Manual for jPSXdec: PlayStation 1 Media Decoder/Converter in Java

Version 0.99.7 (beta)

Michael Sabin jpsxdec@gmail.com

Table of Contents

1 About

jPSXdec is a cross-platform PlayStation 1 media reader, decoder, and converter written using the Java framework.

jPSXdec can read any file or image that comes off PlayStation discs. While you may have some success from ripping or copying individual files off the disc, it is recommended that the entire disc be ripped as a BIN/CUE file. Full disc ripping can be done using various tools on each platform.

2 Minimum system requirements

jPSXdec is almost universally cross-platform, running on Windows, Mac, and Linux, but requires Java to be installed to work. It should easily be able to run on a machine with a 1ghz processor and 512mb of RAM.

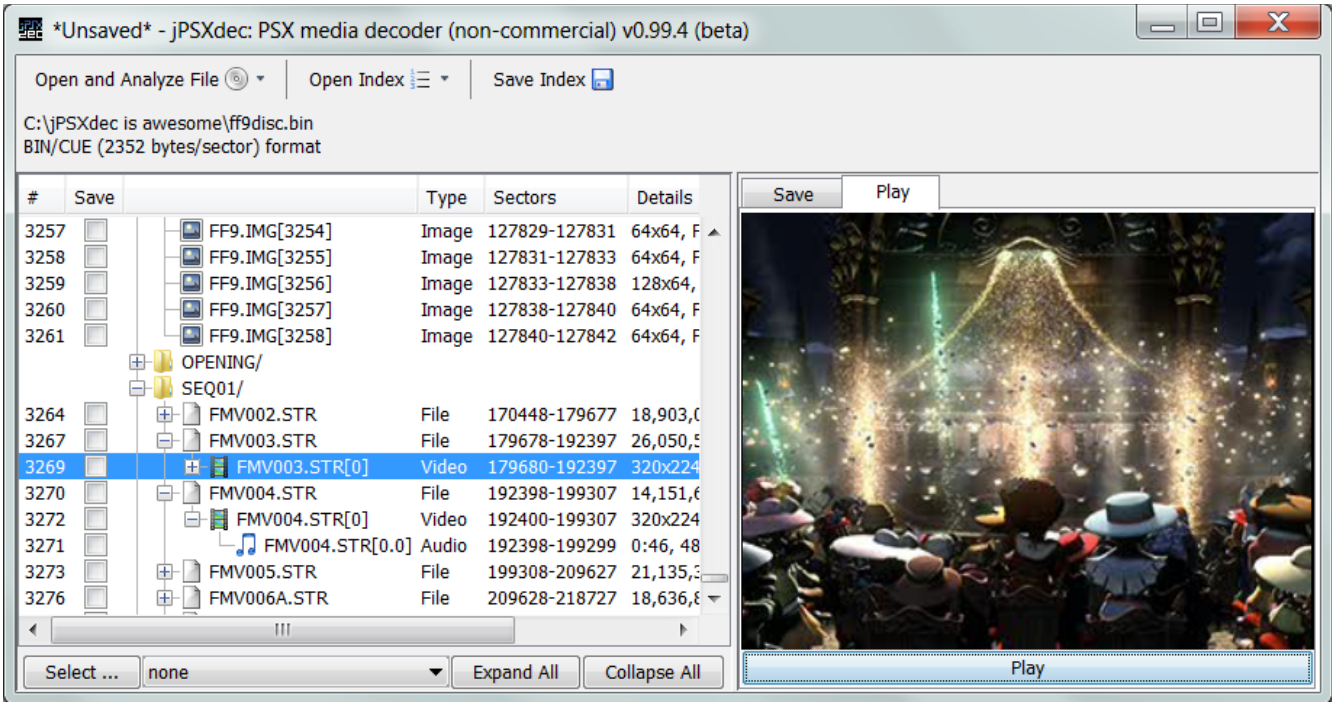
2.1 Java

There are several Java Virtual Machines and frameworks available. jPSXdec has only been tested on Oracle's Java. It requires Java 5 or higher.

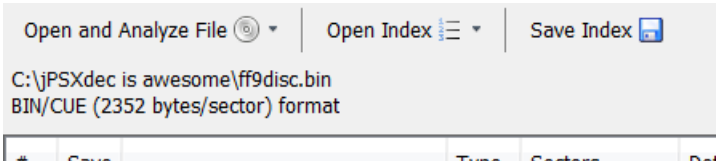
3 GUI (Graphical User Interface)

Starting jPSXdec without arguments will launch the graphical user interface.
On Windows, double-click `jpsxdec.exe`. On Mac and Linux, double-click `jpsxdec.jar`.

If there is only one argument supplied, jPSXdec will auto-detect if it is an index file or a disc image and immediately open it in the GUI.

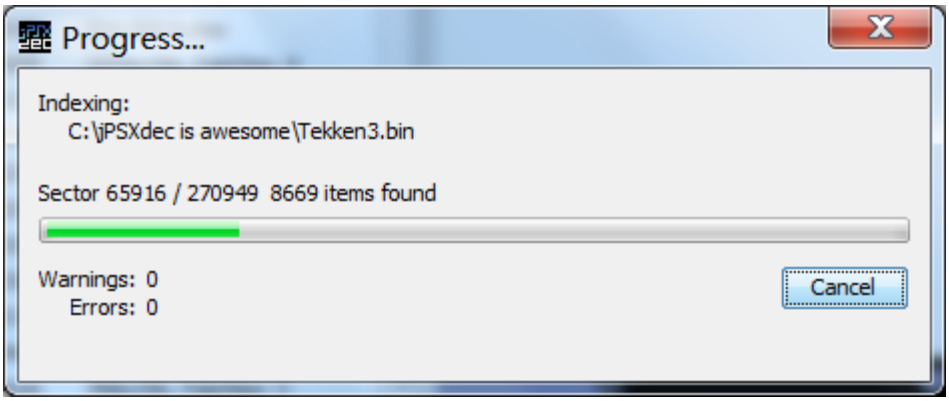


3.1 Toolbar



Open and Analyze File

A file dialog box will let you select a disc image or any other supported format (see chapter 7) to analyze. Or alternatively, a drop-down list with the most recent files opened. The analysis begins immediately.



Open Index

A file dialog box will let you select a previously saved disc index file. Or alternatively, a drop-down list with the most recent files opened.

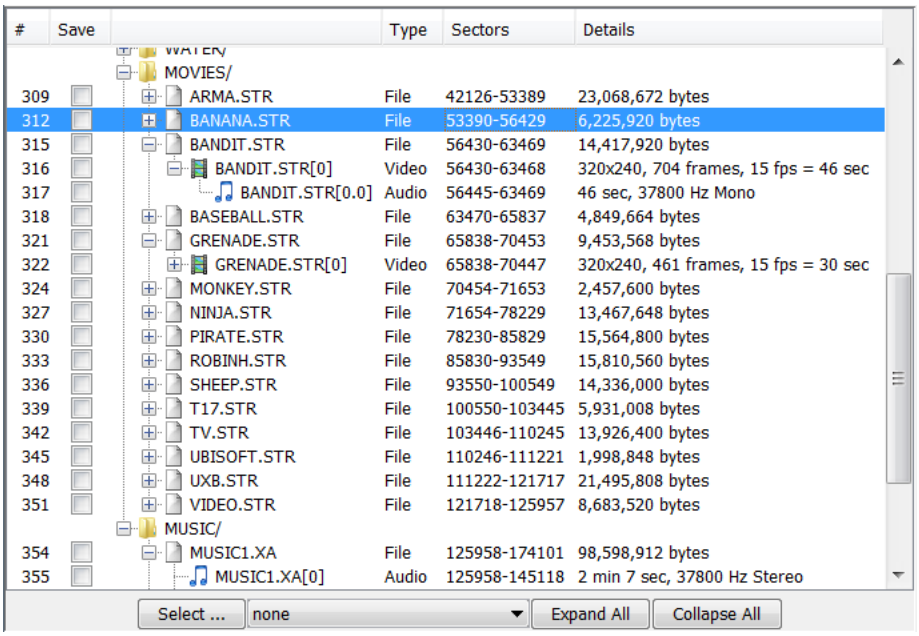
Save Index

After opening and analyzing a disc image, you can save the index it generates so you don't have to re-analyze the disc image.

Disc info

Under the toolbar buttons is a little information about the disc image: the file name, and the format of the disc.

3.2 List of items



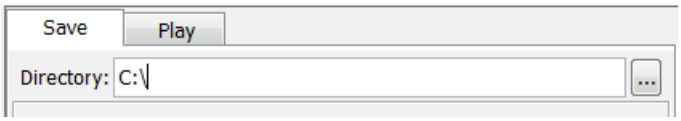
Columns

#:	Unique number associated with the item
Save:	Toggle this check box to include for saving
Type:	Type of item
Sectors:	The disc sectors that the item spans
Details:	More information about the item

Buttons

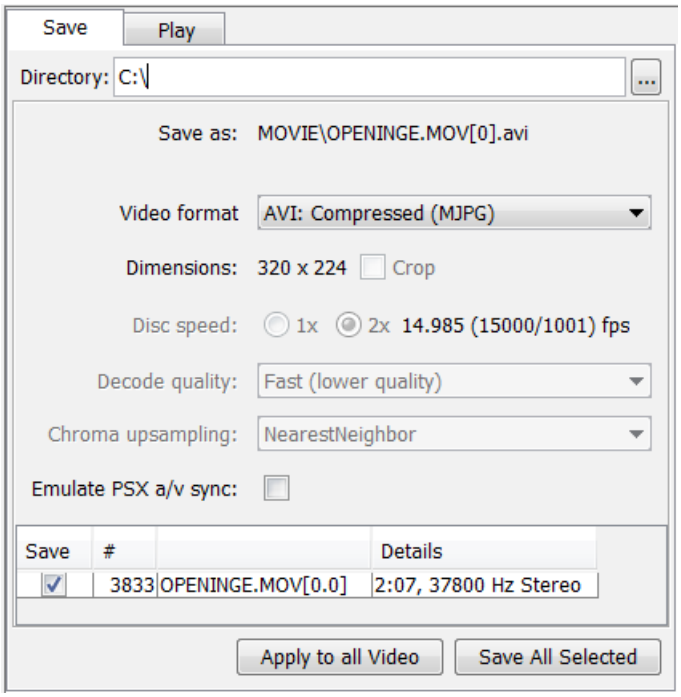
Select...:	Pick an item type in the drop-down menu, then press this button to select (check) all the items of that type.
Expand All/Collapse All:	Full expand or collapse all tree nodes.

3.3 Save panel



At the top of the "Save" panel is the "Directory" where files will be saved to.

Video

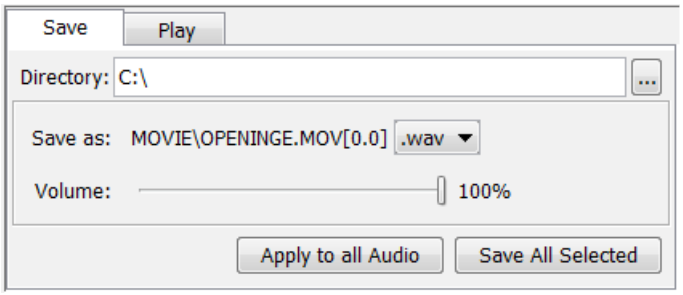


Save as:	The file name(s) that will be created.
Video format:	Format to save the video. See chapter 5 for details about these formats.
Dimensions, Crop:	The dimensions of the video that will be saved. Some videos are cropped by default, but the full dimensions can be saved.
Disc speed, fps:	The frames-per-second of the video. Some videos have an ambiguous frame rate, so the Disc speed may be available to be set.
Decode quality:	The quality of the video decoder.
Chroma upsampling:	In order to decode the image, the color information has to be scaled. This option sets the interpolation method. Only for highest quality decoding.
Emulate PSX a/v sync:	The PlayStation hardware doesn't start the audio and video at the exact same time. Enabling this will add some delay to the audio or video so they are in sync like how they would be on the PlayStation.

List of audio streams (if available):

Movies with audio almost always have only one stream. The only time I've seen a movie with multiple audio streams was due to disc corruption.

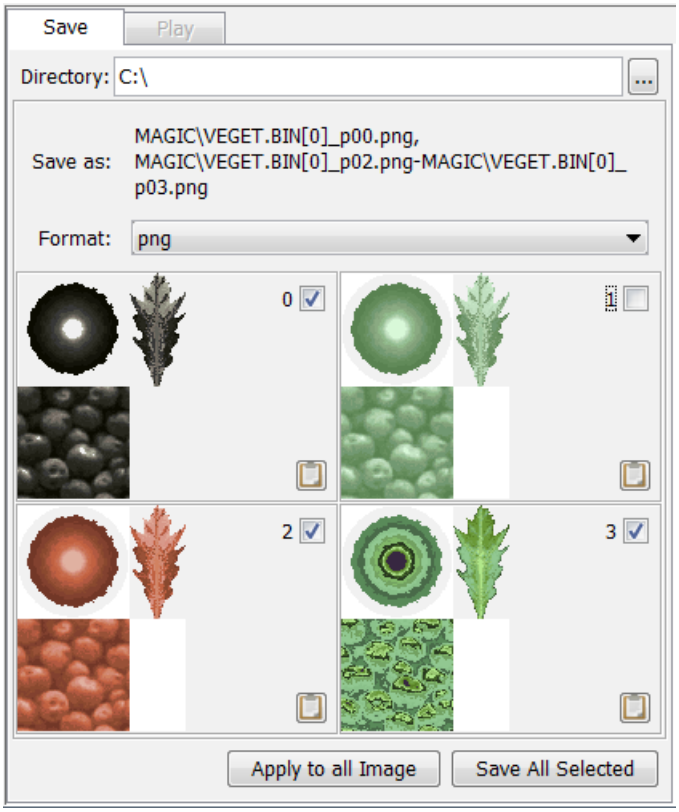
Audio



Save as: Shows the output file name and lets you select the output format.

Volume: This volume adjustment occurs before the final step of the audio decoding, which may improve quality if the audio was being cropped.

Images (TIM)



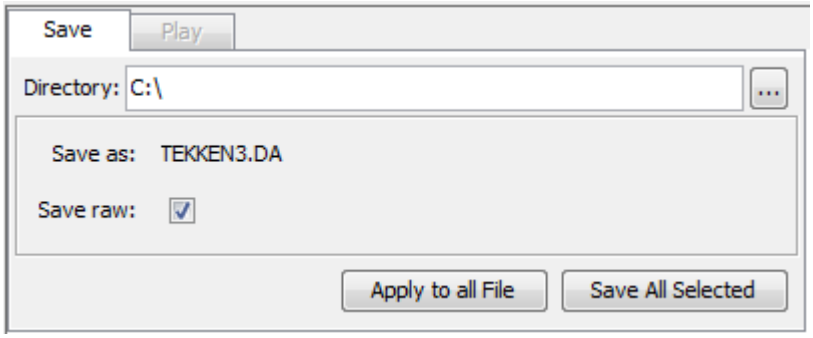
Save as: Summary of files that will be generated.

Format: The format to save the image(s) as.

(palettes): TIM images can contain multiple palettes. You can check which palettes you want to save. A separate image will be saved for each palette.

Click the clipboard button to copy that palette image to the clipboard.

Files



Save as: Shows the output file name.

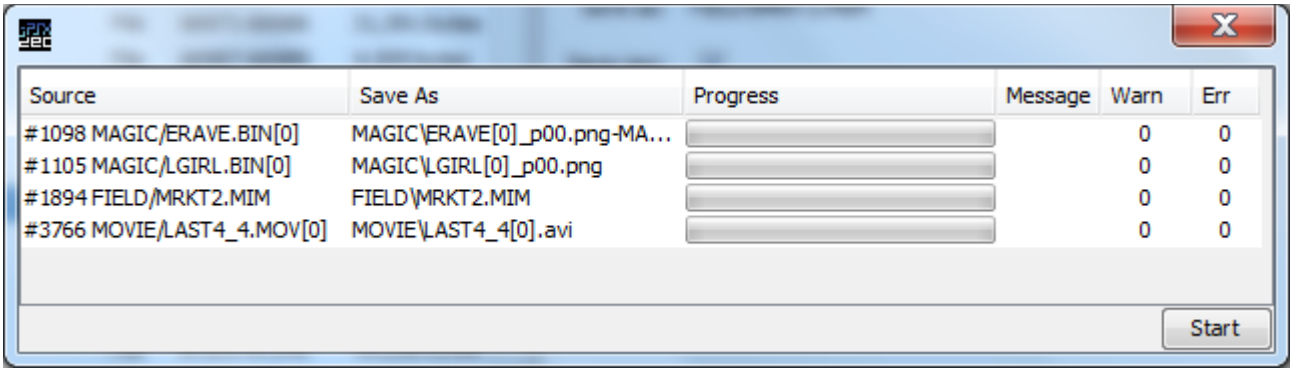
Save raw: If the source disc image contains raw sector data, this option lets you save all the raw data, or just the normal sector data. This option may not be available for some disc image formats. Note that some important data can be lost if Save raw is not checked.

Apply to all <type>

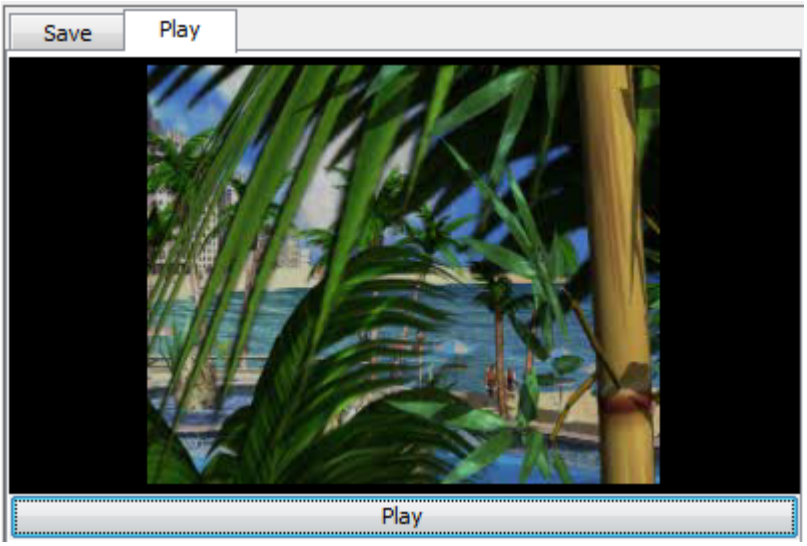
Pressing this button will apply most of the settings of the current type to all other items of that type. e.g. sets all videos to be saved with high quality options.

Save All Selected

This will show the dialog with all the items selected (checked) for saving. Pressing Start will begin the saving process.



3.4 Play panel



Lets you play video and audio straight off the disc image. There's nothing displayed when playing audio.

3.5 Settings

The GUI saves the file jpsxdec.ini in the startup directory so it can maintain the history of previously opened files and folders.

4 Command-line

4.1 Quick start

First, the source file (whether it be a disc image or a single media file) needs to be analyzed and an index generated.

Example to generate an index:

```
java -jar jpsxdec.jar -f MyGameImage.bin -x MyGameImage.idx
```

Once the index is created, open it to review what was found in the source file.

Identify the id-number of an item you are interested in, and run the command to extract/decode it.

Example to extract the first (#0) item with default options:

```
java -jar jpsxdec.jar -x MyGameImage.idx -i 0
```

4.2 Main command-line options

`java -jar jpsxdec.jar -help/-h/-?`
Displays main help.

`java -jar jpsxdec.jar -f <in_file> -x <index_file>`
Build an index of `<in_file>` and save it as `<index_file>`

`java -jar jpsxdec.jar [-f <in_file>] [-x <index_file>] <index_command> [index_command_options]`
Perform a command that requires an index. Either the `<in_file>` or `<index_file>` need to be specified, or both.

If only the `<in_file>` is specified, then it is indexed, but the index is not saved. This is useful for small .str files that only contain one or two streams.

If only the `<index_file>` is specified, then jPSXdec uses the source file stored in the index as the `<in_file>`.

If both the `<in_file>` and `<index_file>` are specified, then jPSXdec uses the index, but uses the `<in_file>` specified on the command-line instead of the file defined in the index.

The available `<index_command>`:

> `-item/-i # [item_command] [item_command_options]`
> Process one disc item (see item's help for options)
>
> `-a <type> [item_command] [item_command_options]`
> Process all disc items of type `audio`, `video`, or `file` (see item's help for options).
>
> Some shared `[item_command]`:
>
> > If no command is given, then it performs the default decoding command
> >
> > `-play`
> > Opens a very simple player to play movies and audio
> >
> > `-?/-h/-help`
> > Displays detailed help about the item
>
> `-visualize <output_pdf>`
> Creates a massive pdf representing the physical layout the disc with color-coded sectors and index items

`java -jar jpsxdec.jar -f <in_file> <in_file_command> [in_file_command_options]`

Perform an operation on `<in_file>`

`-copysect < # | #-# > <out_file>`
Copy sectors to `<out_file>`

`-sectordump <out_file>`
Write list of sector types to `<out_file>` (for debugging)

`-static <type>`
Process special static file types

`tim`
Converts a tim image to .png, one image per palette.

`bs` or `mdec`
Converts a single bitstream or mdec frame to an image.

`-dim` *width*`x`*hight*
Specifies the dimensions of the frame. Required.

`-quality/-q <low, high, psx>`
Decoding quality (default `high`). See `-quality` option in the video section for more details.

`-fmt <png, bmp, jpg, mdec>`
Output format. `mdec` only available for bitstream sources. Default `png`

`-up <upsampling>`
Chroma upsampling interpolation method for `high` quality. Default `Bicubic`. See video section for more details.

`-debug`
Prints the full conversion details (very low level and technical). Requires the java `-ea` (enable assertion) flag.

Additional option (optional):
`-verbose/-v #`
How much info to print: 0 for none, 1 for only errors, 2 for only errors & warnings, 3 for normal (default), 4 for extra

4.3 Options for disc item types

jPSXdec can detect the following data on PlayStation discs.

STR Video

Streaming PlayStation videos.
For the highest quality output, use these flags:

```
-quality high -vf avi:rgb -up Lanczos3
```

The PlayStation doesn't require videos to have a consistent frame rate. As such, some games have movies with variable frame rates, and most games have movies where the frame rate isn't precisely as detected by jPSXdec (but the variance isn't perceivable). When saving videos as an image sequence, frame rate is simply ignored. When saving as AVI, jPSXdec tries to place the frames in the AVI time-line as close to where they belong. This may mean that duplicate frames are inserted into the movie (thankfully AVI files have a convenient feature so duplication adds almost no extra size to files). Silent audio may also be inserted as needed in rare cases to keep the audio in sync. If you want to emulate the PSX timing more precisely, the `-psxav` option is available.

Command-line options

`-frameinfodump`
: Prints out detailed information about each video frame.

`-replaceframes <str_replace_xml>`
: Replaces frames of a video item according to the options in the xml file. See chapter 9 for details.

Options for decoding video items

`-dir <directory>`
: Specifies the output directory where item(s) will be saved. Default is the current directory.

`-quality/-q <low, high, psx>`
: Selects the decoding quality. Default is low. Option only applies to `avi:rgb`, `bmp`, and `png` formats.

  `low` quality is very fast and uses integer based arithmetic.

  `high` is slower, but should create better output because it uses floating point arithmetic (although it may not be noticeable without close inspection). It uses interpolation for chroma upsampling for a noticeable quality improvement. Use `-up` option to set interpolation method.

  `psx` quality tries to duplicate what the PlayStation would generate (about the same quality as `low`). It's not perfect, but is currently better than any other decoder ever written.

  This option is only applicable when saving as `png`, `bmp`, or `avi:rgb`.

`-vidfmt/-vf <format>`

Selects the output format (default is `avi:mjpg`). See chapter 5 for details about these formats.

`avi:mjpg` – Smallest output with only a small loss of quality.

`avi:rgb` – Completely uncompressed, RGB (device independent bitmap, i.e. DIB) AVI. Very large, but lossless.

`avi:yuv` – High quality output and smaller than `avi:rgb`. Uses the fourcc YV12 codec.

`avi:jyuv` – Most similar to PlayStation color spectrum. Uses the fourcc YV12 codec, but pixel values use the full [0-255] range (also known as "pc.601").

`png`, `jpg`, `bmp` – Saves a series of images.

`bs` – Demuxed compressed bitstream frame data found in separate sectors on the disc. 'Demuxing' is the first step in the decoding process, which simply combines the separate frame chunks into a contiguous block.

`mdec` – Generated from the demuxed bitstream data. This data would then be fed into the PlayStation MDEC chip, which would produce viewable images.

`-frame <#> / -frames <#-#>`

Lets you specify that only 1 frame, or a range of frames should be decoded. Audio is ignored when using this option because accurate audio decoded must begin from the start of the audio stream (you can't start in the middle). Frames can be defined using any combination of formats as described in chapter 6 (e.g. `#15.4-@668674.2`).

`-up <upsampling>`

Specifies the chroma upsampling interpolation method. Default is `Bicubic`. Ignored if not using `-q high`. Options (not case-sensitive): `NearestNeighbor`, `Bilinear`, `Bicubic`, `Bell`, `Mitchell`, `BSpline`, `Lanczos3`, `Hermite`.

`-noaud`

Don't decode audio with the video (ignored when video does not have audio, or when not saving as AVI).

`-psxav`

By default, when saving the audio and video as an AVI, they are set to start playing at precisely the same time. While that is probably how the video was intended to be seen, that is not exactly how it is played back on the PlayStation. The audio most likely starts where it first appears on the disc, which could be up to 0.15 seconds after the video starts. This option saves the audio and video to begin exactly when they appear on the disc (option ignored if not saving AVI with audio).

`-nocrop`

PlayStation 1 movies technically must have dimensions that are

multiples of 16. In cases where games need movies that do not have such dimensions, the extra space is filled with garbage and is cropped off before being drawn on screen. When this option is used, that extra data is not cropped. Option ignored if saving as bitstream (`bs`) or `mdec` format.

`-ds <1 or 2>`

"Disc speed" Movie frame rates are entirely dependent on the speed the disc spins. PlayStation 1 can spin discs at 1x speed, or 2x speed. Most games use 2x speed for all their movies. jPSXdec can usually automatically detect the disc during indexing, but if the speed is unclear, it will assume 2x. In the rare case that is incorrect, this option lets you force using 1x disc speed instead (effectively halving the frame-rate). Option ignored if not saving as AVI.

`-num <#, @, index, sector, header>`

When saving a sequence of images, indicates how to number the frame files. See chapter 6 for details about these options. `index` will use the frame index (starting as 0), `sector` (`@`) will use the start sector, `header` (`#`) will use the frame number found in sector headers. Default is `index`. Option is ignored when saving as AVI.

XA Audio & Square Audio

Standard XA ADPCM audio used by Sony PlayStation 1 and Phillips CD-i, and audio from Square (now SquareEnix) games (e.g. Final Fantasy).

Command-line options

`-replacexa <audio_file>`

(Only for XA audio) Converts an audio file into XA ADPCM format and overwrites the XA ADPCM audio stream. The audio file properties must match XA audio properties (channels, sample rate, etc.). Accepted audio formats are are `wav`, `aif`, `au`, and `snd`. This operation cannot be undone, so backup your disc before running it.

`-replacexa <other_index_file> -xa <item#>`

(Only for XA audio) Overwrites the XA ADPCM audio stream with XA data from another file. The XA streams must have the same format. This operation cannot be undone, so backup your disc before running it.

Example:

If you have `GAME.bin`, generate the index for it (e.g. `GAME.idx`). Identify the XA audio stream you want to overwrite in the index (e.g. #23).

Now if you have new XA audio in a file called `NEW.xa`, generate the index for it as well (e.g. `NEW.idx`). Identify the item number of the new audio

stream in `NEW.idx` (e.g. #1). Now you have what you need to replace the audio.

```
java -jar jpsxdec.jar -x GAME.idx -i 23 -replacexa NEW.idx -xa 1
```

Options for decoding audio items

`-dir <directory>`

Specifies the output directory where item(s) will be saved. Default is the current directory.

`-vol <range of 0 to 100>`

This does more than just adjust the volume, because it does the adjustment before clamping the waveform and rounding it to an integer volume. This can also help in cases where a game's audio has distortions due to the clamping. Adjusting the volume at this point should produce better quality than adjusting the volume later using another program. Default is 100.

`-audfmt/-af <format>`

Output audio format. Possible options are `wav`, `aif`, `au`, and `snd`. Default is `wav`.

File (ISO9660)

Normal files on disc images are detected and can be extracted and saved. Note that jPSXdec ignores the file size and just saves the full sectors.

Command-line options

Options for saving file items

`-dir <directory>`

Specifies the output directory where item(s) will be saved. Default is the current directory.

`-iso`

jPSXdec can detect 4 different styles of disc images (raw+subchannel 2448 bytes/sector, raw 2352 bytes/sector, special 2336 bytes/sector, and iso 2048 bytes/sector). By default, files from these disc images are copied out with the same style as the source disc image. With this `-iso` option, all raw sector headers are ignored and the file is saved as if it had been normally copied off a disc. Care must be taken that this option isn't used on files containing XA audio streams, as that raw header information is required for decoding them.

TIM image

TIM images regularly used in PlayStation games.

Command-line options

`-replacetim <image_file>`

Attempts to convert a normal image file (e.g. .png) to TIM and overwrite the TIM on the disc image. This operation cannot be undone, so backup your disc before running it. This operation will only succeed if:

- the image file is the same dimensions as the TIM image
- the TIM image has only 1 palette
- the input image can be successfully converted to the same bit depth as the TIM image it is replacing

Options for saving TIM items

`-dir <directory>`

Specifies the output directory where item(s) will be saved. Default is the current directory.

`-imgfmt/-if <image format>`

Format to save the TIM image. Use item's help for available options. These options will be different depending on if the TIM image has a palette. Default is `png`.

`-pal <palettes to save>`

TIM images may come with one or more palettes. By default, all palettes are saved as a separate output files. This option lets you select exactly which palettes to save using a comma delimited list, or hyphens (no spaces). For example, to save the first and third, fourth and fifth palettes, you could use this: `-pal 1,3-5`

5 Video output format quality

jPSXdec offers several formats to save videos. Each has pros and cons.

AVI with MJPG codec

AVI with JPG compressed frames. See JPEG for details about the pros/cons.

AVI uncompressed RGB (a.k.a. device independent bitmap, or DIB)

Very large, but highest quality, especially when using 'high' quality to improve chroma upsampling.

- Ensures the PlayStation IDCT is used
- Ensures the correct YCbCr → RGB conversion is used

The only possible down side to using this format is some tools might work better or have filters for YUV or MJPG formats (I'm only aware of 1 Avisynth plugin).

AVI YUV (a.k.a. Rec.601) – AVI with YV12 codec (technically it is YcbCr, not YUV)

About 50% smaller than RGB. Unfortunately there are 2 major issues with this format.

- Rec.601 only uses a portion of the available 255 range to store data: [16-235] for Luma (Y), and [16-240] for Chroma (CrCb). The PlayStation's YCbCr format uses the full [0-255] possible range of values. jPSXdec scales the values to fit within the subset, resulting in some quality loss.
- You're left to the mercy of other tools to convert to RGB. Unfortunately all other tools (except for Avisynth) will convert YCbCr → RGB with a ½ pixel "chroma shift." This will cause slightly incorrect colors in the output.

AVI "JYUV" (a.k.a. pc.601)

Like normal AVI YUV format above, except keeps the full [0-255] range of possible values. It's called "JYUV" because JPEG also uses the full range. Most similar to the original PlayStation video colors. Avisynth and ffmepg are the only tools I'm aware of that can handle this specific format. Again, 50% smaller than RGB, but all other tools (except Avisynth) will introduce a chroma shift when converting to RGB.

png, bmp

Highest quality. See AVI uncompressed RGB for details.

JPEG

The PlayStation video format is very similar to the JPEG format. It's so similar that it is possible to translate most of the data directly from one format to the other, without the need to fully decode and then re-encode. There are some caveats however.

- There is still a small amount of quality loss during the translation due do some rounding. However, every other video decoder I've seen also throws away that rounding data, so translating to JPEG should be at least as high quality as any other decoder.
- The PlayStation Inverse Discreet Cosine Transform (IDCT) won't be used by other JPEG decoders. No two IDCTs are identical, even among JPEG decoders. Ideally it's best to use the same IDCT to decode that was used to encode. jPSXdec is the only decoder that uses the PlayStation one. Since ffmpeg doesn't use the PlayStation IDCT for PSX video decoding, the quality here is at least as good as ffmpeg.
- The PlayStation unique YCbCr → RGB conversion won't be used by other JPEG decoders. The PlayStation conversion equation is very similar to the JPEG equation, but not exactly. Since ffmpeg uses the JPEG equation for PSX video decoding, the quality here is at least as good as ffmpeg.

Since this is mostly just a translation, the real decoding will be left to a JPEG decoder. Ideally the decoder you choose will do chroma upsampling and have deblocking and deringing filters.

demuxed bitstream

PlayStation video frames are broken up into multiple chunks. 'Demuxing' is the first step in the decoding process, which simply combines the separate chunks into a contiguous block. Most people have no use for this format.

mdec

The second step in the decoding process is uncompressing the demuxed bitstream. This data would then be fed into the PlayStation MDEC chip, which would produce viewable images. Most people have no use for this format.

6 Frame numbering and look-up

There are a few places in jPSXdec where it is necessary to identify a frame in a video.

- The video command-line `-frame`(s) option (input)
- The `-replaceframes` XML (input)
- When saving a sequence of images (output)

jPSXdec supports referring to frames in 3 different ways, each with different pros and cons.

	Syntax	Pro	Con

Index (i.e. first frame is 0, second frame is 1, etc).
This is the default.
	Just the index number, e.g.
	`0`
	`5`
	`273`
		Simple, always starts at 0.	May not be consistent between disc images if there are corrupted sectors.

Frame number found in the sector headers.	Frame number prefixed by #. May also have a decimal number indicating the duplicate frame number, e.g.
	`#1`
	`#7`
	`#26.3`
		Always consistent between disc images. Can relate sectors to their frame.	Some videos don't store a frame number in the header. Some videos have duplicate numbers in the headers. Some videos first frame is not 1.

Sector where the frame is found.	Sector number prefixed by @. May also have a decimal number indicating more than one frame starting at the sector, e.g.
	`@11235`
	`@403749`
	`@89553.2`
		Always consistent between disc images.	Really only useful for development.

It is very unlikely you would ever need to use anything but the default (index).

7 Disc and file formats recognized by jPSXdec

jPSXdec reads a handful of disc images formats, including ISO, BIN/CUE, and Alcohol 120%. MDF. Since it can read BIN/CUE disc images, it can also read .STR and .XA files because those files are simply extracted portions of BIN/CUE disc images. Using the `-static` command-line option, you can also read TIM images.

8 Index files

Index files contain important information about the contents of a disc image or media file. jPSXdec needs the index before it can perform any operations on a source file.

Index files may be edited by hand. If there's something wrong, jPSXdec will let you know.

The first line of index files contain a header indicating the jPSXdec version used to save the file. jPSXdec won't open index files saved with a different version.

Index files contain a path to the source file that the index was created from which jPSXdec will use to find the disc image (this can be overridden from the command-line).

All audio and video streams are listed independently, but jPSXdec will detect if an audio stream is associated with a video stream.

9 Replacing movies

Using the `-replaceframes` option on video items, jPSXdec can re-encode and replace video frames within streaming movies on the disc or movie file. Example command-line:

```
java -jar jpsxdec.jar -x <indexfile> -i <video#> -replaceframes <xmlfile>
```

9.1 xml

To use this feature, you must create a .xml file that tells jPSXdec what frames to replace, and how to replace them. The format is as follows:

```
<?xml version="1.0"?>
<str-replace version="0.2">

    <replace frame="10">newframe10.bmp</replace>

    <replace frame="#13" format="mdec">newframe13.mdec</replace>

    <partial-replace frame="@22354" tolerance="5"
                     mask="test.png" rect="20,15,200,150">
        newpartialframe.png
    </partial-replace>

</str-replace>
```

The root <str-replace> tag needs the version attribute to equal the current file format version (0.2).

Within the <str-replace> tag, only <replace> and <partial-replace> tags are allowed.

The <replace> tag specifies that a frame should be completely replaced. The frame attribute is required which specifies the frame number to replace. The frame number can be specified by the methods described in chapter 6.

The tag has one optional format attribute that can have the value of either "bs" or "mdec" which indicate that the replacement image isn't a normal image file but a 'bitstream' or 'mdec' file.

Within the <replace> tag is the file name of the image that should be encoded and inserted into the movie. Only .bmp and .png images are supported, unless the format attribute is used.

The <partial-replace> tag specifies that a frame should only be partially replaced. This is the perfect option for adding subtitles or other partial adjustments to a frame. The rest of the frame remains unchanged, so the quality loss due to re-encoding is minimized. This tag also has the required frame attribute. jPSXdec automatically compares the existing frame in the movie with the new image to detect what parts are different. It then has three optional attributes that let you specify extra pixels to ignore, even if they are different.

The tolerance attribute is similar to the 'tolerance' option for the wand selection or fill tools of advanced image editors. When detecting what pixels have changed, jPSXdec will ignore different pixels if they are only different by the tolerance amount (on each RGB channel). 0 means no tolerance: all differences are detected. 255 means all differences are ignored. The default value of this attribute is 0.

The mask attribute lets you use an image that will indicate exactly what pixels to check for differences. Any pixel that is solid black in the mask image will not be compared. Non-black pixels will be compared according to the tolerance option.

The rect attribute is similar to the mask option. It lets you specify a simple rectangle region to compare. Pixels outside the region are ignored and considered unchanged. The value of this attribute is four numbers separated by commas: x, y, width, height. Pixels within the rectangle are compared according to the tolerance option.

Within the <partial-replace> tag is the name of the image that will partially replace the frame. Only .bmp and .png files are supported. Note that if no differences are detected between the frames, nothing is changed in the original.

9.2 Encoding tips and tricks

Encoding video is tricky business. To get the absolute best quality, you probably have to be somewhat of a video format guru. For those that aren't, here is a bit of information that might help you.

Each video frame has a certain amount of space allocated to it. Most video frames don't use all of the space, but the more complicated frames get pretty close.

When encoding new video frames, jPSXdec tries to encode with the best quality that will still fit in the location of the existing frame. During the replacing process, you will see messages when an encoded frame fails to fit within the allotted space. In those cases, jPSXdec will reduce the image quality a little bit and try again. It will continue to do this until the new frame image fits.

In some cases the new frame won't fit even with the worst quality, or the resulting frame is so compressed that the quality is unacceptable. These low quality frames will contain many visual artifacts. So what can you do in those cases?

PlayStation video uses the same compression technique as JPEG. The main factors that affect JPEG size are **contrast** and **color**.

Contrast Hard lines, or extreme differences in light and dark or color has the biggest impact on compression size. To reduce these, you could try smoothing the hard edges with a blur tool. To reduce the color contrast, you could try dimming the bright colors or increasing the brightness of dark colors.

Color Color doesn't affect the size anywhere near as much as contrast, but still plays a part. Any shade of gray is smaller than any kind of color. If you must have color, keeping the colors to shades of pure red or pure blue will also save you some space.

When testing your newly encoded movie, you might find the PlayStation stuttering or even freezing. You could try reducing the image contrast/color even further. Or, if you are trying partial-replace, you could try full frame replace. Unfortunately, even if you do everything right, the game may still have problems with some changed frames. It's hard to say exactly what is wrong without debugging PSX assembly.

9.3 About encoder quality

The jPSXdec video encoder uses double-precision floating-point math. This already makes it at least as good as the AVI2STR/movconv tool found in the PSX Dev kit.

There are however some advanced encoding features I wish the jPSXdec encoder could use that would make it undeniably better than the PSX Dev kit tool.

- Smart chroma sub-sampling calculation
- Pre-anti-aliasing and pre-blockiness reduction
- Temporal spreading of quantization error
- Shorter VLC adjustment searching

If there are any insights you could provide about these very awesome enhancements, please let me know.

10 Reporting issues

Feel free to email me at jpsxdec@gmail.com, or report the issue on the project website. Ideally, please include the index (.idx) of the file you're having issues with, along with the generated debug0.log after reproducing the issue.

11 Building from source

jPSXdec can be built using the readily available Apache Ant build system. An Ant build script has been included. If Ant is installed and configured properly, simply running the `ant` command in the directory with build.xml will generate a directory containing the binary program and everything needed to distribute it.

12 License

13 Disclaimer