Publish server-side live streams with the Wowza Streaming Engine Java API

Originally Published on 01/09/2012 | Updated on 05/02/2019 6:11 pm PDT

The IPublishingProvider in the Wowza Streaming Engine™ Java API enables creation of a server-side live stream that can be a mixture of live and video-on-demand (VOD) content. The Stream class is built on top of this functionality. This lower level functionality does not provide any scheduling or playlists. It only publishes live or video-on-demand content into a live stream.

Below is an example of a class that uses IPublishingProvider to publish the live stream named publishstream to the application named live. The class runs as a thread in the background and switches every 10 seconds between the VOD clip sample.mp4 and the live stream myStream.
run: START
long startTime = System.currentTimeMillis();
long playStartTime = startTime;
try {
 IVHost vhost = VHostSingleton.getInstance(VHost.VHOST_DEFAULT);
 Publisher publisher = Publisher.createInstance(vhost, applicationName);
 publisher.publish(publishStreamName);
 long nextSwitch = playStartTime + cycleTime;
 long nextType = 0;
 IPublishingProvider provider = new PublishingProviderMediaReader(publisher, playStartTime, vodStreamName);
// provider.seek(20000);
 provider.setRealTimeStartTime(startTime);
 WMSLoggerFactory.getLogger(ServerPublisherWorker.class).info("ServerPublisherWorker.run: Start with vod stream: "+vodStreamName);
 while (true) {
  boolean moreInFile = provider != null ? provider.play(publisher) : false;
  long currentTime = System.currentTimeMillis();
  if (!moreInFile || currentTime > nextSwitch) {
   if (provider != null) {
    provider.close();
    provider = null;
   }
   if ((nextType % 2) == 0) {
    provider = new PublishingProviderLive(publisher, publisher.getMaxTimecode(), liveStreamName);
    // ((PublishingProviderLive) provider).setStartOnPreviousKeyFrame(false);
    provider.setRealTimeStartTime(currentTime);
    WMSLoggerFactory.getLogger(ServerPublisherWorker.class).info("ServerPublisherWorker.run: Switch to live stream: "+liveStreamName);
   } else {
    provider = new PublishingProviderMediaReader(publisher, publisher.getMaxTimecode(), vodStreamName);
    // provider.seek(20000);
    provider.setRealTimeStartTime(currentTime);
    WMSLoggerFactory.getLogger(ServerPublisherWorker.class).info("ServerPublisherWorker.run: Switch to vod stream: "+vodStreamName);
   }
  }
 }
To see this example in action, create a server-side listener that's invoked when the server starts and stops when the server shuts down. Here is an example:

```java
nextSwitch = currentTime + cycleTime;
nextType++;
if (nextType == 100)
    break;
else
    sleep(sleepTime);
synchronized(lock) {
    if (!running)
        break;
}
provider.close();
publisher.publish(null);
synchronized(lock) {
    running = false;
}
} catch (Exception e) {
    WMSLoggerFactory.getLogger(ServerPublisherWorker.class).error("ServerPublisherWorker.run: "+e.toString());
e.printStackTrace();
} WMSLoggerFactory.getLogger(ServerPublisherWorker.class).info("ServerPublisherWorker.run: STOP");
```
Then add this server listener to the list in [install-dir]/conf/Server.xml:

```java
package com.wowza.wms.plugin.test.integration;
import com.wowza.wms.server.*;
public class ServerPublisherServerListener implements IServerNotify {
    ServerPublisherWorker worker = null;
    public void onServerCreate(IServer server) {
    }
    public void onServerInit(IServer server) {
        worker = new ServerPublisherWorker();
        worker.start();
    }
    public void onServerShutdownComplete(IServer server) {
    }
    public void onServerShutdownStart(IServer server) {
        if (worker != null)
            worker.quit();
        worker = null;
    }
}
com.wowza.wms.plugin.test.integration.ServerPublisherServerListener
```

With this running, publish a live stream named `myStream` to the application named `live`. When you play the live stream `publishstream`, you'll see that every 10 seconds it switches from the VOD clip to the live stream.