Ando Media Client-Side Tracking

API Description

Ando Media Client-Side Tracking
Document Version 1.0 - Last Modified 1/4/2010

Table of Contents

Ando Media Client-Side Tracking

This document describes the methods that Ando Media clients may use to track listener statistics on the client-side, or player-side. All the methods (except for the Advanced Method) described here require javascript and cookies to be enabled on the listener's browser.

Background

Client-side tracking is used in order to send listener usage statistics to Ando Media from the listener's browser. It is done using standard methods and was created to be as simple to implement as possible. There are two primary methods that can be used, a "javascript only" solution as well as a "javascript + flash" solution.

Tracking API

Regardless of which implementation solution has been chosen, you will interface with the Tracking API in the same fashion. You will make all API calls through a javascript object called "AndoTrack". This object has the following methods:

- AndoStartTracking(stationId, extra);
- AndoPauseTracking();
- AndoResumeTracking();
- AndoEndTracking();
- AndoEvent(eventName, eventValue);
- AndoGetDebug();

AndoStartTracking(stationId, extra)

This function takes a stationId (mandatory) and an string (optional). Station Id is required and represents the station id that will be associated with the tracking statistics. The optional string is a querystring that will be added onto the initial tracking call and is used for special cases. This function begins the listening session and starts a periodic "pingback" to Ando Media to indicate that the listener is still listening. It is the responsibility of the calling code to make sure that this function is called when the listener begins to listen to a stream.

For clients/players that support multiple streams, each one a different station id, you may call AndoStartTracking() with a different station Id (assuming the listener switched stations) and any previous tracking calls will be stopped and the new tracking calls (with the new station id) will begin.

AndoPauseTracking()

This function is intended to be called when a listener pauses a stream. By making this call, the tracking code will stop the automatic "ping" logic that it performs. There is an automatic timeout build into the tracking code so that if the tracking logic is "paused" for more than 3 minutes, and the listener "resumes" listening, a new session will be started, otherwise the existing listening session will be reused.

AndoResumeTracking()

This function is called if a player is resumed from a previously paused state. If the AndoResumeTracking()

call is made within 3 minutes of the AndoPauseTracking() call, then the existing session is used, otherwise a new session is created and the automatic pings will resume.

AndoEndTracking()

This function will halt any active tracking calls that are being performed. If this function is called, no more tracking activity will be recorded. To start a new session, the function AndoStartTracking() must be called.

AndoEvent(eventName, eventValue)

This function can be used to send discrete tracking events that will be tied to the listening session. Currently, we do not support any reporting on these events, and so we do not recommend that it is used at this time.

AndoGetDebug()

This function can be used to retrieve debugging information from the tracking code. This function returns a string that is intended to be displayed on the player page for debugging purposes. This should only be used during the initial development of the tracking code.

Javascript Only Solution

This method uses **only** javascript to send usage information. It should be used only in the case where the player or page doesn't have any existing flash components. Because this solution is purely javascript, there are some security limitations that we were required to handle, and thus the error-handling capability of this solution is not as good as the flash-based solution.

In order to use this solution, the following code should be placed on the page serving the player.

```
<script src="http://lt.andomedia.com/ltjs.php" type="text/javascript"></script>
```

After this has been added, you can now interface with the tracking code via a javascript object named "AndoTrack". See the Code Examples section to see what these calls look like.

Javascript + Flash Solution

This method uses a mixture of javascript and flash to make the tracking calls. Javascript is used only as a simple interface to the flash object that will perform the actual tracking calls. By using this method, we are not limited by the Javascript security issues that come with the pure javascript method, and thus the flash tracking object can be more "error-resilient" because it can directly communicate with Ando Media's servers.

In order to use this solution, the following code should be placed on the page serving the player.

```
<script src="http://lt.andomedia.com/ltflash.php" type="text/javascript"></script>
```

After this has been added, you can now interface with the tracking code via a javascript object named "AndoTrack". See the Examples section to see what these calls look like.

Code Examples

Windows Media Player Embedded Control

In this example, the player that will be implementing the tracking logic is a standard webpage that contains a Windows Media Embedded Control. In this case, we are using the pure Javascript method so we don't add the requirement of downloading and installing flash. Also, we will be using JScript (Microsoft's version of Javascript) to interface with the player itself to receive events such as "player stop" and "player start".

```
<script src="http://lt.andomedia.com/ltjs.php" type="text/javascript"></script>

<script id="clientEventHandlersJS" language="JScript"
FOR="WM" EVENT="playStateChange(NewState)">
<!--
var paused = false;
var andoStationId = 2797;
switch (NewState) {

      case 1:
            AndoTrack.AndoEndTracking();
            paused = false;
            break;
      case 2:
            AndoTrack.AndoPauseTracking();
            paused = true;
            break;
      case 3:
            if (paused) {
                  AndoTrack.AndoResumeTracking();
                  paused = false;
            }
            else {
                  AndoTrack.AndoStartTracking(andoStationId);
            }
            break;
      default:
}
//-->
</script>
```

Flash Player Embedded Control

In this example, the player that will be implementing the tracking logic is a standard webpage that contains an embedded flash player. In this case, we will use the Javascript+Flash method. In this example, we will create Javascript "wrapper functions" which will invoke the underlying tracking API. If the flash player that is being used has javascript-callback capabilities for player events (like start, stop, pause), then you will simply invoke the tracking API functions from those callbacks. If the flash player does not expose a javascript callback for these events, you must make explicit getURL() calls within the Actionscript of the player to invoke the javascript functions for the corresponding player events.

```
<script src="http://lt.andomedia.com/ltflash.php" type="text/javascript"></script>

<script language="Javascript">
var andoStationId = 2797;
function StartTracking()
{
      AndoTrack.AndoStartTracking(andoStationId);
}
function EndTracking()
{
      AndoTrack.AndoEndTracking();
}
function PauseTracking()
{
      AndoTrack.AndoPauseTracking();
}
function ResumeTracking()
{
      AndoTrack.AndoResumeTracking();
}
</script>
```

Advanced Method - Direct HTTP Calls

The section describes the "Advanced Method" of listener tracking. It is designed for embedded devices or custom applications that do not run in a web browser. To use this method, the client application must be able to make a standard HTTP GET request, passing in parameters, and also be able to parse the result (which is a comma-delimited plain text result string).

This method requires two different calls, NEW LISTENER and PING. NEW LISTENER is called when a listen session has begun, and then based on the return value from NEW LISTENER, the PING request will be performed on a periodic basis. These two calls are described on the following page

New Listener

```
http://lt.andomedia.com/lt?sid=1234&dev=My%20Device&dist=My%20Distribution
%20ID&ss=My%20Source&ps=My%20Player&vid=12345678
```

Parameter	Description
sid	Station Identifier (this is an Ando Media station id)
dev	Device String (optional)
dist	Distribution String (optional)
ss	Source String (optional)
ps	Player String (optional)
vid	Visitor ID (optional)

The URL call will return (in plain text) a single result line that will look like the following :

```
60,MTAuMjQ4LjIxNS4xOTd+ODIzNTdBRkMtQkQ0Mi00MjMzLUEyREItQUM4RDVDMEQ1NTMw
```

There are two fields in the result, separated by a comma. The first field represents the "ping frequency" and is the amount of time, in seconds, that the client should make a call to the "ping" form of the URL (described below). The second field is a unique identifier that MUST be passed into each and every ping call. This string will be unique for each listener.

PING

```
http://lt.andomedia.com/lt?guid=MTAuMjQ4LjIxNS4xOTd%2BODIzNTdBRkMtQkQ0Mi00MjMzLUEyRE
ItQUM4RDVDMEQ1NTMw
```

This URL call takes only a single parameter, which is the unique identifier that was returned via the NEW LISTENER call. Note that as with all HTTP calls, each parameter must be URL encoded.

The Ping URL call returns the same form as the NEW LISTENER call.

```
60,MTAuMjQ4LjIxNS4xOTd+ODIzNTdBRkMtQkQ0Mi00MjMzLUEyREItQUM4RDVDMEQ1NTMw
```

where the first field represents the time (in seconds) to submit the next ping call.

If any errors result in the initial NEW LISTENER call, the client code MUST re-initiate the NEW LISTENER call until a successful NEW LISTENER call is made. Any errors making PING requests can be safely ignored, and the previous successful "ping frequency" should be used.

Interaction Data

Additionally, we can support "session interaction" data. This is data that will be associated with the listener session demarcated by the NEW/PING calls and provides a method by which specific "interaction points" can be reported. These are actions that a client-defined and are the responsibility of each client to manage and maintain.

Some examples of "interaction points" might be :

- Listener clicks on a banner ad
- Listener switches "sub-channels" within the player.

The form of this call is as follows :

```
http://lt.andomedia.com/lt?guid=MTAuMjQ4LjIxNS4xOTd%2BODIzNTdBRkMtQkQ0Mi00MjMzLUEyRE
ItQUM4RDVDMEQ1NTMw&a=i&in=genre_change&iv=Classic%20Rock
```

Parameter	Description
guid	The unique session identifier from the initial NEW call
a	Action code (i = interaction)
in	Interaction point name
iv	Interaction point value

Note that the interaction data calls should be performed in ADDITION to the normal PING calls that are required to continue to keep track of the listening session.

Currently, we do not support any reporting on these events, and so we do not recommend their use at this time.

Implementation Guidelines

This section describes the general implementation guidelines that should be followed in order to provide a problem-free implementation of the conversation flow.

The general conversation flow for listener tracking is as follows :

1. On behalf of the listener, a HTTP request is made from the IP of the listener that will perform a "NEW LISTENER" action (described above).
2. This HTTP call will return 2 fields, a frequency and a unique identifier. Both values must be saved and used for future calls.
3. The "NEW LISTENER" call should return very quickly (< 500ms, but could be more depending on the connection speed of the listener), however it should be good practice to have a appropriate timeout value (5-10 seconds) for making this call. It is also suggested that the HTTP call be done in a separate thread if at all possible in order to not impede the listening experience.
4. In the event of an error (either a HTTP return of 500 or a return value that does not contain the required 2 fields), the call must be made again until a successful "NEW LISTENER" call is achieved. It is recommended that in the event of a failure, subsequent calls to "NEW LISTENER" should be limited to once a minute.
5. After a successful call of "NEW LISTENER", the calling application will need to perform periodic "PING" calls on behalf of the listener (described above). The frequency of these calls (in seconds) is returned as part of the "NEW LISTENER" call. In addition to the frequency, a unique identifier (guid) is also returned as part of the "NEW LISTENER" call. This guid is unique for each listener and should be passed as a single parameter to the "PING" call. A successful "PING" call will return 2 fields just like the "NEW LISTENER" call, representing the frequency of the next "PING" as well as an echo of the guid that was passed in.
6. In the event of an error on the "PING" call (either a HTTP return of 500 or a return value that does not contain the required 2 fields), the call should be abandoned, and the application running on behalf of the listener should use the previous frequency value to determine the next time to submit the "PING" request.
7. After a successful call of "PING", the new frequency returned by the call should be used for the next "PING" call, and also a validation of the returned guid should be performed in order to make sure that it matches the original one sent in.
8. In the event of a "long term" failure (i.e. one that lasts more than an hour), the PING calls should be abandoned and the application should attempt to send in a "NEW LISTENER" request.

Visitor ID

A visitor id is a unique identifier that can be used to track listener activity across listening sessions. It is a value that can be managed either by the calling application or (if cookies are supported) can be managed automatically by LT. If a visitor ID is passed in on the NEW LISTENER request then it will be saved and stored as part of the listening session. If it is not passed in, then LT will look for a cookie that is set (called vid) and will then use that. If no cookie is passed in, then one is set on the HTTP response so that future requests by that listener can be tracked.