Services (Web Services) account.
password	`ops`	Specifies the password used to access the Interaction Recording Web Services (Web Services) account. **Note:** • If the "Configuring the Secure Password Storage" step was performed, leave this value empty. • The password can be overridden by the `HTCC_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Interaction Recording Web Services (RWS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in

Parameter Name	Default Value	Description
		PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Interaction Recording Web Services on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.
rws_timeout	30	Specifies the timeout duration, in seconds, for Recording Muxer Script while sending a request to Interaction Recording Web Services. **Note:** The timeout value must be greater than or equal to 30.

Configuring the Connection to Recording Crypto Server

To configure the connection to the Recording Crypto Server, set the following parameters in the **[rcs]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
base_uri	Empty	Specifies the host and port of the Recording Crypto Server: `https://<Recording Crypto Server host>:<Recording Crypto Server port>`
username	Empty	Specifies the contact center admin username used to access the Recording Crypto Server account belonging to the contact center specified by the **contact_center_id** option in the **[htcc]** section. **Note:** The user must have the media decrypt permission.
password	Empty	Specifies the contact center admin password used to access the Recording Crypto Server account belonging to the contact center specified by the the **contact_center_id** option in the **[htcc]** section.

Parameter Name	Default Value	Description
		Note: • If the Configuring the Secure Password Storage step was performed, leave this value empty. • The password can be overridden by the `RCS_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Recording Crypto Server (RCS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Recording Crypto Server on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Processing Commands

1. The Recording Muxer uses libraries for analyzing and handling multimedia data. To configure these commands, set the following parameters in the **muxer.cfg** file, the **[processing]** section:
   - **ffmpeg** = The path to the ffmpeg executable file.

> **Important**
>
> The ffmpeg executable is located under the directory where the thirdparty ffmpeg package was unzipped/ untarred.

   - **ffprobe** = The path to the ffprobe executable file.

> **Important**
>
> The ffprobe executable is located under the directory where the thirdparty ffmpeg package was unzipped/

untarred.

2. To enable Muxer to read multiple screen recordings metadata with one request, configure the following parameters using the **muxer.cfg** configuration file (optional):
   - **batch_read_screen_recording_metadata**: Determines how screen recording metadata is received. The new algorithm reads multiple screen recordings metadata in one request. The previous algorithm reads one request at a time.
     Valid Values: Using Bulk API = 1 / Using previous algorithm the integer <>1
     Default Value: 1
   - **query_slice_size**: Defines the maximum number of call recording records whose screen recordings should be queried.
     Valid Values: all integers > 0
     Default Value: `100`

3. Configure the **openssl** parameter to set the path to the openssl executable.

   Important

   - The openssl executable is located under the directory where the thirdparty openssl package was unzipped/untarred.
   - On Linux, specifying the absolute path to the openssl executable path is recommended to ensure that the default installed openssl (for example, `/usr/bin/openssl`) is not executed instead.

4. Configure the **window_past** and **window_past_older_than** parameters to set the time in the past to search for the call recordings to multiplex with the screen recordings. See the "Configure HA" section for the recommended values for these parameters.

5. Configure the **clean_temp_folder_timeout** parameter in the **[processing]** section to determine how often the recording files are cleaned up in the **temp folder**. **clean_temp_folder** should only be configured when **auto_clean_temp_folder** is set to 1. By default the **clean_temp_folder** value is 43200 (that is, cleanup occurs every 12 hours). If this value is set to `-1`, Muxer will attempt to perform a cleanup when it is idle.

For more information about the **[processing]** section parameters, see the Genesys Interaction Recording Options Reference.

Configuring Sharding (Optional)

Sharding can be used to increase the capacity of the Recording Muxer Script solution. When configured, the muxing workload is divided among multiple active instances. By default, Sharding is

disabled and `muxer_id = -1`.

When Sharding is in use, a Muxer instance can be configured to run in primary or in backup mode:

- In primary mode, the Muxer should be configured to query for call records from the last `n` minutes (`window_past_older_than=0`, `window_past=n minutes`), based on configuration in the `muxer.cfg` file for that instance.
- In backup mode, the Muxer should be configured to query for call records that are older than the last `n` minutes but newer than `m` minutes (`window_past_older_than= n`, `window_past= m` minutes), based on configuration in the `muxer.cfg` file for that instance.

Sharding is configured based on the following command line or configuration file parameters within the `[processing]` section:

- **muxer_id:** A unique Muxer ID.
  Valid values: A non-negative integer starting with 0 (the Muxer ID should be incremented by 1 for each additional instance).
  If you are not using Sharding, the value should be empty or `-1`.
- **total_muxers:** The total number of primary Muxer instances deployed (excluding the backup).
  Valid Values: `max(muxer_id) + 1`
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.
- **muxer_type:** indicates if the Muxer is operating in primary mode or backup mode.
  Valid Values: primary, backup
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.

To specify Sharding parameters using the command line, the following arguments are used:

- muxer-type
- muxer-id
- total-muxers

Note: The Sharding parameter values passed in the command line overrides the corresponding values specified within the configuration file. The following is the supported command line: `python.exe muxer_process.py --config-file=CONFIG_FILE --muxer-type=MUXER_TYPE --muxer-id=MUXER_ID --total-muxers=TOTAL_MUXERS`

For example: When using the following values, the system will have two instances of Muxer running:

- muxer_type=primary
- muxer_id=0 (for the first instance)
- muxer_id=1 (for the second instance)
- total_muxers=2

The following is the command line example for running the first instance: `python.exe ../muxer/muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=0 --total-muxers=2`

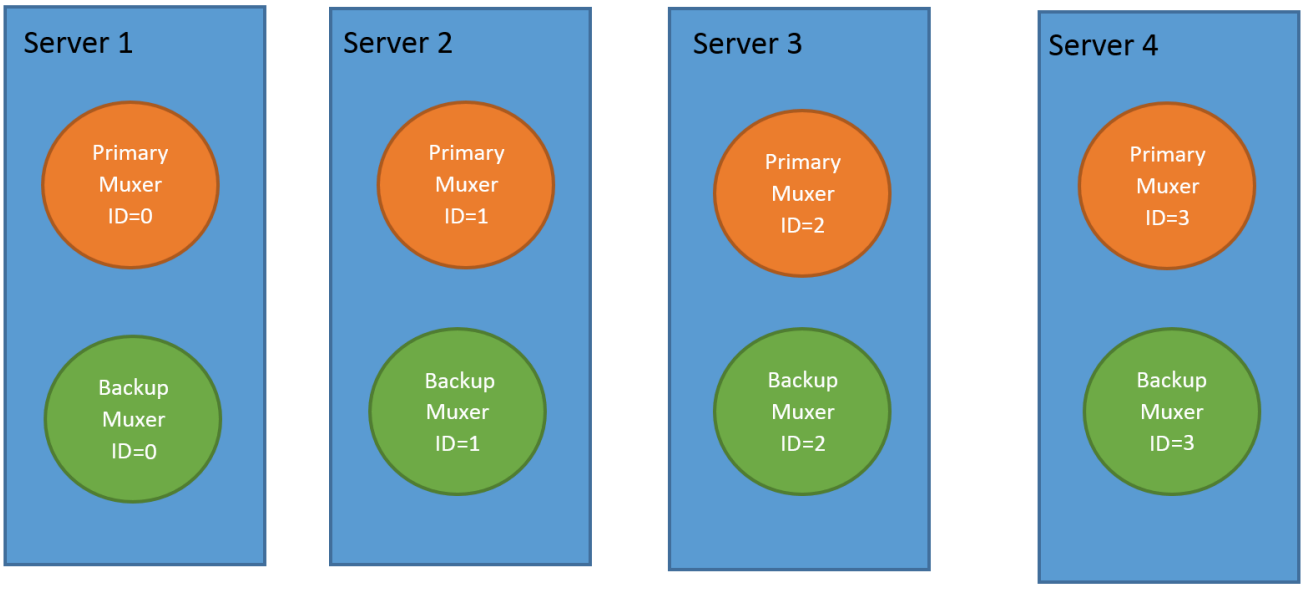
The following is the command line example for running the second instance: `python.exe ../muxer/`

```
muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=1 --total-muxers=2
```

Note: When there are multiple instances of Muxers deployed on the same machine, then, a different **temp_dir** value for each instance of the Muxer must be configured in the `[processing]` section of the `muxer.cfg` file, and each Muxer instance must use a separate Muxer configuration file. This avoids the issues of one Muxer deleting the temporary files for the other instances.

Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that each active primary instance be run on a separate machine. For a high availability deployment, a primary instance and a backup instance can be run on the same machine; however, in this case the instances should be configured so that the node IDs overlap (so that a single machine does not provide primary and backup coverage for the same `muxer_id`).



When running in backup mode, the Muxer will automatically calculate the `muxer_id` to be used to support this deployment mechanism, based on the specified `muxer_id`. The configured `muxer_id` used for the backup instance should match the `muxer_id` that is configured for the primary instance on the same machine, if both primary mode and backup mode instances are deployed together. For example, if `muxer_id=2` and `total_muxers=4` in the Muxer configuration file:

- When `muxer_type=primary`, the `muxer_id` used will be 2.

- When `muxer_type=backup`, the `muxer_id` used will be 3.

> **Important**
>
> If a Muxer instance is added or removed:
>
> - The `total_muxers` value must be changed for each existing Muxer instance.
> - All muxer instances must be restarted.
> - Before starting the Muxer application, create and configure the **temp_dir** and **logfile_path** folders for both the Primary Muxer instance and the Backup Muxer instances running on the same machine.

Configuring High Availability (HA)

> **Important**
>
> The content in the Configure HA tab only applies if the Sharding configuration is not in use (see: Configure Sharding (Optional) tab). If Sharding is in use, refer to the high availability configuration described in the Configure Sharding (Optional) tab.

Recording Muxer Cluster

The Recording Muxer Script provides High Availability support using multiple instances of the Recording Muxer Script (all active). HA supports:

- Active/active pairs with the aim to load balance equally between the Recording Muxer nodes by splitting and configuring the time window on each node, so that it is close to equal the number of recordings found on each time window.
- When one of the node dies, recordings are still multiplexed.

Limitations:

- If the node with time window, now - N/2, dies, multiplexing will still occur, but a slower rate since the second node's time window is from N/2 to N.
- If the node with time window, N/2 - N, dies, screen recordings that are uploaded with the delay more than N/2 might not be multiplexed.
- Nodes should be configured so that the time windows are exclusive of each other, otherwise it may result in two multiplexed files being uploaded.

To configure HA:

1. In each Recording Muxer's **muxer.cfg** configuration file, in the **[processing]** section, set the following values for each node. For example,

- On first node:
  - **window_past**= 720
  - **window_past_older_than** = 5

2. On second node:
   - **window_past** = 1440
   - **window_past_older_than** = 725

   The above will multiplex all recordings that were recorded within the last 1 day.
3. As a general rule, if the screen recording upload occurs with a delay of *N*, the configuration on each node can be set to:
   - On first node:
     - **window_past** = *N* / 2
     - **window_past_older_than** =
   - **min-poll_interval** = N/200
4. On second node:
   - **window_past**= *N*
   - **window_past_older_than** = *N* / 2
   - **min-poll_interval** = N/200

   Ensure that all Recording Muxer instances have the same configuration other than the above.

Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that multiple Recording Muxer instances be deployed on different hosts to provide better HA and also not to have machine resource contentions.
- If the recording upload is delayed by more than the time window configured for the Recording Muxer Script, it is possible that the recording will be outside of the processing window and not be multiplexed. For such cases, the Recording Muxer Script can be run as a migration tool to batch process the records matching any desired criteria. For more information see the **call_recording_query_string** parameter under **Configuring the Advanced Options** in the **Advanced Configuration** tab.
- If the screen recording upload is delayed longer than 24 hours, configure a separate

Muxer instance or Muxer sharding group for every 12 hours. When the Screen Recording Service is provisioned to upload files during non-business hours, the actual delay can be a couple of days if the agent workstation is shut down when the agent signs off from the Agent Desktop.

Configuring the Connection to WebDAV

To configure the connection to WebDAV, set the following parameters in the **[webdav]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
username	Empty	Specifies the username to allow read/write access to the WebDAV storage server.
password	Empty	Specifies the password to allow read/write access to the WebDAV storage server. **Note:** • If multiple WebDAV storage are used for same contact center region, make sure to use the same username and password. • If the "Configuring the Secure Password Storage" step was performed, leave the password value empty. • A password can be overridden by the `WEBDAV_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to WebDAV. Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to WebDAV on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Advanced Options

The following advanced options can be configured in the **[advanced]** section of the **muxer.cfg** file:

- **worker_threads** = The number of parallel processing threads.
- **pagination** = The maximum number of records returned with each Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) query.
- **max_overlap_allowed** = The overlap time before truncating.
- **video_padding_slice_length_ms** = If the video starts later or ends earlier than the audio, set the duration needed to prepend or append a padded video slice. Genesys recommends to set it to `5000`.
- **mark_screen_recording_label** = Whether to apply the label "screenRecording" to the associated call recording metadata after muxing. This configuration is optional. The default value is 1.
- **call_recording_extra_query_string** = Used to specify parameter value pairs other than `startTime`, `endTime`, and `limit`.
  If left empty, the **call_recording_extra_query_string** value will be defaulted internally to `userData=SRSScreenRecordingStateStarted>anAndScroll=true`, if the RWS version is >= 8.5.201.14, otherwise, it remains an "" (empty string).
  Specify "disable" (without quotes) to force it to be an empty string without checking the RWS version.
  When the final value of this configuration is not empty, the Recording Muxer Script will continually poll for records that match the searching criteria according to the final value of the configuration that should be processed.
  Genesys recommends that this parameter be left empty. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`.
  The following table describes values (query parameters) that are available (except `startTime` and `endTime`).
- **call_recording_query_string** = When not empty, `[call_recording_query_string]` queries Interaction Recording Web Services (Web Services) with the given string for records to process. Instead of continually polling for records to process, the Recording Muxer script will exit once the returned records are processed. Genesys recommends that this parameter be left empty unless the Muxer script is to be used for batch migrating the old recordings. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`. The following table describes values (query parameters) that are available:

Parameter Name	Description
callerPhoneNumber	Retrieves all recordings which apply to any call containing the specified ANI attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard which can substitute any number of any symbols in the request. Search is case-sensitive.
dialedPhoneNumber	Retrieves all recordings which apply to any call containing the specified DNIS attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard - which can substitute any number of any symbols in request. Search is case-sensitive.
startTime	Retrieves all recordings that started >= the specified time.
endTime	Retrieves all recordings that ended <= the

Parameter Name	Description
	specified time.
userName	Retrieves all recordings in eventHistory->contacts of which the passed userName/firstName/Lastname is present. User can use wildcards to specify only part of the username/firstname/lastname. If more than 1 word is used (divided by spaces) -the records containing any of provided terms as username, firstname or lastname will be included. If user wants to retrieve records containing ALL terms - the AND keyword should be used. Sample: ?userName=Alice AND Amber - will seek for recording with events->contact-> username/firstName/lastName containing Alice and Amber (possible - in different users). Search is case-insensitive.
userData	Retrieves all recordings in eventHistory->data of which the passed userData is present as value of HashMap. These matches are supported: • Exact match - match the entire value (for example, "tom" will find "tom"). • Wildcarded value (for example, "tom*" will find a record with "tomas"). • Combination of matches - If the query terms are separated by spaces (for example, "tom jerry" will look for recordings that contain "tom" or "jerry").

Configuring the Recording Muxer Using Genesys Administrator Extension (Optional)

The Recording Muxer uses a configuration file instead of a specific application object in Configuration Server. However, it is possible to configure the Recording Muxer as a "third-party server" application enabling Genesys Administrator Extension to monitor, start, and stop the process.

The following steps describe how to setup Recording Muxer as a "third party server" application in Genesys Administrator Extension. For more information, see the *Using the Management Layer* section of the Framework 8.5.1 Management Layer User's Guide

Configuring Recording Muxer Script to Start/Stop via LCA using Genesys Administrator Extension:

1. Install and deploy the latest Recording Muxer script.
2. Make sure that the Local Control Agent (LCA) is running.
3. Create a new application template in Genesys Administrator Extension called `Recording Muxer script` of type `Third Party Server`.
4. Create a new application (for example, `myRecordingMuxer`) in Genesys Administrator Extension using this new application template.
5. On Windows:
   a. Set the `Command Line` parameter to the python executable (for example, `C:\Python311\python.exe`).

b. Set the `Host` parameter in the application's server info to the correct Host object.

c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>\muxer` directory. For example, `C:\Program Files\GCTI\Recording Muxer Script\muxer`.

d. Set the `Command Line Arguments` parameter to the python arguments: `muxer_process.py --config-file=muxer.cfg`.

6. On Linux:

a. Set the `Command Line` parameter to `env`.

b. Set the `Host` parameter in the application's server info to the correct Host object.

c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>/muxer` directory. For example, `/opt/genesys/Recording_Muxer_Script_8.5/muxer/`.

d. Set the `Command Line Arguments` parameter. The `LD_LIBRARY_PATH` must be set to include the openssl binary directory before muxer script execution. For example, `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<untarred openssl directory> /opt/python311/python muxer_process.py –-config-file=muxer.cfg`.

> **Important**
>
> The Recording Muxer does not support configuration through Genesys Administrator Extension. Configuration is acquired using a local configuration file.

Configuring Transport Layer Security (TLS) Connections (Optional)

Python provides the OpenSSL library that is used to establish TLS connections. The OpenSSL library that Python uses is not related to the OpenSSL library installed during installation of third-party libraries, which are used to encrypt muxed recording files.

Configuring TLS connection to Interaction Recording Web Services

1. Set up TLS on Interaction Recording Web Services (RWS). For more information, see Configuring TLS on the Server-Side for Interaction Recording Web Services section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**htcc**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use `https`.
3. In the [**htcc**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the

certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, then set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

Configuring TLS connection to Recording Crypto Server

1. Set up TLS on Recording Crypto Server. For more information, see Configuring an HTTP Port section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**rcs**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use the secure port.
3. In the [**rcs**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

Important

If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

Configuring TLS connection to WebDAV

1. Set up TLS on WebDAV. For more information, see Configuring TLS for the WebDAV Server section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**webdav**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set

trusted_ca to `true`.

- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

For more information about the Recording Muxer Script parameters, see the Genesys Interaction Recording Options Reference.

Starting the Recording Muxer Script

> **Important**
>
> For **muxer.cfg**, if **temp_dir** is configured, verify that the path exists and is writable by the muxer process.

To launch the Recording Muxer script, run the following command from the `<Recording Muxer Install Directory>` (where `x = 6`):

On Windows:

```
<python3.11.5 executable> muxer_process.py --config-file=muxer.cfg
```

On Linux:

```
env LD_LIBRARY_PATH=<untarred openssl directory>:$LD_LIBRARY_PATH <python3.11.5
 executable> muxer_process.py --config-file=muxer.cfg
```

By default the Recording Muxer's log file is stored in the working directory. This can be changed by specifying a preexisting folder in the **logfile_path** parameter in the **[logfile]** section of the configuration file. For example, in Windows:

logfile_path = `C:\logs\recordingMuxer`

Recording Muxer Script Legacy (Python 2) Deprecated

> **Important**
> Recording Muxer Script Legacy (based on Python 2) has been discontinued as of March 31, 2024.

Prerequisites

Before installing and configuring the Recording Muxer Script, you must have the following prerequisites:

- An Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) instance where the call recording and screen recording metadata is stored.
- A Recording Crypto Server instance to decrypt the encrypted recordings.
- Network access to the WebDAV storage where the recordings are stored.

Installing Recording Muxer Script

Installing on Windows

1. Install 32 bit Python 2.7.x from the Python website.
2. Install the Recording Muxer Script IP.

   Note: Install the following third party libraries in the order they appear.

3. Untar the `<Recording Muxer Install Directory>/thirdparty/setuptools-1.3.2.tar.gz` file.
4. From the `<Recording Muxer Install Directory>/thirdparty/setuptools-1.3.2` directory, run `python setup.py install`.
5. Untar the `<Recording Muxer Install Directory>/thirdparty/requests-2.4.1.tar.gz` file.
6. From the `<Recording Muxer Install Directory>/thirdparty/requests-2.4.1` directory, run `python setup.py install`.
7. Untar the `<Recording Muxer Install Directory>/thirdparty/boto-2.32.1.tar.gz` file.
8. From the `<Recording Muxer Install Directory>/thirdparty/boto-2.32.1` directory, run `python setup.py install`.
9. Untar the `<Recording Muxer Install Directory>/thirdparty/easywebdav-1.2.0.tar.gz` file.
10. From the `<Recording Muxer Install Directory>/thirdparty/easywebdav-1.2.0` directory, run `python setup.py install`.
11. Untar the `<Recording Muxer Install Directory>/thirdparty/filechunkio-1.5.tar.gz` file.
12. From the `<Recording Muxer Install Directory>/thirdparty/filechunkio-1.5` directory, run

`python setup.py install`.

13. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.1.7.tar.gz` file.
14. From the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.1.7` directory, run `python setup.py install`.
15. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-modules-0.0.5.tar.gz` file.
16. From the `<Recording Muxer Install Directory>/thirdparty/pyasn1-modules-0.0.5` directory, run `python setup.py install`.
17. Unzip the `<Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-2.4.3-win64-static-gpl3.0.zip`.
18. Unzip the `<Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.0.2j-win64.zip`. This OpenSSL library is used to encrypt the resulting muxed recording file when required.

> **Important**
>
> The following steps are only applicable for Muxer 8.5.265.66 or higher.

19. Untar the `<Recording Muxer Install Directory>/thirdparty/docutils-0.13.1.tar.gz` file.
20. From the `<Recording Muxer Install Directory>/thirdparty/docutils-0.13.1` directory, run `python setup.py install`.
21. Untar the `<Recording Muxer Install Directory>/thirdparty/six-1.10.0.tar.gz` file.
22. From the `<Recording Muxer Install Directory>/thirdparty/six-1.10.0` directory, run `python setup.py install`.
23. Untar the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.6.0.tar.gz` file.
24. From the `<Recording Muxer Install Directory>/thirdparty/python-dateutil-2.6.0` directory, run `python setup.py install`.
25. Untar the `<Recording Muxer Install Directory>/thirdparty/jmespath-0.9.1.tar.gz` file.
26. From the `<Recording Muxer Install Directory>/thirdparty/jmespath-0.9.1` directory, run `python setup.py install`.
27. Untar the `<Recording Muxer Install Directory>/thirdparty/botocore-1.4.57.tar.gz` file.
28. From the `<Recording Muxer Install Directory>/thirdparty/botocore-1.4.57` directory, run `python setup.py install`.
29. Untar the `<Recording Muxer Install Directory>/thirdparty/futures-3.0.5.tar.gz` file.
30. From the `<Recording Muxer Install Directory>/thirdparty/futures-3.0.5` directory, run `python setup.py install`.
31. Untar the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.1.10.tar.gz` file.
32. From the `<Recording Muxer Install Directory>/thirdparty/s3transfer-0.1.10` directory, run `python setup.py install`.
33. Untar the `<Recording Muxer Install Directory>/thirdparty/boto3-1.4.0.tar.gz` file.
34. From the `<Recording Muxer Install Directory>/thirdparty/boto3-1.4.0` directory, run `python setup.py install`.

Installing on Linux (RHEL)

1. Install Python 2.7.6 or later:
   - Download the software from the Python website.
     It is recommend that newer versions of Python are installed separately from an existing versions (do not update).
2. Install the Recording Muxer Script IP.

   Note: Install the following third party libraries in the order they appear.
3. Untar the `<Recording Muxer Install Directory>/thirdparty/setuptools-1.3.2.tar.gz` file.
4. From the `<Recording Muxer Install Directory>/thirdparty/setuptools-1.3.2` directory, run `python setup.py install`.
5. Untar the `<Recording Muxer Install Directory>/thirdparty/requests-2.4.1.tar.gz` file.
6. From the `<Recording Muxer Install Directory>/thirdparty/requests-2.4.1` directory, run `python setup.py install`.
7. Untar the `<Recording Muxer Install Directory>/thirdparty/boto-2.32.1.tar.gz` file.
8. From the `<Recording Muxer Install Directory>/thirdparty/boto-2.32.1` directory, run `python setup.py install`.
9. Untar the `<Recording Muxer Install Directory>/thirdparty/easywebdav-1.2.0.tar.gz` file.
10. From the `<Recording Muxer Install Directory>/thirdparty/easywebdav-1.2.0` directory, run `python setup.py install`.
11. Untar the `<Recording Muxer Install Directory>/thirdparty/filechunkio-1.5.tar.gz` file.
12. From the `<Recording Muxer Install Directory>/thirdparty/filechunkio-1.5` directory, run `python setup.py install`.
13. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.1.7.tar.gz` file.
14. From the `<Recording Muxer Install Directory>/thirdparty/pyasn1-0.1.7` directory, run `python setup.py install`.
15. Untar the `<Recording Muxer Install Directory>/thirdparty/pyasn1-modules-0.0.5.tar.gz` file.
16. From the `<Recording Muxer Install Directory>/thirdparty/pyasn1-modules-0.0.5` directory, run `python setup.py install`.
17. Untar the `<Recording Muxer Install Directory>/thirdparty/ffmpeg/ffmpeg-2.4.3-centos5-x86_64-static-gpl3.0.tar.bz2`.
18. Execute `chmod a+x ffmpeg` and `chmod a+x ffprobe`.
19. Untar the `<Recording Muxer Install Directory>/thirdparty/openssl/openssl-1.0.2j-centos5-x86_64.tar.bz2`. This OpenSSL library is used to encrypt the resulting muxed recording file when required.
20. Execute `chmod a+x openssl`.

> **Important**
> The following steps are only applicable for Muxer 8.5.265.66 or higher.

21. Untar the <Recording Muxer Install Directory>/thirdparty/docutils-0.13.1.tar.gz file.
22. From the <Recording Muxer Install Directory>/thirdparty/docutils-0.13.1 directory, run python setup.py install.
23. Untar the <Recording Muxer Install Directory>/thirdparty/six-1.10.0.tar.gz file.
24. From the <Recording Muxer Install Directory>/thirdparty/six-1.10.0 directory, run python setup.py install.
25. Untar the <Recording Muxer Install Directory>/thirdparty/python-dateutil-2.6.0.tar.gz file.
26. From the <Recording Muxer Install Directory>/thirdparty/python-dateutil-2.6.0 directory, run python setup.py install.
27. Untar the <Recording Muxer Install Directory>/thirdparty/jmespath-0.9.1.tar.gz file.
28. From the <Recording Muxer Install Directory>/thirdparty/jmespath-0.9.1 directory, run python setup.py install.
29. Untar the <Recording Muxer Install Directory>/thirdparty/botocore-1.4.57.tar.gz file.
30. From the <Recording Muxer Install Directory>/thirdparty/botocore-1.4.57 directory, run python setup.py install.
31. Untar the <Recording Muxer Install Directory>/thirdparty/futures-3.0.5.tar.gz file.
32. From the <Recording Muxer Install Directory>/thirdparty/futures-3.0.5 directory, run python setup.py install.
33. Untar the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.1.10.tar.gz file.
34. From the <Recording Muxer Install Directory>/thirdparty/s3transfer-0.1.10 directory, run python setup.py install.
35. Untar the <Recording Muxer Install Directory>/thirdparty/boto3-1.4.0.tar.gz file.
36. From the <Recording Muxer Install Directory>/thirdparty/boto3-1.4.0 directory, run python setup.py install.

Upgrading Recording Muxer Script

1. Backup the Recording Muxer Script installation directory including logs and configuration file.
2. Uninstall the Recording Muxer Script component.
3. Install the new Recording Muxer Script component.
4. Update the Recording Muxer Script configuration according to the standard installation procedures.

> **Important**
> Uninstalling the previous Recording Muxer Script is optional.

Configuring Recording Muxer Script

This section describes how to configure the Recording Muxer Script for your environment.

Configure Passwords (Optional)

> **Important**
> In a Linux or Windows environment, Muxer supports the use of environment variables instead of parameters in the configuration file for certain parameters. When both are available, the environment variable take precedence.
>
> The following definitions describe the mapping of the environment variables to the corresponding configuration parameter:
>
> - **HTCC_PASSWORD**—maps to the existing configuration parameter under the `htcc` section, `password` value.
> - **RCS_PASSWORD**— maps to the existing configuration parameter under the `rcs` section, `password` value.
> - **WEBDAV_PASSWORD**—maps to the existing configuration parameter under the `webdav` section, `password` value.

In a Windows only environment, Recording Muxer Script supports storing all passwords in a secure keystore instead of storing in plain-text in the **muxer.cfg** file.

1. From the **muxer** directory folder in the Recording Muxer installation folder (for example, **<Recording Muxer Install Directory>\muxer**), execute the following command:
   `python encryptPassword.py`
   The command will prompt for the appropriate values to be entered for the password/key configuration parameters. See the Genesys Interaction Recording Options Reference for the descriptions of the parameters.
2. Configure the **muxer.cfg** file leaving the following parameter values empty:

```
[webdav]
password =

[htcc]
password=

[rcs]
password =
```

Configuring the Connection to Interaction Recording Web Services (Web Services)

To configure the Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) connection, set the following parameters in the **[htcc]** section of the Recording Muxer **muxer.cfg** configuration file:

Parameter Name	Default Value	Description
base_uri		Specifies the host and port of the Interaction Recording Web Services (Web Services) server—for example, `https://<web services host>:<web services port>/`.
contact_center_id		Specifies the unique identifier of the contact center.
username	`ops`	Specifies the username used to access the Interaction Recording Web Services (Web Services) account.
password	`ops`	Specifies the password used to access the Interaction Recording Web Services (Web Services) account. **Note:** • If the "Configuring the Secure Password Storage" step was performed, leave this value empty. • The password can be overridden by the `HTCC_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Interaction Recording Web Services (RWS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Interaction Recording Web Services on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Parameter Name	Default Value	Description
rws_timeout	30	Specifies the timeout duration, in seconds, for Recording Muxer Script while sending a request to Interaction Recording Web Services. **Note:** The timeout value must be greater than or equal to 30.

Configuring the Connection to Recording Crypto Server

To configure the connection to the Recording Crypto Server, set the following parameters in the **[rcs]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
base_uri	Empty	Specifies the host and port of the Recording Crypto Server: `https://<Recording Crypto Server host>:<Recording Crypto Server port>`
username	Empty	Specifies the contact center admin username used to access the Recording Crypto Server account belonging to the contact center specified by the **contact_center_id** option in the **[htcc]** section. **Note:** The user must have the media decrypt permission.
password	Empty	Specifies the contact center admin password used to access the Recording Crypto Server account belonging to the contact center specified by the the **contact_center_id** option in the **[htcc]** section. **Note:** • If the Configuring the Secure Password Storage step was performed, leave this value empty. • The password can be overridden by the `RCS_PASSWORD` environment variable.

Parameter Name	Default Value	Description
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to Recording Crypto Server (RCS). Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to Recording Crypto Server on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Processing Commands

1. The Recording Muxer uses libraries for analyzing and handling multimedia data. To configure these commands, set the following parameters in the **muxer.cfg** file, the **[processing]** section:

   - **ffmpeg** = The path to the ffmpeg executable file.

   > **Important**
   > The ffmpeg executable is located under the directory where the thirdparty ffmpeg package was unzipped/ untarred.

   - **ffprobe** = The path to the ffprobe executable file.

   > **Important**
   > The ffprobe executable is located under the directory where the thirdparty ffmpeg package was unzipped/ untarred.

2. To enable Muxer to read multiple screen recordings metadata with one request, configure the following parameters using the **muxer.cfg** configuration file (optional):

   - **batch_read_screen_recording_metadata**: Determines how screen recording metadata is received. The new algorithm reads multiple screen recordings metadata in one request. The previous algorithm reads one request at a time.
     Valid Values: Using Bulk API = 1 / Using previous algorithm the integer <>1
     Default Value: 1
   - **query_slice_size**: Defines the maximum number of call recording records whose screen recordings should be queried.

Valid Values: all integers > 0
Default Value: 100

3. Configure the **openssl** parameter to set the path to the openssl executable.

> **Important**
>
> - The openssl executable is located under the directory where the thirdparty openssl package was unzipped/untarred.
> - On Linux, specifying the absolute path to the openssl executable path is recommended to ensure that the default installed openssl (for example, `/usr/bin/openssl`) is not executed instead.

4. Configure the **window_past** and **window_past_older_than** parameters to set the time in the past to search for the call recordings to multiplex with the screen recordings. See the "Configure HA" section for the recommended values for these parameters.

5. Configure the **clean_temp_folder_timeout** parameter in the **[processing]** section to determine how often the recording files are cleaned up in the **temp folder**. **clean_temp_folder** should only be configured when **auto_clean_temp_folder** is set to 1. By default the **clean_temp_folder** value is 43200 (that is, cleanup occurs every 12 hours). If this value is set to `-1`, Muxer will attempt to perform a cleanup when it is idle.

For more information about the **[processing]** section parameters, see the Genesys Interaction Recording Options Reference.

Configuring Sharding (Optional)

Sharding can be used to increase the capacity of the Recording Muxer Script solution. When configured, the muxing workload is divided among multiple active instances. By default, Sharding is disabled and `muxer_id = -1`.

When Sharding is in use, a Muxer instance can be configured to run in primary or in backup mode:

- In primary mode, the Muxer should be configured to query for call records from the last `n` minutes (`window_past_older_than=0`, `window_past=n minutes`), based on configuration in the `muxer.cfg` file for that instance.
- In backup mode, the Muxer should be configured to query for call records that are older than the last `n` minutes but newer than `m` minutes (`window_past_older_than= n`, `window_past= m` minutes), based on configuration in the `muxer.cfg` file for that instance.

Sharding is configured based on the following command line or configuration file parameters within the `[processing]` section:

- **muxer_id:** A unique Muxer ID.
  Valid values: A non-negative integer starting with 0 (the Muxer ID should be incremented by 1 for each additional instance).
  If you are not using Sharding, the value should be empty or `-1`.
- **total_muxers:** The total number of primary Muxer instances deployed (excluding the backup).
  Valid Values: `max(muxer_id) + 1`
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.
- **muxer_type:** indicates if the Muxer is operating in primary mode or backup mode.
  Valid Values: primary, backup
  If you are not using Sharding, (indicated by `muxer_id` not being set, or being set to `-1`), the Muxer ignores this value.

To specify Sharding parameters using the command line, the following arguments are used:

- muxer-type
- muxer-id
- total-muxers

Note: The Sharding parameter values passed in the command line overrides the corresponding values specified within the configuration file. The following is the supported command line:
`python.exe muxer_process.py --config-file=CONFIG_FILE --muxer-type=MUXER_TYPE --muxer-id=MUXER_ID --total-muxers=TOTAL_MUXERS`

For example: When using the following values, the system will have two instances of Muxer running:

- muxer_type=primary
- muxer_id=0 (for the first instance)
- muxer_id=1 (for the second instance)
- total_muxers=2

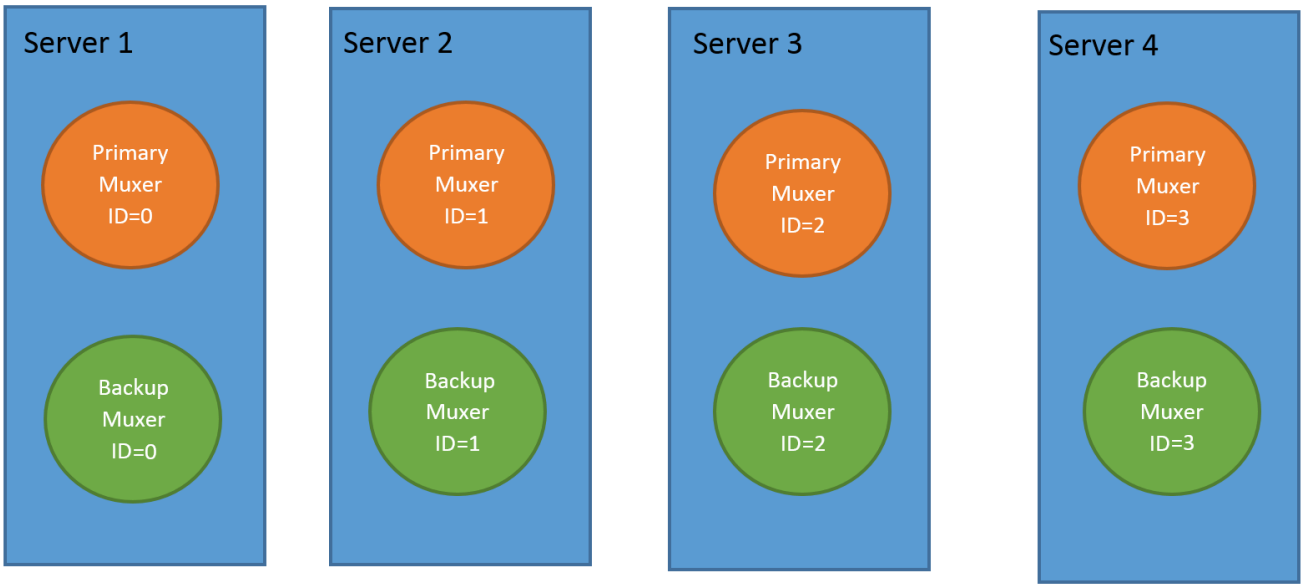
The following is the command line example for running the first instance: `python.exe ../muxer/muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=0 --total-muxers=2`

The following is the command line example for running the second instance: `python.exe ../muxer/muxer_process.py --config-file=muxer.cfg --muxer-type=primary --muxer-id=1 --total-muxers=2`

Note: When there are multiple instances of Muxers deployed on the same machine, then, a different **temp_dir** value for each instance of the Muxer must be configured in the `[processing]` section of the `muxer.cfg` file, and each Muxer instance must use a separate Muxer configuration file. This avoids the issues of one Muxer deleting the temporary files for the other instances.

> Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that each active primary instance be run on a separate machine. For a high availability deployment, a primary instance and a backup instance can be run on the same machine; however, in this case the instances should be configured so that the node IDs overlap (so that a single machine does not provide primary and backup coverage for the same `muxer_id`).



When running in backup mode, the Muxer will automatically calculate the `muxer_id` to be used to support this deployment mechanism, based on the specified `muxer_id`. The configured `muxer_id` used for the backup instance should match the `muxer_id` that is configured for the primary instance on the same machine, if both primary mode and backup mode instances are deployed together. For example, if `muxer_id=2` and `total_muxers=4` in the Muxer configuration file:

- When `muxer_type=primary`, the `muxer_id` used will be 2.
- When `muxer_type=backup`, the `muxer_id` used will be 3.

Important

If a Muxer instance is added or removed:

- The `total_muxers` value must be changed for each existing Muxer instance.
- All muxer instances must be restarted.
- Before starting the Muxer application, create and configure the **temp_dir** and **logfile_path** folders for both the Primary Muxer instance and the Backup Muxer instances running on the same machine.

Configuring High Availability (HA)

Important

The content in the Configure HA tab only applies if the Sharding configuration is not in use (see: Configure Sharding (Optional) tab). If Sharding is in use, refer to the high availability configuration described in the Configure Sharding (Optional) tab.

Recording Muxer Cluster

The Recording Muxer Script provides High Availability support using multiple instances of the Recording Muxer Script (all active). HA supports:

- Active/active pairs with the aim to load balance equally between the Recording Muxer nodes by splitting and configuring the time window on each node, so that it is close to equal the number of recordings found on each time window.
- When one of the node dies, recordings are still multiplexed.

Limitations:

- If the node with time window, now - N/2, dies, multiplexing will still occur, but a slower rate since the second node's time window is from N/2 to N.
- If the node with time window, N/2 - N, dies, screen recordings that are uploaded with the delay more than N/2 might not be multiplexed.
- Nodes should be configured so that the time windows are exclusive of each other, otherwise it may result in two multiplexed files being uploaded.

To configure HA:

1. In each Recording Muxer's **muxer.cfg** configuration file, in the **[processing]** section, set the following values for each node. For example,
   - On first node:
     - **window_past**= `720`
     - **window_past_older_than** = `5`

2. On second node:
   - **window_past** = `1440`
   - **window_past_older_than** = `725`

   The above will multiplex all recordings that were recorded within the last 1 day.
3. As a general rule, if the screen recording upload occurs with a delay of *N*, the configuration on each node can be set to:
   - On first node:
     - **window_past** = *N* `/ 2`
     - **window_past_older_than** =
   - **min-poll_interval** = `N/200`
4. On second node:
   - **window_past**= *N*
   - **window_past_older_than** = *N* `/ 2`
   - **min-poll_interval** = `N/200`

   Ensure that all Recording Muxer instances have the same configuration other than the above.

Important

- Genesys recommends that the maximum window length configured in each Recording Muxer Script instance be 12 hours (720 minutes). That is, the difference between the **window_past** and **window_past_older_than** parameters should be a maximum of 720 minutes. If the window length is greater than 12 hours, the configuration may cause problems with Elasticsearch.
- Genesys recommends that multiple Recording Muxer instances be deployed on different hosts to provide better HA and also not to have machine resource contentions.
- If the recording upload is delayed by more than the time window configured for the Recording Muxer Script, it is possible that the recording will be outside of the processing window and not be multiplexed. For such cases, the Recording Muxer Script can be run as a migration tool to batch process the records matching any desired criteria. For more information see the **call_recording_query_string** parameter under **Configuring the Advanced Options** in the **Advanced Configuration** tab.
- If the screen recording upload is delayed longer than 24 hours, configure a separate Muxer instance or Muxer sharding group for every 12 hours. When the Screen Recording Service is provisioned to upload files during non-business hours, the actual delay can be a couple of days if the agent workstation is shut down when the agent signs off from the Agent Desktop.

Configuring the Connection to WebDAV

To configure the connection to WebDAV, set the following parameters in the **[webdav]** section of the Recording Muxer **muxer.cfg** file:

Parameter Name	Default Value	Description
username	Empty	Specifies the username to allow read/write access to the WebDAV storage server.
password	Empty	Specifies the password to allow read/write access to the WebDAV storage server. **Note:** • If multiple WebDAV storage are used for same contact center region, make sure to use the same username and password. • If the "Configuring the Secure Password Storage" step was performed, leave the password value empty. • A password can be overridden by the `WEBDAV_PASSWORD` environment variable.
trusted-ca	`false`	Configures TLS certificate validation when making a secure outbound connection to WebDAV. Valid values are `true`, `false`, and the path to a trusted certificate authority (CA) bundle. The CA file must be in PEM format. Muxer will exit during initialization under the following conditions: CA path does not exist, CA file is not a valid PEM file, or CA file is corrupted. For more information, see Configuring TLS connection to WebDAV on the Configuring Transport Layer Security (TLS) Connections (Optional) tab.

Configuring the Advanced Options

The following advanced options can be configured in the **[advanced]** section of the **muxer.cfg** file:

- **worker_threads** = The number of parallel processing threads.
- **pagination** = The maximum number of records returned with each Interaction Recording Web Services (or Web Services if you're using version 8.5.210.02 or earlier) query.

- **max_overlap_allowed** = The overlap time before truncating.
- **video_padding_slice_length_ms** = If the video starts later or ends earlier than the audio, set the duration needed to prepend or append a padded video slice. Genesys recommends to set it to `5000`.
- **mark_screen_recording_label** = Whether to apply the label "screenRecording" to the associated call recording metadata after muxing. This configuration is optional. The default value is 1.
- **call_recording_extra_query_string** = Used to specify parameter value pairs other than `startTime`, `endTime`, and `limit`.
  If left empty, the **call_recording_extra_query_string** value will be defaulted internally to `userData=SRSScreenRecordingStateStarted>anAndScroll=true`, if the RWS version is >= 8.5.201.14, otherwise, it remains an "" (empty string).
  Specify "disable" (without quotes) to force it to be an empty string without checking the RWS version.
  When the final value of this configuration is not empty, the Recording Muxer Script will continually poll for records that match the searching criteria according to the final value of the configuration that should be processed.
  Genesys recommends that this parameter be left empty. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`.
  The following table describes values (query parameters) that are available (except `startTime` and `endTime`).
- **call_recording_query_string** = When not empty, `[call_recording_query_string]` queries Interaction Recording Web Services (Web Services) with the given string for records to process. Instead of continually polling for records to process, the Recording Muxer script will exit once the returned records are processed. Genesys recommends that this parameter be left empty unless the Muxer script is to be used for batch migrating the old recordings. Query parameters have to be formatted as: `<parameter name>=<value>[&<parameter name>=<value>…]`. The following table describes values (query parameters) that are available:

Parameter Name	Description
callerPhoneNumber	Retrieves all recordings which apply to any call containing the specified ANI attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard which can substitute any number of any symbols in the request. Search is case-sensitive.
dialedPhoneNumber	Retrieves all recordings which apply to any call containing the specified DNIS attribute. The exact match of stored number (alphanumeric-only) and request parameter (alphanumeric-only) is used. The request string can contain * wildcard - which can substitute any number of any symbols in request. Search is case-sensitive.
startTime	Retrieves all recordings that started >= the specified time.
endTime	Retrieves all recordings that ended <= the specified time.
userName	Retrieves all recordings in eventHistory->contacts of which the passed userName/ firstName/Lastname is present. User can use wildcards to specify only part of the username/ firstname/lastname. If more than 1 word is used (divided by spaces) -the records containing any of provided terms as username, firstname or lastname will be included. If user wants to

Parameter Name	Description
	retrieve records containing ALL terms - the AND keyword should be used. Sample: ?userName= Alice AND Amber - will seek for recording with events->contact-> username/firstName/ lastName containing Alice and Amber (possible - in different users). Search is case-insensitive.
userData	Retrieves all recordings in eventHistory->data of which the passed userData is present as value of HashMap. These matches are supported: • Exact match - match the entire value (for example, "tom" will find "tom"). • Wildcarded value (for example, "tom*" will find a record with "tomas"). • Combination of matches - If the query terms are separated by spaces (for example, "tom jerry" will look for recordings that contain "tom" or "jerry").

Configuring the Recording Muxer Using Genesys Administrator Extension (Optional)

The Recording Muxer uses a configuration file instead of a specific application object in Configuration Server. However, it is possible to configure the Recording Muxer as a "third-party server" application enabling Genesys Administrator Extension to monitor, start, and stop the process.

The following steps describe how to setup Recording Muxer as a "third party server" application in Genesys Administrator Extension. For more information, see the *Using the Management Layer* section of the Framework 8.5.1 Management Layer User's Guide

Configuring Recording Muxer Script to Start/Stop via LCA using Genesys Administrator Extension:

1. Install and deploy the latest Recording Muxer script.
2. Make sure that the Local Control Agent (LCA) is running.
3. Create a new application template in Genesys Administrator Extension called `Recording Muxer script` of type `Third Party Server`.
4. Create a new application (for example, `myRecordingMuxer`) in Genesys Administrator Extension using this new application template.
5. On Windows:
   a. Set the `Command Line` parameter to the python executable (for example, `C:\Python27\python.exe`).
   b. Set the `Host` parameter in the application's server info to the correct Host object.
   c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>\muxer` directory. For example, `C:\Program Files\GCTI\Recording Muxer Script\muxer`.
   d. Set the `Command Line Arguments` parameter to the python arguments: `muxer_process.py --config-file=muxer.cfg`.
6. On Linux:

a. Set the `Command Line` parameter to `env`.

b. Set the `Host` parameter in the application's server info to the correct Host object.

c. Set the `Working Directory` parameter to the `<Recording Muxer Install Directory>/muxer` directory. For example, `/opt/genesys/Recording_Muxer_Script_8.5/muxer/`.

d. Set the `Command Line Arguments` parameter. The `LD_LIBRARY_PATH` must be set to include the openssl binary directory before muxer script execution. For example, `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<untarred openssl directory> /opt/python27/python muxer_process.py –-config-file=muxer.cfg`.

> **Important**
>
> The Recording Muxer does not support configuration through Genesys Administrator Extension. Configuration is acquired using a local configuration file.

Configuring Transport Layer Security (TLS) Connections (Optional)

Python provides the OpenSSL library that is used to establish TLS connections. To use a newer version of OpenSSL, upgrade the version of Python being used (within the 2.7.x family). The OpenSSL library that Python uses is not related to the OpenSSL library installed during installation of third-party libraries, which are used to encrypt muxed recording files.

Configuring TLS connection to Interaction Recording Web Services

1. Set up TLS on Interaction Recording Web Services (RWS). For more information, see Configuring TLS on the Server-Side for Interaction Recording Web Services section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**htcc**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use `https`.
3. In the [**htcc**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, then set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject

alternative name.

Configuring TLS connection to Recording Crypto Server

1. Set up TLS on Recording Crypto Server. For more information, see Configuring an HTTP Port section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**rcs**] section of the Recording Muxer Script configuration file, set the **base_uri** parameter to use the secure port.
3. In the [**rcs**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

Configuring TLS connection to WebDAV

1. Set up TLS on WebDAV. For more information, see Configuring TLS for the WebDAV Server section. For information on acquiring TLS certificates and private keys, see Genesys Security Deployment Guide.
2. In the [**webdav**] section of the Recording Muxer Script configuration file, configure the **trusted_ca** parameter as follows:

- If the TLS certificate was issued by a well-known certificate authority such as Verisign, set **trusted_ca** to `true`.
- If the TLS certificate was issued by a certificate authority, set **trusted_ca** to the path of the CA certificate. The file containing the certificate must be in PEM format.

> **Important**
>
> If there are intermediate certificate authorities forming a chain of trust, then append all certificates in the chain into a single file. All the files containing certificates must be in PEM format. The file should have the certificates in order of lowest in the chain to the root of the chain. The root certificate authority should be the

last certificate listed in the file. Set **trusted_ca** to the path to this file.

- If the TLS certificate is a self-signed certificate, set **trusted_ca** to the path of the CA that generated the self-signed certificate. The file containing the certificate must be in PEM format.
- If you do not wish to verify the TLS certificate and use TLS only for encrypted transmission, set **trusted_ca** to `false`. If certificate verification is not configured, certificates will not be checked if they have expired or the server hostname is matching the certificate's common name or subject alternative name.

For more information about the Recording Muxer Script parameters, see the Genesys Interaction Recording Options Reference.

Starting the Recording Muxer Script

Important

For **muxer.cfg**, if **temp_dir** is configured, verify that the path exists and is writable by the muxer process.

To launch the Recording Muxer script, run the following command from the <Recording Muxer Install Directory> (where x = 6):

On Windows:

```
<python2.7.x executable> muxer_process.py --config-file=muxer.cfg
```

On Linux:

```
env LD_LIBRARY_PATH=<untarred openssl directory>:$LD_LIBRARY_PATH <python2.7.x
 executable> muxer_process.py --config-file=muxer.cfg
```

By default the Recording Muxer's log file is stored in the working directory. This can be changed by specifying a preexisting folder in the **logfile_path** parameter in the **[logfile]** section of the configuration file. For example:

logfile_path = `C:\logs\recordingMuxer`