PSP2PSARC User's Guide

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Document serial number: 000004892117

Table of Contents

About This Document

PSP2PSARC is used to generate compressed archives from multiple input files. PSARC archives are similar to `.tar.Z` or `.zip` files. PSARC archives are used in conjunction with File IO Scheduler (FIOS) to transparently support access to compressed data.

This document provides an overview of PSP2PSARC and describes how to generate and manipulate PSARC archives.

1 Overview

PSARC is an extensible archive format that is designed to provide:

- A small, fast table of contents
- File contents compressed in blocks
- Random access to large compressed files
- Support for incremental decompression in constrained memory situations
- An extended file list interface for detailed control over each file archived

PSARC archives are compressed using ZLIB compression.

Using Compression

Compressing your data files can significantly improve game performance by decreasing load times. You can also use compressed data files to lower network bandwidth requirements and conserve space when reading game data or caching data in memory. Because using compression can also reduce iteration times during development, incorporating compression into your project early can even speed up the game development process.

Note: FIOS2 provides a dearchiver that enables transparent, real-time, access to compressed files. To compress files, you need to use the PSP2PSARC tool, FIOS2 does not provide an I/O filter for writing compressed files.

Performance gains and space savings depend on the type of data you are compressing. Highly redundant data compresses very well and using compression can have a dramatic impact. In an analysis of Insomniac Games's *Resistance: Fall of Man* (2006), compression ratios of geometry and lip-sync data ranged from 75% to a peak of 98.2%. In the most extreme case, using compression dropped the file load time from 2.8 seconds to less than 0.1 seconds. Most game data, such as general level data and textures, typically compresses at about 40–60%. In addition, by decompressing some data from a PSARC archive and reading other data from relatively slow I/O media allows FIOS2 to provide excellent overall I/O performance.

There is no benefit to compressing already-compressed data such as movies and MP3 files, but they can be stored in an uncompressed PSARC archive to maintain consistent data transfer procedures. (Audio data can be stored in compressed archives, but in general it's best to use a specialized audio compressor like MP3.)

Incorporating Compression into Your Project

When you implement a compression-based solution, it is important to be able to evaluate the resulting changes in performance. To do that, you need to create performance benchmarks and establish a pre-compression baseline against which you can compare.

When you incorporate compression into your project:

(1) Create a benchmark.
Establish a performance benchmark and baseline you can use to evaluate changes in performance. For example, determine how long it takes to load Level 2 of your game in its uncompressed form.

(2) Determine what to compress.
In particular, look for data that might be highly redundant, since compressing it can result in a big improvement in performance. Consider compressing any non-A/V data that you have to load, transfer across the network, or write to disk.

(3) Create the archive.
Use the PSP2PSARC command-line tool to generate the archive, as described in "Using the PSP2PSARC Tool".

(4) Check compression ratios on your data.
As a first step, verify that compression has significantly reduced the size of the data by comparing the uncompressed and compressed file sizes. (You can display the compression ratios for an archive with the PSP2PSARC tool's list command. For more information, see "Listing the Contents of an Archive".)

(5) Add a FIOS2 dearchiver I/O filter.
When you add a dearchiver, you will need to change your access paths to read from the archive instead of the uncompressed source. For more information, see the *libfios2 Overview*.

(6) Measure the benchmark again to verify that it is really faster.
Repeat the same test you performed on the uncompressed data to determine whether the compressed data does, in fact, load faster.

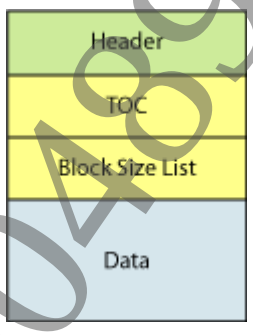
ZLIB Compression

PSARC archives are compressed using ZLIB compression. While archives compressed using ZLIB aren't as small as those compressed using algorithms such as LZMA, ZLIB offers fast compression and decompression and has modest memory requirements. You can configure the amount of memory used during compression and ZLIB only uses stack space and the output buffer during decompression.

PSARC Format

A PSARC archive consists of a Header, a table of contents (TOC) and a list that contains the sizes of each block in the archive, as well as the compressed blocks of data.

Figure 1 PSARC Format



Header

The Header identifies a file as a PSARC archive and specifies the compression method, information about the structure of the archive, and what options were set when the archive was created. It consists of eight unsigned long words, stored in network (big-endian) byte order (Table 1).

Table 1 Header Structure

Position	Content	Type/Value	Description
0	Magic Number	Unsigned long `0x50534152 'PSAR'`	Identifies the file as a PSARC archive.
1	Version number	Unsigned long `0xMMMMmmmm`	(M)ajor revision number (currently 0001) (m)inor revision number. Implemented to ensure that future versions maintain backward compatibility.
2	Compression method	Unsigned long	A 4-byte tag that identifies the compression scheme in use. For example: `0x7A6C6962 = 'zlib'`

Position	Content	Type/Value	Description
3	Total TOC size	Unsigned long	The size of the complete TOC, including the header. This is the amount of disk space needed to read in the entire directory structure.
4	TOC entry size	Unsigned long	The size of a single entry in the TOC, currently 30. This allows the size to be expanded in the future while maintaining backward compatibility with older PSARC file reading code.
5	`numFiles`	Unsigned long	The number of files in the archive, including the manifest (a plain-text file directory).
6	`blockSize`	Unsigned long	The size, in bytes, of each (compressed) block before compression and after decompression, which is the required size of a decompression buffer. Partial blocks are possible and are used to handle small files and the remaining bytes at the end of a large file.
7	`archiveFlags`	Unsigned long (bit field)	Information about the options that were specified when the archive was created. Current flags are: `IGNORECASE` – File names are upper-cased before the hash digest is created (default is case-sensitive). `ABSOLUTEPATHS` – Full path names are used to create the hash digest.

TOC

The table of contents (TOC) has one entry for each file in the archive plus an entry for the manifest file (Table 2). The manifest file contains a plain-text directory listing of the archive contents.

The TOC is meant to be cached and available when a PSARC file is open. File names in the table of contents are hashed, so are not human-readable.

Table 2 TOC Entry Structure

Bytes	Content	Description
0 to 15	`nameDigest`	128-bit md5 hash of the full path name of the file. A null hash (all zeroes) is used for the manifest file.
16 to 19	`blockListStart`	Index of the entry in the block size list for the first block of the file. This is used to get the compressed block sizes from the block size list.
20 to 24	`originalSize`	The original size of file, as a 40-bit integer. PSARC supports files up to (1 terabyte – 1) in size.
25 to 29	`startOffset`	The offset of the first compressed block of this file from the start of the PSARC header, as a 40-bit integer. This is used to access the file data.

Block Size List

The block size list is an *N*-byte array of compressed block sizes that directly follows the table of contents. It is used to locate each block of a file.

The block list is written in order from the first block of the first file through the last block of the last file. For every block, there is a block list entry that provides the size of the block after it is compressed. When you begin at `startOffset`, you can access and load each compressed block using the block size information in the block list.

The size of each entry in the block list depends on the maximum block size specified in the header. For block sizes up to and including 64KB, each block size entry is stored as two bytes. For block sizes up to and including 16MB, each block size entry is stored as three bytes, and so on. A block size value of 0

equates to the maximum block size and indicates that the block is not compressed at all and will be read as-is (raw) without being decompressed.

Note: If the `compressedBlockSize` equals the `uncompressedBlockSize` then it won't fit in *N* bytes. In this case, the `blockSize` is set to zero and interpreted as equal to `uncompressedBlockSize`, except when the actual file size is zero.

Data

The Data region of a PSARC archive contains the individual compressed blocks of data for each file in the archive. The first file is the Manifest file created by PSP2PSARC.

Manifest File

The Manifest file lists the names of all of the files that are in the archive. Each file name is separated by a `0x0A` line terminator. There is no terminator after the last file.

The file names in the Manifest are normalized. In the normalized file names:

- The forward slash (`/`) is used as a separator.
- The drive letter (`C:`) is removed.

2 Using the PSP2PSARC Tool

The PSP2PSARC command-line tool enables you to create compressed archives. You can also use it to verify an archive, list archive contents, and extract files from an archive.

Creating an Archive

When you create an archive, you can either specify the individual files or directories you want to add to the archive or use a text input file to specify a list of files.

Note: You can also use an XML specification file to identify the files you want to archive and control creation of the archive. For more information, see "Creating an Archive Using an XML File".

Specifying Individual Files and Directories

If you are archiving a small number of files or the contents of one or more directories, you can simply list them on the command line. When you specify the name of a directory, every file in that directory and its subdirectories are added recursively to the archive.

You can also specify an input file when you list individual files and directories on the command line. Files specified on the command line are added to the archive before files specified in the input file.

You can use the `-o` option to specify the name of the archive you are creating. If no output file name is specified, the name of the first specified file or directory is used with the extension `.psarc`. If an output file already exists, it is not overwritten unless you specify `--overwrite`. Instead, archive creation fails, nothing is written, and an error message displays. For example:

```
Archive 'foo.psarc' already exists. Use -y/--overwrite to force.
```

Note: When creating an archive file, if the file to be archived has the same name as the output file, and the `--overwrite` option is specified, the file to be archived is overwritten, and will be unrecoverable.

For example, to create an archive called `fbarchive.psarc` that contains the file `foo.txt` and all of the files in the `testfiles` directory:

```
% psp2psarc create -o fbarchive.psarc somewhere/foo.txt somewhere/testfiles
```

Note: The `-o` option is optional when you are listing the individual files and directories you want to add to the archive.

When you create an archive, the file compression ratios are displayed in the command output. Note that the compression ratio shown when you create an archive is calculated as `((original-final)/original)`, so larger percentages indicate a higher degree of compression. For uncompressed files, the compression ratio will be 0% (or a small negative value due to the size of the header information.)

Using a File List

If you are archiving a large number of files that reside in different locations, you can use a text file to specify a list of files to add to the archive. You use the `-input-file` option to specify the name of the input file. Note that you should explicitly set the name of the output file with the `--output` option. If you do not, PSARC will attempt to auto-generate a name based on that of the first input file. This attempt may fail in some cases, such as when the input file is already a PSARC file.

You can also list individual files and directories on the command line when you use an input file. Files specified on the command line are added to the archive before files specified in the input file.

The input file can only contain one file per line. You cannot list directories, only individual files and wildcards. For example:

```
somewhere/foo.txt
somewhere/bar.txt
```

For example, to create an archive called `flarchive.psarc` from the files listed in `filelist.txt`:

```
% psp2psarc create -o flarchive.psarc -I somewhere/filelist.txt
```

Changing Compression Settings

PSARC archives are compressed using ZLIB compression. You can use the `--level` option to control the compression quality. There's a trade off between the quality of compression and performance. Setting the level to 1 creates the archive in the fastest time possible, but produces a larger archive. Setting the level to the maximum level creates the smallest possible archive, but it takes longer to generate.

The maximum compression level for ZLIB archives is 9. The default compression level is the maximum level.

For example, to create a ZLIB archive of the files listed in `filelist.txt` in the fastest time possible:

```
% psp2psarc create --level 1 -o flarchive.psarc -I somewhere/filelist.txt
```

Stripping Paths from File Names

To strip part of the path name from a file's name in the PSARC table of contents, you can use the `--strip` option. You can specify multiple `--strip` options to selectively strip multiple paths. To strip all path names stored in the archive, use the `--strip-all` option.

For example, if you are adding the `X:/assets/images/trees` directory to the archive, you could use the `--strip` option to remove the path prefix from all of the files in that directory:

```
% psp2psarc create -o treearchive.psarc --strip=assets/images
    X:/assets/images/trees
```

The file `X:/assets/images/trees/elm.png` would then be archived as `trees/elm.png`.

If you stripped all paths from files in the archive:

```
% psp2psarc create -o treearchive.psarc --strip-all X:/assets/images/trees
```

The file `X:/assets/images/trees/elm.png` would be archived as `elm.png`.

It is also possible to strip from the middle of a path name:

```
% psp2psarc create -o treearchive.psarc --strip=images/
    X:/assets/images/trees
```

In this example, the file `X:/assets/images/trees/elm.png` would be archived as `assets/trees/elm.png`.

The strip string is a Perl regular expression. (Note that regular expressions often require escape strings to prevent them from being corrupted by the shell command line parser.)

```
% psp2psarc create -o treearchive.psarc --strip=[a-z]+es/
    X:/assets/images/trees
```

In this example, images/ matches the regular expression, so the file `X:/assets/images/trees/elm.png` would be archived as `assets/trees/elm.png`.

Verifying an Archive

You can check the structural integrity of an existing archive by specifying the `verify` command when you run the PSP2PSARC tool. This operation does not validate the data itself, but it traverses the archive and decompresses each file to RAM to make sure the archive is not corrupted and the data can be extracted.

To verify a PSARC archive, you specify the `verify` command and the name of the archive:

```
% psp2psarc verify fbarchive.psarc
```

The PSP2PSARC tool validates each file in the archive and prints "`Archive OK`" and returns 0 if the archive passes the validation checks. If the archive does not pass the validation checks, the `verify` command prints "`Archive Failed`" and returns an error code of 1.

Listing the Contents of an Archive

You can display the contents of an existing archive by specifying the `list` command when you run the PSP2PSARC tool. This operation returns the name of each file, followed by the compression ratio.

Note: The compression ratio is calculated as (final/original), so uncompressed files will have a compression ratio of 100%.

To list the contents of a PSARC archive, you specify the `list` command and the name of the archive:

```
% psp2psarc list fbarchive.psarc
```

Creating an Archive Using an XML File

You can use an XML file to control the archive creation process. In addition to specifying the files and directories that you want to archive, the XML file can be used to set options such as the compression type and path type (absolute or relative). This simplifies archive generation when you need to override the default options or exclude multiple files or groups of files.

Note: Options that control the execution of the PSP2PSARC tool, such as `--help` and `--verbose` are specified using command line arguments even when you're using an XML specification file to create your archive.

Creating a PSARC Specification File

A PSARC specification file consists of a `<psarc>` element that contains a list of command elements such as `<create>`, `<extract>`, `<append>`, or `<delete>`. For example:

```
<psarc>
  <create>
    <directory path="assets" />
  </create>
</psarc>
```

This specification file would load the files in the `assets` directory and output the archive to the output file specified on the command line.

Note: The `<create>` and `<extract>` commands are currently the only supported commands. Support for the modification of archives will be available once the `<append>`, `<delete>`, and `<resolve>` commands are available.

Controlling Archive Options

To control how the archive is generated, you specify options on the `<create>` element. For example, the following `<create>` command would generate an archive called `foo.psarc` that ignores case when matching file names:

```
<create archive="foo.psarc" ignorecase="true">
```

The key attributes you can specify for the <create> element are shown in Table 3. Use the psp2psarc dtd command to see the complete list of supported attributes. The <create> attributes correspond to the command line options described in "PSP2PSARC Command Line Tool Reference".

Table 3 Key Attributes for the <create> element

Attribute	Description
archive	Sets the output file name. This is equivalent to setting the -o option on the command line.
mergedups	Causes identical files to share the same data in the archive. For example if "level4/hero1.model" and "/level7/dude2.model" contain identical data then they will each have their own entry in the Table of Contents. But the entries will point to the same data blocks in the archive. This can reduce the size of the archive.
ignorecase	Ignores case when matching file names.
stripall	Strips the path of the current directory from the paths of the file names in the archive.
skipmissingfiles	Ignore it and continue when a file cannot be found. Set to true to ignore missing files.
jobs	Specifies the number of CPU threads to use when compressing. The default is the number of processors available.
blocksize	Sets the block size in bytes. This is equivalent to setting the --block-size option on the command line.
format	Specifies the type of archive to create.

Adding Files to the Archive

To specify the files you want to add to the archive, you add <file>, <filelist>, and <directory> tags to the <create> element.

Use <file> tags to add individual files or any file that matches a regular expression:

- Specify the name of a file with the path attribute:

```
<file path="data/stuff.txt" />
```

- Specify multiple files using a wildcard. For example to include all MP3 files:

```
<file wildcard="songs/*.mp3" />
```

- Specify a regular expression to match with the regex attribute. In the following regular expression, .* is used to match any character and the \ is used to escape the dot delimiter in .txt so that it is not treated as a special character.

```
<file regex=".*\.txt" />
```

- For file names that include characters that could be interpreted as part of a regular expression, add the escape delimiter \ to treat the characters as part of the file name. For example, to specify a file named "data/stuff(123)", use the \ escape character for each of the parenthesis characters:

```
<file path="data/stuff\(123\).txt" />
```

- Specify a different name to use in the archive with the archivepath attribute. (The name of the file in the table of contents is normally derived from the input path.)

```
<file path="X:/assets/changed.path"
    archivepath="/somewhere/else/changed.path" />
```

- Control whether or not files are compressed by setting the compressed attribute:

```
<file regex=".*\.png" compressed="false" />
```

Use a <filelist> tag to specify a text file that lists the files you want to archive. List one file per line in the text file. Wildcards are supported in this text file, but regular expressions are not. (This is the equivalent of specifying the --input-file option on the command line.)

- Specify the name of the input file with the path attribute:

```
<filelist path="myfiles.txt" />
```

Use a <directory> tag to add the entire contents of a directory. PSARC will recursively include subdirectories. If you do not want to include subdirectories, use the <file> tag with a wildcard instead.

- Specify the name of the directory with the path attribute:

```
<directory path="assets/level5" />
```

Excluding Files from the Archive

To prevent selected files from being added to the archive, you can add <exclude> tags to the <create> element.

- Specify the name of the file you want to exclude with the path attribute. For example, to prevent biguselessfile.raw from being included:

```
<exclude path="somewhere/biguselessfile.raw" />
```

- Specify the wildcard attribute to use a wildcard expression to exclude a group of files. The wildcard will only be matched with the file name and not the entire path. For example, to exclude all PSARC archives:

```
<exclude wildcard="*.psarc" />
```

- Specify the regex attribute to use a regular expression to specify the files to exclude. Use this attribute to exclude files based on the path. For example, to exclude any files in a "notes" directory:

```
<exclude regex=".*/notes/.*" />
```

Setting Compression Options

To set the level of compression, you add a <compression> tag to the <create> element and specify the level attribute to set the compression level. This is the equivalent of setting the --level option on the command line.

To explicitly enable or disable compression, you specify the enabled attribute. This can be used in conjunction with the compression attribute on the <file> element to enable compression only for specific files. For example, to only compress text files:

```
<compression enabled="false"/>
<file regex=".*/.txt" compressed="true" />
```

Stripping Paths

To strip part of the path name from a file's name in the PSARC table of contents, you add a <strip> tag to the <create> element. Note that this works after the path name has been converted to canonical form.

- Specify the regex attribute to set the part of the path you want to strip. For example, if you've added the file X:/assets/images/trees/elm.png to the archive, the following <strip> tag results in the file being named trees/elm.png when it's loaded into the archive:

```
<strip regex="/assets/images" />
```

Using a PSARC Specification File to Generate an Archive

To use your PSARC Specification File to create an archive, you use the --xml option to specify the name of your XML file:

```
% psp2psarc create --xml=somewhere/fbspec.xml
```

When using an XML specification file, it is an error to specify command line options that affect the output file. You can only specify command line options that control the operation of PSARC, such as `--verbose` or `--jobs`.

Extracting Files from a PSARC Archive

You can extract the contents of an existing archive by specifying the `extract` command when you run the PSP2PSARC tool. By default the files are extracted to the current directory.

To extract all of the files from a PSARC archive, you specify the `extract` command and the name of the archive file:

```
% psp2psarc extract fbarchive.psarc
```

Note: You can only extract files from one archive at a time. If multiple files are specified, extract only processes the first file and the rest are treated as archive paths.

You can also use the `--input` option to specify the archive file. For example:

```
% psp2psarc extract --input=fbarchive.psarc
```

Extracting Selected Files

You can also extract specific files from an archive by specifying the files you want to extract as additional arguments. For example:

```
% psp2psarc extract fbarchive.psarc archivedir/foo.txt archivedir/bar.txt
```

If you use the `--input` option, all of the file arguments are treated as archive paths. For example, the following command will extract both the `foo.txt` file and the `bar.txt` file from `fbarchive.psarc`:

```
% psp2psarc extract --input=fbarchive.psarc archivedir/foo.txt
archivedir/bar.txt
```

Note: If the `--input` option is not specified, `psarc` treats the first file argument as the archive file, and any subsequent file arguments as paths to the specific files you want to extract.

Specifying an Output Directory

By default, `psp2psarc` extracts files to the current directory. To specify a different output directory, use the `--to` option:

```
% psp2psarc extract --input=fbarchive.psarc --to=extracted
```

If the directory does not already exist in the filesystem, it will be created.

Extracting Files Using an XML File

You can use an XML file to control the archive extraction process.

Archive Extract Options

To control how files are extracted, you specify options on the `<extract>` element. For example, the following `<extract>` command would extract all of the files in the archive `foo.psarc` into the current directory:

```
<extract archive="foo.psarc">
```

The key attributes you can specify for the `<extract>` element are shown in Table 4. Use the `psarc dtd` command to see the complete list of supported attributes.

Table 4 Key Attributes for the <extract> element

Attribute	Description
`archive` or `input`	Sets the archive file name, this is equivalent to setting the `--input` option on the command line.
`to`	Directory to extract files into, this is equivalent to setting the `--to` option on the command line.
`stripall`	Strips the leading path from the paths of the file names in the archive.
`skipmissingfiles`	Ignore all missing files, and continue if any specified file cannot be found.
`overwrite`	Allow existing files to be overwritten.

Extracting Files from the Archive

Use the `<file>` tags to add individual files, and to specify the behavior for missing files (on a per-file basis). If no `<file>` element is specified, by default all files are extracted.

- Specify the name of a file with the `path` or `archivepath` attribute:

```
<file path="data/stuff.txt" />
```

- Specify the action to take if the file does not exist using the `skipifmissing` attribute, by default, `skipifmissing` is `false`:

```
<file path="data/stuff.txt" skipifmissing="true" />
```

PSP2PSARC Command Line Tool Reference

To invoke the tool, you specify:

```
psp2psarc verb [options] [files]
```

The following tables describe all of the options you can specify when invoking the PSP2PSARC tool from the command line. You can also use the help option to display information about each option:

```
% psp2psarc -h
```

Table 5 PSARC Actions

Verb	Description
`create`	Create an archive.
`extract`	Extract contents of an existing archive.
`list`	List contents of an existing archive.
`verify`	Verify an existing archive's structural integrity and ensure that all files can be decompressed.
`dtd`	Show the XML input file syntax (DTD).

Table 6 PSARC General Options

Option	Description
`-h` `--help`	Show the help information for the `psarc` command and exit.
`--version`	Display the PSP2PSARC tool version and exit.
`-d` `--debug`	Display debug messages.
`-v` `--verbose`	Display status messages.
`-q` `--quiet`	Don't show progress information.
`-y` `--overwrite`	Overwrite existing files when creating, or extracting from, an archive.

Table 7 PSARC Create Options

Option	Description
`-b BLOCKSIZE` `--block-size=BLOCKSIZE`	Specify the block size in bytes. The default is 65536 (64KB).
`-C` `--compress-manifest`	Compress the manifest, even if the no-compress (`-N`) option is used.
`--exclude=EXCLUDEFILE`	Exclude the specified file. You can use wildcard matching to exclude groups of files. For example, to exclude all archives specify `--exclude=*.psarc`. Use multiple exclude options to exclude multiple files or groups of files. Note that you must use an = and cannot put any spaces before the wildcard or it will be interpreted by the shell: `--exclude=*.png` is OK, but `--exclude *.png` won't work.
`-i` `--ignore-case`	Make the archive directory case-insensitive. By default, archives are case sensitive. If in doubt, do not set this option and leave the archive case sensitive.
`-I INPUTFILE` `--input-file=INPUTFILE`	Read the names of files to archive from the specified file. The input file must be a plain text file that lists one file path per line.
`-j JOBS` `--jobs=JOBS`	Set the number of compression threads to run at a time. The default is 4, which works well for dual-core and quad-core machines. In general, it is best to set `JOBS` to the number of cores or processors you have.
`--level`	Set the compression quality to a value between 1 and 9. To generate archives faster, use a lower number. To generate smaller archives, use a higher number. The default compression level is 9.
`--merge-dups`	Request that identical files share the same data in the archive. For example if `level4/hero1.model` and `/level7/dude2.model` contain identical data, they each have their own entry in the Table of Contents but the entries point to the same data blocks in the archive. This can reduce the size of the archive.
`-N` `--no-compress`	Store everything (including the manifest) uncompressed in the archive. Use this option with the compress-manifest (`-C`) option.
`-o OUTPUT` `--output=OUTPUT`	Set the output file name. If no output file is specified when you create an archive, the first specified file or directory name is used as the file name and given a `.psarc` extension.
`-R` `--relative`	Make all paths in the archive relative. This is the default behavior.
`-s REGEX` `--strip=REGEX`	Prefix to strip from the pathnames stored in the archive. Specified as a Perl-compatible regex. This option can be specified more than once. Defaults to current working directory.
`-S` `--strip-all`	Strip all paths from files stored in the archive.
`--skip-missing-files`	Ignore it and continue when a file cannot be found.
`--xml=filename.xml`	Specify an XML input file to control the archive creation.
`--zlib`	Compress the archive using the ZLIB compression algorithm. This is the default.

Table 8 PSARC Extract Options

Option	Description
`--input=FILE`	Specify the archive to extract files from. By default, an extract operation uses the first file argument.
`--to=DIRECTORY`	Specify the directory to write extracted files to. The default is the current directory.

Table 9 PSARC Error Codes

Error Code (Hexadecimal)	Error Code (Decimal)	Description
0x80420001	-2143158271	Out of memory.
0x80420002	-2143158270	Out of range.
0x80420003	-2143158269	Read/Write error.
0x80420004	-2143158268	Corrupt file.
0x80420005	-2143158267	File not found.
0x80420006	-2143158266	File position error.
0x80420007	-2143158265	Internal error.
0x80420008	-2143158264	Invalid format.
0x80420009	-2143158263	Unexpected End of File (EOF).
0x8042000A	-2143158262	Invalid attribute.
0x8042000B	-2143158261	Missing attribute.
0x8042000C	-2143158260	Invalid element.
0x8042000D	-2143158259	Invalid number.
0x8042000E	-2143158258	Invalid argument.
0x8042000F	-2143158257	Invalid directory.
0x80420010	-2143158256	Not supported.
0x80420011	-2143158255	Cannot create archive.

3 Using FIOS2 with PSARC Archives

You use FIOS2 to access files in compressed archives created with the PSP2PSARC tool. When you add a FIOS2 dearchiver I/O filter, the dearchiver transparently handles requests for compressed and archived data. It receives normal read requests and automatically remaps them into compressed reads and decompresses the retrieved data as needed.

For more information about using FIOS2, refer to the *libfios2 Overview*.