NAME

jfr − parse and print Flight Recorder files

SYNOPSIS

To print the contents of a flight recording to standard out:

jfr print [*options*] *file*

To print metadata information about flight recording events:

jfr metadata *file*

To assemble chunk files into a flight recording file:

jfr assemble *repository file*

To disassemble a flight recording file into chunk files:

jfr disassmble [*options*] *file*

To view the summary statistics for a flight recording file:

jfr summary *file*

options Optional: Specifies command−line options separated by spaces. See the individual subcomponent sections for descriptions of the available options.

file Specifies the name of the target flight recording file (**.jfr**).

repository
Specifies the location of the chunk files which are to be assembled into a flight recording.

DESCRIPTION

The **jfr** command provides a tool for interacting with flight recorder files (**.jfr**). The main function is to filter, summarize and output flight recording files into human readable format. There is also support for merging and splitting recording files.

Flight recording files are created and saved as binary formatted files. Having a tool that can extract the contents from a flight recording and manipulate the contents and translate them into human readable format helps developers to debug performance issues with Java applications.

Subcommands

The **jfr** command has several subcommands:

- **print**
- **summary**
- **assemble**
- **disassemble**
- **metadata**

jfr print subcommand

Use **jfr print** to print the contents of a flight recording file to standard out. The syntax is:

jfr print [**--xml**|**--json**] [**--categories** <*filters*>] [**--events** <*filters*>] [**--stack-depth** <*depth*>] <*file*>

where:

--xml Print the recording in XML format

--json
Print the recording in JSON format

--categories <*filters*>
Select events matching a category name. The filter is a comma−separated list of names, simple and/or qualified, and/or quoted glob patterns

`--events` *<filters>*
: Select events matching an event name. The filter is a comma−separated list of names, simple and/or qualified, and/or quoted glob patterns

`--stack-depth` *<depth>*
: Number of frames in stack traces, by default 5

<file> Location of the recording file (**`.jfr`**)

The default format for printing the contents of the flight recording file is human readable form unless either **`xml`** or **`json`** is specified. These options provide machine−readable output that can be further parsed or processed by user created scripts.

Use **`jfr --help print`** to see example usage of filters.

To reduce the amount of data displayed, it is possible to filter out events or categories of events. The filter operates on the symbolic name of an event, set by using the **`@Name`** annotation, or the category name, set by using the **`@Category`** annotation. If multiple filters are used, events from both filters will be included. If no filter is used, all the events will be printed. If a combination of a category filter and event filter is used, the selected events will be the union of the two filters.

For example, to show all GC events and the CPULoad event, the following command could be used:

```
jfr print --categories GC --events CPULoad recording.jfr
```

Event values are formatted according to the content types that are being used. For example, a field with the **`jdk.jfr.Percentage`** annotation that has the value 0.52 is formatted as 52%.

Stack traces are by default truncated to 5 frames, but the number can be increased/decreased using the **`--stack-depth`** command−line option.

`jfr summary` subcommand
: Use **`jfr summary`** to print statistics for a recording. For example, a summary can illustrate the number of recorded events and how much disk space they used. This is useful for troubleshooting and understanding the impact of event settings.

The syntax is:

`jfr summary` *<file>*

where:

<file> Location of the flight recording file (**`.jfr`**)

`jfr metadata` subcommand
: Use **`jfr metadata`** to view information about events, such as event names, categories and field layout within a flight recording file. The syntax is:

`jfr metadata` *<file>*

where:

<file> Location of the flight recording file (**`.jfr`**)

jfr `assemble` subcommand
: Use jfr **`assemble`** to assemble chunk files into a recording file.

The syntax is:

`jfr assemble` *<repository> <file>*

where:

<repository>
: Directory where the repository containing chunk files is located

<file> Location of the flight recording file (**`.jfr`**)

Flight recording information is written in chunks. A chunk contains all of the information necessary for

parsing. A chunk typically contains events useful for troubleshooting. If a JVM should crash, these chunks can be recovered and used to create a flight recording file using this **jfr assemble** command. These chunk files are concatenated in chronological order and chunk files that are not finished (.part) are excluded.

jfr disassemble subcommand

Use **jfr disassemble** to decompose a flight recording file into its chunk file pieces. The syntax is:

jfr disassemble [**--max-chunks** *<chunks>*] [**--output** *<directory>*] *<file>*

where:

--output *<directory>*
: The location to write the disassembled file, by default the current directory

--max-chunks *<chunks>*
: Maximum number of chunks per file, by default 5. The chunk size varies, but is typically around 15 MB.

--max-size *<size>*
: Maximum number of bytes per file.

<file> Location of the flight recording file (**.jfr**)

This function can be useful for repairing a broken file by removing the faulty chunk. It can also be used to reduce the size of a file that is too large to transfer. The resulting chunk files are named**myfile_1.jfr**, **myfile_2.jfr**, etc. If needed, the resulting file names will be padded with zeros to preserve chronological order. For example, the chunk file name is **myfile_001.jfr** if the recording consists of more than 100 chunks.

jfr version and help subcommands

Use **jfr --version** or **jfr version** to view the version string information for this jfr command.

To get help on any of the jfr subcommands, use:

jfr <--help|help> [*subcommand*]

where:

[*subcommand*] is any of:

- **print**
- **metadata**
- **summary**
- **assemble**
- **disassemble**