

# Mobilize your content for Kinoma Play

It's the recreation of the Internet, it's the recreation of the PC story, and it is before us...and it is very likely that it will happen in the next year.

— Google CEO Eric Schmidt to the World Economic Forum, 1/25/2008

## Overview

This whitepaper is for content providers who want to publish their content for Kinoma Play.

There are two parts to publishing content for Kinoma products — the **feed**, and the **media**.

The **feed** is the “directory” to your media, and can be either RSS (which you already use for podcasts) or OPML . For simple podcast-like feeds, RSS works great. If you want to organize your content into folders, you’ll use OPML.

The **media** is your video and/or audio compressed to a mobile-friendly format. You’re probably already publishing media in a supported format, in which case you’ll just need to create another “flavor” at a mobile-friendly size and bitrate.

# Table of contents

---

About.....	4
Introduction .....	4
What's involved?.....	4
Overview .....	4
RSS feeds vs. OPML feeds .....	4
When to use RSS .....	5
When to use OPML .....	5
Creating a basic RSS (podcast) feed .....	6
A podcast feed with a single program .....	6
A podcast feed with multiple programs.....	6
Character encoding .....	7
Enhancing your RSS feed.....	7
Adding thumbnail icons .....	7
Adding text summaries .....	9
Creating a basic OPML feed .....	10
Character encoding.....	10
Enhancing your OPML feed.....	10
Adding thumbnail icons .....	10
Using Kinoma extensions to OPML.....	11
Validate your feeds .....	13
Feed Validator.....	13
OPML Validator.....	13
Internet Explorer.....	13
Feed compression .....	14
Resources .....	14
Media RSS specification .....	14
Base64 Encoder (web-based Base64 encoder).....	14
Base64 Online (web-based Base64 encoder) .....	14
Encoding your content.....	15
Encoding your audio .....	16
Universal .....	16
Encoding video.....	20
Universal .....	20
Best Quality.....	21
Encoding photos .....	25

Universal .....	25
Target size.....	25
Create your content feed.....	26
A feed with multiple programs .....	26
A feed with both Universal and Best Quality content .....	26
Add UPnP classes .....	27
Add descriptions .....	28
Add thumbnails.....	29
Search.....	31
Add a Search item to your guide.....	31
Parsing Search requests.....	32
Returning Search results .....	32
Submit your feed to Kinoma Guide.....	34
Thank you! .....	35

# About

---

This whitepaper describes the incremental changes you can make to your production and publishing workflows to syndicate content to **Kinoma Play** for Windows Mobile, and for other platforms that Kinoma Play becomes available for.

Content prepared according to these guidelines is will also be compatible with Kinoma Player 4 EX on Palm OS.

## Introduction

By targeting **Kinoma Play** you can extend your content's reach to affluent, educated professionals and consumers on their most intimate device — their phone.

Every copy of Kinoma Play features **Kinoma Guide**, a content discovery system that allows users to quickly find and play the world's best mobile-ready content.

You can publish one or more “channels” of content to Kinoma Guide by making small changes to your encoding and publishing workflows, and then notifying Kinoma that your content is Kinoma-ready.

## What's involved?

To publish your content to Kinoma Play:

- Encode your content for mobile
- Create a feed for your mobile content
- Submit your feed to Kinoma

## Overview

Among its many capabilities, Kinoma Play is a podcatcher. A podcatcher — also known as a podcast receiver or a podcast feed aggregator — are simply feed readers optimized for video/audio. As a podcatcher, Kinoma Play allows users to bookmark and play podcasts right from within Kinoma Play.

## RSS feeds vs. OPML feeds

An RSS feed is a file that contains a list of items, typically blog posts or links to video/audio podcast content. We're interested in RSS feeds that contain a list of links to podcast content, generally just called podcast feeds. Kinoma Play also supports RSS feeds with links to photos, also known as photocasts or photostreams.

An OPML feed is a file that contains a hierarchical list, typically used to aggregate multiple feeds from one or more sources. We're interested in OPML feeds that contain a list of podcast feeds.

## **When to use RSS**

To create one or more channels intended for use in the Kinoma Media Guide, create one or more podcast feeds.

## **When to use OPML**

If you want to make several podcast feeds available via a single URL, create an OPML file containing links to the podcast feeds.

When used in Kinoma Play — either via the Kinoma Media Guide, or by users who manually open the URL to your OPML file — the OPML file will be represented as a category containing multiple channels.

Most podcasters just create individual RSS feeds. Organizations with large number of feeds, such as National Public Radio, often also create OPML feeds.

# Creating a basic RSS (podcast) feed

---

## A podcast feed with a single program

At its simplest, a one-item podcast feed looks like this:

```
<?xml version="1.0"?>
<rss version="2.0">
<channel>
  <title>My Podcast</title>
  <link>http://www.mydomain.com/</link>
  <description>My latest crazy-cool podcast</description>
  <item>
    <title>My Podcast</title>
    <enclosure url="http://mydomain.com/media/myPodcast.mp3" ←
      length="640561" type="audio/mp3" />
    <guid>http://www.mydomain.com/podcasts/media/myPodcast.mp3</guid>
  </item>
</channel>
</rss>
```

Note that this feed contains a single `<item>`, representing a single program in Kinoma Play.

When users choose a podcast feed with a single program, Kinoma Play will begin playing the item immediately.

## A podcast feed with multiple programs

To include multiple programs in a podcast, simply add an additional `<item>` for each program.

```

<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <title>My Podcast</title>
    <link>http://www.mydomain.com/</link>
    <description>My latest crazy-cool podcasts</description>
  <item>
    <title>My Podcast - 2007-07-01</title>
    <enclosure url="http://www.mydomain.com/media/myPodcast.mp3"
      length="441540" type="audio/mp3" />
    <guid>http://www.mydomain.com/podcasts/media/podcast-07-07-01.mp3</guid>
  </item>
  <item>
    <title>My Podcast - 2007-07-08</title>
    <enclosure url="http://mydomain.com/media/myPodcast.mp3"
      length="427639" type="audio/mp3" />
    <guid>http://www.mydomain.com/podcasts/media/podcast-07-07-08</guid>
  </item>
  <item>
    <title>My Podcast - 2007-07-15</title>
    <enclosure url="http://mydomain.com/media/myPodcast.mp3"
      length="431086" type="audio/mp3" />
    <guid>http://www.mydomain.com/podcasts/media/podcast-07-07-15</guid>
  </item>
</channel>
</rss>

```

## Character encoding

For widest compatibility, feeds should use UTF-8 character encoding. Not only should the text itself be UTF-8 encoded, but the XML header should explicitly confirm this as shown here:

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

## Enhancing your RSS feed

### Adding thumbnail icons

Kinoma Play can display a thumbnail icon for programs in RSS feeds. To add a thumbnail icon to a program, use a `<media:thumbnail>` element.

The `<media:thumbnail>` element is standardized as part of Yahoo's Media RSS extensions to RSS. Note that when you use Media RSS extensions, you must add the Media RSS namespace declaration to your feed's `<rss>` element, like this:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
```

Kinoma Play requires the following three attributes for the `<media:thumbnail>` element:

- `url` – A URI containing a Base64-encoded JPEG thumbnail image
- `width` – The width (in pixels) of the JPEG thumbnail image
- `height` – The height (in pixels) of the JPEG thumbnail image

In order for the thumbnail images to fit within the media browser entries, the width must be 44 pixels or less, and the height must be 22 pixels or less. In addition, the height should be evenly divisible by two.

The following example specifies a 20 x 20 thumbnail image:

```
<media:thumbnail width="20" height="20" ←  
url="data://image/jpeg;base64,/9j/4AAQSkZJRgABAQAAZABkAAD/7AARRHVja3 ←  
kAAQAEAAAAZAAA/+4ADkFkb2JlAGTAAAAAaf/bAIQAAQEBAQEBAQEBAQEBAQEBA ←  
QEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBA ←  
AwEBAQEBAQECAQECAGIBAgIDAwmDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMD ←  
DAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMD ←  
DAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMD ←  
8AAEQgAFAAUAwERAAIRAQMRAF/EAIUAAQADAA ←  
AAAAAAAAAAAAAAAAAgGBwoBAQEAAwEAAAAAAAAAAAAAAAAAcIBAUJBhAAAAQFAwMEA ←  
gMAAAAAAAAAAwQFBgITFBUHARYXABIiESMkJSEYMyCIEQAABAQEBQMDBQEAAAAA ←  
AAABAgMEERITBSIUfQYAIRYHFzEyIzMkCFFCUKMNZP/aAAwDAQACEQMRAD8A2oZ ←  
Sy26DL9M4kxss0TU0oaGXcWRcjR5GBXAZ5A9DqMnJCEimByqeecZkhDwjJHhKIk ←  
TiD11gGjG00BkTvN34vTDuK37H9tHLJpvRRmV08fOETogYoKnAiKbduU6eZelCM ←  
5SqqERInLEtNkUBVhbG0bWl1tw29t0kVVtVUyaDchwSrGIEVFFVoGFNentwAJzGA ←  
eZS1ETQkZyZSbAIS61f9CJGYzRNvE3SYTTub+PEoFytox01DV0Ncx6kIyokBqsm ←  
OAidECPkdRYfKESHSLTU5v2++/Hay3K71X3STdFqt5zpvLfcLM2ttRRuUTukWz1 ←  
kBBbvCEgJEHCS4FMJao1LER2duJ293Q6JZgthbau4KU6S7Z4q4EpFBgkodFc5wU ←  
REfU5DJzAAyDhHb8zNHh/mn522dv3qhkh3ium0G2aebIv1++t7Jkqr/ABM7PPqo ←  
/N+w/C/niuboTSNQmgFSX6MsYV6v28k0tbDNDFx4foydYdEy11fNUfXB+tSMI ←  
05P1jCMnOEeXAMz21LjezP14rAjt9TPZUTmw42gluw8cRmu+EVMabYYb5Z5tbJh ←  
mdSasInoxmA00IIaApGeKcJBRAX693M78kHF670fmE77gPjGZW7cjG3LW1+ABKk ←  
7tqaKQpgobACpFUSkyjDCdMRCBgEaE2E1Yt3drkNu3Soe1t802fETKCipE3J1VE ←  
1QTESiZMRUADgUxTHKVQpDacAhVaS40S4JaymYVN4gQ2fjtOYi87H8tta7BMFvQ ←  
HDbYx2xEx1vh2V+gC7rAMKE0iFy4BUOLSHQyKJBoGebu/Kd9uHaV3Z3u+L3C30b ←  
26u6rVRuZLM3h+mKKzkpVG6J1TrFCSEyAepQLHnn2TsvtFZ1zb301qNXV/TYJW ←  
5oDUHJiJM0TgYAcncpJgRNMomkIUp1zqCE5y1IMV/x46/0w23ZB9xz+RNu0g1yo ←  
uXelwPb/A0e/WL2af0m1nt9vf49Vj4P35/nB41yp+utFzmV1GpNqus5ST1r0ft6 ←  
fuq4IR5cHnVdm876/WLpFbL1Y4f8AgyVWb0p1Mc3pJijDnwiMzcP7R/unb+2a4G ←  
hvU2uvEsWn2zQffX6RM7Lb8uVM9PDv6sPvf4X6DX88aR0JME2oSyVIDLR/trwmk ←  
y/zS5S8puC/ZnWGtF6JzWryj9H+HKNSOCnGEauCMI84cHfHn6YbrRntz9xzym3e ←  
ROXaK5TPR7Pyz9Ffp/pTyfmTe2X59vUedj/84OvGviXReupy5X0arNUjgym/DX ←  
j9PL/ACzQk5w4UN1+d9Gw1+tpEo1cvk/b+6rkvkpw90+CEZuUeHH10h4A+P/Z" />
```

As shown above, the URI scheme must be "data", and the MIME type and encoding must immediately follow the scheme. Hence each `<media:thumbnail>` element must start with:

```
data://image/jpeg;base64,
```

The Base64-encoded JPEG image data must follow. (While the above example spans multiple lines, the Base64-encoded string in the XML document must not include any line breaks.)

## What is this "data://" thing in my URL?

Just like http:// and ftp://, data:// is another standard URL scheme. It is defined in RFC 2397.

Data URLs are great for small images because it allows them to be embedded directly into the feed, which can substantially reduce the time it takes to retrieve the feed. This makes them especially useful with mobile networks.

## How do I Base64-encode a JPEG or PNG file?

We used a web-based tool. The Resources section [link] at the end of this tech note lists two web-based tools that you can use to convert your JPEG file to a Base64-encoded string.

## Adding text summaries

Kinoma Play supports the `<itunes:summary>` RSS extension, as shown below.

```
<itunes:summary>Eloquent summary of the July 1 podcast</itunes:summary>
```

Any text within the `<itunes:summary>` element will be used as the item's description both in Kinoma Play's detailed list view, and in any program's Details screen.

If you use `<itunes:summary>` extension, you must declare that the feed uses the iTunes namespace as shown here:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
```

Note that if you're using both Media RSS extensions and iTunes extensions, you must to declare both as shown here:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss" ←  
xmlns:media="http://search.yahoo.com/mrss">
```

# Creating a basic OPML feed

---

At its simplest, an OPML looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<opml xmlns:kinoma="http://schema.kinoma.com/opml">
<head>
  <title>Sample</title>
  <dateCreated>Wed, 16 May 2007 14:48:38</dateCreated>
  <dateModified>Wed, 16 May 2007 14:48:38</dateModified>
  <ownerName>John Doe</ownerName>
  <ownerEmail>john@doe.com</ownerEmail>
</head>
<body>
  <outline text="News">
    <outline text="Podcast 1" type="rss" xmlUrl="http://domain.com/feeds/podcast-1.xml" />
    <outline text="Podcast 2" type="rss" xmlUrl="http://domain.com/feeds/podcast-2.xml" />
  </outline>
</body>
</opml>
```

This example OPML will create a Kinoma Play category called “Sample” which contains two RSS feeds (podcast-1.xml and podcast-2.xml).

## Character encoding

For widest compatibility, OPML feeds should use UTF-8 character encoding. Not only should the text itself be UTF-8 encoded, but the XML header should explicitly confirm this as shown here:

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

## Enhancing your OPML feed

### Adding thumbnail icons

As with RSS items, you can add icons to OPML items using the `<media:thumbnail>` element.

The `<media:thumbnail>` element must be a child of an `<outline>` element, as shown below. In this example, a thumbnail is associated with the “News” category. (Note the Base64 data has been truncated for readability.)

```
<?xml version="1.0" encoding="UTF-8"?>
<opml xmlns:kinoma=http://schema.kinoma.com/opml xmlns:media="http://search.yahoo.com/mrss/">
  <head>
    <title>Sample</title>
    <dateCreated>Wed, 16 May 2007 14:48:38</dateCreated>
    <dateModified>Wed, 16 May 2007 14:48:38</dateModified>
    <ownerName>John Doe</ownerName>
    <ownerEmail>john@doe.com</ownerEmail>
  </head>
  <body>
    <outline text="News">
      <media:thumbnail url="data://image/jpeg;base64,/9j/...+P/Z" width="20" height="20"/>
    </outline>
  </body>
</opml>
```

## Using Kinoma extensions to OPML

Kinoma has defined three extensions to OPML, which offer you additional control over how your OPML data appears in Kinoma Play, Kinoma FreePlay and Kinoma Play v4.2 or later.

If you use the Kinoma OPML extensions, you must declare the Kinoma namespace in your OPML file. This is done by adding the proper `xmlns` attribute to your `<opml>` element, as shown here:

```
xmlns:kinoma="http://schema.kinoma.com/opml">
```

## Using the `kinoma:description` extension

We've defined an OPML extension called `kinoma:description` that allows you to provide a detailed description of an entry. This description appears on the second line in the media browser detailed view, as well as in the Kinoma Play Details form.

```
<body>
  <outline text="News" kinoma:description="A best-of collection of my excellent podcasts">
    <outline text="Podcast 1" type="rss" xmlUrl="http://domain.com/feeds/podcast-1.xml" />
    <outline text="Podcast 2" type="rss" xmlUrl="http://domain.com/feeds/podcast-2.xml" />
  </outline>
</body>
```

If you don't define `kinoma:description`, the description will be empty.

## Using the `kinoma:hideurl` extension

The OPML extension `kinoma:hideurl` is used to suppress the display of URLs in the Kinoma Play user interface. If you want to hide the URL to any of your feeds, set the value of this attribute to "true".

```
<outline text="My podcast" type="rss" kinoma:hideurl="true"
xmlUrl="http://www.mydomain.com/podcasts/myPodcast.rss" />
```

## Using the kinoma:stemming extension

Stemming is a feature of Kinoma Play that tries to make titles more easily viewable on a small-screen device. For example, the example OPML below uses channel titles that are too long for a small-screen device, making it likely that the most important parts would be truncated.

```
<body>
<outline text="CNN Headline News">
  <outline text="CNN Headline News - World" type="rss" ↵
    xmlUrl="http://cnn.com/podcasts/hln-world.rss" />
  <outline text="CNN Headline News - U.S." type="rss" ↵
    xmlUrl="http://cnn.com/podcasts/hln-us.rss" />
  <outline text="CNN Headline News - Politics" type="rss" ↵
    xmlUrl="http://cnn.com/rss/hln-pol.rss" />
  <outline text="CNN Headline News - Entertainment" type="rss" xmlUrl="http://cnn.com/rss/hln-
ent.rss" />
  <outline text="CNN Headline News - Health" type="rss" xmlUrl="http://cnn.com/rss/hln-
health.rss" />
  <outline text="CNN Headline News - Tech" type="rss" ↵
    xmlUrl="http://http://cnn.com/rss/hln-tech.rss" />
  <outline text="CNN Headline News - Travel" type="rss" ↵
    xmlUrl="http://cnn.com/rss/hln-travel.rss" />
  <outline text="CNN Headline News - Living" type="rss" ↵
    xmlUrl="http://cnn.com/rss/hln-living.rss" />
  <outline text="CNN Headline News - Business" type="rss" ↵
    xmlUrl="http://cnn.com/rss/hln-buz.rss" />
  <outline text="CNN Headline News - Sports" type="rss" ↵
    xmlUrl="http://cnn.com/rss/hln-sports.rss" />
</outline>
</body>
```

To avoid this, Kinoma Play will automatically identify the part that's common to all channel titles ("CNN Headline News -") and trim it off in order to maintain readability. The result is exactly the same as if you'd defined the OPML like this:

```
<body>
  <outline text="CNN Headline News">
  <outline text="World" type="rss" xmlUrl="http://cnn.com/podcasts/hln-world.rss" />
  <outline text="U.S." type="rss" xmlUrl="http://cnn.com/podcasts/hln-us.rss" />
  <outline text="Politics" type="rss" xmlUrl="http://cnn.com/rss/hln-pol.rss" />
  <outline text="Entertainment" type="rss" xmlUrl="http://cnn.com/rss/hln-ent.rss" />
  <outline text="Health" type="rss" xmlUrl="http://cnn.com/rss/hln-health.rss" />
  <outline text="Tech" type="rss" xmlUrl="http://http://cnn.com/rss/hln-tech.rss" />
  <outline text="Travel" type="rss" xmlUrl="http://cnn.com/rss/hln-travel.rss" />
  <outline text="Living" type="rss" xmlUrl="http://cnn.com/rss/hln-living.rss" />
  <outline text="Business" type="rss" xmlUrl="http://cnn.com/rss/hln-buz.rss" />
  <outline text="Sports" type="rss" xmlUrl="http://cnn.com/rss/hln-sports.rss" />
  </outline>
</body>
```

The OPML extension **kinoma:stemming** is used to prevent Kinoma Play from “stemming” the OPML entries. This extension is represented as an attribute of the **<opml>** element.

To force Kinoma Play to show full channel titles regardless of length, set the **kinoma:stemming** to “false”.

```
<opml xmlns:kinoma="http://schema.kinoma.com/opml" kinoma:stemming="false">
```

If you don’t define **kinoma:stemming**, the default is “true”.

## Validate your feeds

It’s crucial to create valid podcast feeds, so that users see and hear what you intended. We like to use these validators:

### Feed Validator

<http://feedvalidator.org/>

Feed Validator works for all RSS and Atom feeds, including podcast feeds.

### OPML Validator

<http://validator.opml.org/>

This is technically in beta, but it’s the best OPML validator that we know of. When you successfully validate your OPML, you’ll see this image in the results:

### Internet Explorer

Both RSS and OPML feeds must be valid XML documents.

On Windows, a quick way to confirm that a feed is valid XML is to open it with Internet Explorer. If the file is not valid XML, Internet Explorer will tell you where the problem is.

## Feed compression

Kinoma products support gzip-compressed RSS and OPML feeds. This can reduce the amount of data transferred by up to 80%, so we strongly recommend it for delivering feeds.

Most web servers (including Apache and Microsoft IIS) support on-the-fly gzip compression, which means that you won't need to make any changes to the feeds themselves.

Your IT department can help set this up. If you are the IT department, it's reasonably straightforward, and you can learn the details in your web server's documentation.

With Apache 2.x, for example, this can be as easy as putting an `.htaccess` file containing `SetOutputFilter DEFLATE` in the same directory as the feeds that you want to deliver compressed.

## Resources

### Media RSS specification

<http://search.yahoo.com/mrss>

### Base64 Encoder (web-based Base64 encoder)

<http://www.opinionatedgeek.com/dotnet/tools/Base64Encode/>

To convert a file to a Base64 string suitable for icon data, click the Browse button, choose your JPEG file, and then click Encode to create the Base64 string representing the encoded version of your file.

### Base64 Online (web-based Base64 encoder)

<http://www.motobit.com/util/base64-decoder-encoder.asp>

Click the Browse button to choose your JPEG file, then click Convert the source data to create the Base64 string representing the encoded version of your file.

# Encoding your content

---

To create Kinoma-optimized content, you'll create one or two "flavors" for each of your content items:

The **Universal** flavor will offer good quality, and work with both Kinoma FreePlay (Kinoma's free player) and Kinoma Play

The **Best Quality** flavor will offer the best-possibly quality, and be used automatically when Kinoma Play is used to play your content.

If you're already encoding and distributing content on the web, it's likely that you're already encoding your content in the formats specified in this whitepaper.

The next three sections describe how to prepare your **audio**, **video** and **pictures**.

# Encoding your audio

---

## Universal

For the **Universal** flavor of your audio content, use **MP3 files**.

Encode your stereo content at **64 kbps, 44.1 kHz**.

CBR (constant bitrate) encoding will work great, although you can use VBR (variable bitrate) encoding if you prefer. At low bitrates like 64 kbps, “joint stereo” encoding must be enabled for highest quality.

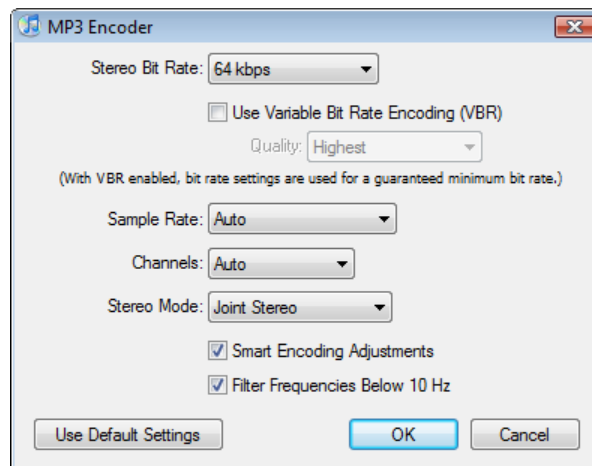


Figure 1 — iTunes settings for 64 kbps encoding

## Metadata

Make sure your **Title** and **Author/Artist** metadata is correct, and appears how you’ll want it to appear during playback.

The metadata should be as complete as possible, but short enough to fit comfortably on smaller mobile screens.

If you want to control what gets displayed during audio playback, include album art. You can add album art to your MP3 files using iTunes or similar music management applications.



Figure 2 — Adding artwork metadata using iTunes

**Note:** Because of the design of the MP3 file format, album art will be downloaded before the audio can begin playing. Keep the album art size small —16–32 KB is ideal, while 100 KB is too large — to avoid unnecessary delays at start-up.

## Best Quality

For the **Best Quality** flavor of your audio content, use **AAC (.m4a) files**.

Encode your stereo content at **48-64 kbps, 44.1 kHz**. Use 48 kbps to match the quality of your Universal flavor at a lower bitrate. Use 64 kbps if you want noticeably better quality than the Universal flavor at the same bitrate.

CBR (constant bitrate) encoding will work great, although you can use VBR (variable bitrate) encoding if you prefer.

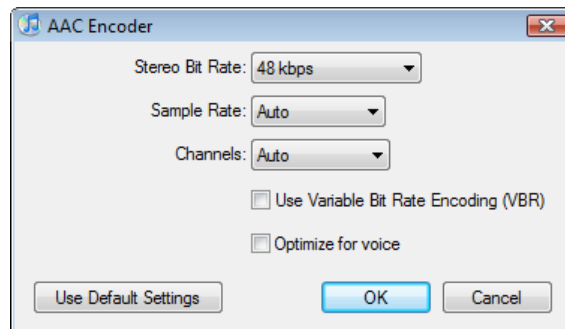


Figure 3 — iTunes settings for 48 kbps AAC (MPEG-4) encoding

## Metadata

Make sure your **Title** and **Author/Artist** metadata is correct, and appears how you'll want it to appear during playback.

The metadata should be as complete as possible, but short enough to fit comfortably on smaller mobile screens.

If you want to control what gets displayed during audio playback, include album art. You can add album art to your AAC files using iTunes or similar music management applications.

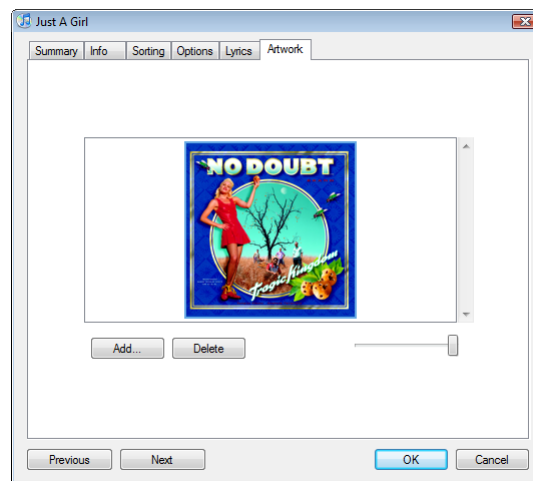


Figure 4 — Adding artwork metadata using iTunes

**Note:** Because of the design of the AAC (MPEG-4) file format, album art will be downloaded before the audio can begin playing. Keep the album art size small —16–32 KB is ideal, while 100 KB too large — to avoid unnecessary delays at start-up.



# Encoding video

---

## Universal

For the **Universal** flavor of your video content, create **FLV** files with **Sorenson Spark** (Flash 7) video and **MP3** audio.

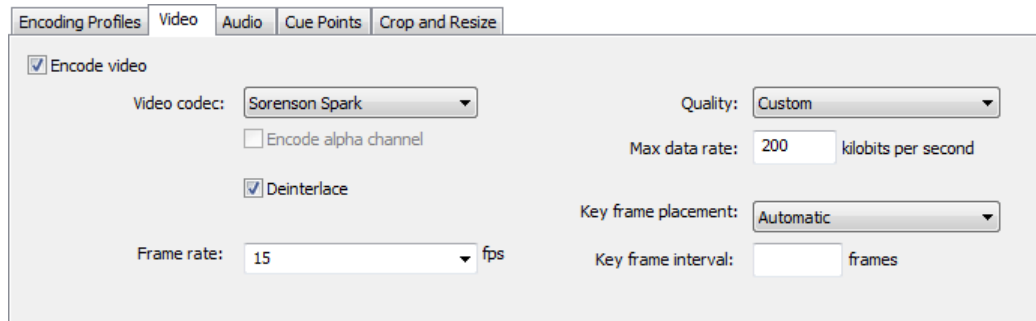


Figure 5 — Video encoding settings in Adobe Flash CS3 Video Encoder

We recommend that you keep the video bitrate to under 300 kbps if possible. You can use higher bitrates, but playback will not be as smooth under challenging network conditions or on older phones.

Generally, keep your frame rate at or under **15 fps**. To avoid jerky video, always set your frame rate to a sub-multiple of your source frame rate — for example, 25 fps PAL video will actually play smoother at 12.5 fps (which is the same as saying “every other frame”) than 15 fps.

## Size

We recommend that you encode your content at a size that will fit within a 320x240 box. For example, standard non-widescreen 4:3 video should be encoded at 320x240.

For other aspect ratios, round the video height to the nearest 8 pixels. For example, 16:9 video should be encoded at 320x184.

$$320 \times ( 9 / 16 ) = 180$$

$$180 \text{ rounded to nearest } 8 = 184$$

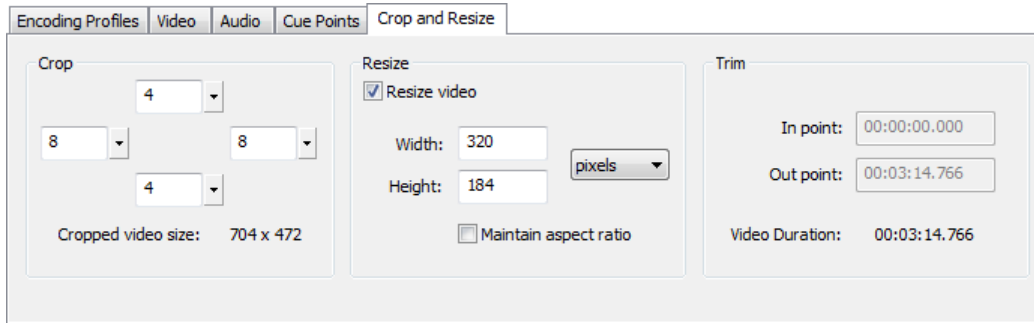


Figure 6 — Video size settings in Adobe Flash CS3 Video Encoder

## Audio

For the video stream's audio, you'll use **MP3**.

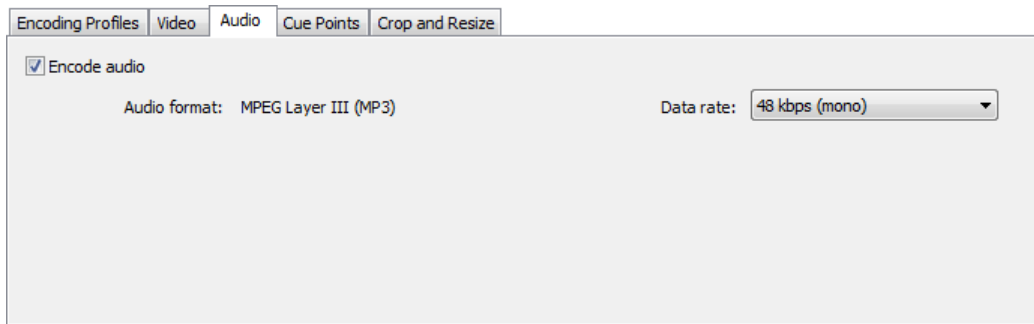


Figure 7 — Audio encoding settings in Adobe Flash CS3 Video Encoder

Encode in mono at **48 kbps**, or at **64 kbps** if stereo is required. You can encode the audio at a higher bitrate, the tradeoff being that playback will not be as smooth under adverse network conditions.

## Best Quality

For the **Best Quality** flavor of your video content, create **MPEG-4** files with **AVC (H.264)** video and **AAC** audio.

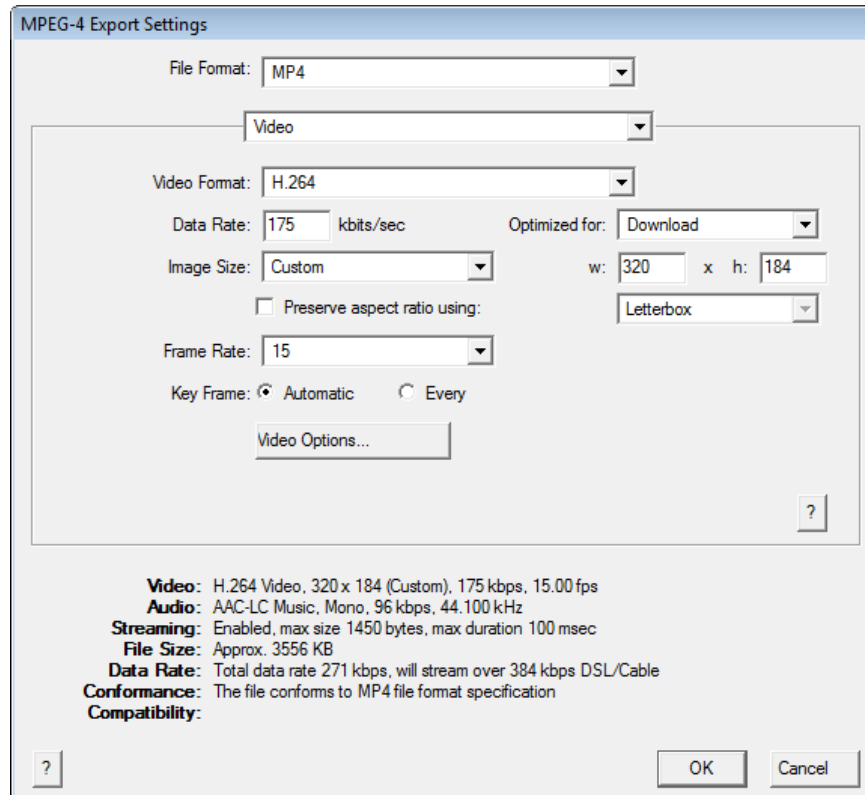


Figure 8 — Video encoding settings in QuickTime Player export

Keep the video bitrate to **175 kbps** if possible. You can encode the video at a higher bitrate, but playback will not be as smooth under challenging network conditions.

Generally, keep your frame rate at or under **15 fps**. To avoid jerky video, always set your frame rate to a sub-multiple of your source frame rate — for example, 25 fps PAL video will actually play smoother at 12.5 fps (which is the same as saying “every other frame”) than 15 fps.

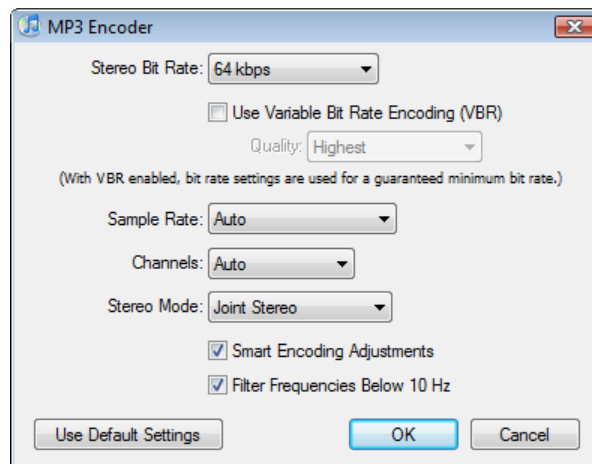


Figure 9 — iTunes settings for 64 kbps encoding

## Size

We recommend that you encode your content at a size that will fit within a 320x240 box. For example, standard non-widescreen 4:3 video should be encoded at 320x240.

For other aspect ratios, round the video height to the nearest 8 pixels. For example, 16:9 video should be encoded at 320x184.

$$320 \times (9 / 16) = 180$$

$$180 \text{ rounded to nearest } 8 = 184$$

## Audio

For the video stream's audio, use AAC.

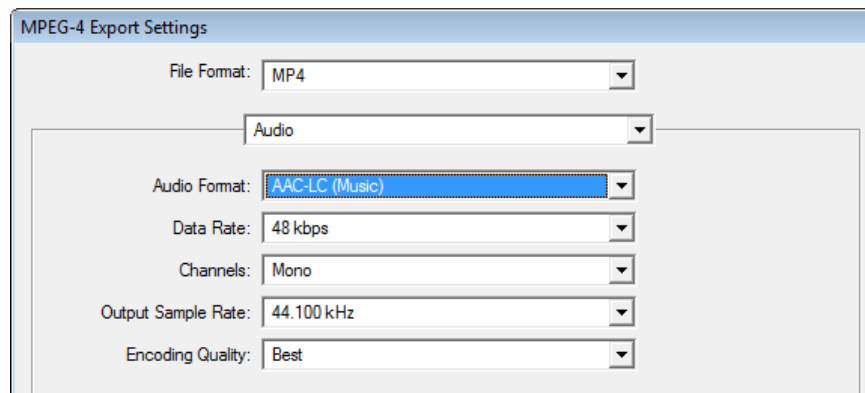


Figure 10 — Audio encoding settings in QuickTime Player export

Encode in mono at **48 kbps** if possible, or at **64 kbps** if stereo is required. You can encode the audio at a higher bitrate, the tradeoff being that playback will not be as smooth under adverse network conditions.



# Encoding photos

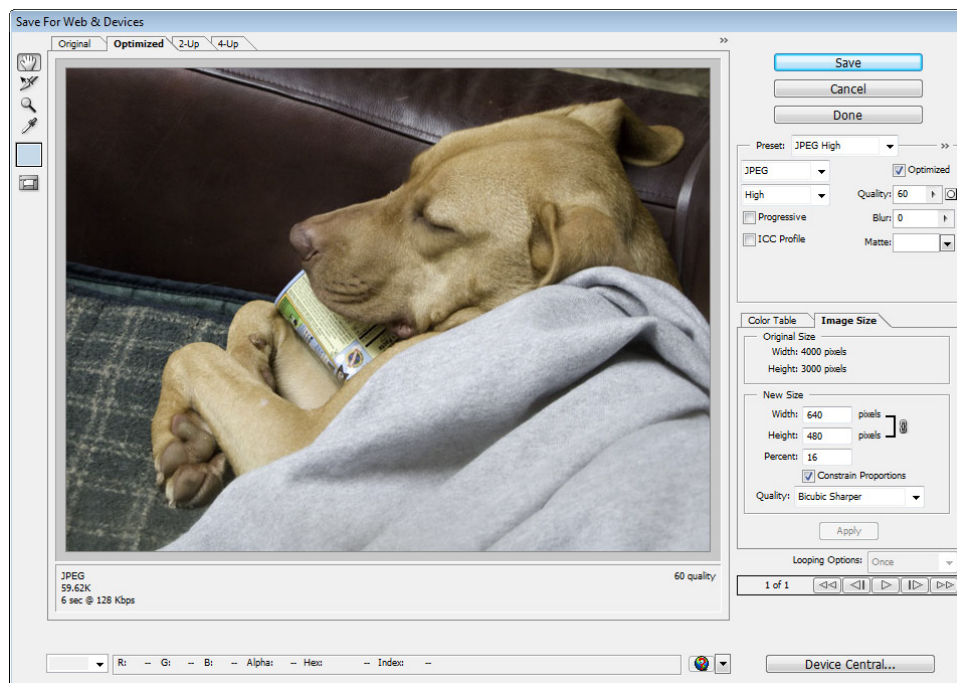
## Universal

For photos, the Universal and Best Quality flavors are the same — just create standard **JPEG** files.

## Target size

For the best experience, scale your photos so that they fit within a 640x640 box. For example, a standard 3504x2336 (8 megapixel) should be scaled to 640x427.

For most scenarios, applications' **High** or **Medium** compression settings should be perfectly acceptable. For photos smaller than 640x640, file sizes of around 50 KB will look great.



# Create your content feed

---

**RSS** is the standard you're already using to publish blog and podcast content.

You'll use the same RSS standard to create your own content channel for Kinoma Play.

## A feed with multiple programs

Here's an example RSS feed containing three programs. Additional programs would simply be represented as additional `<item>` elements.

```
<?xml version="1.0"?>
<rss version="2.0">
<channel>
  <title>My Podcast</title>
  <link>http://www.mydomain.com/</link>
  <description>My latest crazy-cool podcasts</description>
  <item>
    <title>My Podcast - 2007-07-01</title>
    <enclosure url="http://domain.com/v/ep1.flv" length="41540" type="video/x-flv" />
    <guid>http://mydomain.com/v/ep1.wmv</guid>
  </item>
  <item>
    <title>My Podcast - 2007-07-08</title>
    <enclosure url="http://domain.com/v/ep2.flv " length="27639" type=" video/x-flv" />
    <guid>http://mydomain.com/v/ep2.wmv</guid>
  </item>
  <item>
    <title>My Podcast - 2007-07-15</title>
    <enclosure url="http://domain.com/v/ep3.flv " length="31086" type=" video/x-flv" />
    <guid> http://mydomain.com/v/ep3.wmv</guid>
  </item>
</channel>
</rss>
```

To create a hierarchical list of programs — for example, a list of programs sorted into folders by program name, month, etc. — can you use another standard format called OPML.

You can find complete details on creating and testing RSS and OPML feeds for Kinoma Play in our *Optimizing Feeds* technote at [http://kinoma.com/Tech\\_Note\\_5](http://kinoma.com/Tech_Note_5).

## A feed with both Universal and Best Quality content

Here's an example RSS feed containing one program in both **Universal** and **Best Quality** flavors.

To create a feed that supports two flavors of content for each program, use the standard **Media RSS** extensions to RSS. You must declare that the feed uses Media RSS by including the Media RSS **namespace declaration** to your feed's `<rss>` element, like this:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
```

Note that the `<item>` is a group containing the Universal and Best Quality `<media:content>` elements.

As usual, additional programs would simply be represented as additional `<item>` elements.

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
<channel>
  <title>My Podcast</title>
  <link>http://www.mydomain.com/</link>
  <description>My latest crazy-cool podcasts</description>
  <item>
    <title>My Podcast - 2007-07-01</title>
    <media:group>
      <media:content url="http://www.foo.com/episode-1.flv"
        fileSize="10000" bitrate="348" type="video/x-flv"
        isDefault="true" />
      <media:content url="http://www.foo.com/episode-1.mp4"
        fileSize="6500" bitrate="225" type="video/mp4" />
    </media:group>
  </item>
</channel>
</rss>
```

## Add UPnP classes

For a better user experience, add UPnP class to your RSS feeds.

This example is the same as the previous, but adds (1) a UPnP namespace declaration to the top of the feed, and (2) a UPnP class description that describes the item as a video.

```

<?xml version="1.0"?>
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss" xmlns:upnp="urn:schemas-upnp-
org:metadata-1-0/upnp/">
<channel>
  <title>My Podcast</title>
  <link>http://www.mydomain.com/</link>
  <description>My latest crazy-cool podcasts</description>
  <item>
    <title>My Podcast - 2007-07-01</title>
    <media:group>
      <media:content url="http://www.foo.com/episode-1.flv"
        fileSize="10000" bitrate="348" type="video/x-flv"
        isDefault="true" />
      <media:content url="http://www.foo.com/episode-1.mp4"
        fileSize="6500" bitrate="225" type="video/mp4" />
    </media:group>
    <upnp:class>object.item.videoItem</upnp:class>
  </item>
</channel>
</rss>

```

Here are the UPnP classes to use for various types of content:

UPnP class	Use For
<b>object.item.videoItem</b>	Video files
<b>object.item.videoItem.videoBroadcast</b>	Live video streams
<b>object.item.videoItem.webcam</b>	Webcams
<b>object.item.audioItem</b>	Audio files
<b>object.item.audioItem.audioBook</b>	Audiobook
<b>object.item.audioItem.audioBroadcast</b>	Live audio streams (radio, etc.)
<b>object.container.internetFolder</b>	Folders
<b>object.container.playlistContainer.feed</b>	RSS feeds

## Add descriptions

For a better user experience, add **descriptions** to your programs.

To add a description, use the `<itunes:summary>` RSS extension as shown here:

```
<itunes:summary>Description of this program</itunes:summary>
```

Any text within the `<itunes:summary>` element will be used as the item's description both in Kinoma Play's detailed list view, and in any program's Details screen.

If you use `<itunes:summary>` extension, you must declare that the feed uses the iTunes namespace as shown here:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
```

Note that if you're using both Media RSS extensions and iTunes extensions, you must to declare both as shown here:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss"
xmlns:media="http://search.yahoo.com/mrss">
```

## Add thumbnails

For a better user experience, create **thumbnails** for your programs.

To add thumbnail images to your programs, use the `<media:thumbnail>` element.

The `<media:thumbnail>` element is standardized as part of Yahoo's **Media RSS** extensions to RSS.

Note that when you use Media RSS extensions, you must add the Media RSS **namespace declaration** to your feed's `<rss>` element, like this:

```
<rss version="2.0" xmlns:media="http://search.yahoo.com/mrss">
```

`<media:thumbnail>` elements need three things:

- **url** — The URL to the thumbnail image
- **width** — The width (in pixels) of the thumbnail image
- **height** — The height (in pixels) of the thumbnail image

In order for the thumbnail images to fit within the media browser entries, the width must be 44 pixels or less, and the height must be 22 pixels or less. The following example specifies a 20 x 20 thumbnail image:

```
<media:thumbnail width="64" height="64" url="http://domain.com/thumbnails/mythumbnail.jpg" />
```

To improve performance, `<media:thumbnail>` elements can include Base64-encoded thumbnails directly. For more details, see [###].

# Advertising

---

To insert advertising in your content, simply point to an ASX playlist URL. The URL can just as easily point to a script that dynamically creates the ASX playlist.

This example contains a non-skippable entry for a pre-roll advertisement, followed by a skippable entry for the main video clip.

```
<asx version="3.0">
  <title>Ultimate Diamonds</title>
  <entry clientskip="no">
    <ref href="http://mysite.com/video/ads/ultimate-diamonds.flv" />
  </entry>
  <entry clientskip="yes">
    <ref href=" http://mysite.com/video/main-content.flv " />
  </entry>
</asx>
```

We recommend ads that are duration-appropriate for the medium — a 30 second advertisement is great for TV, but fairly excruciating for mobile. A 10-second ad works well, but a 5-second ad may be an even better choice (especially if the content has its own title sequence).

# Search

---

When you have a lot of content, providing search functionality is a must. Here's how to do it:

## Add a Search item to your guide

In your guide OPML or RSS, add an item called **“Search...”** with a URL that points to your search script. The URL will use the URL scheme **x-search**, and have an attribute called **where** set to the “double-encoded” URL to your script.

```
<outline text="Search..." kinoma:description="Find everything my content service" ←  
type="link" url="x-search://ask?where=http%3A%2F%2Fsearch.mysite.com%2Fsearch" ←  
upnp:class="object.container.kinoma.search" />
```

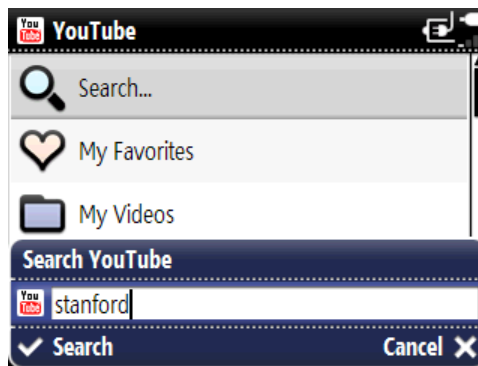


Figure 11 — Kinoma search in action

## “Double-encoded”?

Double-encoding is simply taking an already-valid URL, and then URL encoding it again so it can be used as a value in another URL.

If you're generating your guide data with scripts, most scripting languages can do this for you. For example, if you're using PHP, use `urlencode()`. For ASP.NET, use `Server.URLEncode()`.

Otherwise, you can use a site like <http://www.albionresearch.com/misc/urlencode.php> to double-encode your URL. Just paste the URL into the “Plain” field and click **URLEncode**.

## An example

If your search URL is this:

```
http://search.mysite.com/search.php
```

Then the double-encoded version is this:

```
http%3A%2F%2Fsearch.mysite.com%2Fsearch.php
```

To get the final search URL for the feed, just prepend “x-search://ask?where=”, like this:

```
x-search://ask?where=http%3A%2F%2Fsearch.mysite.com%2Fsearch.php
```

## Parsing Search requests

Search requests will be sent with the following key/value pairs:

- **what** — The search string entered by the user.
- **offset** — The index in the search results to start at (either 0 or 1 based counting) when there is “More...”. The value of **offset** will be either `[OFFSET_0]` or `[OFFSET_1]`.
- **max-results** — The maximum number of items to return.

For example, the URL might look like this:

```
http://search.mysite.com/search.php?what=foo&offset=[OFFSET_1]&max-results=20
```

Translated to English, this means “Return all items containing ‘foo’. Start with item 1, and return up to 20 items.”

The **offset** and **max-results** that your search script receives determines what “chunk” of the search results you should deliver. For example, if **offset** is 20 and **max-results** is 20, and the value of offset is `[OFFSET_1]`, then you should return results 21–41.

## Getting the values

The way that you get the HTTP values depends on your scripting language. If you’re using PHP, you can assign the values to PHP variables like this:

```
$searchWhat = $_GET["what"];
$searchOffset = $_GET["offset"];
$searchMaxResults = $_GET["max-results"];
```

For ASP.NET, this will look something like:

```
string searchWhat=Request["what"];
string searchOffset=Request["offset"];
string searchMaxResults=Request["max-results"];
```

## Returning Search results

The search results should be returned as RSS, using the same guidelines that exist for guide data.

## A bit “More...” detail

To make the “More...” item appear in your search results, the RSS feed with the search results needs to return the total number of search results. Kinoma Play uses [OpenSearch](#) RSS extensions for that.

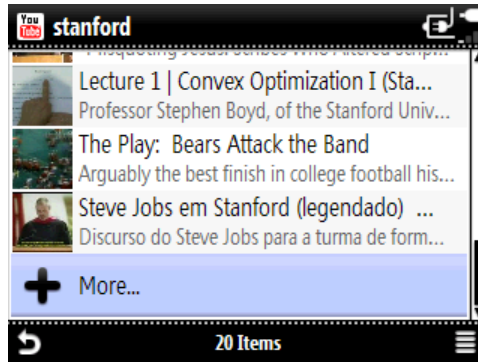


Figure 12 — Using OpenSearch RSS extension to support “More...” results

[http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft\\_3#The\\_.22totalResults.22\\_element](http://www.opensearch.org/Specifications/OpenSearch/1.1/Draft_3#The_.22totalResults.22_element)

For example:

```
<rss xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/" ...>
  <channel ...
    <opensearch:totalResults>123</opensearch:totalResults>
    ...
  </channel>
```

For consistency and completeness, we also recommend using the OpenSearch [startIndex](#) and [itemsPerPage](#) elements.

# Submit your feed to Kinoma Guide

---

To submit your feed for inclusion into Kinoma Guide, contact us at [content@kinoma.com](mailto:content@kinoma.com).

Please include its URL and a short description.

# Thank you!

---

If you have any questions or just want more information, we'd consider ourselves lucky to talk with you. Just let us know how and when we can reach you by sending us a note at [content@kinoma.com](mailto:content@kinoma.com).