Swift News

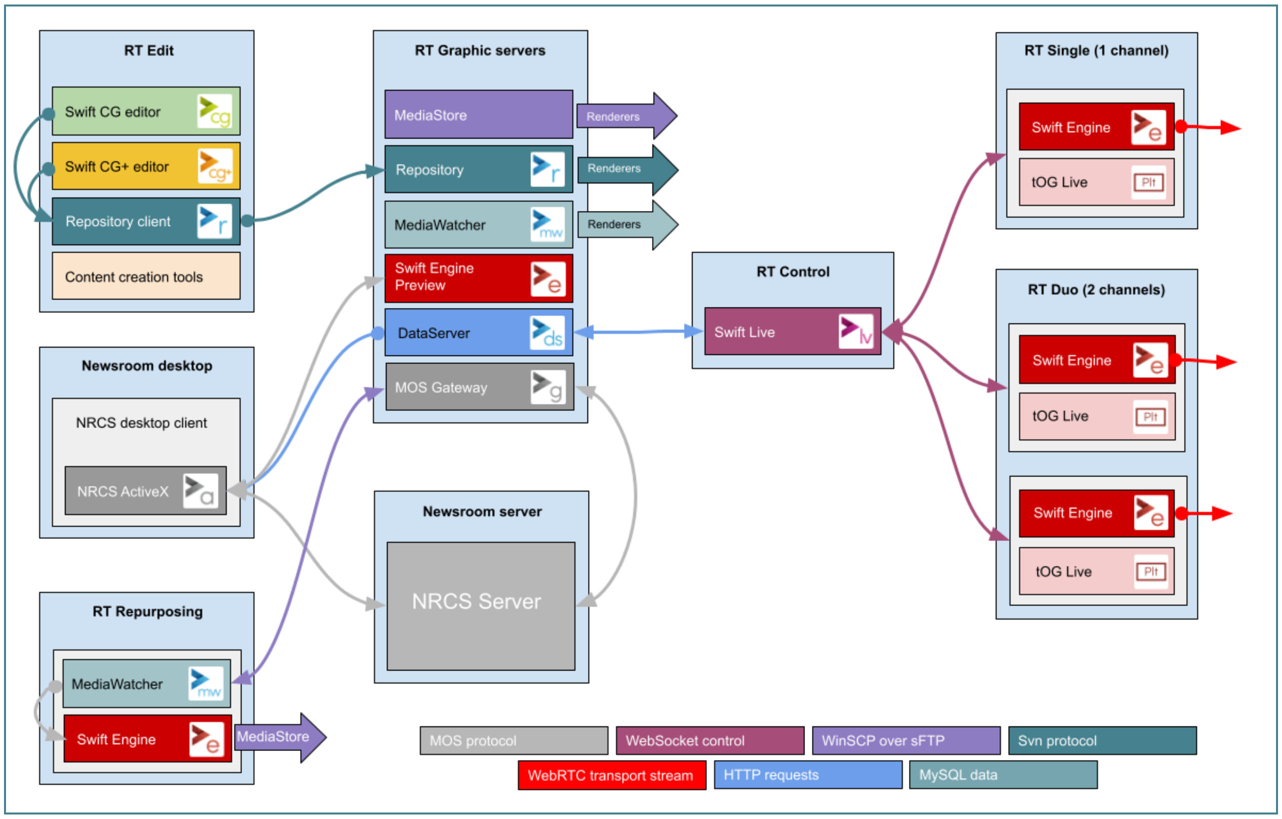
Manual and overview

ORIGINAL DATE	28/02/2017
PRINCIPAL AUTHOR	Sean Kirwan
SECONDARY AUTHORS	Justin Avery
VERSION	1.0; 2.0
UPDATE DATES	17/06/22

Overview

This document sets out the overall configuration and the constituent applications of the product. It follows the life-cycle of a graphic template from creation to transmission - from the creation of graphic templates by the design department through the completion of those templates by journalists at their desktops finishing in the transmission of the completed templates manually in a gallery or using automation systems and the MOS protocol.

The newsroom system is a distributed system. It consists of many disparate applications running on a network of computers. The diagram below sets out the platforms, applications and protocols involved. It does not represent an actual newsroom. There is for instance only one journalist desktop platform - in an actual newsroom there would be many.



Data and control flow in the diagram above moves from left to right - creation on the left, storage/distribution on the centre-left, control on the centre-right and playout on the right.

Note: RT graphic servers are grouped into a single box in the diagram but would usually be run on several boxes - so the MediaStore would be run on an RT MediaStore platform etc. The following will refer to these separate platforms. A platform here means a computer that will host one or more RT applications. For example, the **RT DataServer** will host the DataServer and DataClient applications - see below for platform descriptions.

The workflows are as follows:

1. Designers and producers use Swift CG and Swift CG+ to create graphic templates on the **RT Edit** platform (on the left of the diagram). They use the Repository application to store these assets in the Visual SVN repository on the **RT Repository** platform (part of RT Servers).

2. Journalists incorporate the graphic templates into running orders using the **RT NRCS ActiveX** (on the left of the diagram) plugged into the NRCS Journalist Client. These rundowns are then sent to the NRCS server.

3. The NRCS server will send these rundowns (or any modification of them) to a MOSGateway instance (which is registered with it a MOS device) running on the **RT MOSGateway** platform. This instance will in turn synchronise the new/updated rundowns to the **RT DataServer** platform and the new/updated mosObjects to the **RT Repurposing** platform.

4. The **RT ActiveXPreview** platform has the specific purpose of generating final frame previews of graphics configured by the **RT NRCS ActiveX**. It runs multiple Swift Engine renderers.

5. The **RT Repurposing** platform will layoff the graphics described in mosObjects created in 2 as clips (using a combination of the MediaWatcher and Swift Engine applications) and will send these clips to the **RT MediaStore** platform.

6. The **RT MediaStore** platform hosts a folder - assets written to this folder will be automatically synchronised to the appropriate machine. For example, images/movies and data files will be copied to the renderers (to be used when graphics are rendered out) using WinSCP over sFTP.

7. The **RT MediaWatcher** platform runs the MediaWatcher application which continually ingests external data and “puts” it in a location accessible to the renderers (either for instance, a local MySQL database or the **RT MediaStore** platform).

8. The DataServer instance running on the **RT DataServer** platform serves up the HTML5 control app Swift Live to a chrome instance running on an **RT Control** platform.

Related Manuals

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

Applications

This section lists and describes the applications used in the RT Newsroom product. These applications can be used in smaller subsets (for example, Swift CG and Swift Playout can be used together as a Character Generator - a simple low cost platform for creating and playing out graphics) but work together in a distributed fashion to provide newsroom functionality.

The descriptions below are cursory and meant as a guide. Please refer to the manuals for the individual applications for a full description of the application's features and functionality.

All systems have a "Basic Install" which consists of the following:

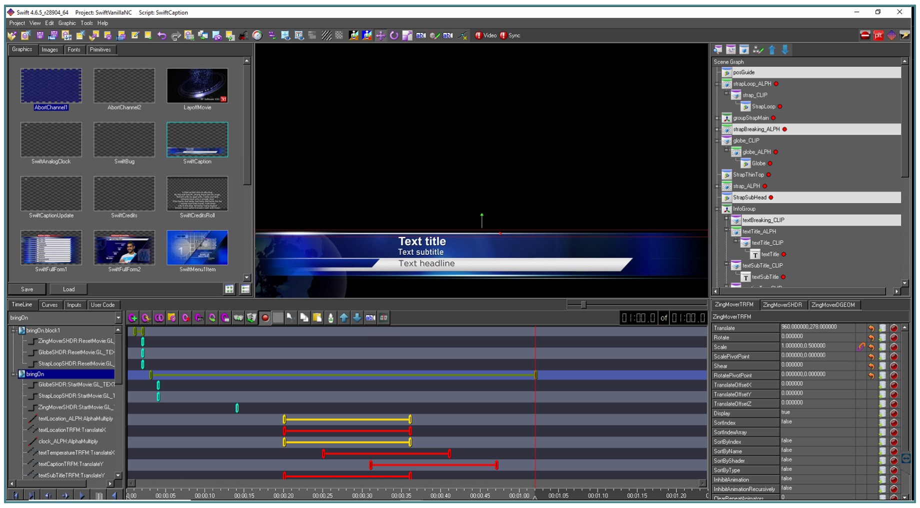
DataClient: All systems will have a DataClient service installed. The Repository application sends this service commands to update the local version of the graphic project etc.

WinSCP and sFTP: All systems have the OpenSSH service enabled - this will run an sFTP server. Applications on other machines can use WinSCP (also installed) to transfer data and asset files to any other machine.

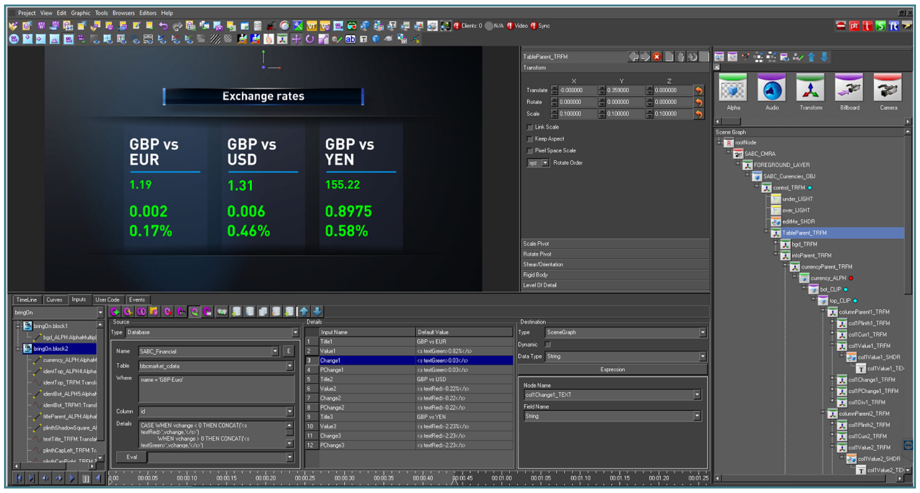
Miscellaneous: The following utility applications are installed as part of the basic install: Ruby, 7zip, MPC-HC, Tortoise, Subversion, Vim, Notepad++, ffmpeg, and wget.

Swift CG and Swift CG+

Swift CG creates 2D graphics with simpler structure and more limited data requirements.



Swift CG+ is the full editor and can be used to create any form of broadcast graphic of any complexity.



These are the standard RTSW tools for creating and authoring graphics. The final outcome from using these tools is a graphics project. This is a folder which contains all the assets used by the graphics (font, images, geometries etc.) and the scripts and manifests that encapsulate the structure and behaviour of the graphics.

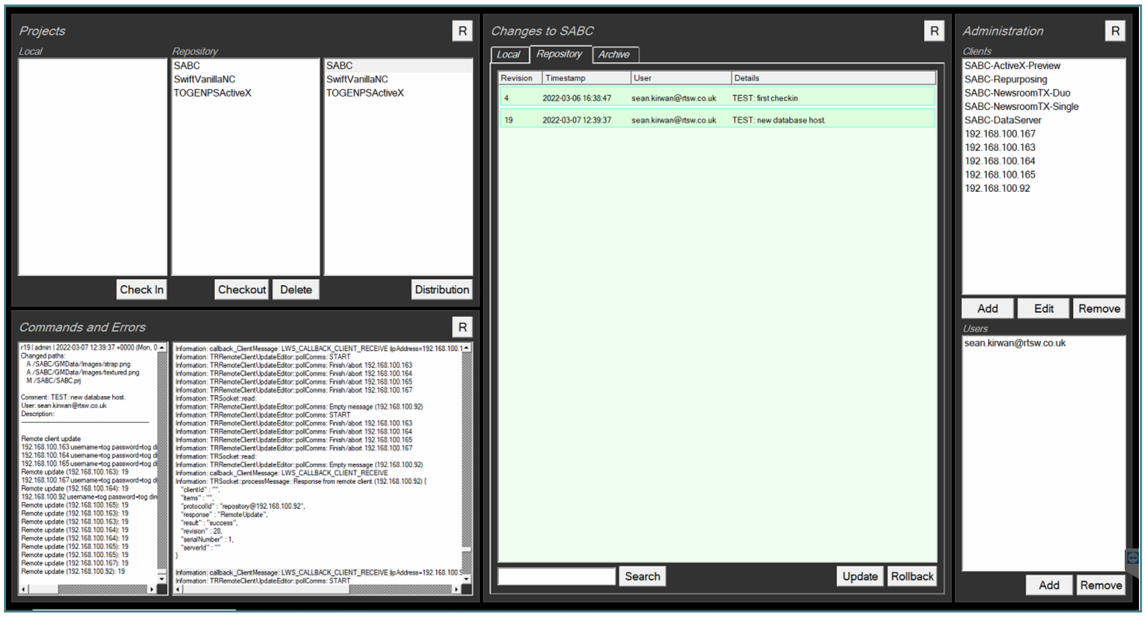
Repository

RT divides graphic assets into three kinds:

- assets that make up a project of graphics which have a very long life, typically being current until the look of a program or channel is redesigned). These are managed by the Repository application.

- image/movie assets that are relevant for just a single day. These are distributed by being dragged onto the Swift Live interface (see below). This sends them to the DataServer application which in turn sends them to the renderers for immediate use.

- image/movie assets that are relevant for a long period of time - images of politicians, sports club logos would be examples. These are usually handled by a MAM but RT provides a simple solution using a central folder of assets called MediaStore which is synchronised using WinSCP over sFTP to similar folders on other renderers and servers.



SABC
SwiftVanillaNC
TOGENPSActiveX

SABC
SwiftVanillaNC
TOGENPSActiveX

Revision	Timestamp	User	Details
4	2022-03-06 16:38:47	sean.kirwan@rtsw.co.uk	TEST: first checkin
19	2022-03-07 12:39:37	sean.kirwan@rtsw.co.uk	TEST: new database host.

SABC-ActiveX-Preview
SABC-Repurposing
SABC-NewsroomTX-Duo
SABC-NewsroomTX-Single
SABC-DataServer
192.168.100.167
192.168.100.163
192.168.100.164
192.168.100.165
192.168.100.92

sean.kirwan@rtsw.co.uk

```
r19 | admin | 2022-03-07 12:39:37 +0000 (Mon, 0
Changed paths:
   A /SABC/GMData/Images/strap.png
   A /SABC/GMData/Images/textured.png
   M /SABC/SABC.prj

Comment: TEST: new database host.
User: sean.kirwan@rtsw.co.uk
Description:
------------------------------------------------

Remote client update
192.168.100.163 username=tog password=tog di
192.168.100.164 username=tog password=tog di
192.168.100.165 username=tog password=tog di
Remote update (192.168.100.163): 19
192.168.100.167 username=tog password=tog di
Remote update (192.168.100.164): 19
192.168.100.92 username=tog password=tog dir
Remote update (192.168.100.165): 19
Remote update (192.168.100.163): 19
Remote update (192.168.100.163): 19
Remote update (192.168.100.164): 19
Remote update (192.168.100.164): 19
Remote update (192.168.100.165): 19
Remote update (192.168.100.165): 19
Remote update (192.168.100.167): 19
Remote update (192.168.100.92): 19
```

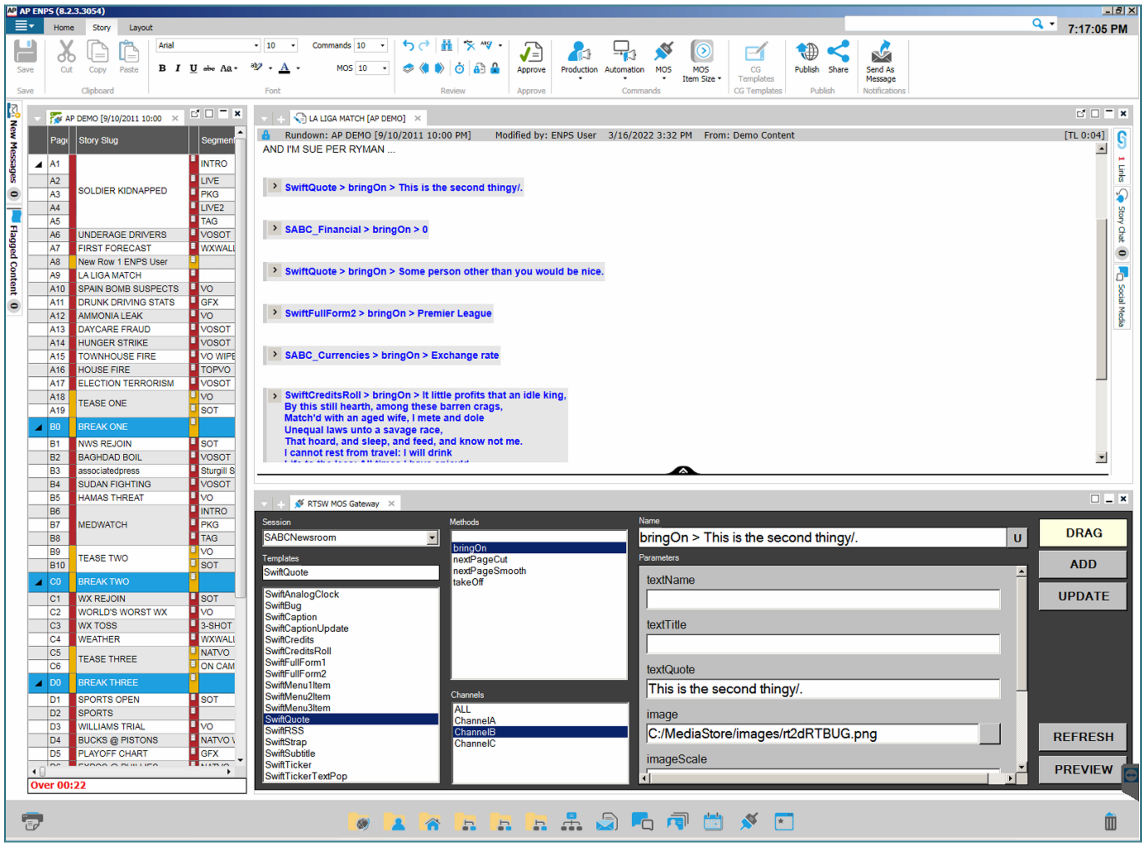
```
Information: callback_ClientMessage: LWS_CALLBACK_CLIENT_RECEIVE (ipAddress=192.168.100.1
Information: TRRemoteClientUpdateEditor::pollComms: START
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.163
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.164
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.165
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.167
Information: TRSocket::read:
Information: TRRemoteClientUpdateEditor::pollComms: Empty message (192.168.100.92)
Information: TRRemoteClientUpdateEditor::pollComms: START
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.163
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.164
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.165
Information: TRRemoteClientUpdateEditor::pollComms: Finish/abort 192.168.100.167
Information: TRSocket::read:
Information: TRRemoteClientUpdateEditor::pollComms: Empty message (192.168.100.92)
Information: callback_ClientMessage: LWS_CALLBACK_CLIENT_RECEIVE
Information: TRSocket::processMessage: Response from remote client (192.168.100.92) {
  "clientId" : "",
  "items" : "",
  "protocolId" : "repository@192.168.100.92",
  "response" : "RemoteUpdate",
  "result" : "success",
  "revision" : 28,
  "serialNumber" : 1,
  "serverId" : ""
}

Information: callback_ClientMessage: LWS_CALLBACK_CLIENT_RECEIVE (ipAddress=192.168.100.9
Information: TRRemoteClientUpdateEditor::pollComms: START
```

The Repository application is used to maintain a "versioned database" of graphic templates (and associated assets) where each change is logged and to distribute the latest version of these graphics to an editable list of client machines - in the newsroom case this would be some of the servers and all the rendering machines.

NRCS ActiveX

NRCS ActiveX implements the ActiveX functionality for NRCS desktop clients as described in the MOS Protocol. The purpose of these controls is to enable a non-expert user (for example, a journalist) to add graphic templates to a story in an NRCS running order.



AP DEMO [9/10/2011 10:00

Page	Story Slug	Segment
A1	SOLDIER KIDNAPPED	INTRO
A2		LIVE
A3		PKG
A4		LIVE2
A5		TAG
A6	UNDERAGE DRIVERS	VOSOT
A7	FIRST FORECAST	WXWALL
A8	New Row 1 ENPS User	
A9	LA LIGA MATCH	
A10	SPAIN BOMB SUSPECTS	VO
A11	DRUNK DRIVING STATS	GFX
A12	AMMONIA LEAK	VO
A13	DAYCARE FRAUD	VOSOT
A14	HUNGER STRIKE	VOSOT
A15	TOWNHOUSE FIRE	VO WIPE
A16	HOUSE FIRE	TOPVO
A17	ELECTION TERRORISM	VOSOT
A18	TEASE ONE	VO
A19		SOT
B0	BREAK ONE	
B1	NWS REJOIN	SOT
B2	BAGHDAD BOIL	VOSOT
B3	associatedpress	Sturgill S
B4	SUDAN FIGHTING	VOSOT
B5	HAMAS THREAT	VO
B6	MEDWATCH	INTRO
B7		PKG
B8		TAG
B9	TEASE TWO	VO
B10		SOT
C0	BREAK TWO	
C1	WX REJOIN	SOT
C2	WORLD'S WORST WX	VO
C3	WX TOSS	3-SHOT
C4	WEATHER	WXWALL
C5	TEASE THREE	NATVO
C6		ON CAM
D0	BREAK THREE	
D1	SPORTS OPEN	SOT
D2	SPORTS	
D3	WILLIAMS TRIAL	VO
D4	BUCKS @ PISTONS	NATVO
D5	PLAYOFF CHART	GFX

Over 00:22

LA LIGA MATCH [AP DEMO]

Rundown: AP DEMO [9/10/2011 10:00 PM] Modified by: ENPS User 3/16/2022 3:32 PM From: Demo Content [TL 0:04]

AND I'M SUE PER RYMAN ...

SwiftQuote > bringOn > This is the second thingy/.

SABC_Financial > bringOn > 0

SwiftQuote > bringOn > Some person other than you would be nice.

SwiftFullForm2 > bringOn > Premier League

SABC_Currencies > bringOn > Exchange rate

SwiftCreditsRoll > bringOn > It little profits that an idle king,
By this still hearth, among these barren crags,
Match'd with an aged wife, I mete and dole
Unequal laws unto a savage race,
That hoard, and sleep, and feed, and know not me.
I cannot rest from travel: I will drink

RTSW MOS Gateway

Session: SABCNewsroom

Templates: SwiftQuote

SwiftAnalogClock
SwiftBug
SwiftCaption
SwiftCaptionUpdate
SwiftCredits
SwiftCreditsRoll
SwiftFullForm1
SwiftFullForm2
SwiftMenu1Item
SwiftMenu2Item
SwiftMenu3Item
SwiftQuote
SwiftRSS
SwiftStrap
SwiftSubtitle
SwiftTicker
SwiftTickerTextPop

Methods:
bringOn
nextPageCut
nextPageSmooth
takeOff

Channels:
ALL
ChannelA
ChannelB
ChannelC

Name: bringOn > This is the second thingy/.

Parameters:

textName:

textTitle:

textQuote: This is the second thingy/.

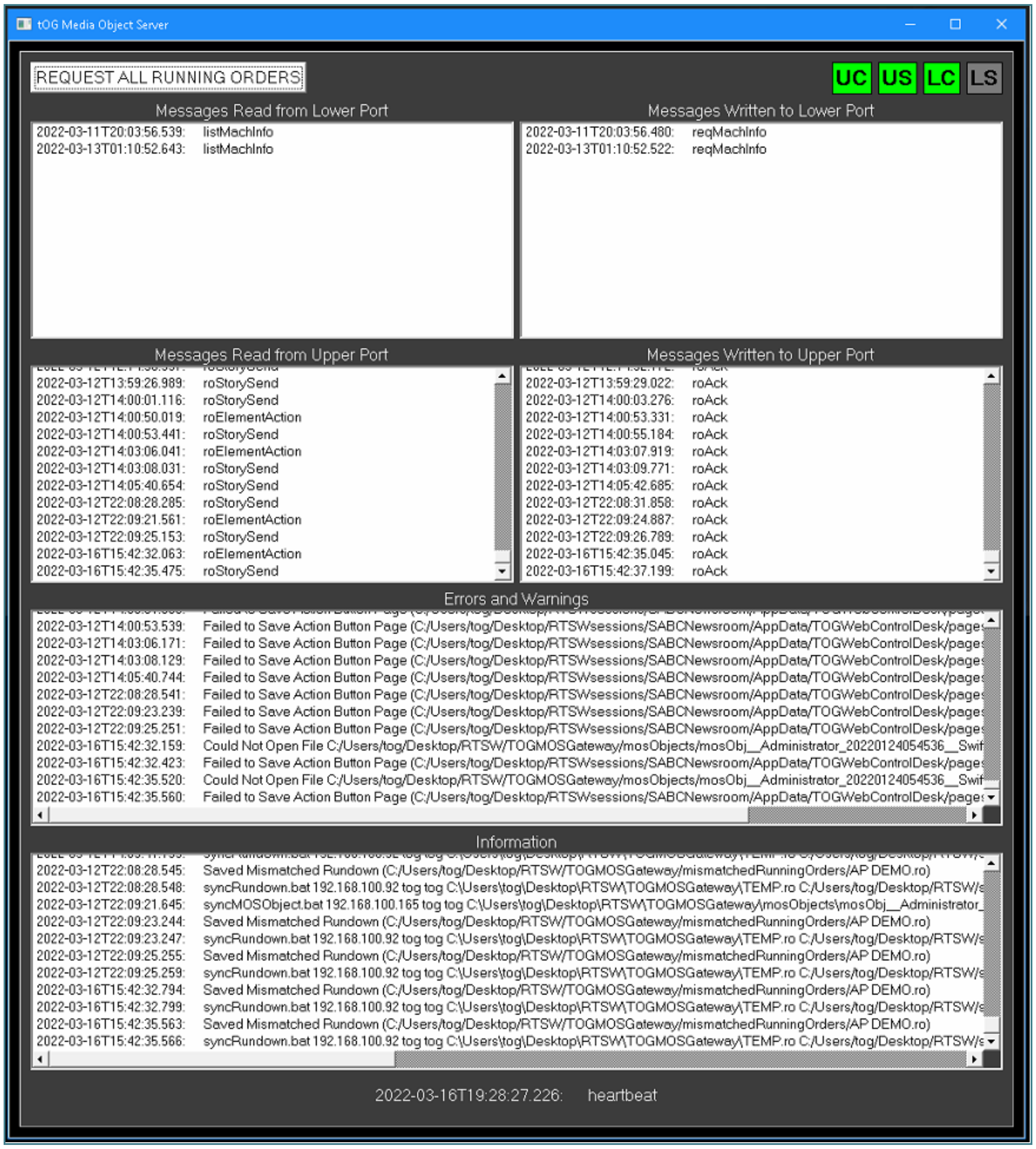
image: C:/MediaStore/images/rt2dRTBUG.png

imageScale:

These ActiveX controls allow users to select a graphic template, complete the template (by entering values for template parameters), preview the completed graphic template and then add it to a story as a new item. The user can specify the objSlug and mosAbstract for the item. RT maintains separate ActiveX controls for each of the supported NRCS Desktop Clients.

NRCS MOSGateway

MOSGateway communicates with the Newsroom Control System.


tOG Media Object Server
REQUEST ALL RUNNING ORDERS
UC
US
LC
LS
Messages Read from Lower Port
Messages Written to Lower Port
Messages Read from Upper Port
Messages Written to Upper Port
Errors and Warnings
Information

Messages Read from Lower Port

```
2022-03-11T20:03:56.539:    listMachInfo
2022-03-13T01:10:52.643:    listMachInfo
```

Messages Written to Lower Port

```
2022-03-11T20:03:56.480:    reqMachInfo
2022-03-13T01:10:52.522:    reqMachInfo
```

Messages Read from Upper Port

```
2022-03-12T13:59:26.989:    roStorySend
2022-03-12T14:00:01.116:    roStorySend
2022-03-12T14:00:50.019:    roElementAction
2022-03-12T14:00:53.441:    roStorySend
2022-03-12T14:03:06.041:    roElementAction
2022-03-12T14:03:08.031:    roStorySend
2022-03-12T14:05:40.654:    roStorySend
2022-03-12T22:08:28.285:    roStorySend
2022-03-12T22:09:21.561:    roElementAction
2022-03-12T22:09:25.153:    roStorySend
2022-03-16T15:42:32.063:    roElementAction
2022-03-16T15:42:35.475:    roStorySend
```

Messages Written to Upper Port

```
2022-03-12T13:59:29.022:    roAck
2022-03-12T14:00:03.276:    roAck
2022-03-12T14:00:53.331:    roAck
2022-03-12T14:00:55.184:    roAck
2022-03-12T14:03:07.919:    roAck
2022-03-12T14:03:09.771:    roAck
2022-03-12T14:05:42.685:    roAck
2022-03-12T22:08:31.858:    roAck
2022-03-12T22:09:24.887:    roAck
2022-03-12T22:09:26.789:    roAck
2022-03-16T15:42:35.045:    roAck
2022-03-16T15:42:37.199:    roAck
```

Errors and Warnings

```
2022-03-12T14:00:53.539:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T14:03:06.171:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T14:03:08.129:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T14:05:40.744:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T22:08:28.541:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T22:09:23.239:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-12T22:09:25.251:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-16T15:42:32.159:    Could Not Open File C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mosObjects/mosObj__Administrator_20220124054536__Swi
2022-03-16T15:42:32.423:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
2022-03-16T15:42:35.520:    Could Not Open File C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mosObjects/mosObj__Administrator_20220124054536__Swi
2022-03-16T15:42:35.560:    Failed to Save Action Button Page (C:/Users/tog/Desktop/RTSWsessions/SABCNewsroom/AppData/TOGWebControlDesk/page
```

Information

```
2022-03-12T22:08:28.545:    Saved Mismatched Rundown (C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mismatchedRunningOrders/AP DEMO.ro)
2022-03-12T22:08:28.548:    syncRundown.bat 192.168.100.92 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\TEMP.ro C:/Users/tog/Desktop/RTSW/
2022-03-12T22:09:21.645:    syncMOSObject.bat 192.168.100.165 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\mosObjects\mosObj__Administrator_
2022-03-12T22:09:23.244:    Saved Mismatched Rundown (C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mismatchedRunningOrders/AP DEMO.ro)
2022-03-12T22:09:23.247:    syncRundown.bat 192.168.100.92 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\TEMP.ro C:/Users/tog/Desktop/RTSW/
2022-03-12T22:09:25.255:    Saved Mismatched Rundown (C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mismatchedRunningOrders/AP DEMO.ro)
2022-03-12T22:09:25.259:    syncRundown.bat 192.168.100.92 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\TEMP.ro C:/Users/tog/Desktop/RTSW/
2022-03-16T15:42:32.794:    Saved Mismatched Rundown (C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mismatchedRunningOrders/AP DEMO.ro)
2022-03-16T15:42:32.799:    syncRundown.bat 192.168.100.92 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\TEMP.ro C:/Users/tog/Desktop/RTSW/
2022-03-16T15:42:35.563:    Saved Mismatched Rundown (C:/Users/tog/Desktop/RTSW/TOGMOSGateway/mismatchedRunningOrders/AP DEMO.ro)
2022-03-16T15:42:35.566:    syncRundown.bat 192.168.100.92 tog tog C:\Users\tog\Desktop\RTSW\TOGMOSGateway\TEMP.ro C:/Users/tog/Desktop/RTSW/
```

```
2022-03-16T19:28:27.226:    heartbeat
```

It uses the standard MOS protocol to query the NRCS about the status and contents of the running orders compiled using the NRCS. It also handles MOS messages from the NRCS reflecting changes to the current running order - especially the addition of graphics. It creates graphic stacks and pages for later playout from the running orders it receives from the NRCS.

Automated newsroom systems can control Swift Engine renderers indirectly. They do this by sending the MOS Protocol roCtrl messages to the MOSGateway.

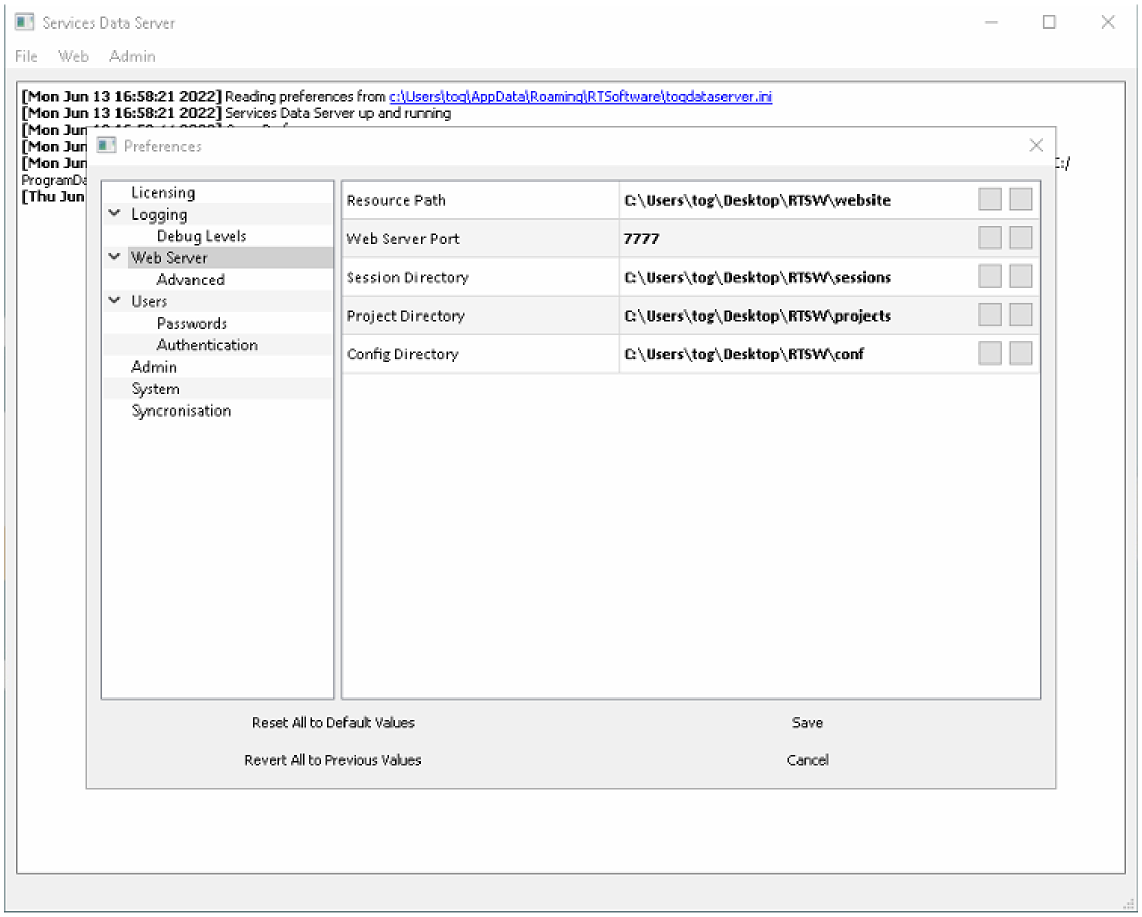
This message commands MOSGateway to play out an item from a story in a running order (the running order, story and item are specified in the message). The item will have been created by dragging from a NRCS ActiveX embedded in a NRCS Desktop Client onto a story in a running order. The graphic template and parameters

The graphics that have implemented the cueGraphic, bringOn and takeOff methods. These map to the READY, EXECUTE and STOP commands in the MOS message.

The url of the Swift Engine to control is specified in the MOSGateway .conf file. Only one Swift Engine instance can be controlled by a MOSGateway instance.

DataServer

The DataServer application is essentially a web server with broadcast graphic specific extra functionality.



```
[Mon Jun 13 16:58:21 2022] Reading preferences from c:\Users\tog\AppData\Roaming\RTSoftware\togdataserver.ini
[Mon Jun 13 16:58:21 2022] Services Data Server up and running
```

Resource Path	C:\Users\tog\Desktop\RTSW\website
Web Server Port	7777
Session Directory	C:\Users\tog\Desktop\RTSW\sessions
Project Directory	C:\Users\tog\Desktop\RTSW\projects
Config Directory	C:\Users\tog\Desktop\RTSW\conf

DataServer serves up project manifest data to the NRCS ActiveX. The ActiveX actually makes a HTTP request for the data to the DataServer.

DataServer serves up the Swift Live control application to a chrome running on an RT Control platform. The DataServer will also handle HTTP requests from Swift Live to save and load preference, page and other file types stored on the RT DataServer platform.

DataClient

The DataClient is a service which takes command from the Repository application over tcp/ip. It runs on any platform that needs graphic projects to be available locally - this includes all renderers and even some servers e.g. DataServer. There commands are:

RemoteVersion: return the current revision of the specified project in the locally checked out projects repository.

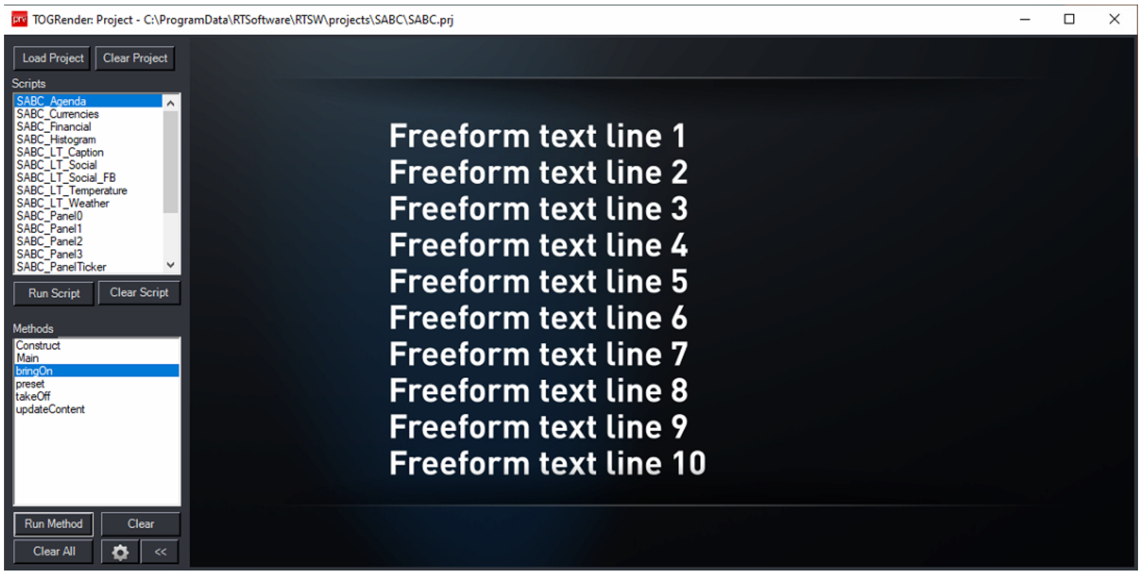
RemoteUpdate: update the specified project in the locally checked out projects repository. This will drag down changes to the project in the central repository and apply them to the local project - essentially bringing all the files and folders in the project on disk up to the latest version.

RemoteSynchronize: the local renderers will load a project when they are run up or when told to do so under external control. This command will send a command to the renderer to update its loaded version of the project with any of the changes made to the project on disk by the RemoteUpdate command.

RemoteDelete: delete the specified project in the locally checked out projects repository.

Swift Engine

The Swift Engine is a rendering application - it is a lightweight wrapper around the RT DLL renderer.



Swift Engine is used on a variety of platforms.

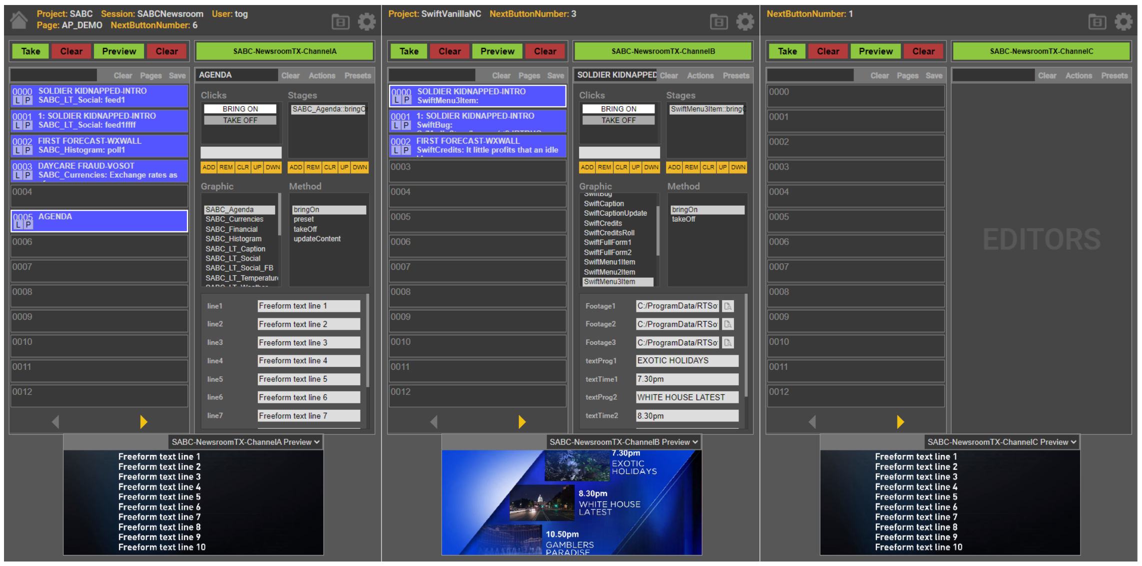
It is used on the RT Single and RT Duo platforms to provide preview channels to the Swift Live control application by streaming WebRTC. It can also be used to stream NDI but doesnt output SDI or 2110 - so it cannot be used for the main transmission channels at the moment if those output formats are required (Swift CG in Live mode is currently used).

It is used on the RT ActiveXPreview platform to render the final frame of a graphic (the last frame of the bringOn method) for use in preview in the NRCS ActiveX plugin.

It is used on the RT Repurposing platform to render out clips of the graphics described in MOS Objects created in the NRCS ActiveX plugin.

Swift Live

Swift Live is a HTML5 application served by DataServer (dataServer is a web server like apache or IIS but with broadcast graphic specific features). It allows the operator to control up to three channels of graphics with preview and NRCS integration.



Project: SABC Session: SABCNewsroom User: tog
Page: AP_DEMO NextButtonNumber: 6

SABC-NewsroomTX-ChannelA

0000 SOLDIER KIDNAPPED-INTRO
SABC_LT_Social: feed1

0001 1: SOLDIER KIDNAPPED-INTRO
SABC_LT_Social: feed1ffff

0002 FIRST FORECAST-WXWALL
SABC_Histogram: poll1

0003 DAYCARE FRAUD-VOSOT
SABC_Currencies: Exchange rates as

0004

0005 AGENDA

0006
0007
0008
0009
0010
0011
0012

AGENDA
Stages: SABC_Agenda::bringOn
Graphic: SABC_Agenda, SABC_Currencies, SABC_Financial, SABC_Histogram, SABC_LT_Caption, SABC_LT_Social, SABC_LT_Social_FB, SABC_LT_Temperature
Method: bringOn, preset, takeOff, updateContent

line1	Freeform text line 1
line2	Freeform text line 2
line3	Freeform text line 3
line4	Freeform text line 4
line5	Freeform text line 5
line6	Freeform text line 6
line7	Freeform text line 7

SABC-NewsroomTX-ChannelA Preview

Freeform text line 1
Freeform text line 2
Freeform text line 3
Freeform text line 4
Freeform text line 5
Freeform text line 6
Freeform text line 7
Freeform text line 8
Freeform text line 9
Freeform text line 10

Project: SwiftVanillaNC NextButtonNumber: 3

SABC-NewsroomTX-ChannelB

0000 SOLDIER KIDNAPPED-INTRO
SwiftMenu3Item:

0001 1: SOLDIER KIDNAPPED-INTRO
SwiftBug:

0002 FIRST FORECAST-WXWALL
SwiftCredits: It little profits that an idle

0003
0004
0005
0006
0007
0008
0009
0010
0011
0012

SOLDIER KIDNAPPED
Stages: SwiftMenu3Item::bringOn
Graphic: SwiftBug, SwiftCaption, SwiftCaptionUpdate, SwiftCredits, SwiftCreditsRoll, SwiftFullForm1, SwiftFullForm2, SwiftMenu1Item, SwiftMenu2Item, SwiftMenu3Item
Method: bringOn, takeOff

Footage1	C:/ProgramData/RTSo
Footage2	C:/ProgramData/RTSo
Footage3	C:/ProgramData/RTSo
textProg1	EXOTIC HOLIDAYS
textTime1	7.30pm
textProg2	WHITE HOUSE LATEST
textTime2	8.30pm

SABC-NewsroomTX-ChannelB Preview

7.30pm EXOTIC HOLIDAYS
8.30pm WHITE HOUSE LATEST
10.50pm GAMBLERS PARADISE

NextButtonNumber: 1

SABC-NewsroomTX-ChannelC

0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012

SABC-NewsroomTX-ChannelC Preview

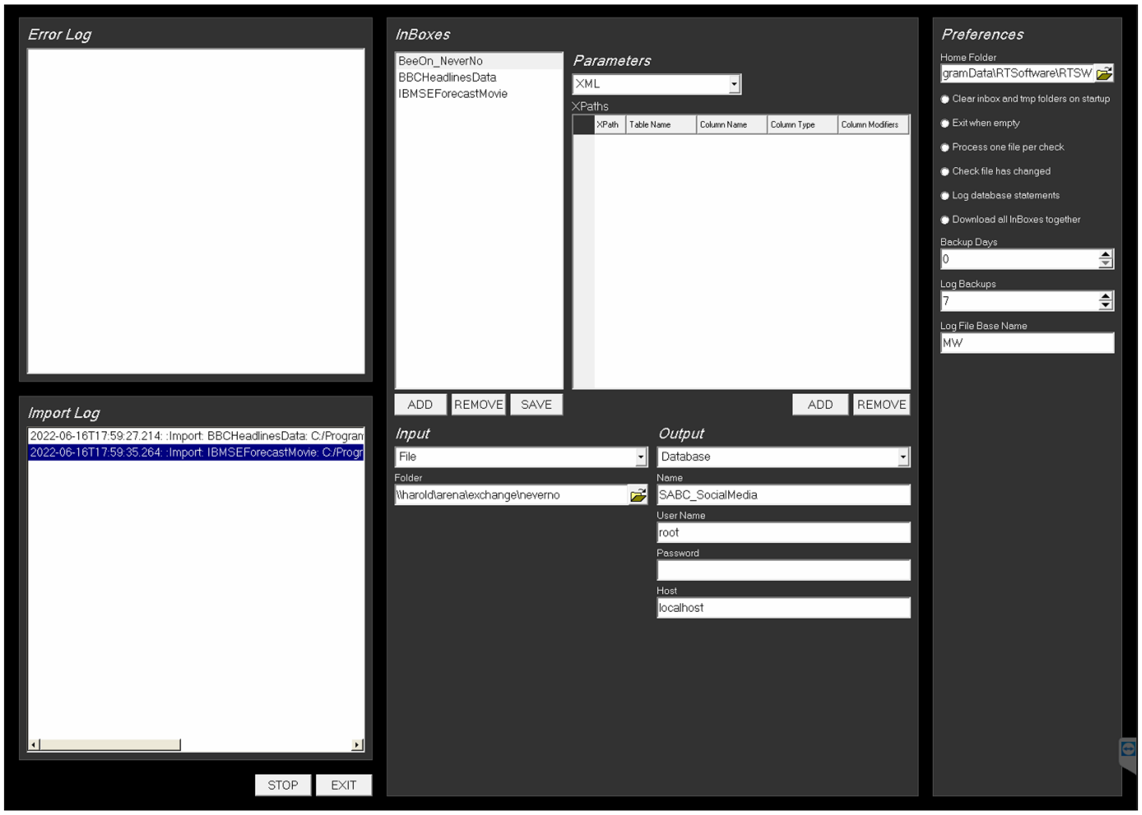
Freeform text line 1
Freeform text line 2
Freeform text line 3
Freeform text line 4
Freeform text line 5
Freeform text line 6
Freeform text line 7
Freeform text line 8
Freeform text line 9
Freeform text line 10

Swift Live can load the playlists and pages obtained from the NRCS via MOSGateway and DataServer. Swift Engine is a lightweight version of Swift CG+ which outputs a WebRTC transport stream. This is used for preview in the web control application.

The Swift Live application is accessible from any desktop (that has network access to the server). Playlists or action button pages of graphic templates can be compiled and previewed at the desktop - these playlists and pages are automatically available to operators for transmission. This route to transmission by-passes the NRCS completely.

MediaWatcher

The MediaWatcher application ingests data from external sources, processes it and stores it away ready for use by the renderers.



InBoxes: BeeOn_NeverNo, BBCHeadlinesData, IBMSEForecastMovie

Parameters: XML

XPath	Table Name	Column Name	Column Type	Column Modifiers

Input: File

Folder: \\harold\arena\exchange\neverno

Output: Database

Name: SABC_SocialMedia

User Name: root

Password:

Host: localhost

Home Folder: gramData\RTSoftware\RTSW

- Clear inbox and tmp folders on startup
- Exit when empty
- Process one file per check
- Check file has changed
- Log database statements
- Download all InBoxes together

Backup Days: 0

Log Backups: 7

Log File Base Name: MW

```
2022-06-16T17:59:27.214: :Import: BBCHeadlinesData: C:/Progran
2022-06-16T17:59:35.264: :Import: IBMSEForecastMovie: C:/Progr
```

To configure the application, the user creates a list of “InBoxes”. These match up a source of data (file, url, web server or serial port) with a destination (file, database, MOS). The application periodically checks the inboxes inputs (checking a watch folder, downloading a file etc). The application checks a list of internal plugins to see which one will accept the input data. The plugin will then process the data and write to the specified output. The plugins handle many standard and third party data types but new plugins can be added to handle specific customer data requirements on a services basis.

Communication Protocols

These protocols are used to move assets, data and newsroom rundowns between machines, to control renderers and stream graphic template preview to the Swift Live application.

WebSocket

This is a communication protocol based on websockets. It was developed by RT Software to support the development of HTML5 control applications. The messages are json packets. As well as the basic communications protocol (heartbeats etc), the WebSocket Protocol includes apis for running graphics, accessing databases, accessing graphic assets and manifests etc.

This protocol is used in three ways:

- to allow DataServer to serve and support the Swift Live control application.
- to allow Swift Live to control Swift Engine (to render graphics to air)
- to allow DataServer to serve graphic project manifests and icons to NRCS ActiveX so the journalist can select and configure graphics.

svn

The Repository application uses the svn command (part of the Collab Subversion library) to communicate with a Visual SVN server on the RT Repository platform - to store and access graphic project assets in the repository etc.

The DataClient uses the svn command to update checked out projects on the local machine.

WinSCP over sFTP

sFTP can be enabled on all machines by enabling the OpenSSH server under Windows 10 and WinSCP can be used to distribute files either putting the file on a remote machine or by synchronising folders across machines.

MOSGateway can be configured to send rundowns to the server running DataServer - from which it can be served up to Swift Live. It can also send MOSObjects to a server (the RT Repurposing platform) running a MediaWatcher configured to process MOSObjects and use the graphic info in them to drive a Swift Engine to render those graphics and save them to a movie file.

Images and movies can be distributed to renderers using Swift Live. The files are dragged onto the Manage Assets section of the application - this automatically sends those assets to the DataServer machine. They can then be sent to the renderers using the Synchronise option - this uses WinSCP to synchronise the folders containing these assets on the DataServer machine with the corresponding folders on the renderers.

Images and movies can be distributed to renderers using the RT MediaStore platform. They are copied onto the MediaStore folder on the platform. From there they are copied onto registered target machines by synchronising the folder with corresponding folders on those machines - using WinSCP over sFTP.

MOS

MOS is a protocol for communications between Newsroom Computer Systems (NRCS) and Media Object Servers (MOS) such as Video Servers, Audio Servers, Still Stores, Character Generators, Automation Servers, and Prompters. The MOS Protocol development is supported through cooperative collaboration among equipment vendors, software vendors and end users.

An NRCS is a centralised system that enables the creation of running orders of news stories. RT supports three so far (iNews, ENPS and OpenMedia).

It has three uses in the newsroom configuration:

- It allows MOSGateway to communicate with the NRCS. MOSGateway uses MOS messages to obtain running orders and to respond to MOS messages from the NRCS to update current running orders and MOS objects
- It allows the NRCS ActiveX (NRCS desktop client plugin) to insert graphic templates selected and configured by the journalist into running order stories as items (which are in turn passed onto MOSGateway).

WebRTC

WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. The RT renderers can create a WebRTC (unbuffered) video stream of the graphics it renders.

The Swift Engine application (a lightweight wrap of the full RT renderer) uses it to provide a desktop preview of graphics for Swift Live.

The WebSocket Protocol has been extended to support this protocol to facilitate the inclusion of a movie window for preview in an HTM5 application.

Platforms

The platforms involved vary from high-end graphics workstations used to create graphic assets to low-end desktop computers used to host the playout HTML5 application in a Chrome browser. The hardware recommended by RT can be found on the RT website at: https://rtsw.co.uk/support_specs/#EnablingTechnology.

The types of platform are:

RT Edit

Applications: Swift CG, Swift CG+, Repository
Hardware: high-end graphics workstation.

Designers will use tools like Photoshop and AfterEffects to create images and movies and Maya and Studio Max to create geometries and shaders. These can be imported into Swift CG and CG+ (via the fbx exchange format file) along with other assets like fonts and used to author graphics.

Once ready the graphics can be stored in a central repository (on the RT Repository platform) and then distributed to the RT Single, RT Duo, RT ActiveXPreview, RT Repurposing and RT DataServer platforms using the Repository application.

RT Repository

Applications: Visual SVN
Hardware: file server

This hosts Visual SVN repository server - the graphics assets repository. It stores the initial version of a graphics project and then every change (with a comment) made to it. The repository will grow in size over time as graphics are added and changed. Projects are typically less than a Gigabyte.

RT Single and RT Duo

Applications: Swift Engine, Swift CG+, DataClient, OpenSSH sFTP
Hardware: high-end graphics server with SDI/IP video card

The graphics templates and associated assets are held locally - distributed here by DataClient dragging down project changes from RT Repository platform. Any transient assets (movies and images mainly) needed by playout are copied locally (to ensure speed and security of access) into the MediaStore folder over sFTP.

Swift Engine is launched by clicking a desktop icon. It loads all the graphic assets on startup (templates are loaded when required and then cached), opens up SDI video inputs and outputs and any network sockets needed for control.

It then waits for a control application to connect. Under control it will play graphics and methods and render the output to the SDI video out. It can either produce a fill and key output and be keyed onto the transmission stream downstream or it can take the transmission stream as input and render the graphics onto it.

The RT Duo platform runs four renderers (two tOG Lives producing 2110 video outputs for the live channels and two Swift Engines producing WebRTC transport streams for the preview channels on the Swift LIve control interface).

The RT Single platform runs two renderers (one tOG Live producing a 2110 video output for the live channel and one Swift Engine producing a WebRTC transport stream for the preview channel on the Swift LIve control interface).

Windows bat files and desktop icons which start and stop the renderers are also provided.

Newsroom desktop and NRCS ActiveX

Applications: NRCS Journalist Client, NRC ActiveX plugin
Hardware: desktop computer

This hosts the desktop client of the NRCS which journalists can use to create running orders, enter their stories and store them in the central database of the NRCS (hosted on the NRCS server).

These NRCS Desktop Clients all have support for MOS Protocol compliant ActiveX controls. The NRCS ActiveX can be loaded as part of the NRCS Client (selected from a list of controls or by clicking on an item in a story). The journalist can then select a graphic template and enter data for it (which can then be previewed using a still supplied by a Swift Engine on the RT ActiveXPreview server). The template and data can then be dragged onto a story (creating a mosObject). This creates an item within the story with a mosPayload that holds a description of the graphics template and a mosObject reference.

RT ActiveXPreview

Applications: Swift Engine, DataClient, OpenSSH sFTP
Hardware: high-end graphics server

The NRCS ActiveX plugin is used by the journalist to select and configure a graphic (to be added to the NRCS rundown). To preview the graphic the ActiveX can request a final frame from a Swift Engine instance running on this platform. It runs up to 10 instances and the ActiveX randomly picks between them.

RT MOS Gateway

Applications: MOS Gateway, WinSCP
Hardware: file server

This hosts the MOSGateway application which runs in the background without interaction reading, saving and converting running orders and messages from the NRCS. This uses very little disk (running orders are typically small).

It also copies NRCS Rundowns to the RT DataServer platform and NRCS MOS Objects to the RT Repurposing platform.

RT Repurposing

Applications: MediaWatcher, Swift Engine, DataClient, OpenSSH sFTP
Hardware: high-end graphics server

The purpose of this platform is to automatically render out graphics as movie files (with alpha) for use in NLEs. The graphics are specified by the journalists in the ENPS Client and passed to the MOS Gateway server as MOS Objects. The MOS Gateway passes them to this server using WinSCP over sFTP. A MediaWatcher application watches a folder for these MOS Object files. It sends commands to a Swift Engine instance to render the graphics described in the MOS Object files to movie files. The movie files are then moved to the Facilis SAN using WinSCP over sFTP.

RT MediaWatcher

Applications: MediaWatcher, WinSCP, MySQL
Hardware: file server

This hosts the MediaWatcher application which handles external data sources - acquiring the data, processing it and storing the results away. For example it will read social media files pushed by NeverNo. It processes the xml files containing the message info into a MySQL database and the avatar images or associated images/movies to the Facilis SAN (from where it is distributed to the renderers).

RT DataServer

Applications: DataServer, OpenSSH sFTP
Hardware: file server

This hosts the DataServer application - this is a Web server that serves up graphic manifest data to the NRCS ActiveX or the Swift Live HTML5 application running in a chrome browser on an RT Control platform. It also serves up MOS Rundowns copied onto the platform using sFTP (from the RT MOS Gateway platform) to Swift Live.

RT MediaStore

Applications: WinSCP
Hardware: file server

This platform hosts a folder called MediaStore (containing image/movie or data files) and it synchronises with corresponding folders on other platforms (mainly the renderers). The folder may exist on a local disk or on a SAN mounted on the platform.

Note: the image/movie folder should also be available to the NRCS ActiveX running inside the NRCS Desktop Client on the journalist's desktop.

RT Control desktop

Applications: Chrome
Hardware: desktop computer

There are two applications that do playout control and the application used will decide which platform is appropriate.

Swift Live

> This requires the simplest platform - any pc that can run the Chrome browser. The operator starts the browser and directs it to the DataServer on the data/preview server machine. The operator can then login and select which session to use. This will load the Swift Live application and connect it to the Swift Engine (for preview) and Swift Engine (for transmission) specified in the session. The operator can then select, configure, preview and playout graphics.

Swift Playout

> This is usually a high-end graphics workstation. Stacks derived from the running orders in the NRCS by MOSGateway (copied to this machine by the synchronisation of transient assets) can be loaded into Swift Playout. Items on the stack can be previewed and then taken to air. The preview is done within Swift Playout and the application (using the MOS Protocol) commands a connected Swift Engine to transmit it.