How to Create Custom MRSS Using The Flexible Feed XSLT

Last Modified on 01/23/2022 11:50 am IST

Kaltura provides various out of the box syndication feeds (variations of MRSS) to suite different syndication destinations such as Google's WebMasters Tools, Yahoo!, Bing, iTunes and more. Often, publishers will need a customized syndication feed to suite a different format and destination. Using Kaltura's "Flexible Feed" Syndication format, developers are able create custom syndication feeds. It is possible to customize a syndication feed by submiting an XSLT to Kaltura.

For example -

- Currently there is no one agreed-upon standard for syndication feeds. Although the MRSS format is gaining popularity it is not strictly defined, and there are different ways to implement MRSS as well as various variations of it in use in the market. Issues with the standard includes: optional elements that will not always be validate and the structure/hierarchy of elements is varied. To learn more about the MRSS spec visit: https://www.bing.com/webmaster/help/how-to-submit-sitemaps-82a15bd4
- In addition to MRSS iTunes's format has also gained popularity. Much like the MRSS format there are slight variations for iTunesU that require additional tags.
- Boxee's compliant MRSS feed specification include slight variations from Yahoo!'s original format.

The Flexible Feed provides support for different feed formats, including MRSS variations, itunes variations and etc. with minimal development work. By applying XSLT transformation to the base Kaltura feed, custom feeds can be created without the need to write server code. The XSLT transformation is performed separately on each item (entry) in the feed, improving transformation performance.

The XSLT is executed item by item in order to support large XMLs (up to 10K items), therefore:

- There must be an item tag for each item.
- No values will be retrieved prior to the item tag.

The KalturaGenericSyndicationFeed

For every Kaltura account, a generic feed is created outlining all the available fields in the specific account. Fields include Custom Metadata fields associated with the Kaltura account, information about video flavors (renditions) and more.
The KalturaGenericSyndicationFeed is the base feed from which the Flexible Feed will perform the desired transformation (by applying the given XSLT file) and generate the

custom feed format desired.

To retrieve the Generic Kaltura Syndication Feed, open the Kaltura API Test Console and follow these steps:

1. Generate a valid Kaltura Session using the session.start or user.loginByLoginId actions.
2. Select the ***syndicationFeed*** service.
3. Select the ***add*** action.
4. Under syndicationFeed, select ***KalturaGenericSyndicationFeed*** and click Edit next to it.
5. In the ***syndicationFeed:name*** field, enter a name (e.g: myGenericFeed) and make sure the checkmark next to the field is marked.
6. If you are not using End-User Entitlements, make sure to set **syndicationFeed:enforceEntitlement** to 0 (default is 1).
7. Click the ***Send*** button.
8. In the result, locate the ***feedUrl*** node, and copy the URL inside it (it should look like this: http://www.kaltura.com/api_v3/getFeed.php?partnerId=309&feedId=1_xw6gi6b4)
9. Download the XML from the URL described in ***feedURL***. This is the Generic Feed of that Kalura account.

Creating The Flexible Feed (The XSLT)

After using the KalturaGenericSyndicationFeed to generate a base feed for the Kaltura account, use it to map the desired fields to any custom format of choice by creating an XSLT file.

Additionally, Kaltura provides a generic schema for the syndication feed. The schema can be used as a reference to map field names and structure.

The XSLT should be divided to 2 templates:

- The Feed Header and Footer part:

```
<xsl:template name="rss" match="/">
<xsl:apply-templates name="item" select="channel/items/item" />
```

- The Feed Item part (Item = each entry processed):

```
<xsl:template name="item" match="item">
```

Click below to see full example of valid XSLT including both Header and Item templates:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:media="http://search.yahoo.com/mrss/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
exclude-result-prefixes="xs">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" />
  <xsl:template name="rss" match="/">
    <rss xmlns:media="http://search.yahoo.com/mrss/">
      <xsl:for-each select="rss">
        <xsl:attribute name="version">
          <xsl:value-of select="string(number(string(@version)))" />
        </xsl:attribute>
        <xsl:apply-templates name="item"
        select="channel/items/item" />
      </xsl:for-each>
    </rss>
  </xsl:template>
  <xsl:template name="item" match="item">
    <xsl:variable name="var1_content" select="content" />
    <xsl:variable name="var2_resultof_cast"
    select="string(title)" />
    <item>
      <title>
        <xsl:value-of select="$var2_resultof_cast" />
      </title>
      <link>
        <xsl:value-of select="string(link)" />
      </link>
      <xsl:for-each select="$var1_content">
        <media:content>
          <xsl:attribute name="url">
            <xsl:value-of select="string(@url)" />
          </xsl:attribute>
        </media:content>
      </xsl:for-each>
      <media:title>
        <xsl:value-of select="$var2_resultof_cast" />
      </media:title>
      <media:thumbnail>
        <xsl:value-of select="string(thumbnailUrl)" />
      </media:thumbnail>
      <media:description>
        <xsl:value-of select="string(description)" />
      </media:description>
      <media:keywords>
        <xsl:call-template name="implode">
          <xsl:with-param name="items"
          select="$var1_content/tags/tag" />
        </xsl:call-template>
      </media:keywords>
    </item>
  </xsl:template>
  <xsl:template name="implode">
    <xsl:param name="items" />
    <xsl:param name="separator" select="','" />
    <xsl:for-each select="$items">
      <xsl:if test="position() > 1">
        <xsl:value-of select="$separator" />
      </xsl:if>
      <xsl:value-of select="." />
    </xsl:for-each>
  </xsl:template>
```

```
</xsl:stylesheet>
```

Roku Channel MRSS

Roku uses a standard Yahoo! MRSS format, with slight additions. The following XSLT provides the correct format for syndication of content in a Ruko channel.

This XSLT is provided as a sample reference only, note that there are more steps required to enable sharing of content in a Roku channel. Please contact your Kaltura Account Manager for more details.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:media="http://search.yahoo.com/mrss/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:php="http://php.net/xsl"
exclude-result-prefixes="xs">
  <xsl:output method="xml" />
  <xsl:template match="*">
   <xsl:element name="{local-name()}">
     <xsl:apply-templates/>
   </xsl:element>
</xsl:template>

  <xsl:template name="rss" match="/">
    <rss xmlns:media="http://search.yahoo.com/mrss/">
      <xsl:for-each select="rss">
        <xsl:variable name="var1_channel" select="channel" />
        <xsl:attribute name="version">
          <xsl:value-of select="string(@version)" />
        </xsl:attribute>
        <channel>
          <title>
            <xsl:value-of select="string($var1_channel/title)" />
          </title>
          <link>
            <xsl:value-of select="string($var1_channel/link)" />
          </link>
          <description>
            <xsl:value-of select="string($var1_channel/description)" />
          </description>
          <xsl:apply-templates name="item"
          select="channel/items/item" />
        </channel>
      </xsl:for-each>
    </rss>
  </xsl:template>
  <xsl:template name="item" match="item">
    <xsl:variable name="var1_content" select="content" />
<xsl:variable name="var1_media" select="media" />
    <xsl:variable name="var2_resultof_cast" select="string(title)" />
    <xsl:variable name="var2_thumbnailUrl" select="thumbnailUrl" />
<xsl:variable name="var2_createdAt" select="createdAt" />
<xsl:variable name="var2_player" select="player" />
    <item>
      <title>
        <xsl:value-of select="$var2_resultof_cast" />
```

```
      <xsl:value-of select="$var2_resultof_cast" />
    </title>
    <link>
      <xsl:value-of select="string(link)" />
    </link>
    <xsl:for-each select="$var1_content">
       <media:content>
        <xsl:attribute name="url">
          <xsl:value-of select="string(@url)" />
        </xsl:attribute>
  <xsl:attribute name="duration">
   <xsl:value-of select="string(floor(number(string($var1_media/duration))))" />
  </xsl:attribute>
       </media:content>
    </xsl:for-each>
    <media:title>
      <xsl:value-of select="$var2_resultof_cast" />
    </media:title>
  <media:description>
      <xsl:value-of select="string(description)" />
    </media:description>
  <media:keywords>
      <xsl:call-template name="implode">
        <xsl:with-param name="items" select="tags/tag" />
      </xsl:call-template>
    </media:keywords>
    <media:thumbnail>
      <xsl:copy-of select="$var2_thumbnailUrl/@node()" />
      <xsl:copy-of select="$var2_thumbnailUrl/node()" />
    </media:thumbnail>
  <media:player>
  <xsl:attribute name="url">
          <xsl:value-of select="string($var2_player/@url)" />
        </xsl:attribute>
      <xsl:copy-of select="$var2_player/node()" />
    </media:player>
  <media:rating scheme="urn:simple">b</media:rating>
  <pubdate>
      <xsl:value-of select="php:function('date', 'Y-m-d H:i:s', sum($var2_createdAt))"/>
    </pubdate>
   </item>
  </xsl:template>
  <xsl:template name="implode">
   <xsl:param name="items" />
   <xsl:param name="separator" select="','" />
   <xsl:for-each select="$items">
    <xsl:if test="position() &gt; 1">
     <xsl:value-of select="$separator" />
    </xsl:if>
    <xsl:value-of select="." />
   </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Boxee Channel MRSS

To successfully submit Kaltura feeds to Boxee's library, a new format has to be created, pointing the user to a hosted player page and including Boxee's special fields.

This XSLT is provided as a sample reference only, note that there are more steps

required to enable sharing of content in Boxee. Please contact your Kaltura Account Manager for more details.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:boxee="http://boxee.tv/spec/rss/"
xmlns:media="http://search.yahoo.com/mrss/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:php="http://php.net/xsl" version="1.0"
exclude-result-prefixes="xs">
 <xsl:output method="xml" />
 <xsl:template match="*">
  <xsl:element name="{local-name()}">
   <xsl:apply-templates />
  </xsl:element>
 </xsl:template>
 <xsl:template name="rss" match="/">
  <rss>
   <xsl:for-each select="rss">
    <xsl:variable name="var1_channel" select="channel" />
    <xsl:attribute name="version">
     <xsl:value-of select="string(@version)" />
    </xsl:attribute>
    <channel>
     <title>
      <xsl:value-of select="string($var1_channel/title)" />
     </title>
     <link>
      <xsl:value-of select="string($var1_channel/link)" />
     </link>
     <description>
      <xsl:value-of select="string($var1_channel/description)" />
     </description>
     <xsl:apply-templates name="item"
     select="channel/items/item" />
    </channel>
   </xsl:for-each>
  </rss>
 </xsl:template>
 <xsl:template name="item" match="item">
  <xsl:variable name="var1_content" select="content" />
  <xsl:variable name="var1_media" select="media" />
  <xsl:variable name="var2_resultof_cast"
  select="string(title)" />
  <xsl:variable name="var2_thumbnailUrl" select="thumbnailUrl" />
  <xsl:variable name="var2_createdAt" select="createdAt" />
  <item>
   <title>
    <xsl:value-of select="$var2_resultof_cast" />
   </title>
   <boxee:media-type>
    <xsl:attribute name="type">movie</xsl:attribute>
    <xsl:attribute name="name">Feature</xsl:attribute>
    <xsl:attribute name="expression">full</xsl:attribute>
   </boxee:media-type>
   <link>
    <xsl:value-of select="string(link)" />
   </link>
   <media:content>
    <xsl:attribute name="url">
     <xsl:value-of select="string(link)" />
```

```
      </xsl:attribute>
      <xsl:attribute name="type">
        <xsl:value-of select="'application/x-shockwave-flash'" />
      </xsl:attribute>
      <xsl:attribute name="duration">
        <xsl:value-of select="string(number(string($var1_media/duration)) div number('1000'))" />
      </xsl:attribute>
    </media:content>
    <media:title>
      <xsl:value-of select="$var2_resultof_cast" />
    </media:title>
    <media:description>
      <xsl:value-of select="string(description)" />
    </media:description>
    <media:keywords>
      <xsl:call-template name="implode">
        <xsl:with-param name="items" select="tags/tag" />
      </xsl:call-template>
    </media:keywords>
    <media:thumbnail>
      <xsl:copy-of select="concat($var2_thumbnailUrl/@node(),'/width/300')" />
      <xsl:copy-of select="concat($var2_thumbnailUrl/node(),'/width/300')" />
    </media:thumbnail>
    <media:rating scheme="urn:simple">b</media:rating>
    <pubdate>
      <xsl:value-of select="php:function('date', 'Y-m-d H:i:s', sum($var2_createdAt))" />
    </pubdate>
  </item>
 </xsl:template>
 <xsl:template name="implode">
  <xsl:param name="items" />
  <xsl:param name="separator" select="','" />
  <xsl:for-each select="$items">
    <xsl:if test="position() &gt; 1">
      <xsl:value-of select="$separator" />
    </xsl:if>
    <xsl:value-of select="." />
  </xsl:for-each>
 </xsl:template>
</xsl:stylesheet>
```

JSONified Feed (JSON representation of a feed)

The sample below emphasizes the strength and flexibility of the Flexible Feed. Leveraging XSLT, it is possible to completely transform the format of the feed, and even generate non-XML output. In this case, the feed is outputted as JSON to be easily parsed by Javascript applications.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text" omit-xml-declaration="yes"/>
 <xsl:param name="KalturaHasNextItem" />

   <xsl:template name="rss" match="/">{<xsl:for-each select="rss">
  <xsl:variable name="indent"><xsl:text>   </xsl:text></xsl:variable>
  <xsl:variable name="channel-title"><xsl:call-template name="safe-json-string"><xsl:with-param name="text"
select="string(channel/title)" /></xsl:call-template></xsl:variable>
  <xsl:variable name="channel-link"><xsl:call-template name="safe-json-string"><xsl:with-param name="text"
select="string(channel/link)" /></xsl:call-template></xsl:variable>
```

```
  <xsl:variable name="channel-description"><xsl:call-template name="safe-json-string"><xsl:with-param
name="text" select="string(channel/description)" /></xsl:call-template></xsl:variable>

  <xsl:attribute name="version">
        <xsl:value-of select="string(@version)"/>
       </xsl:attribute>
    "channel":{
   "title":"<xsl:value-of select="$channel-title"/>",
   "link":"<xsl:value-of select="$channel-link"/>",
   "description":"<xsl:value-of select="$channel-description"/>",
   "items": {<xsl:apply-templates name="item" select="channel/items/item"/>
   }
       }
     </xsl:for-each>
 }</xsl:template>

 <!-- Item Property-->
 <xsl:template name="item" match="item">
    "<xsl:value-of select="string(entryId)"/>":{
  <xsl:apply-templates select="*"><xsl:with-param name="indent"><xsl:text>   </xsl:text></xsl:with-param>
</xsl:apply-templates>
    }<xsl:if test="($KalturaHasNextItem = '1') or (count(/rss/channel/items/item) &gt; 1 and position() &lt; last())">,
</xsl:if>
    </xsl:template>



    <!-- Object or Element Property-->
    <xsl:template match="*">
     <xsl:param name="indent"/>
     <xsl:value-of select="$indent"/>
     <xsl:text>"</xsl:text>
        <xsl:value-of select="name()"/>" : <xsl:call-template name="Properties"><xsl:with-param name="indent"
select="$indent"/></xsl:call-template>
    </xsl:template>

    <!-- Array Element -->
    <xsl:template match="*" mode="ArrayElement">
     <xsl:param name="indent"/>
     <xsl:value-of select="$indent"/>
        <xsl:call-template name="Properties"><xsl:with-param name="indent" select="$indent"/></xsl:call-template>
    </xsl:template>

    <!-- Object Properties -->
    <xsl:template name="Properties">
     <xsl:param name="indent"/>
        <xsl:variable name="childName" select="name(*[1])"/>
        <xsl:choose>
           <xsl:when test="not(*|@*)">
    <xsl:variable name="myVar1">
     <xsl:text>"</xsl:text>
     <xsl:call-template name="safe-json-string">
      <xsl:with-param name="text" select="." />
     </xsl:call-template>
     <xsl:text>"</xsl:text>
    </xsl:variable>
    <xsl:value-of select="$myVar1" disable-output-escaping="yes"/>
   </xsl:when>
          <xsl:when test="count(*[name()=$childName]) > 1">
           <xsl:text>{
   </xsl:text>
    <xsl:value-of select="$indent"/>
```

```
    <xsl:text>"</xsl:text>
    <xsl:value-of select="$childName"/>
    <xsl:text>" : [
  </xsl:text>
    <xsl:apply-templates select="*" mode="ArrayElement">
     <xsl:with-param name="indent"><xsl:value-of select="$indent"/><xsl:text>  </xsl:text></xsl:with-param>
    </xsl:apply-templates>
    <xsl:text>
   </xsl:text>
    <xsl:value-of select="$indent"/>
    <xsl:text>]
  </xsl:text>
    <xsl:value-of select="$indent"/>}
          </xsl:when>
          <xsl:otherwise>
           <xsl:text>{
  </xsl:text>
    <xsl:apply-templates select="@*">
     <xsl:with-param name="indent"><xsl:value-of select="$indent"/><xsl:text> </xsl:text></xsl:with-param>
    </xsl:apply-templates>
    <xsl:if test="(count(@*) > 0) and (count(*[name()=$childName]) > 0)"><xsl:text>,
  </xsl:text>
    </xsl:if>
    <xsl:apply-templates select="*"><xsl:with-param name="indent"><xsl:value-of select="$indent"/><xsl:text>
</xsl:text></xsl:with-param></xsl:apply-templates>
    <xsl:text>
  </xsl:text>
    <xsl:value-of select="$indent"/>
    <xsl:text>}</xsl:text>
   </xsl:otherwise>
       </xsl:choose>
       <xsl:if test="following-sibling::*">,
       </xsl:if>
    </xsl:template>

    <!-- Attribute Property -->
    <xsl:template match="@*">
     <xsl:param name="indent"/>
  <xsl:variable name="myVar">
   <xsl:call-template name="safe-json-string">
    <xsl:with-param name="text" select="." />
   </xsl:call-template>
  </xsl:variable>
  <xsl:value-of select="$indent"/>
  <xsl:text>"</xsl:text>
  <xsl:value-of select="name()"/>" : "<xsl:value-of select="$myVar"/>
  <xsl:text>"</xsl:text>
  <xsl:if test="position() != last()">,
  </xsl:if>
    </xsl:template>


 <!-- replace string -->
 <xsl:template name="string-replace-all">
    <xsl:param name="text" />
    <xsl:param name="replace" />
    <xsl:param name="by" />
    <xsl:choose>
      <xsl:when test="contains($text, $replace)">
        <xsl:value-of select="substring-before($text,$replace)" />
        <xsl:value-of select="$by" />
        <xsl:call-template name="string-replace-all">
```

```
        <xsl:with-param name="text" select="substring-after($text,$replace)" />
        <xsl:with-param name="replace" select="$replace" />
        <xsl:with-param name="by" select="$by" />
      </xsl:call-template>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$text" />
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>

 <xsl:template name="safe-json-string">
  <xsl:param name="text" />
  <xsl:variable name="myVar1Temp">
   <xsl:call-template name="string-replace-all">
     <xsl:with-param name="text" select="$text" />
     <xsl:with-param name="replace" select="'''" />
     <xsl:with-param name="by" select="'\'" />
   </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="myVar1">
   <xsl:call-template name="string-replace-all">
     <xsl:with-param name="text" select="$myVar1Temp" />
     <xsl:with-param name="replace" select="'&amp;quot;'" />
     <xsl:with-param name="by" select="'&amp;quot;'" />
   </xsl:call-template>
  </xsl:variable>
  <xsl:value-of select="$myVar1"/>
 </xsl:template>

</xsl:stylesheet>
```