Swift News

MOS Gateway manual

DATE	28/02/2017
PRINCIPAL AUTHOR	Sean Kirwan
SECONDARY AUTHORS	Justin Avery
VERSION	1.0
UPDATE	09/06/22

Overview

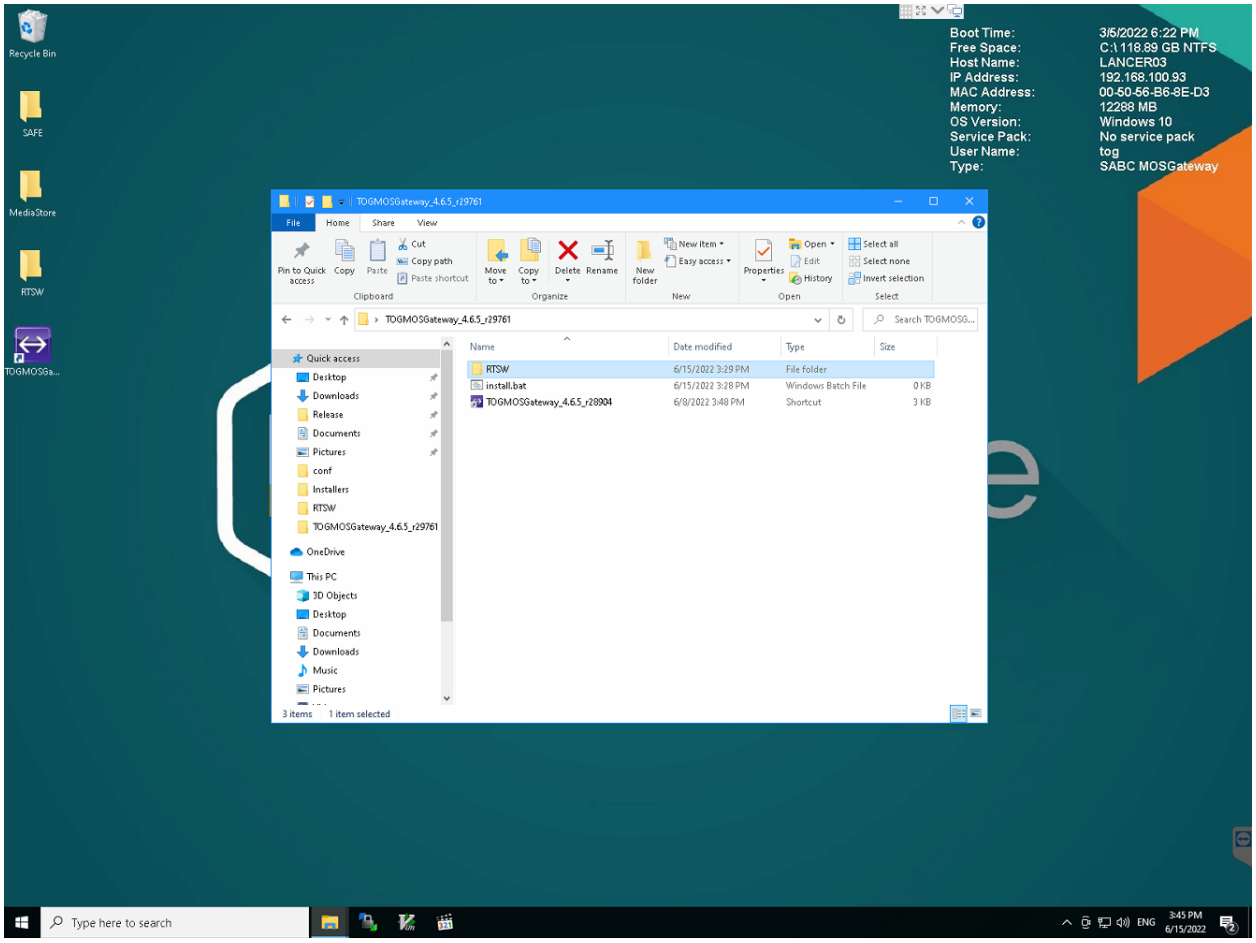
This document describes the RT Newsroom product **RT Newsroom: MOSGateway**.

The purpose of the MOS Gateway application is to connect to the NRCS server, retrieve current running orders and handle any MOS commands received from the NRCS server. These running orders are compiled by the journalists and contain graphic templates inserted into the running order stories using the tOG NRCSActiveX for the NRCS Desktop Client. These running orders are then converted into stacks and pages which can be used by RT playout applications (Swift Playout and Swift Live).

Related Manuals

- Swift CG
- Swift CG+
- Repository
- RT Newsroom
- Swift Engine
- MediaWatcher
- DataServer
- RT Newsroom: MOSGateway (this manual)
- RT Newsroom: ActiveX
- Swift Live
- Swift Playout
- Swift Engine

Installation



The MOS Gateway application is installed by copying a zip file for the appropriate version to the desktop, unzipping it and running the install.bat file it contains. The bat file will create and populate the RTSW folder and copy a launcher icon to the desktop. The RTSW folder will be created in C:/ProgramData/RTSoftware if that folder exists - otherwise it will be created on the desktop. The RTSW/conf folder contains the configuration file - TOGMOSGateway.conf. The RTSW/logs folder contains the logs for the application (unless specified in the configuration file).

It will be necessary to edit the configuration file and at least set the ncsId, ncsHostName, remoteRundownTarget and remoteMOSObjectTarget.

Operation

The application is runup from a desktop icon. It reads all its parameters from a configuration file (see below) and connects to the NRCS using the url specified. It creates four sockets (server/client for the upper/lower ports) and indicates the status of the connections on the interface (using grey/green/red buttons). The **reqMachInfo** MOS command is sent to the NRCS.

The application heartbeats on all the sockets at a configurable period (if there is no other message). The NRCS will send messages when running orders become current, are modified, deleted or set as ready-to-air. The application reads, processes and replies to these messages. They are saved to log files in **<homeDirectory>/logs/TOGMOSGateway.log** and listed on the interface.

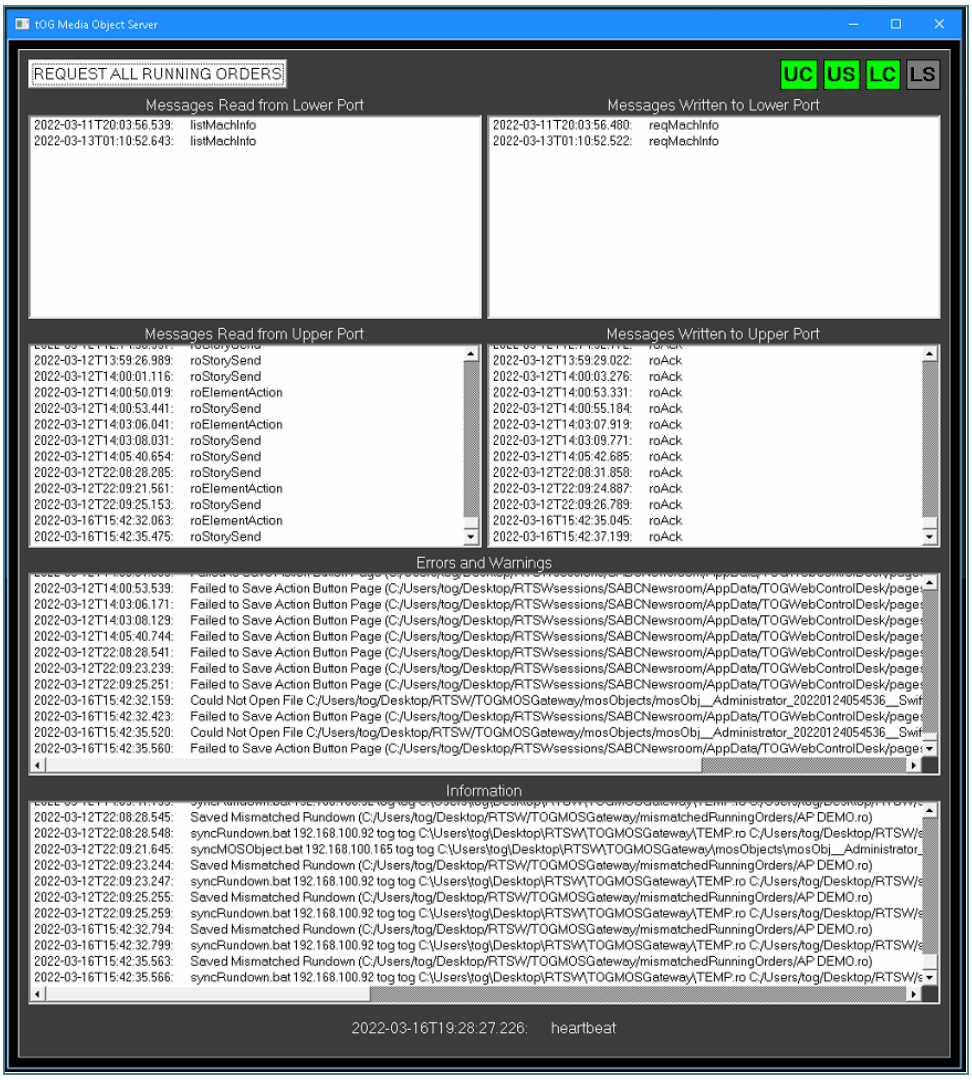
The application maintains (on the data/preview server) a local folder of running orders in **<homeDirectory>/TOGMOSGateway/runningOrders** and another folder of MOS objects in **<homeDirectory>/TOGMOSGateway/mosObjects**.

These are also converted to stacks and pages and saved locally (to **<homeDirectory>/projects/<projectName>/GMScripts/Stacks** and **<homeDirectory>/sessions/<sessionName>/AppData/TOGWebControlDesk/pages)** every time a running order is changed.

If there is a locally running DataServer, it will serve up the pages directly to the Swift Live control app. If not, they are sent over FTP using WinSCP. The stacks are synchronised to playout control desktops running Swift Playout. The pages are synchronised to DataServer platforms (using the syncMOSObject.bat script in the RTSW/bin folder). The mosObjects are synchronised to DataServer platforms (using the syncRundown.bat script in the RTSW/bin folder). These synchronizations are listed in the Information tab.

The application also processes roCtrl messages. The MOSGateway finds the running order, story and item specified in the message (from the files in the runningOrders folder) and extracts the mosPayload from the item. This contains the name of a graphic template and a list of parameter values. It then sends messages to a machine hosting a Swift Engine with the url specified by the parameter liveHostName to run methods in the graphic (possibly using the parameters). The values for the command tag in the roCtrl message READY, EXECUTE and STOP map to cueGraphic, bringOn and takeOff methods.

Interface



Request All Running Orders Button

This button will request all currently open running orders from the NRCS - it sends a roReqAll command. This is answered with an roListAll message from the NRCS. The application then sends a roReq for each running order listed in the roListAll message. The NRCS then replies with an roList message containing the full details of the running order which is saved to disk.

UC Button/US Button/LC Button/LS Button

These buttons indicated the status of the four sockets the application

opens with the NRCS - white for unconnected, green for connected and red for disconnection.

Messages Read from Lower Port ListBox
Messages Read from Upper Port ListBox
Messages Written to Lower Port ListBox
Messages Written to Upper Port ListBox

All messages sent to the NRCS or received from the NRCS are listed here with the time and type of command. The full message is written to the log file.

Errors and warnings
Information

Any errors or warnings are displayed here. These may report on a failure to connect to the NRCS, a badly formed message or failure to save pages/stacks etc. The Information box displays any file successfully saved locally or synchronised to remote servers.

Heartbeat

This shows the time of the last heartbeat.

Configuration File

The file is called TOGMOSGateway.conf and located at **/RTSW/conf.**

homeDirectory
rtswDirectory

This is **/Desktop/RTSW** by default. It contains the TOGMOSGateway folder. All the running orders and MOS objects.

mosID

The ID of the MOS gateway - **rtsw.togmosgateway.fulham.rtsw.mos**.

nrcsID

The ID of the NRCS.

nrcsHostName

The ip address or url of the machine hosting the NRCS server.

liveHostName

The ip address of the machine hosting a Swift Engine to be controlled by the NRCS via the MOS Gateway to playout graphics to air.

sessionName

The name of the session (from the Swift Live) to which action button pages derived from running orders will be saved.

projectName

The name of the project to which stacks derived from running orders will be saved.

logFileName

The name (and location) of the log file. Default value: RTSW/logs.

lowerPort
upperPort

The value of the lower and upper ports (these rarely have to be

changed). Default value: 10540, 10541

vendor

Some vendors include unique tags in their messages.

heartbeatsPeriod

This specifies how often a heartbeat message is sent to the NRCS. Default value: 200

requireNCSConnectToLowerPort
requireNCSConnectToUpperPort

If only some profiles are supported by the NRCS (or not needed by RT applications), then it may not be necessary to connect to all the NRCS ports. Default value: true, true.

extractSessionNameFromROName

When saving a page from a running order, dont use the session name specified within the running order or the one in the configuration file, use the running order name. Default value: false.

replyToHeartbeat

This suppress replies to hDefault value: true

checkIfAlreadyRunning

If this is set to true then the application will check if another MOS Gateway is running, inform the user and exit. Default value: false.

remoteRundownTarget
remoteMOSObjectTarget

These specify the machines to which to synchronise the rundowns and mosObjects. They consist of a comma separated list of “identifiers”. Each identifier is a vertical-bar separated list - the first item is the ipAddress of the remote machine, the second is the destination folder on that machine. It is also possible to include a username and password (for example, 192.168.100.121|tog|tog|C:/Users/tog/Desktop/RTSW).

Profile Support

The MOS Gateway supports MOS Protocol v2.8.4.

Profile 0

- heartbeat - heart-beating on all sockets.
- reqMachInfo - returns listMachInfo.
- listMachInfo - used to reply to a reqMachInfo from the NRCS.

Profile 1

- mosReqObj - read the mosObject file and return if it exists; otherwise mosAck.
- mosObj - used to reply to a mosReqObj from the NRCS.
- mosReqAll - read the mosObject files and return in a mosListAll.
- mosListAll - used to reply to a mosReqAll from the NRCS.

Profile 2

- roCreate - saves the running order to a file and returns roAck.
- roReplace - saves the running order to a file and returns roAck.
- roDelete - deletes the running order file and returns roAck.
- roReq - returns the running order if the file exists; otherwise roAck with status of NACK.
- roList - saves the running order in the messages to a file.
- roMetadataReplace - unused; returns roAck.
- roElementStat - unused; returns roAck.

- roElementAction - the running order file is modified and saved; returns roAck.
- roReadyToAir - creates a readyToAir file for the running order; returns roAck.

These messages are included in the protocol for backwards compatibility with previous versions of the protocol:

- roStoryAppend - adds the supplied story to the file of the running order specified; return roAck.
- roStoryInsert - inserts the supplied story into the file of the running order specified; return roAck.
- roStoryReplace - replaces the specified story with the supplied story in the file of the running order specified; return roAck.
- roStoryMove - unused.
- roStoryMoveMultiple - unused.
- roStorySwap - unused.
- roItemInsert - unused.
- roItemReplace - unused.
- roItemMoveMultiple - unused.
- roItemDelete - unused.
- roStat - unused.
- roItemStat - unused.

Profile 3

- mosObjCreate - derive a mosObject from the message and save to a file; returns mosAck and the mosObj to the NRCS.
- mosItemReplace - unused.
- mosReqObjList - unsupported; returns mosAck.
- mosReqSearchableSchema - unsupported; returns mosAck.

- mosListSearchableSchema - unsupported; returns mosAck.
- mosReqObjList - unsupported; returns mosAck.
- mosObjList - unsupported; returns mosAck.
- mosReqObjAction - unsupported; returns mosAck.

Profile 4

- roReqAll - returns roListAll containing all the running order files.
- roListAll - used to reply to a roReqAll from the NRCS.
- roStorySend - replace the story in the running order file; returns roAck.

Profile 5

- roItemCue - unsupported; returns roAck.
- roCtrl - passes on commands to Swift Engine on liveHostName; return roAck

Profile 6

- unsupported

Profile 7

- roReqStoryAction - unused.