

Filters settings documentation

by SuperBonBon

1. Introduction

This document will describe all the [uploadFilters](#) or [downloadFilters filters](#) settings that are provided with JAFS.

These filters can be used to accept, reject or notify files uploads or downloads on the server. The content of the file can be used by the filter to determine if the files can be sent or received.

Once a filter has processed the defined file content (stored in a buffer), the implementation can throw a filter exception if the processed data is not matching the filter criterias. The filter exception message will be returned to the client and the global transfert will be unsuccessful. For upload filters only the uploaded data will be deleted from the filesystem automatically.

Currently two filters are provided with JAFS :

- The audio MPEG filter: This filter can be used to reject mp3 file uploads not matching certain criteria such as bitrate.
- The JMS notification filter: This filter fires messages on an MQ containing the file name that has been transferred and some other useful info on a defined JMS message queue or topic.

2. Filters common settings

Basically there are 2 mandatory settings for upload or download filters.

2.1. fileType

Comma delimited file extensions (txt,bin,foo,bar) : This is the setting that defines the extension of the file uploaded or downloaded which can be processed by the filter. For an MPEG audio files filter, the value would be : mp3,mp2,mp1

The wildcard character (*) matchs all file types.

2.2. triggerOffset

Integer : This is the filter buffer size in bytes. Once the buffer is filled with incoming or outgoing data, it will be given to the filter for the data processing. The size is quite important since the filter result depends of this size. I.E For an MPEG audio file upload filter, a too small value (10 bytes) will make the processing almost always returning false since there are few chances that an MPEG header is included in the first 10 bytes (usually used by ID3 tags.). A too large value (10 mb) will waist bandwidth if the filter returns false since the filter will be only triggered when the buffer has reached 10 mb and the file automatically deleted from the hard disc.

Warning:

Do not provide a too big value (I.E 10 mb) since this buffer size is allowed during each transferts for the filter. The server could run out of memory quite rapidly with too large settings

If you need to only trigger the filter when the transfert is successfully (meaning no abort or io error) finished set the triggerOffset value to -1, the only restriction is that the buffer will be not provided (since the file could be huge) but a file object will be if you need the file data for some notification work.

Note:

Files are deleted automatically by upload filters only when the filters is returning false

Note:

Offsets larger than file size being sent or received will never trigger the filter.

3. Create your own filter

To create you own filter you simply need to create a class implementing the *net.sbbi.jafs.filters.UploadFilter* interface for an upload filter or the *net.sbbi.jafs.filters.DownloadFilter* interface for download filters. Note that both interfaces can be implemented by the same filter class.

We also recommend to extend the *net.sbbi.jafs.filters.GenericFilter* class to avoid to have to implement some generic filter work.

Once created, simply compile the class and drop the jar in the lib dir and add a new [filter](#) config entry in the server config file.

4. MPEG upload filter settings

Classname: *net.sbbi.jafs.filters.Mp3Filter*

This filter can be used to accept or reject MPEG audio files being received. The files can be rejected based on their bitrate and frequency.

Note:

Make sur that you provide at least 10kb of buffer size to make sure that the filter will have enough data to work with and return a correct result.

4.1. allowVBR

True or false : define if Variable Bit Rate files are accepted on the server or not

4.2. rejectUnparsable

True or false : if set to true all MP3 files that can't be correctly parsed are rejected during upload.

4.3. bitrateSmaller

An integer : filter will match all files that have a bit rate smaller that the defined value.

4.4. bitrateGreater

An integer : filter will match all files that have a bit rate greater that the defined value.

4.5. bitrateEquals

An integer : filter will match all files that have a bit rate equals to the defined value.

4.6. bitrateNotEquals

An integer : filter will match all files that have a bit rate different than the defined value.

4.7. frequencyGreater

An integer : filter will match all files that have frequency greater than the defined value.

4.8. frequencySmaller

An integer : filter will match all files that have frequency smaller than the defined value.

4.9. frequencyEquals

An integer : filter will match all files that have frequency equals to the defined value.

4.10. frequencyNotEquals

An integer : filter will match all files that have frequency not equals to the defined value.

5. JMS notifications filter settings

Classname: *net.sbbi.jafs.filters.JMSNotificationFilter*

This filter can be used to send JMS messages once a file has been uploaded or downloaded. The filter can define a topic and/or a queue where the message will be fired. The message (MapMessage) contains the following keys :

- filePath: The upload or downloaded file path
- user: The user name
- userIp: the user IP address
- serverCode: The ftp server code on which the transfert has been made
- serverRoot: The ftp server root path

5.1. JNDIInitialContext

A string : The JNDI initial context to access the JMS connection factory

5.2. JNDIProviderUrl

A string : The JNDI provider URL

5.3. queueConnectionFactoryJNDIName

A string : The JNDI name of the queue connection factory

5.4. topicConnectionFactoryJNDIName

A string : The JNDI name of the topic connection factory

5.5. queueJNDIName

A string : The JNDI name of the queue where the message will be fired

5.6. topicJNDIName

A string : The JNDI name of the topic where the message will be fired

6. Create your own filters

To create your own filters you'll simply need to implement the *net.sbbi.jafs.filters.DownloadFilter* interface for a download filter and/or the *net.sbbi.jafs.filters.UploadFilter* interface for an upload filter.

We also recommend you to extend the *net.sbbi.jafs.filters.GenericFilter* class to have less work to do, here is a small implementation example :

```
package net.sbbi.jafs.filters;

import java.io.File;

import net.sbbi.jafs.config.FtpServerConfig;
import net.sbbi.jafs.session.SessionContext;
import net.sbbi.jafs.utils.Translator;

import org.apache.avalon.framework.parameters.ParameterException;
import org.apache.avalon.framework.parameters.Parameters;

/**
 * Simple sample hello world filter
 */
public class HelloWorldFilter extends GenericFilter
    implements UploadFilter, DownloadFilter {

    public void filter( Translator translator, byte[] buffer, File processedFile,
        SessionContext session, FtpServerConfig s

        throws FilterException {
        String msg = "Hello world from file '" + processedFile.getPath() +
            "' transferred by user '" + session.getUserName() +
            "' on server '" + serverConfig.getServerCode() + "'";
        System.out.println( msg );
    }

    public void customParameterize( Parameters parameters )
        throws ParameterException {
        // no params to parse
    }
}
```

```
}
```