Wowza nDVR: Query recording duration with the Wowza Streaming Engine Java API

This article demonstrates how to use the Wowza Streaming Engine™ Java API to investigate DVR recordings by querying the duration in dvrTime, packetTime, and UTC time using an HTTP provider.

**Note:** This functionality is supported in Wowza Streaming Engine or Wowza Media Server™ 3.0.4 or later.

**Time Units**

nDVR chunks have time information for three time units:

- **DVR time:** This is a time in milliseconds of the DVR recording. The first chunk always starts at time 0 and time progresses forward. If there are gaps in the stream, either because the encoder stopped sending data for a while or because nDVR is being run in append mode, the DVR time will not contain the gaps.

- **Packet time:** The encoder timestamps the outgoing audio and video packets. The nDVR chunks preserve this packet time.

- **UTC time.** As the live stream comes into the nDVR module, the packets are time stamped with the current UTC time. Note this isn't an accurate timestamp of the time when the packets left the encoder; it’s a rough timestamp of when the packets are received by Wowza nDVR.

**Primary interfaces**

Each `IDvrStreamStore` contains a single `IDvrManifest`, which represents the information that makes up the DVR recording.

The `IDvrManifest` contains several channels of information, including audio and video channels. This class is `DvrChannelManifest`. These channels are comprised of manifest records, which includes a timestamp for each of the 3 time units.

By navigating from the `IApplicationInstance` to the audio and video channels, we can inspect the first and last recorded entries and determine the start and stop times in all 3 time units.

The example HTTP provider
The example below demonstrates querying a stream for all its associated DVR recordings using the **HTTPProvider** mechanism. The example shows whether the DVR recording is live, contains audio and or video, and reports the start and stop times in DVR, packet and UTC timescales.

The provided example is only example code and can easily be modified to suit your specific purposes.

An HTTP provider is an extension to the server that can be attached to a **HostPort** definition in `[install-dir]/conf/VHost.xml` and can be controlled via a URL. There is more detailed information on this subject in the Wowza Media Server User’s Guide.

Use the **Wowza IDE** to compile this class into a JAR file and copy it to the `[install-dir]/lib` folder. Then add this HTTP provider to the `/conf/VHost.xml` /HostPort (Port 8086) /HTTPProviders container. This HTTP provider should be added **above** the ServiceVersion HTTP provider.

Using the HTTP provider

The **nDVR** recording can be controlled with the HTTP provider via a URL. The format is:

```
http://[wowza-ip-address]:8086/dvrstreamquery?action=query&app=[application-name]&streamname=[stream-name][&forceload=true]
```

Where:

- **[wowza-ip-address]**: The IP address of the server running Wowza Streaming Engine.
- **app**: The application name the stream is running on.
- **streamname**: The stream name of the live source stream to be recorded.
- **forceload**: (Optional) The provider by default only looks at DVR recordings already loaded by the media server. Add this parameter to force the loading DVR recordings for the stream.

The HTTP provider will return HTML that in the browser looks like this:
Sample code

Below is the sample code for querying DVR store information:

```java
package com.wowza.wms.plugin.test.dvr.api;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.*;
import com.wowza.wms.application.IApplicationInstance;
import com.wowza.wms.dvr.*;
import com.wowza.wms.http.*;
import com.wowza.wms.logging.WMSLoggerFactory;
import com.wowza.wms.plugin.test.dvr.api.HTTPDvrStreamQuery.StartEndTimes;
import com.wowza.wms.stream.mediacaster.MediaStreamMediaCasterUtils;
import com.wowza.wms.vhost.*;

// Usage: http://[wowza-ip-address]:8086/dvrstreamquery?action=query&app=[application-name]&streamname=[stream-name][&forceload=true]

class HTTPDvrStreamQuery extends HTTProvider2Base {
    private static final String CLASSNAME = "HTTPDvrStreamQuery";
    private static final Class CLASS = HTTPDvrStreamQuery.class;

    public void onHTTPRequest(IVHost vhost, IHTTPRequest req, IHTTPResponse resp) {
        if (!doHTTPAuthentication(vhost, req, resp)) {
            return;
        }

        WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " HTTPRequest");
        Map params = req.getParameterMap();
        String action = "";
        String app = "";
        String streamName = "";
        String report = "";
        boolean forceLoad = false;
```
boolean forceLoad = false;

if (req.getMethod().equalsIgnoreCase("get") || req.getMethod().equalsIgnoreCase("post")) {
    req.parseBodyForParams(true);
    try {
        if (params.containsKey("action")) {
            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);
        } else {
            report += "streamname" + " is required";
        }
        WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " streamName: " + streamName);
        if (params.containsKey("forceload")) {
            String forceLoadString = params.get("forceload").get(0);
            forceLoad = Boolean.parseBoolean(forceLoadString);
            WMSLoggerFactory.getLogger(CLASS).info(CLASSNAME + " forceload: " + forceLoadString);
        }
    } catch (Exception ex) {
        report = "Error: " + ex.getMessage();
    }
} else {
    report = "Nothing to do.";
}
try {
    IApplicationInstance appInstance = vhost.getApplication(app).getAppInstance("_definst_");
    // If no error
    if (report.equalsIgnoreCase("") {
" + result;

    } else {
        report = "Action: "+action + " is not valid.";
    }
}
}

} catch (Exception e) {
    report = "Error: " + e.getMessage();
}

String retStr = "
" + report + "
"
public String queryDvrStreamInfo(IApplicationInstance appInstance, String streamName, boolean forceLoad) {
    StringBuffer sb = new StringBuffer();
    IDvrStreamManager dvrMgr = DvrStreamManagerUtils.getStreamManager(appInstance, IDvrConstants.DVR_STREAMING_PACKETIZER_ID, streamName, forceLoad);
    if (dvrMgr == null) {
        sb.append("Stream "+streamName+" not loaded.");
    } else {
        List stores = dvrMgr.getStreamStores();
        for (IDvrStreamStore store : stores) {
            sb.append(queryDvrStoreInfo(store));
        }
    }
    return sb.toString();
}

public String queryDvrStoreInfo(IDvrStreamStore store) {
    StringBuffer sb = new StringBuffer();
    if (store != null) {
        boolean hasAudio = store.hasAudio();
        boolean hasVideo = store.hasVideo();
        boolean isLive = store.isLive();
        sb.append(String.format("Store:%s", store.getStreamName()));
        String result = String.format("isLive: %s   hasAudio: %s   hasVideo: %s", isLive, hasAudio, hasVideo);
        sb.append(result);
        long duration = -1;
        StartEndTimes dvrTimes = queryDvrStartStop(store);
        if (dvrTimes != null) {
            duration = dvrTimes.end - dvrTimes.start;
            String s = String.format("dvrStart: %s  dvrEnd: %s  duration: %s", formatDvrTime(dvrTimes.start), formatDvrTime(dvrTimes.end), formatDvrTime(duration));
            sb.append(s);
        }
    }
    return sb.toString();
}
protected StartEndTimes queryDvrStartStop(IDvrStreamStore store) {
    if (store == null) {
        return null;
    }
    DvrChannelManifest manifest = getVideoOrAudioManifest(store);
    if (manifest == null) {
        return null;
    }
    DvrManifestEntry firstEntry = manifest.getFirstEntry();
    DvrManifestEntry lastEntry = manifest.getLastRecordedEntry();
    long start = firstEntry.getStartTimecode();
    long end = lastEntry.getStopTimecode();
    return new StartEndTimes(start, end);
}
protected StartEndTimes queryPacketStartStop(IDvrStreamStore store) {  
    if (store == null) {  
        return null;  
    }  
    DvrChannelManifest manifest = getVideoOrAudioManifest(store);  
    if (manifest == null) {  
        return null;  
    }  
    DvrManifestEntry firstEntry = manifest.getFirstEntry();  
    DvrManifestEntry lastEntry = manifest.getLastRecordedEntry();  
    long start = firstEntry.getPacketStartTime();  
    long end = lastEntry.getPacketStartTime() + lastEntry.getDuration();  
    return new StartEndTimes(start, end);  
}  
protected StartEndTimes queryUtcStartStop(IDvrStreamStore store) {  
    if (store == null) {  
        return null;  
    }  
    DvrChannelManifest manifest = getVideoOrAudioManifest(store);  
    if (manifest == null) {  
        return null;  
    }  
    DvrManifestEntry firstEntry = manifest.getFirstEntry();  
    DvrManifestEntry lastEntry = manifest.getLastRecordedEntry();  
    long start = firstEntry.getUtcStartTime();  
    long end = lastEntry.getUtcStartTime() + lastEntry.getDuration();  
    return new StartEndTimes(start, end);  
}  
private DvrChannelManifest getVideoOrAudioManifest(IDvrStreamStore store) {  
    IDvrManifest manifest = store.getManifest();
boolean hasAudio = store.hasAudio();
boolean hasVideo = store.hasVideo();

DvrChannelManifest channelManifest = null;

if (hasVideo) {
    channelManifest = manifest.getManifestChannel(IVHost.CONTENTTYPE_VIDEO);
} else if (hasAudio) {
    channelManifest = manifest.getManifestChannel(IVHost.CONTENTTYPE_AUDIO);
}
return channelManifest;

protected String formatUtcTime(long utc) {
    final String UTC_FORMAT = "yyyy-MM-dd HH:mm:ss";
    SimpleDateFormat dateFormatLocal = new SimpleDateFormat(UTC_FORMAT);
    return dateFormatLocal.format(new Date(utc));
}

protected String formatDvrTime(long t) {
    return String.format("%.3f", t/1000.0);
}

protected String formatPacketTime(long t) {
    return String.format("%s", t);
}

public class StartEndTimes {
    long start = -1;
    long end = -1;

    public StartEndTimes(long start, long end) {
        this.start = start;
        this.end = end;
    }
}