AdminAuthorizeAttribute : AuthorizeAttribute

Fronted:

RecordingUploadedController

UnzipRecording Queue [

RecordingUploadedMessage (zipFile name , s3 folder)

]

BackEnd:

RecordingUploadedMessageHandler

-Download recording from bucket of temp recordings to EC2 local folder (needed role\token)

-unzip recording

-upload unzipped recording folder to bucket of constant recordings (need role\token)

-update recordings collection s3Path=[recording path],

Status = “unzipped”

Copy\Move

Fronted:

RecordingController

TransferRecordingFiles Queue [

BeginCopyRecordingMessage(

recordingId, destination course id, user id

]

BeginCopyRecordingMessageHandler:

(Recording)

entity.Status.ProcessingTasks.CopyTasks.Add(sourceTask);

foreach(var destination in message.Destination)

-can be copied to one or more courses

CopyRecordingMessage

-Download recording files from s3 [source course]

-upload recording files to s3 [target course]

-modify recording xml files

EndCopyRecordingMessage

Transcoding if required

Removing copy tasks

Push notification

-

{

.when('/courses', {

templateUrl: '/content/templates/courses/coursesList.html',

controller: 'coursesListCtrl',

title: 'Tegrity - Courses',

resolve: {

security: ["routeChangeSvc", function (routeChangeSvc) {

return routeChangeSvc.checkRegularUserOrGuestPermissions(Constants.Protocols.HTTPS);

}]

}

})

courseSvc.get

coursesController.get