Use nDVR recording with the Wowza Streaming Engine Java API

This article explains how to control nDVR recording using a Wowza Streaming Engine HTTP provider.

Configure nDVR

By default, nDVR starts recording a stream as soon as the stream starts. However, the `startRecordingOnStartup` property can be set to `false` in your `Application.xml` file (under `/`). This will cause the recorder to initialize, but it won’t start recording until `ILiveStreamDvrRecorder.startRecording()` is called.

nDVR recording process

The recording process consists of:

1. Getting the `ILiveStreamDvrRecorder` recorder class from the live stream using `IMediaStream.getDvrRecorder()`.
2. (Optional) Setting the recording name via `ILiveStreamDvrRecorder.setRecordingName()`.
3. Start the recording via `ILiveStreamDvrRecorder.startRecording()`.
4. Stop the recording using `ILiveStreamDvrRecorder.stopRecording()`.

Example HTTP provider

The example below demonstrates nDVR recording using an HTTP provider. An HTTP provider is an extension to Wowza Streaming ENgine that can be attached to a `HostPort` definition in
The class below is an HTTP provider that controls starting and stopping DVR recording.

Use the Wowza IDE to compile this class into a .jar file and copy it to the \[install-dir\]/lib folder in your Wowza Streaming Engine installation. Then add this HTTP provider to the (Port 8086) / container in \[install-dir\]/conf/VHost.xml. This HTTP provider should be added before the HTTPServerVersion HTTP provider, which is typically last in the container for the host port.

Using the HTTP provider

nDVR recording can be controlled with the HTTP provider with a URL that uses the format:

http://[wowza-ip-address]:8086/dvrstreamrecord?app=[application-name]&streamname=[stream-name]&recordingname=[recording-name]&action=[start|stop]

Where:

- [wowza-ip-address] is the IP address of the Wowza Streaming Engine server.
- [application-name] is the application name the stream is running on.
- [stream-name] is the stream name of the live source stream to be recorded.
- [recording-name] is the name of the recorded store. This is optional and defaults to [stream-name] if not specified.

Set RecordingName

If [recording-name] is set, instead of recording the stream to \[install-dir\]/dvr/[application-name]/[app-inst]/[stream-name].[version], the recording is made to \[install-dir\]/dvr/[application-name]/[app-inst]/[recording-name].[version].

To play the recording, you would specify [recording-name] instead of [stream-name] in the playback URL. For example, if the incoming stream name is myStream and the recording name is myRecording, you would specify this URL for iOS devices:
Code example

Here's the code example for controlling nDVR recording:
action = params.get("action").get(0);

} else {
    report += 
    "action" + " is required;"
    WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " action: " + action);
    if (params.containsKey("app")) {
        app = params.get("app").get(0);
    } else {
        report += 
        "app" + " is required;"
    }
    WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " app: " + app);
    if (params.containsKey("streamname")) {
        streamName = params.get("streamname").get(0);
        recordingName = streamName; // default to stream name
    } else {
        report += 
        "streamname" + " is required;"
    }
    WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " streamName: " + streamName);
    // If recordingName is specified, use it instead
    if (params.containsKey("recordingname")) {
        recordingName = params.get("recordingname").get(0);
        WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " recordingName: " + recordingName);
    }
} catch (Exception ex) {
    report = "Error: " + ex.getMessage();
}
else {
    report = "Nothing to do;";
}
try {
    IApplicationInstance appInstance = vhost.getApplication(app).getAppInstance("_definst_");
    if (!appInstance.getPublishStreamNames().contains(streamName)) {
        report = "Live stream " + streamName + " does not exist;"
    }
    if (action.equalsIgnoreCase("start") && report.equalsIgnoreCase("") {
        WMSLoggerFactory.getLogger(CLASS).info(String.format("%s.%s: %s", CLASSNAME, "start", streamName));
    } else {
        report += 
        "action" + " is required;"
    }
    WMSLoggerFactory.getLogger(CLASS).info(String.format("%s.%s: %s", CLASSNAME, "start", streamName));
}
String streamTypeStr = appInstance.getStreamType();
boolean isLiveRepeaterEdge = false;
while (true) {
    StreamList streamDefs = appInstance.getVHost().getStreamTypes();
    StreamItem streamDef = streamDefs.getStreamDef(streamTypeStr);
    if (streamDef == null)
        break;
    isLiveRepeaterEdge = streamDef.getProperties().getPropertyBoolean(
        "isLiveRepeaterEdge",
        isLiveRepeaterEdge);
    break;
}
if (isLiveRepeaterEdge)
    streamName = MediaStreamMediaCasterUtils.mapMediaCasterName(appInstance, null, streamName);
IMediaStream stream = appInstance.getStreams().getStream(streamName);
if (stream != null) {
    startRecording(stream, recordingName);
    report = action + " " + streamName + " as " + recordingName;
} else {
    WMSLoggerFactory.getLogger(CLASS).warn(String.format("%s.%s: stream '%s' not found.", CLASSNAME, "start", streamName));
    report = "Stream Not Found: " + streamName;
}
else if (action.equalsIgnoreCase("stop") & report.equalsIgnoreCase("") {
    WMSLoggerFactory.getLogger(CLASS).info(String.format("%s.%s: %s", CLASSNAME, "stop", streamName));
    String path = stopRecording(streamName);
    report = action + " " + streamName + " " + path;
}
} catch (Exception e) {
    report = "Error: " + e.getMessage();
}
String retStr = " + report + ";
try {
    OutputStream out = resp.getOutputStream();
    byte[] outBytes = retStr.getBytes();
    out.write(outBytes);
public void startRecording(IMediaStream stream, String recordingName) {
    String streamName = stream.getName();
    // add it to the recorders list
    synchronized (dvrRecorders) {
        // Stop previous recorder
        ILiveStreamDvrRecorder prevRecorder = dvrRecorders.get(streamName);
        if (prevRecorder != null && prevRecorder.isRecording()) {
            prevRecorder.stopRecording();
        }
        // get the stream's DVR recorder and save it in a map of recorders
        ILiveStreamDvrRecorder dvrRecorder = stream.getDvrRecorder("DVR_DEFAULT_RECORDER_ID");
        if (dvrRecorder != null) {
            if (dvrRecorder.isRecording()) {
                dvrRecorder.stopRecording();
            }
            // start recording
            dvrRecorder.setRecordingName(recordingName);
            dvrRecorder.startRecording();
            dvrRecorders.put(streamName, dvrRecorder);
        } else {
            WMSLoggerFactory.getLogger(CLASS).warn(String.format("%s.%s: DVR Recorder not found for stream '%s'.", CLASSNAME, "start", streamName));
        }
    }
}

public String stopRecording(String streamName) {
    String path = 
    ILiveStreamDvrRecorder dvrRecorder = null;
    synchronized (dvrRecorders) {
        dvrRecorder = dvrRecorders.remove(streamName);
    }
    if (dvrRecorder != null) {
        IDvrStreamManager dvrManager = dvrRecorder.getDvrManager();
        if (dvrManager != null) {
            // some other code
        }
    }
    return path;
}
if (dvrManager != null) {
    IDvrStreamStore store = dvrManager.getRecordingStreamStore();
    IDvrFileSystem fs = store.getFileSystem();
    path = fs.getBasePath();
}

// stop recording
dvrRecorder.stopRecording();

try {
    Thread.sleep(500);
} catch (InterruptedException e) {
    e.printStackTrace();
}

} else {
    WMSLoggerFactory.getLogger(CLASS).warn(String.format("%s.%s: DVR Manager not found for stream '%s'.", CLASSNAME, "stop", streamName));
}

return path;
}