Heritrix Documentation

Internet Archive and contributors

Jul 27, 2022

Contents:

Note: More Heritrix documentation currently lives on the Github wiki. We're in the process of editing some of the structured guides and migrating them here.

CHAPTER 1

Getting Started with Heritrix

1.1 System Requirements

Heritrix is primarily used on Linux. It may run on other platforms but is not regularly tested or supported on them.

Heritrix requires Java 8 or 11. We recommend using your Linux distribution's OpenJDK 11 packages. Alternatively up to date builds of OpenJDK 8 and 11 for several platforms are available from Adoptium.

The default Java heap for Heritrix is 256MB RAM, which is usually suitable for crawls that range over hundreds of hosts. Assign more of your available RAM to the heap if you are crawling thousands of hosts or experience Java out-of-memory problems. You can use the JAVA_OPTS variable to configure memory

1.2 Installation

Download the latest Heritrix distribution package linked from the Heritrix releases page and unzip it somewhere.

The installation will contain the following subdirectories:

bin contains shell scripts/batch files for launching Heritrix.

lib contains the third-party .jar files the Heritrix application requires to run.

conf contains various configuration files (such as the configuration for Java logging, and pristine versions of the bundled profiles)

jobs the default location where operator-created jobs are stored

1.3 Environment Variables

1. Set the `JAVA_HOME` environment variable. The value should point to your Java installation.

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk
```

2. Set the `HERITRIX_HOME` environment variable. The value should be set to the path where Heritrix is installed.

   ```
   export HERITRIX_HOME=/home/user/heritrix3.1
   ```

3. Set execute permission on the Heritirix startup file.

   ```
   chmod u+x $HERITRIX_HOME/bin/heritrix
   ```

4. To change the amount of memory allocated to Heritrix (the Java heap size), set the `JAVA_OPTS` environment variable. The following example allocates 1GB of memory to Heritrix.

   ```
   export JAVA_OPTS=-Xmx1024M
   ```

1.4 Runnning Heritrix

To launch Heritrix with the Web UI enabled, enter the following command. The username and password for the Web UI are set to "admin" and "admin", respectively.

```
$HERITRIX_HOME/bin/heritrix -a admin:admin
```

By default, the Web UI listening address is only bound to the 'localhost' address. Therefore, the Web UI can only be accessed on the same machine from which it was launched. The '-b' option may be used to listen on different/additional addresses. See *Security Considerations* before changing this default.

If the parameter supplied to the `-a` option is a file path beginning with "@", the admin username and password will be read from a file. This adds an additional layer of protection to the admin username and password by ensuring they don't appear directly in the command-line and can't be seen by other users running the `ps` command.

1.5 Accessing the User Interface

After Heritrix has been launched, the Web-based user interface (WUI) becomes accessible.

The URI to access the Web UI is typically

https://localhost:8443/

The initial login page prompts for the username and password. After login, your session will time-out after a period of non-use.

Access to the WUI is through HTTPS. Heritrix is installed with a keystore containing a self-signed certificate. This will cause your browser to display a prompt, warning that a self-signed certificate is being used. Follow the steps below for your browser to login to Heritrix for the first time.

Chrome: The message "Your connection is not private" is displayed. Click the "Advanced" button and then click "Proceed to localhost (unsafe)."

Firefox: The message "Warning: Potential Security Risk Ahead" is displayed. Click the "Advanced…" button and then click "Accept the Risk and Continue."

1.6 Your First Crawl

1. Enter the name of the new job in the text box with the "create new job" label. Then click "create".
2. Click on the name of the new job and you will be taken to the job page.

3. Click on the "Configuration" link at the top and the contents of the job configuration file will be displayed.
4. At this point you must enter several properties to make the job runnable.
   1. First, add the URL of page explaining how webmasters can contact you to the metadata.operatorContactUrl property.
   2. Next, populate the `<prop>` element of the `longerOverrides` bean with the seed values for the crawl. A test seed is configured for reference.
   3. When done click "save changes" at the top of the page.

   For more detailed information on configuring jobs see Configuring Jobs and Profiles
5. From the job screen, click "build." This command will validate the job configuration and load it into memory. In the Job Log the following message will display: "INFO JOB instantiated."
6. Next, click the "launch" button. This command launches the job in "paused" mode. At this point the job is ready to run.
7. To run the job, click the "unpause" button. The job will now begin sending requests to the seeds of your crawl. The status of the job will be set to "Running." Refresh the page to see updated statistics.
8. When you want to stop your crawl, click the "terminate" button to finish and then "teardown" to unload the job configuration from memory.

1.7 Exiting Heritrix

To exit Heritrix get back to the main page by clicking "Engine" in the top bar. Then check the "I'm sure" box under "Exit Java" and click the "exit java process" button.

CHAPTER 2

Operating Heritrix

2.1 Running Heritrix

To launch Heritrix with the Web UI enabled, enter the following command. The username and password for the Web UI are set to “admin” and “admin”, respectively.

```
$HERITRIX_HOME/bin/heritrix -a admin:admin
```

By default, the Web UI listening address is only bound to the ‘localhost’ address. Therefore, the Web UI can only be accessed on the same machine from which it was launched. The ‘-b’ option may be used to listen on different/additional addresses. See *Security Considerations* before changing this default.

2.1.1 Command-line Options

-a, --web-admin ARG **(Required)** Sets the username and password required to access the Web UI.

The argument may be a `USERNAME:PASSWORD` such as `admin:admin`. If the argument is a string beginning with “@”, the rest of the string is interpreted as a local file name containing the operator login and password.

-b, --web-bind-hosts HOST Specifies a comma-separated list of hostnames/IP-addresses to bind to the Web UI. You may use ‘/’ as a shorthand for ‘all addresses’. **Default**: `localhost/127.0.0.1`

-j, --job-dirs PATH Sets the directory Heritrix stores jobs in. **Default:** `$HERITRIX_HOME/jobs`

-l, --logging-properties PATH Reads logging configuration from a file. **Default:** `$HERITRIX_HOME/conf/logging.properties`

-p, --web-port PORT Sets the port the Web UI will listen on. **Default:** `8443`

-r, --run-job JOBNAME Runs the given Job when Heritrix starts. Heritrix will exit when the job finishes.

-s, --ssl-params ARG Specifies a keystore path, keystore password, and key password for HTTPS use. Separate the values with commas and do not include whitespace. By default Heritrix will generate a self-signed certificate the first time it is run.

2.1.2 Environment Variables

The Heritrix launch script `./bin/heritrix` obeys the following environment variables:

FOREGROUND Set to any value – e.g. 'true' – if you want to run heritrix in the foreground.

JAVA_HOME Directory where Java is installed.

JAVA_OPTS Additional options to pass to the JVM. For example specify `-Xmx1024M` to allocate 1GB of memory to Heritrix.

HERITRIX_HOME Directory where Heritrix is installed.

HERITRIX_OUT Path messages will be logged to when running in background mode. **Default:** `$HERITRIX_HOME/heritrix_out.log`

2.2 Running Heritrix under Docker

Heritrix can also be run under Docker. The Web UI is enabled by default and exposed via port 8443. The following command creates and runs a detached container with username and password for the Web UI set to "admin" and "admin", respectively. It also maps the local `jobs` directory into the container. Please ensure that mounted directories already exist or have the correct permissions!

```
mkdir jobs
docker run --init --rm -d -p 8443:8443 -e "USERNAME=admin" -e "PASSWORD=admin" -v
↪$(pwd)/jobs:/opt/heritrix/jobs iipc/heritrix
```

See *Security Considerations* about securely running Heritrix.

2.2.1 Configurations

To allow Heritrix to be run within a container, the environment variables (`FOREGROUND`, `JAVA_HOME`, `HERITRIX_HOME`) are already set and should not be changed. The Heritrix command-line options can't be accessed directly and are only exposed via environment variables:

USERNAME, PASSWORD (Required) The Web UI username and password. Will be forwarded to `-a, --web-admin ARG`.

CREDSFILE If either `USERNAME` or `PASSWORD` are not set or empty, `CREDSFILE` can alternatively be used to supply the Web UI credentials. It should be a path within the Heritrix container which can be used to bind-mount local files or docker secrets. See "@" description for `-a, --web-admin ARG`.

JOBNAME This forwards the jobname to the `-r, --run-job JOBNAME` command-line option, to run a single job and then quit. Note that your container should not have a restart policy set to automatically restart on exit.

2.3 Security Considerations

Heritrix is a large and active network application that presents security implications, both on the local machine, where it runs, and remotely, on machines it contacts.

2.3.1 Understanding the Risks

It is important to recognize that the Web UI allows remote control of the crawler in ways that could potentially disrupt a crawl, change the crawler's behavior, read or write locally-accessible files, and perform or trigger other actions in the Java VM or local machine by the execution of arbitrary operator-supplied scripts.

Unauthorized access to the Web UI could end or corrupt a crawl. It could also change the crawler's behavior to be a nuisance to other network hosts. Files accessible to the crawler process could potentially be deleted, corrupted, or replaced, which could cause extensive problems on the crawling machine.

Another potential risk is that worst-case or maliciously-crafted content, in conjunction with crawler issues, could disrupt the crawl or other files and operations on the local system. For example, in the past, without malicious intent, some rich-media content has caused runaway memory use in third-party libraries used by the crawler. This resulted in memory-exhaustion that stopped and corrupted the crawl in progress. Similarly, atypical input patterns have caused runaway CPU use by crawler link-extraction regular expressions, causing severely slow crawls. Crawl operators should monitor their crawls closely and use the project discussion list and issue database to stay current on crawler issues.

2.3.2 Network Access Control

Launched without any specified bind-address ('-b' flag), the crawler's Web UI only binds to the localhost/loopback address (127.0.0.1), and therefore is only network-accessible from the same machine on which it was launched.

If practical, this default setting should be maintained. A technique such as SSH tunneling could be used by authorized users of the crawling machine to enable Web access from their local machine to the crawling machine.For example, consider Heritrix running on a machine 'crawler.example.com', with its Web UI only listening/bound on its localhost address. Assuming a user named 'crawloperator' has SSH access to 'crawler.example.com', she can issue the following SSH command from her local machine:

```
ssh -L localhost:9999:localhost:8443 crawloperator@crawler.example.com -N
```

This tells SSH to open a tunnel which forwards conections to "localhost:9999" (on the local machine) to the remote machines' own idea of "localhost:8443". As a result, the crawler's Web UI will be available via "https://localhost:9999/" for as long as the tunnel exists (until the ssh command is killed or connection otherwise broken). No one else on the network may directly connect to port 8443 on 'crawler.example.com' (since it is only listening on the local loopback address), and no one elsewhere on the net may directly connect to the operator's port 9999 (since it also is only listening on the local loopback address).

If you need Heritrix's listening port bound to a public address, the '-b' command-line flag may be used. This flag takes, as an argument, the hostname/address to use. The '/' character can be used to indicate all addresses.

If you use this option, you should take special care to choose an even more unique/unguessable/brute-force-search-resistant set of login credentials. You may still want to consider using other network/firewall policies to block access from unauthorized origins.

2.3.3 Login Authentication Access Control

The administrative login and password only offer rudimentary protection against unauthorized access. For best security, you should be sure to:

1. Use a strong, unique username and password combination to secure the Web UI. Heritrix uses HTTPS to encrypt communication between the client and the Web UI. Keep in mind that setting the username and password on the command-line may result in their values being visible to other users of the crawling machine – for example, via the output of a tool like 'ps' that shows the command-lines used to launch processes. Additionally, note that these values are echoed in plain text in the `heritrix_out.log` for operator reference. As of Heritrix 3.1, the administrative username and password are no longer echoed to `heritrix_out.log`. Also, if the parameter

supplied to the -a command line option is a string beginning with "@", the rest of the string is interpreted as a local file name containing the operator login and password. Thus, the credentials are not visible to other machines that use the process listing (ps) command.

2. Launch the Heritrix-hosting Java VM with a user-account that has the minimum privileges necessary for operating the crawler. This will limit the damage in the event that the Web UI is accessed maliciously.

2.4 Log Files

Each crawl job has its own set of log files found in the `logs` subdirectory of a job launch directory.

Logging can be configured by modifying the `logging.properties` file that is located under the `$HERITRIX_HOME/conf` directory. For information on using logging properties, visit http://logging.apache.org/log4j/.

2.4.1 alerts.log

This log contains alerts that indicate problems with a crawl.

2.4.2 crawl.log

Each URI that Heritrix attempts to fetch will cause a log line to be written to the `crawl.log` file. Below is a two line extract from the log.

Field 1. Timestamp The timestamp in ISO8601 format, to millisecond resolution. The time is the instant of logging.

Field 2. ***Fetch Status Code*** Usually this is the HTTP response code but it can also be a negative number if URI processing was unexpectedly terminated.

Field 3. Document Size The size of the downloaded document in bytes. For HTTP, this is the size of content only. The size excludes the HTTP response headers. For DNS, the size field is the total size for the DNS response.

Field 4. Downloaded URI The URI of the document downloaded.

Field 5. Discovery Path The breadcrumb codes (discovery path) showing the trail of downloads that lead to the downloaded URI. The length of the discovery path is limited to the last 50 hop-types. For example, a 62-hop path might appear as "12+LLRLLLRELLLLRLLLRELLLLRLLLRELLLLRLLLRELLLLRLLLRELE".

The breadcrumb codes are as follows.

R	Redirect
E	Embed
X	Speculative embed (aggressive/Javascript link extraction)
L	Link
P	Prerequisite (as for DNS or robots.txt before another URI)

Field 6. Referrer The URI that immediately preceded the downloaded URI. This is the referrer. Both the discovery path and the referrer will be empty for seed URIs.

Field 7. Mime Type The downloaded document mime type.

Field 8. Worker Thread ID The id of the worker thread that downloaded the document.

Field 9. Fetch Timestamp The timestamp in RFC2550/ARC condensed digits-only format indicating when the network fetch was started. If appropriate the millisecond duration of the fetch is appended to the timestamp with a ";" character as separator.

Field 10. SHA1 Digest The SHA1 digest of the content only (headers are not digested).

Field 11. Source Tag The source tag inherited by the URI, if source tagging is enabled.

Field 12. Annotations If an annotation has been set, it will be displayed. Possible annotations include: the number of times the URI was tried, the literal "lenTrunc"; if the download was truncanted due to exceeding configured size limits, the literal "timeTrunc"; if the download was truncated due to exceeding configured time limits or "midFetchTrunc"; if a midfetch filter determined the download should be truncated.

Field 13. WARC Filename The name of the WARC/ARC file to which the crawled content is written. This value will only be written if thelogExtraInfo property of the loggerModule bean is set to true. This logged information will be written in JSON format.

2.4.3 progress-statistics.log

This log is written by the StatisticsTracker bean. At configurable intervals, a log line detailing the progress of the crawl is written to this file.

Field 1. timestamp Timestamp in ISO8601 format indicating when the log line was written.

Field 2. discovered Number of URIs discovered to date.

Field 3. queued Number of URIs currently queued.

Field 3. downloaded Number of URIs downloaded to date.

Field 4. doc/s(avg) Number of document downloaded per second since the last snapshot. The value in parenthesis is measured since the crawl began.

Field 5. KB/s(avg) Amount in kilobytes downloaded per second since the last snapshot. The value in parenthesis is measured since the crawl began.

Field 6. dl-failures Number of URIs that Heritrix has failed to download.

Field 7. busy-thread Number of toe threads busy processing a URI.

Field 8. mem-use-KB Amount of memory in use by the Java Virtual Machine.

Field 9. heap-size-KB The current heap size of the Java Virtual Machine.

Field 10. congestion The congestion ratio is a rough estimate of how much initial capacity, as a multiple of current capacity, would be necessary to crawl the current workload at the maximum rate available given politeness settings. This value is calculated by comparing the number of internal queues that are progressing against those that are waiting for a thread to become available.

Field 11. max-depth The size of the Frontier queue with the largest number of queued URIs.

Field 12. avg-depth The average size of all the Frontier queues.

2.4.4 runtime-errors.log

This log captures unexpected exceptions and errors that occur during the crawl. Some may be due to hardware limitations (out of memory, although that error may occur without being written to this log), but most are probably due to software bugs, either in Heritrix's core but more likely in one of its pluggable classes.

2.4.5 uri-errors.log

This log stores errors that resulted from attempted URI fetches. Usually the cause is non-existent URIs. This log is usually only of interest to advanced users trying to explain unexpected crawl behavior.

2.5 Reports

Reports are found in the “reports” directory, which exists under the directory of a specific job launch.

2.5.1 Crawl Summary (crawl-report.txt)

This file contains useful metrics about completed jobs. The report is created by the StatisticsTracker bean. This file is written at the end of the crawl.

Below is sample output from this file:

```
Crawl Name: basic
Crawl Status: Finished
Duration Time: 1h33m38s651ms
Total Seeds Crawled: 1
Total Seeds not Crawled: 0
Total Hosts Crawled: 1
Total URIs Processed: 1337
URIs Crawled successfully: 1337
URIs Failed to Crawl: 0
URIs Disregarded: 0
Processed docs/sec: 0.24
Bandwidth in Kbytes/sec: 4
Total Raw Data Size in Bytes: 23865329 (23 MB)
Novel Bytes: 23877375 (23 MB)
```

Crawl Name The user-defined name of the crawl.

Crawl Status The status of the crawl, such as “Aborted” or “Finished.”

Duration Time The duration of the crawl to the nearest millisecond.

Total Seeds Crawled The number of seeds that were successfully crawled.

Total Seeds Not Crawled The number of seeds that were not successfully crawled.

Total Hosts Crawled The number of hosts that were crawled.

Total URIs Processed The number of URIs that were processed.

URIs Crawled Successfully The number of URIs that were crawled successfully.

URIs Failed to Crawl The number of URIs that could not be crawled.

URIs Disregarded The number of URIs that were not selected for crawling.

Processed docs/sec The average number of documents processed per second.

Bandwidth in Kbytes/sec The average number of kilobytes processed per second.

Total Raw Data Size in Bytes The total amount of data crawled.

Novel Bytes New bytes since last crawl.

2.5.2 Seeds (seeds-report.txt)

This file contains the crawling status of each seed.

This file is created by the StatisticsTracker bean and is written at the end of the crawl.

Below is sample output from this report:

```
[code] [status] [seed] [redirect]
200 CRAWLED http://www.smokebox.net
```

code *Status code* for the seed URI

status Human readable description of whether the seed was crawled. For example, "CRAWLED."

seed The seed URI.

redirect The URI to which the seed redirected.

2.5.3 Hosts (hosts-report.txt)

This file contains an overview of the hosts that were crawled. It also displays the number of documents crawled and the bytes downloaded per host.

This file is created by the StatisticsTracker bean and is written at the end of the crawl.

Below is sample output from this file:

```
1337 23877316 www.smokebox.net 0 0
1 59 dns: 0 0
0 0 dns: 0 0
```

#urls The number of URIs crawled for the host.

#bytes The number of bytes crawled for the host.

host The hostname.

#robots The number of URIs, for this host, excluded because of `robots.txt` restrictions. This number does not include linked URIs from the specifically excluded URIs.

#remaining The number of URIs, for this host, that have not been crawled yet, but are in the queue.

#novel-urls The number of new URIs crawled for this host since the last crawl.

#novel-bytes The amount of new bytes crawled for this host since the last crawl.

#dup-by-hash-urls The number of URIs, for this host, that had the same hash code and are essentially duplicates.

#dup-by-hash-bytes The number of bytes of content, for this host, having the same hashcode.

#not-modified-urls The number of URIs, for this host, that returned a 304 status code.

#not-modified-bytes The amount of of bytes of content, for this host, whose URIs returned a 304 status code.

2.5.4 SourceTags (source-report.txt)

This report contains a line item for each host, which includes the seed from which the host was reached.

Below is a sample of this report:

```
[source] [host] [#urls]
http://www.fizzandpop.com/ dns: 1
http://www.fizzandpop.com/ www.fizzandpop.com 1
```

source The seed.

host The host that was accessed from the seed.

#urls The number of URIs crawled for this seed host combination.

Note that the SourceTags report will only be generated if the `sourceTagSeeds` property of the `TextSeedModule` bean is set to true.

```
<bean id="seeds" class="org.archive.modules.seeds.TextSeedModule">
  <property name="sourceTagsSeeds" value="true" />
</bean>
```

2.5.5 Mimetypes (mimetype-report.txt)

This file contains a report displaying the number of documents downloaded per mime type. Also, the amount of data downloaded per mime type is displayed.

This file is created by the StatisticsTracker bean and is written at the end of the crawl.

Below is sample output from this report:

```
624 13248443 image/jpeg
450 8385573 text/html
261 2160104 image/gif
1 74708 application/x-javascript
1 59 text/dns
1 8488 text/plain
```

#urls The number of URIs crawled for a given mime-type.

#bytes The number of bytes crawled for a given mime-type.

mime-types The mime-type.

2.5.6 ResponseCode (responsecode-report.txt)

This file contains a report displaying the number of documents downloaded per status code. It covers successful codes only. For failure codes see the crawl.log file.

This file is created by the StatisticsTracker bean and is written at the end of the crawl.

Below is sample output from this report:

```
[#urls] [rescode]
1306 200
31 404
1 1
```

#urls The number of URIs crawled for a given response code.

rescode The response code.

2.5.7 Processors (processors-report.txt)

This report shows the activity of each processor involved in the crawl. For example, the `FetchHTTP` processor is included in the report. For this processor the number of URIs fetched is displayed. The report is organized to report on each Chain (Candidate, Fetch, and Disposition) and each processor in each chain. The order of the report is per the configuration order in the `crawler-beans.cxml` file.

Below is sample output from this report:

```
CandidateChain - Processors report - 200910300032
  Number of Processors: 2

Processor: org.archive.crawler.prefetch.CandidateScoper

Processor: org.archive.crawler.prefetch.FrontierPreparer

FetchChain - Processors report - 200910300032
  Number of Processors: 9

Processor: org.archive.crawler.prefetch.Preselector

Processor: org.archive.crawler.prefetch.PreconditionEnforcer

Processor: org.archive.modules.fetcher.FetchDNS

Processor: org.archive.modules.fetcher.FetchHTTP
  Function:          Fetch HTTP URIs
  CrawlURIs handled: 1337
  Recovery retries:   0

Processor: org.archive.modules.extractor.ExtractorHTTP
  Function:          Extracts URIs from HTTP response headers
  CrawlURIs handled: 1337  Links extracted:   0

Processor: org.archive.modules.extractor.ExtractorHTML
  Function:          Link extraction on HTML documents
  CrawlURIs handled: 449
  Links extracted:   6894
...
```

2.5.8 FrontierSummary (frontier-summary-report.txt)

This link displays a report showing the hosts that are queued for capture. The hosts are contained in multiple queues. The details of each Frontier queue is reported.

2.5.9 ToeThreads (threads-report.txt)

This link displays a report showing the activity of each thread used by Heritrix. The amount of time the thread has been running is displayed as well as thread state and thread Blocked/Waiting status.

2.6 Action Directory

Each job directory contains an action directory. By placing files in the action directory you can trigger actions in a running crawl job, such as the addition of new URIs to the crawl.

At a regular interval (by default less than a minute), the crawl will notice any new files in this directory, and take action based on their filename suffix and their contents. When the action is done, the file will be moved to the nearby 'done' directory. (For this reason, files should be composed outside the action directory, then moved there as an atomic whole. Otherwise, a file may be processed-and-moved while still being composed.)

The following file suffixes are supported:

.seeds A .seeds file should contain seeds that the Heritrix operator wants to include in the crawl. Placing a .seeds file in the action directory will add the seeds to the running crawl. The same directives as may be used in seeds-lists during initial crawl configuration may be used here.

If seeds introduced into the crawl this way were already in the frontier (perhaps already a seed) this method does not force them.

.recover .recover file will be treated as a traditional recovery journal. (The recovery journal can approximately reproduce the state of a crawl's queues and already-included set, by repeating all URI-completion and URI-discovery events. A recovery journal reproduces less state than a proper checkpoint.) In a first pass, all lines beginning with Fs in the recovery journal will be considered included, so that they can not be enqueued again. Then in a second pass, lines starting with F+ will be re-enqueued for crawling (if not precluded by the first pass).

.include A .include file will be treated as a recovery journal, but all URIs no matter what their line-prefix will be marked as already included, preventing them from being re-enqueued from that point on. (Already-enqueued URIs will still be eligible for crawling when they come up.) Using a .include file is a way to suppress the re-crawling of URIs.

.schedule A .schedule file will be treated as a recovery journal, but all URIs no matter what their line-prefix will be offered for enqueueing. (However, if they are recognized as already-included, they will not be enqueued.) Using a .schedule file is a way to include URIs in a running crawl by inserting them into the Heritrix crawling queues.

.force A .force file will be treated as a recovery journal with all the URIs marked for force scheduling. Using a .force file is a way to guarantee that already-included URIs will be re-enqueued and (and thus eventually re-crawled).

Any of these files may be gzipped. Any of the files in recovery journal format (`.recover`, `.include`, `.schedule`, `.force`) may have a `.s` inserted prior to the functional suffix (for example, `frontier.s.recover.gz`), which will cause the URIs to be scope-tested before any other insertion occurs.

For example you could place the following `example.schedule` file in the action directory to schedule a URL:

```
F+ http://example.com
```

In order to use the action directory, the `ActionDirectory` bean must be configured in the `crawler-beans.cxml` file as illustrated below.

```
<bean id="actionDirectory" class="org.archive.crawler.framework.ActionDirectory">
  <property name="actionDir" value="action" />
  <property name="initialDelaySeconds" value="10" />
  <property name="delaySeconds" value="30" />
</bean>
```

The recovery journal directives are listed below:

F+	Add
Fe	Emit
Fi	Include
Fd	Disregard
Fr	Re-enqueued
Fs	Success
Ff	Failure

Note that the recovery journal format's 'F+' lines may include a 'hops-path' and 'via URI', which are preserved when a URI is enqueued via the above mechanisms, but that this may not be a complete representation of all URI state from its discovery in a normal crawl.

2.7 Crawl Recovery

During normal operation, the Heritrix Frontier keeps a journal. The journal is kept in the logs directory. It is named `frontier.recovery.gz`. If a crash occurs during a crawl, the `frontier.recover.gz` journal can be used to recreate the approximate status of the crawler at the time of the crash. In some cases, recovery may take an extended period of time, but it is usually much quicker than repeating the crashed crawl.

If using this process, you are starting an all-new crawl, with your same (or modified) configuration, but this new crawl will take an extended detour at the beginning where it uses the prior crawl's frontier-recover.gz output(s) to simulate the frontier status (discovered-URIs, enqueued-URIs) of the previous crawl. You would move aside all ARC/WARCs, logs, and checkpoints from the earlier crawl, retaining the logs and ARC/WARCs as a record of the crawl so far.

Any ARC/WARC files that exist with the `.open` suffix were not properly closed by the previous run, and may include corrupt/truncated data in their last partial record. You may rename files with a `.warc.gz.open` suffix to `.warc.gz`, but consider validating such ARC/WARCs (by zcat'ing the file to /dev/null to check gzip validity, or other ARC/WARC tools for record completeness) before removing the ".open" suffix.

2.7.1 Full recovery

To run the recovery process, relaunch the crashed crawler and copy the `frontier.recover.gz` file into the *Action Directory*. Then re-start the crawl. Heritrix will automatically load the recovery file and begin placing its URIs into the Frontier for crawling.

If using a `.recover.gz` file, a single complete file must be used. (This is so that the action directory processing of one file at a time can do the complete first pass of 'includes', then the complete full pass of 'schedules', from one file. Supplying multiple `.recover.gz` files in series will result in an includes/schedules, includes/schedules, etc. cycle which will not produce the desired effect on the frontier.)

While the file is being processed, any checkpoints (manual or auto-periodic) will **not** be a valid snapshot of the crawler state. (The frontier-recovery log process happens via a separate thread/path outside the newer checkpointing system.) Only when the file processing is completed (file moved to 'done') will the crawler be in an accurately checkpointable state.

Once URIs start appearing in the queues (the recovery has entered the 'schedules' pass), the crawler may be unpaused to begin fetching URIs while the rest of the 'schedules' recovery pass continues. However, the above note about checkpoints still applies: only when the frontier-recovery file-processing is finished may an accurate checkpoint occur. Also, unpausing the crawl in this manner may result in some URIs being rediscovered via new paths before the original discovery is replayed via the recovery process. (Many crawls may not mind this slight deviation from the recovered' crawls state, but if your scoping is very path- or hop- dependent it could make a difference in what is scope-included.)

Note: Feeding the entire frontier back to the crawler is likely to produce many *"Problem line"* warnings in the job log. Some operators find it useful to allow the entire recovery file to be ingested by the crawler before attempting to resume (unpause), to help isolate this chatter, and to minimize generating duplicate crawldata during recovery.

2.7.2 Split Recovery

An alternate way to run the recovery process is illustrated below. By eliminating irrelevant lines early (outside the recovery process), it may allow the recovery process to complete more quickly than the standard process. It also allows the process to proceed from many files, rather than a single file, so may give a better running indication of progress, and chances to checkpoint the recover.

To run the alternate recovery process:

1. move aside prior logs and ARCs/WARCs as above

2. relaunch the crashed crawler
3. Split any source `frontier.recover.gz` files using commands like the following:

```
zcat frontier.recover.gz | grep '^Fs' | gzip > frontier.include.gz
zcat frontier.recover.gz | grep '^F+' | gzip > frontier.schedule.gz
```

4. Build and launch the previously failed job (with the same or adjusted configuration). The job will now be paused.
5. Move the `frontier.include.gz` file(s) into the action directory. The `action` directory is located at the same level in the file structure hierarchy as the `bin` directory. (If you have many, you may move them all in at once, or in small batches to better monitor their progress. At any point when all previously-presented files are processed – that is, moved to the 'done' directory – it is possible to make a valid checkpoint.)
6. You may watch the progress of this 'includes' phase by viewing the web UI or `progress-statistics.log` and seeing the `discovered` count rise.
7. When all `.includes` are finished loading, you can repeat the process with all the `.schedule` logs.
8. When you notice a large number (many thousands) of URIs in the `queued` count, you may unpause the crawl to let new crawling proceed in parallel to the enqueueing of older URIs.

You **may** drop all `.include` and `.schedule` files into the action directory before launch, if you are confident that the lexicographic ordering of their names will do the right thing (present all `.include` files first, and the `.schedule` files in the same order as the original crawl). But, that leave little opportunity to adjust/checkpoint the process: the action directory will discover them all and process them all in one tight loop.

Note: To be sure of success and current crawl status against any sort of possible IO/format errors, in large recoveries of millions of records, you may want to wait for each step to complete before moving a file, or unpausing the job. Instead of looking at progress-statistics, simply wait for the file to move from action to action/done. Then add the second file. Wait again. Finally unpause the crawler.

A recovery of 100M URIs may take days, so please be patient.

CHAPTER 3

Configuring Crawl Jobs

3.1 Basic Job Settings

Crawl settings are configured by editing a job's crawler-beans.cxml file. Each job has a `crawler-beans.cxml` file that contains the Spring configuration for the job.

3.1.1 Crawl Limits

In addition to limits imposed on the scope of the crawl it is possible to enforce arbitrary limits on the duration and extent of the crawl with the following settings:

maxBytesDownload Stop the crawl after a fixed number of bytes have been downloaded. Zero means unlimited.

maxDocumentDownload Stop the crawl after downloading a fixed number of documents. Zero means unlimited.

maxTimeSeconds Stop the crawl after a certain number of seconds have elapsed. Zero means unlimited. For reference there are 3600 seconds in an hour and 86400 seconds in a day.

To set these values modify the CrawlLimitEnforcer bean.

```
<bean id="crawlLimitEnforcer" class="org.archive.crawler.framework.CrawlLimitEnforcer
↪">
  <property name="maxBytesDownload" value="100000000" />
  <property name="maxDocumentsDownload" value="100" />
  <property name="maxTimeSeconds" value="10000" />
</bean>
```

Note: These are not hard limits. Once one of these limits is hit it will trigger a graceful termination of the crawl job. URIs already being crawled will be completed. As a result the set limit will be exceeded by some amount.

3.1.2 maxToeThreads

The maximum number of toe threads to run.

If running a domain crawl smaller than 100 hosts, a value approximately twice the number of hosts should be enough. Values larger then 150-200 are rarely worthwhile unless running on machines with exceptional resources.

```
<bean id="crawlController" class="org.archive.crawler.framework.CrawlController">
  <property name="maxToeThreads" value="50" />
</bean>
```

3.1.3 metadata.operatorContactUrl

The URI of the crawl initiator. This setting gives the administrator of a crawled host a URI to refer to in case of problems.

```
<bean id="simpleOverrides" class="org.springframework.beans.factory.config.
↪PropertyOverrideConfigurer">
  <property name="properties">
    <value>
      metadata.operatorContactUrl=http://www.archive.org
      metadata.jobName=basic
      metadata.description=Basic crawl starting with useful defaults
    </value>
  </property>
</bean>
```

3.1.4 Robots.txt Honoring Policy

The valid values of “robotsPolicyName” are:

obey Obey robots.txt directives and nofollow robots meta tags

classic Same as “obey”

robotsTxtOnly Obey robots.txt directives but ignore robots meta tags

ignore Ignore robots.txt directives and robots meta tags

```
<bean id="metadata" class="org.archive.modules.CrawlMetadata" autowire="byName">
...
    <property name="robotsPolicyName" value="obey"/>
...
</bean>
```

Note: Heritrix currently only supports wildcards (*) at the end of paths in robots.txt rules.

The only supported value for robots meta tags is “nofollow” which will cause the HTML extractor to stop processing and ignore all links (including embeds like images and stylesheets). Heritrix does not support “rel=nofollow” on individual links.

```
<meta name="robots" content="nofollow"/>
```

3.2 Crawl Scope

The crawl scope defines the set of possible URIs that can be captured by a crawl. These URIs are determined by DecideRules, which work in combination to limit or expand the set of crawled URIs. Each DecideRule, when presented with an object (most often a URI of some form) responds with one of three decisions:

ACCEPT the object is ruled in

REJECT the object is ruled out

PASS the rule has no opinion; retain the previous decision

A URI under consideration begins with no assumed status. Each rule is applied in turn to the candidate URI. If the rule decides ACCEPT or REJECT, the URI's status is set accordingly. After all rules have been applied, the URI is determined to be "in scope" if its status is ACCEPT. If its status is REJECT it is discarded.

We suggest starting with the rules in our recommended default configurations and performing small test crawls with those rules. Understand why certain URIs are ruled in or ruled out under those rules. Then make small individual changes to the scope to achieve non-default desired effects. Creating a new ruleset from scratch can be difficult and can easily result in crawls that can't make the usual minimal progress that other parts of the crawler expect. Similarly, making many changes at once can obscure the importance of the interplay and ordering of the rules.

3.2.1 Decide Rules

AcceptDecideRule This DecideRule accepts any URI.

ContentLengthDecideRule This DecideRule accepts a URI if the content-length is less than the threshold. The default threshold is 2^63, meaning any document will be accepted.

PathologicalPathDecideRule This DecideRule rejects any URI that contains an excessive number of identical, consecutive path-segments. For example, `http://example.com/a/a/a/a/a/foo.html`.

PredicatedDecideRule This DecideRule applies a configured decision only if a test evaluates to true.

ExternalGeoLocationDecideRule This DecideRule accepts a URI if it is located in a particular country.

FetchStatusDecideRule This DecideRule applies the configured decision to any URI that has a fetch staus equal to the "target-status" setting.

HasViaDecideRule This DecideRule applies the configured decision to any URI that has a "via." A via is any URI that is a seed or some kind of mid-crawl addition.

HopCrossesAssignmentLevelDomainDecideRule This DecideRule applies the configured decision to any URI that differs in the portion of its hostname/domain that is assigned/sold by registrars. The portion is referred to as the "assignment-level-domain" (ALD).

IdenticalDigestDecideRule This DecideRule applies the configured decision to any URI whose prior-history content-digest matches the latest fetch.

MatchesListRegexDecideRule This DecideRule applies the configured decision to any URI that matches the supplied regular expressions.

NotMatchesListRegexDecideRule This DecideRule applies the configured decision to any URI that does not match the supplied regular expressions.

MatchesRegexDecideRule This DecideRule applies the configured decision to any URI that matches the supplied regular expression.

ClassKeyMatchesRegexDecideRule This DecideRule applies the configured decision to any URI class key that matches the supplied regular expression. A URI class key is a string that specifies the name of the Frontier queue into which a URI should be placed.

ContentTypeMatchesRegexDecideRule This DecideRule applies the configured decision to any URI whose content-type is present and matches the supplied regular expression. The regular expression must match the full content-type sequence. Ex.: `s/application/javascript;charset=UTF-8/^application\/javascript.*$/g`; `s/text/html/^.*\/html.*$/g`

ContentTypeNotMatchesRegexDecideRule This DecideRule applies the configured decision to any URI whose content-type does not match the supplied regular expression.

FetchStatusMatchesRegexDecideRule This DecideRule applies the configured decision to any URI that has a fetch status that matches the supplied regular expression.

FetchStatusNotMatchesRegexDecideRule This DecideRule applies the configured decision to any URI that has a fetch status that does not match the suppllied regular expression.

HopsPathMatchesRegexDecideRule This DecideRule applies the configured decision to any URI whose "hops-path" matches the supplied regular expression. The hops-path is a string that consists of characters representing the path that was taken to access the URI. An example of a hops-path is "LLXE".

MatchesFilePatternDecideRule This DecideRule applies the configured decision to any URI whose suffix matches the supplied regular expression.

NotMatchesFilePatternDecideRule This DecideRule applies the configured decision to any URI whose suffix does not match the supplied regular expression.

NotMatchesRegexDecideRule This DecideRule applies the configured decision to any URI that does not match the supplied regular expression.

NotExceedsDocumentLengthThresholdDecideRule This DecideRule applies the configured decision to any URI whose content-length does not exceed the configured threshold. The content-length comes from either the HTTP header or the actual downloaded content length of the URI. As of Heritrix 3.1, this rule has been renamed to ResourceNoLongerThanDecideRule.

ExceedsDocumentLengthThresholdDecideRule This DecideRule applies the configured decision to any URI whose content length exceeds the configured threshold. The content-length comes from either the HTTP header or the actual downloaded content length of the URI. As of Heritrix 3.1, this rule has been renamed to ResourceLongerThanDecideRule.

SurtPrefixedDecideRule This DecideRule applies the configured decision to any URI (expressed in SURT form) that begins with one of the prefixes in the configured set. This DecideRule returns true when the prefix of a given URI matches any of the listed SURTs. The list of SURTs may be configured in different ways: the surtsSourceFile parameter specifies a file to read the SURTs list from. If seedsAsSurtPrefixes parameter is set to true, SurtPrefixedDecideRule adds all seeds to the SURTs list. If alsoCheckVia property is set to true (default false), SurtPrefixedDecideRule will also consider Via URIs in the match. As of Heritrix 3.1, the "surtsSource" parameter may be any ReadSource, such as a ConfigFile or a ConfigString. This gives the SurtPrefixedDecideRule the flexibility of the TextSeedModule bean's "textSource" property.

NotSurtPrefixedDecideRule This DecideRule applies the configured decision to any URI (expressed in SURT form) that does not begin with one of the prefixes in the configured set.

OnDomainsDecideRule This DecideRule applies the configured decision to any URI that is in one of the domains of the configured set.

NotOnDomainsDecideRule This DecideRule applies the configured decision to any URI that is not in one of the domains of the configured set.

OnHostsDecideRule This DecideRule applies the configured decision to any URI that is in one of the hosts of the configured set.

NotOnHostsDecideRule This DecideRule applies the configured decision to any URI that is not in one of the hosts of the configured set.

ScopePlusOneDecideRule This DecideRule applies the configured decision to any URI that is one level beyond the configured scope.

TooManyHopsDecideRule This DecideRule rejects any URI whose total number of hops is over the configured threshold.

TooManyPathSegmentsDecideRule This DecideRule rejects any URI whose total number of path-segments is over the configured threshold. A path-segment is a string in the URI separated by a "/" character, not including the first "//".

TransclusionDecideRule This DecideRule accepts any URI whose path-from-seed ends in at least one non-navlink hop. A navlink hop is represented by an "L". Also, the number of non-navlink hops in the path-from-seed cannot exceed the configured value.

PrerequisiteAcceptDecideRule This DecideRule accepts all "prerequisite" URIs. Prerequisite URIs are those whose hops-path has a "P" in the last position.

RejectDecideRule This DecideRule rejects any URI.

ScriptedDecideRule This DecideRule applies the configured decision to any URI that passes the rules test of a JSR-223 script. The script source must be a one-argument function called decisionFor." The function returns the appropriate DecideResult. Variables available to the script include object (the object to be evaluated, such as a URI), "self" (the ScriptDecideRule instance), and context (the crawl's ApplicationContext, from which all named crawl beans are reachable).

SeedAcceptDecideRule This DecideRule accepts all "seed" URIs (those for which isSeed is true).

3.2.2 DecideRuleSequence Logging

Enable `FINEST` logging on the class `org.archive.crawler.deciderules.DecideRuleSequence` to watch each DecideRule's evaluation of the processed URI. This can be done in the `logging.properties` file:

```
org.archive.modules.deciderules.DecideRuleSequence.level = FINEST
```

in conjunction with the `-Dsysprop` VM argument,

3.3 Frontier

3.3.1 Politeness

A combination of several settings control the politeness of the Frontier. It is important to note that at any given time only one URI from any given host is processed. The following politeness rules impose additional wait time between the end of processing one URI and the start of the next one.

delayFactor This setting imposes a delay between the fetching of URIs from the same host. The delay is a multiple of the amount of time it took to fetch the last URI downloaded from the host. For example, if it took 800 milliseconds to fetch the last URI from a host and the `delayFactor` is 5 (a very high value), then the Frontier will wait 4000 milliseconds (4 seconds) before allowing another URI from that host to be processed.

maxDelayMs This setting imposes a maximum upper limit on the wait time created by the `delayFactor`. If set to 1000 milliseconds, then the maximum delay between URI fetches from the same host will never exceed this value.

minDelayMs This setting imposes a minimum limit on politeness. It takes precedence over the value calculated by the `delayFactor`. For example, the value of `minDelayMs` can be set to 100 milliseconds. If the delayFactor only generates a 20 millisecond wait, the value of minDelayMs will override it and the URI fetch will be delayed for 100 milliseconds.

```
<bean id="disposition" class="org.archive.crawler.postprocessor.DispositionProcessor">
  <property name="delayFactor" value="5.0" />
  <property name="maxDelayMs" value="30000" />
  <property name="minDelayMs" value="3000" />
</bean>
```

3.3.2 Retry Policy

The Frontier can be used to limit the number of fetch retries for a URI. Heritrix will retry fetching a URI because the initial fetch error may be a transitory condition.

maxRetries This setting limits the number of fetch retries attempted on a URI due to transient errors.

retryDelaySeconds This setting determines how long the wait period is between retries.

```
<bean id="frontier" class="org.archive.crawler.frontier.BdbFrontier">
  <property name="retryDelaySeconds" value="900" />
  <property name="maxRetries" value="30" />
</bean>
```

3.3.3 Bandwidth Limits

The Frontier allows the user to limit bandwidth usage. This is done by holding back URIs when bandwidth usage has exceeded certain limits. Because bandwidth usage limitations are calculated over a period of time, there can still be spikes in usage that greatly exceed the limits.

maxPerHostBandwidthUsageKbSec This setting limits the maximum bandwidth to use for any host. This setting limits the load placed by Heritrix on the host. It is therefore a politeness setting.

```
<bean id="disposition" class="org.archive.crawler.postprocessor.
↪DispositionProcessor">
  <property name="maxPerHostBandwidthUsageKbSec" value="500" />
</bean>
```

3.3.4 Extractor Parameters

The Frontier's behavior with regard to link extraction can be controlled by the following parameters.

extract404s This setting allows the operator to avoid extracting links from 404 (Not Found) pages. The default is true, which maintains the pre-3.1 behavior of extracting links from 404 pages.

```
<bean id="frontier" class="org.archive.crawler.frontier.BdbFrontier">
<property name="extract404s" value="true" />
</bean>
```

extractIndependently This setting encourages extractor processors to always perform their best-effort extraction, even if a previous extractor has marked a URI as already-handled. Set the value to true for best-effort extraction. The default is false, which maintains the pre-3.1 behavior.

```
<bean id="frontier" class="org.archive.crawler.frontier.BdbFrontier">
   <property name="extractIndependently" value="false" />
</bean>
```

3.4 Sheets (Site-specific Settings)

Sheets provide the ability to replace default settings on a per domain basis. Sheets are collections of overrides. They contain alternative values for object properties that should apply in certain contexts. The target is specified as an arbitrarily-long property-path, which is a string describing how to access the property starting from a beanName in a BeanFactory.

Sheets allow settings to be overlaid with new values that apply by top level domains (com, net, org, etc), by second-level domains (yahoo.com, archive.org, etc.), by subdomains (crawler.archive.org, tech.groups.yahoo.com, etc.) , and leading URI paths (directory.google.com/Top/Computers/, etc.). There is no limit for how long the domain/path prefix which specifies overlays can go; the SURT Prefix syntax is used.

Creating a new sheet involves configuring the `crawler-beans.cxml` file, which contains the Spring configuration of a job.

For example, if you have explicit permission to crawl certain domains without the usual polite rate-limiting, then a Sheet can be used to create a less polite crawling policy that is associated with a few such target domains. The configuration of such a Sheet for the domains example.com and example1.com are shown below. This example allows up to 5 parallel outstanding requests at a time (rather than the default 1), and eliminates any usual pauses between requests.

> **Warning:** Unless a target site has given you explicit permission to crawl extra-aggressively, the typical Heritrix defaults, which limit the crawler to no more than one outstanding request at a time, with multiple-second waits between requests, and longer waits when the site is responding more slowly, are the safest course. Less-polite crawling can result in your crawler being blocked entirely by webmasters.
>
> Finally, even with permission, be sure your crawler's User-Agent string includes a link to valid crawl-operator contact information so you can be alerted to, and correct, any unintended side-effects.

```
<bean id="sheetOverlaysManager" autowire="byType" class="org.archive.crawler.spring.
↪SheetOverlaysManager">
</bean>

<bean class='org.archive.crawler.spring.SurtPrefixesSheetAssociation'>
  <property name='surtPrefixes'>
    <list>
      <value>http://(com,example,www,)/</value>
      <value>http://(com,example1,www,)/</value>
    </list>
  </property>
  <property name='targetSheetNames'>
    <list>
      <value>lessPolite</value>
    </list>
  </property>
</bean>

<bean id='lessPolite' class='org.archive.spring.Sheet'>
  <property name='map'>
```

(continues on next page)

(continued from previous page)

```
    <map>
      <entry key='disposition.delayFactor' value='0.0'/>
      <entry key='disposition.maxDelayMs' value='0'/>
      <entry key='disposition.minDelayMs' value='0'/>
      <entry key='queueAssignmentPolicy.parallelQueues' value='5'/>
    </map>
  </property>
</bean>
```

3.5 Authentication and Cookies

Heritrix can crawl sites behind login by using HTTP authentication, submitting a form or by loading cookies from a file.

3.5.1 Credential Store

Credentials can be added so that Heritrix can gain access to areas of web sites requiring authentication. Credentials need to listed in a CredentialStore.

```
<bean id="credentialStore" class="org.archive.modules.credential.CredentialStore">
  <property name="credentials">
    <map>
      <entry key="exampleHttpCredential" value-ref="exampleHttpCredential"/>
      <entry key="exampleFormCredential" value-ref="exampleFormCredential"/>
    </map>
  </property>
</bean>
```

To enable text console logging of authentication interactions, set the FetchHTTP and PreconditionEnforcer log levels to fine in `conf/logging.properties`:

3.5.2 HTTP Basic and Digest Authentication

In response to a 401 Unauthorized response code Heritrix will do a lookup of a key based on the domain and authentication realm in its CredentialStore. If a match is found, then the credential is loaded into the CrawlURI and the CrawlURI is marked for immediate retry.

When the CrawlURI is retried, the found credentials are added to the request. If the request succeeds with a 200 response code, the credentials are promoted to the CrawlServer and all subsequent requests made against the CrawlServer will preemptively volunteer the credential. If the credential fails with a 401 response code, the URI is no longer retried.

The configured domain should be of the form "hostname:port" unless the port is 80 in which case it must be omitted. For HTTPS URLs without an explicit port use port 443.

```
<bean id="exampleHttpCredential" class="org.archive.modules.credential.
↪HttpAuthenticationCredential">
  <property name="domain" value="www.example.org:443"/>
  <property name="realm" value="myrealm"/>
  <property name="login" value="user"/>
  <property name="password" value="secret"/>
</bean>
```

3.5.3 HTML Form Authentication

Heritrix can be configured to submit credentials to a HTML form using a GET or POST request.

```
<bean id="exampleFormCredential" class="org.archive.modules.credential.
↪HtmlFormCredential">
  <property name="domain" value="example.com"/>
  <property name="loginUri" value="http://example.com/login"/>
  <property name="formItems">
    <map>
      <entry key="login" value="user"/>
      <entry key="password" value="secret"/>
      <entry key="submit" value="submit"/>
    </map>
  </property>
</bean>
```

domain The domain should be of the form “hostname:port” unless the port is 80 in which case it must be omitted. For HTTPS URLs without an explicit port use port 443.

login-uri A relative or absolute URI to which the HTML Form submits. It is not necessarily the page that contains the HTML Form; rather it is the ACTION URI the to which the form submits.

form-items Form-items are a listing of HTML Form key/value pairs. The submit button usually must be included in the form-items.

Note: There is currently no support for successfully submitting forms with dynamic fields whose required name or value changes for each visitor (such as CSRF tokens).

For a site with an HTML Form credential, a login is performed against all listed HTML Form credential login-uris after the DNS and robots.txt preconditions are fulfilled. The crawler will only view sites that have HTML Form credentials from a logged-in perspective. There is no current way for a single Heritrix job to crawl a site in an unauthenticated state and then re-crawl the site in an authenticated state. (You would have to do this in two separately-configured job launches.)

The form login is only run once. Heritrix continues crawling regardless of whether the login succeeds. There is no way of telling Heritrix to retry authentication if the first attempt is not successful. Neither is there a means for the crawler to report success or failed authentications. The crawl operator should examine the logs to determine whether authentication succeeded.

3.5.4 Loading Cookies

Heritrix can be configured to load a set of cookies from a file. This can be used for example to crawl a website behind a login form by manually logging in through the browser and then copying the session cookie.

To enable loading of cookies set the cookiesLoadFile property of the BdbCookieStore bean to a ConfigFile:

```
<bean id="cookieStore" class="org.archive.modules.fetcher.BdbCookieStore">
  <property name="cookiesLoadFile">
     <bean class="org.archive.spring.ConfigFile">
       <property name="path" value="cookies.txt" />
     </bean>
  </property>
</bean>
```

The cookies.txt should be in the 7-field tab-separated Netscape cookie file format. An entry might look like:

```
www.example.org FALSE / FALSE 1311699995 sessionid 60ddb868550a
```

Table 1: Cookie file tab-separated fields

1	DOMAIN	The domain that created and has access to the cookie.
2	FLAG	A TRUE or FALSE value indicating if subdomains within the given domain can access the cookie.
3	PATH	The path within the domain that the cookie is valid for.
4	SECURE	A TRUE or FALSE value indicating if the cookie should be sent over HTTPS only.
5	EXPIRATION	Expiration time in seconds since 1970-01-01T00:00:00Z, or -1 for no expiration
6	NAME	The name of the cookie.
7	VALUE	The value of the cookie.

3.6 Other Protocols

In addition to HTTP Heritrix can be configured to fetch resources using several other internet protocols.

3.6.1 FTP

Heritrix supports crawling FTP sites. Seeds should be added in the following format: `ftp://sftp.example.org/directory.

The FetchFTP bean needs to be defined:

```
<bean id="fetchFTP" class="org.archive.modules.fetcher.FetchFTP">
  <!-- <property name="username" value="anonymous" /> -->
  <!-- <property name="password" value="password" /> -->
  <!-- <property name="extractFromDirs" value="true" /> -->
  <!-- <property name="extractParent" value="true" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
  <!-- <property name="maxLengthBytes" value="0" /> -->
  <!-- <property name="maxFetchKBSec" value="0" /> -->
  <!-- <property name="timeoutSeconds" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
</bean>
```

and added to the FetchChain:

```
<bean id="fetchProcessors" class="org.archive.modules.FetchChain">
  <property name="processors">
    <list>...
    <ref bean="fetchFTP"/>
    ...
   </list>
  </property>
</bean>
```

3.6.2 SFTP

An optional fetcher for SFTP is provided. Seeds should be added in the following format:`sftp://sftp.example.org/directory`.

The FetchSFTP bean needs to be defined:

```
<bean id="fetchSFTP" class="org.archive.modules.fetcher.FetchSFTP">
  <!-- <property name="username" value="anonymous" /> -->
  <!-- <property name="password" value="password" /> -->
  <!-- <property name="extractFromDirs" value="true" /> -->
  <!-- <property name="extractParent" value="true" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
  <!-- <property name="maxLengthBytes" value="0" /> -->
  <!-- <property name="maxFetchKBSec" value="0" /> -->
  <!-- <property name="timeoutSeconds" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
</bean>
```

and added to the FetchChain:

```
<bean id="fetchProcessors" class="org.archive.modules.FetchChain">
  <property name="processors">
    <list>
      ...
      <ref bean="fetchSFTP"/>
      ...
    </list>
  </property>
</bean>
```

3.6.3 WHOIS

An optional fetcher for domain WHOIS data is provided. A small set of well-established WHOIS servers are preconfigured. The fetcher uses an ad-hoc/intuitive interpretation of a 'whois:' scheme URI.

Define the fetchWhois bean:

```
<bean id="fetchWhois" class="org.archive.modules.fetcher.FetchWhois">
  <property name="specialQueryTemplates">
    <map>
      <entry key="whois.verisign-grs.com" value="domain %s" />
      <entry key="whois.arin.net" value="z + %s" />
      <entry key="whois.denic.de" value="-T dn %s" />
    </map>
  </property>
</bean>
```

and add it to the FetchChain:

```
<bean id="fetchProcessors" class="org.archive.modules.FetchChain">
  <property name="processors">
    <list>
      ...
      <ref bean="fetchWhois"/>
      ...
```

(continues on next page)

(continued from previous page)

```
    </list>
  </property>
</bean>
```

To configure a whois seed, enter the seed in the following format: `whois://hostname/path`. For example, `whois://archive.org`. The whois fetcher will attempt to resolve each host that the crawl encounters using the topmost assigned domain and the ip address of the url crawled. So if you crawl `http://www.archive.org/details/texts`, the whois fetcher will attempt to resolve `whois:archive.org` and `whois:207.241.224.2`.

At this time, whois functionality is experimental. The fetchWhois bean is commented out in the default profile.

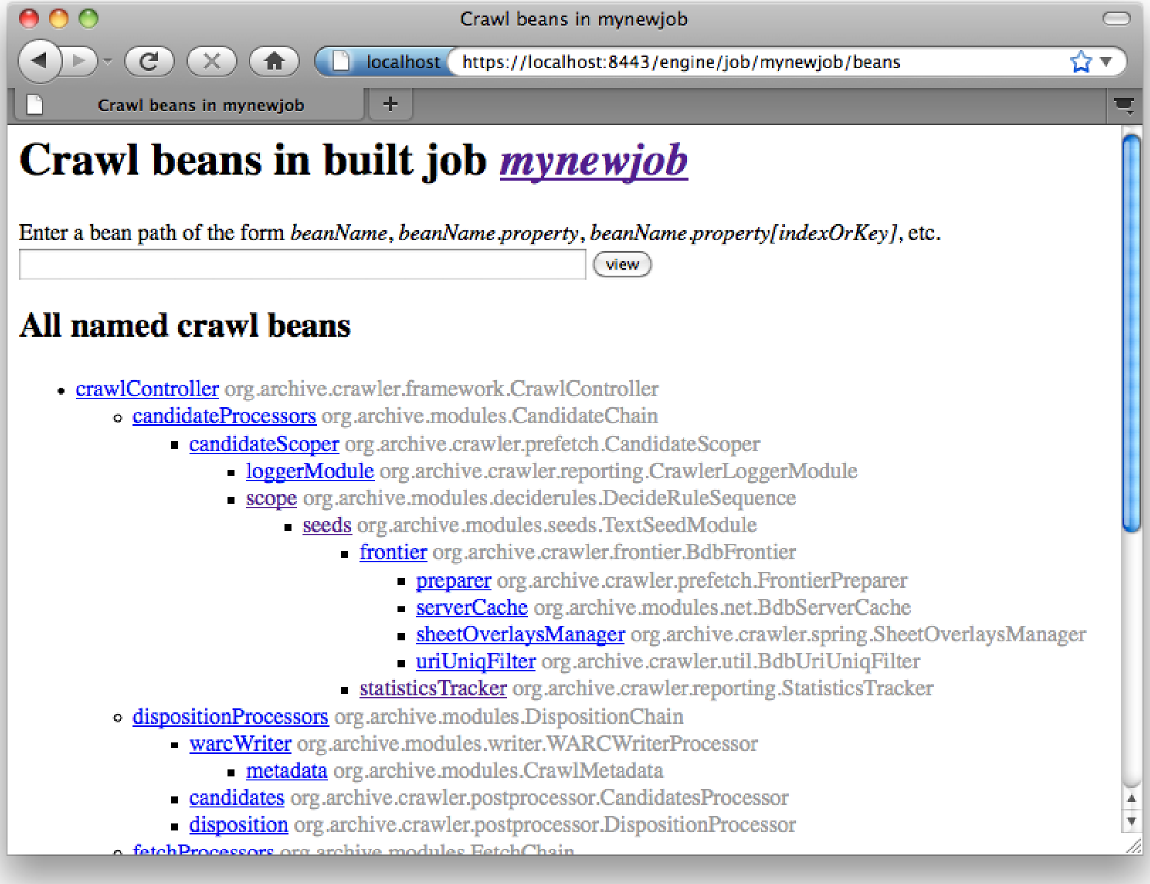
3.7 Modifying a Running Job

While changing a job's XML configuration normally requires relaunching it, some settings can be modified while the crawl is running. This is done through the *Browse Beans* or the *Scripting Console* link on the job page. The Bean Browser allows you to edit runtime properties of beans. You can also use the scripting console to programmatically edit a running job.

If changing a non-atomic value, it is a good practice to pause the crawl prior to making the change, as some modifications to composite configuration entities may not occur in a thread-safe manner. An example of a non-atomic change is adding a new Sheet.

3.7.1 Browse Beans

The WUI provides a way to view and edit the Spring beans that make up a crawl configuration. It is important to note that changing the crawl configuration using the Bean Browser will not update the `crawler-beans.cxml` file. Thus, changing settings with the Bean Browser is not permanent. The Bean Browser should only by used to change the settings of a running crawl. To access the Bean Browser click on the Browse Beans link from the jobs page. The hierarchy of Spring beans will be displayed.



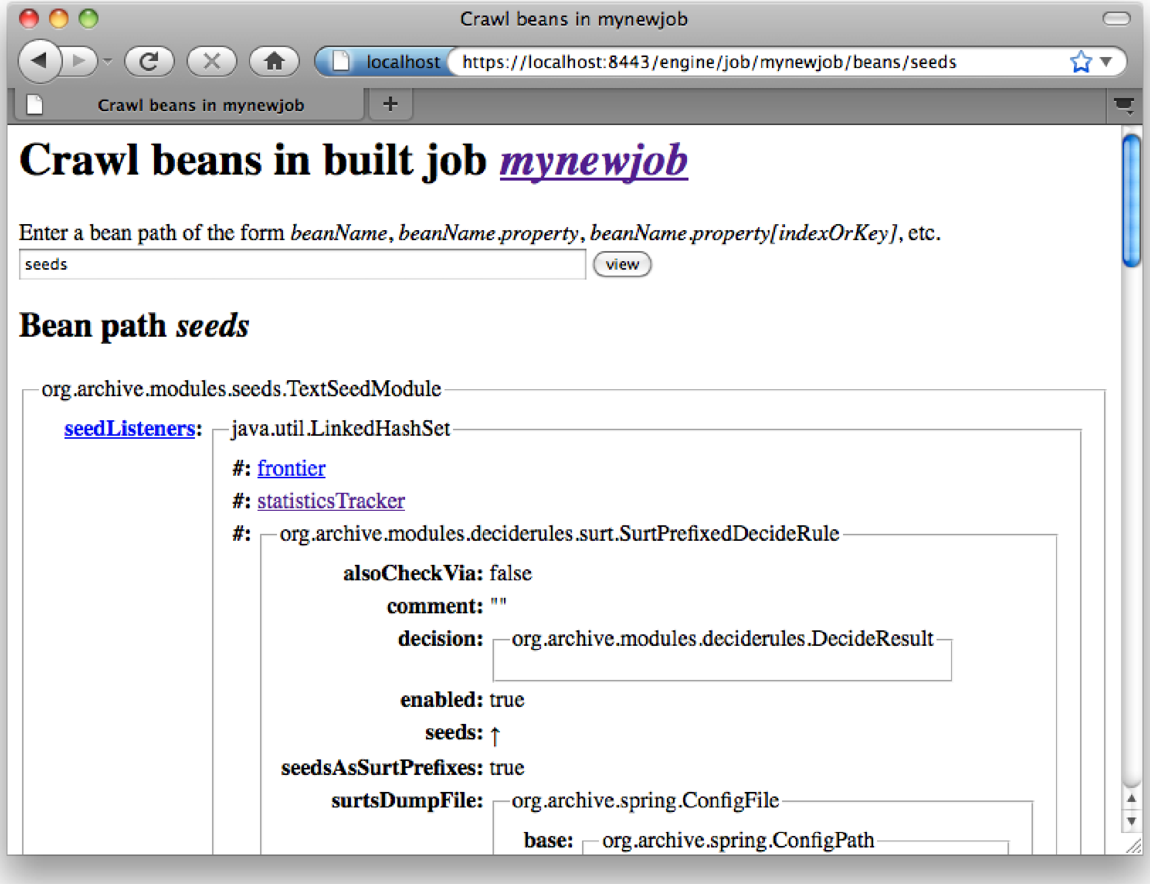
Crawl beans in built job *mynewjob*

Enter a bean path of the form *beanName*, *beanName.property*, *beanName.property[indexOrKey]*, etc.

All named crawl beans

- crawlController org.archive.crawler.framework.CrawlController
  - candidateProcessors org.archive.modules.CandidateChain
    - candidateScoper org.archive.crawler.prefetch.CandidateScoper
      - loggerModule org.archive.crawler.reporting.CrawlerLoggerModule
      - scope org.archive.modules.deciderules.DecideRuleSequence
        - seeds org.archive.modules.seeds.TextSeedModule
          - frontier org.archive.crawler.frontier.BdbFrontier
            - preparer org.archive.crawler.prefetch.FrontierPreparer
            - serverCache org.archive.modules.net.BdbServerCache
            - sheetOverlaysManager org.archive.crawler.spring.SheetOverlaysManager
            - uriUniqFilter org.archive.crawler.util.BdbUriUniqFilter
          - statisticsTracker org.archive.crawler.reporting.StatisticsTracker
  - dispositionProcessors org.archive.modules.DispositionChain
    - warcWriter org.archive.modules.writer.WARCWriterProcessor
      - metadata org.archive.modules.CrawlMetadata
    - candidates org.archive.crawler.postprocessor.CandidatesProcessor
    - disposition org.archive.crawler.postprocessor.DispositionProcessor
  - fetchProcessors org.archive.modules.FetchChain

You can drill down on individual beans by clicking on them. The example below shows the display after clicking on the seeds bean.



Crawl beans in built job *mynewjob*

Enter a bean path of the form *beanName*, *beanName.property*, *beanName.property[indexOrKey]*, etc.

seeds

Bean path *seeds*

org.archive.modules.seeds.TextSeedModule

seedListeners: java.util.LinkedHashSet

#: frontier

#: statisticsTracker

#: org.archive.modules.deciderules.surt.SurtPrefixedDecideRule

alsoCheckVia: false

comment: ""

decision: org.archive.modules.deciderules.DecideResult

enabled: true

seeds: ↑

seedsAsSurtPrefixes: true

surtsDumpFile: org.archive.spring.ConfigFile

base: org.archive.spring.ConfigPath

3.7.2 Scripting Console

[This section to be written. For now see the Heritrix3 Useful Scripts wiki page.]

3.7.3 Configuring HTTP Proxies

There are two options to specify a proxy for crawling.

The command line options `--proxy-host` and `--proxy-port` can be used to define a proxy for all jobs. If only the `--proxy-host` option is given, a default value of 8000 is used for the proxy port. These proxy settings are also used when connecting to a "DNS-over-HTTP" server (see the *section on DNS-over-HTTP* below).

Alternatively one can define a per-job proxy via a the `httpProxyHost` and `httpProxyPort` properties of the `fetchHttp` bean. These settings, if both defined, will overwrite the global options. These setting also allow for a user and password in the `httpProxyUser` and `httpProxyPassword` properties, which the global options do not support, due to incompatibilities of the different supported Java versions.

Also the optional "SOCKS5" proxy documented in the next section is used on a per-job basis; there are currently no global options to define it.

3.7.4 Configuring SOCKS5 Proxy

An optional configuration value to route Heritrix crawler traffic through a SOCKS5 proxy. This will override any set HTTP proxy configuration. It is facilitated by extending the *org.archive.modules.fetcher.FetchHTTP* bean with *socksProxyHost* and *socksProxyPort* values, as in the example below:

```
<bean class="org.archive.modules.fetcher.FetchHTTP" id="fetchHttp">
    <!--  ... -->
    <property name="socksProxyHost" value="127.0.0.1"/>
    <property name="socksProxyPort" value="24000"/>
</bean>
```

3.7.5 Configuring DNS over HTTP (DoH)

If the local DNS on the server running Heritrix is not able to resolve the DNS names of the crawled sites, e.g. because the server is sitting behind an enterprise firewall and can only resolve names of the local network, then using DNS-over-HTTP (DoH) might be an alternative to fetch DNS information.

To activate this, one needs to set the `dnsOverHttpServer` setting of the `fetchDns` bean to the URL of an DoH server. If one has configured a global proxy via the `--proxy-host` and `--proxy-port` command line options, these proxy settings will be used to contact the DoH server as well. However due to limitation of the library in use, username and password information for the proxy are not supported. Also any per-job defined proxy settings in the `fetchHttp` bean are not used when contacting the DoH server.

As the implementation relies on the corresponding client in the “dnsjava” library, which is currently labeled as experimental, this option comes with some limitations:

- If you use Java 11 then due to a well known bug it will not close connections to the DoH server unless Heritrix shuts down. As the DoH server might not accept new connections after some limits while these connections are still open, it is not recommended to use this feature when running Heritrix with Java 11.
- For other Java versions, the connection to the DoH server will be closed when the garbage collector runs. Depending on the garbage collector used this will cause a delay of anything between a few seconds and several minutes before closing the connection. Also note that if the garbage collector is explicitely triggered via the Heritrix UI one needs to add the `-XX:-DisableExplicitGC` option in the `JAVA_OPTS` for Java versions 13 and up as otherwise this action has no effect.

Without making a recommendation the following DoH servers have been tested with the DoH feature:

- https://dns.google/dns-query
- https://cloudflare-dns.com/dns-query

However servers implementing the official RFC 8484 specification unfortunately do not work with the current implementation. This includes e.g. the following server:

- https://dns.digitale-gesellschaft.ch/dns-query

This limitation might be overcome by a newer version of the “dnsjava” library.

CHAPTER 4

Bean Reference

Note: This reference is a work in progress and does not yet cover all available beans. For a more complete list of Heritrix beans please refer to the javadoc.

4.1 Core Beans

4.1.1 ActionDirectory

Directory watched for new files. Depending on their extension, will process with regard to current crawl, and rename with a datestamp into the 'done' directory. Currently supports: - .seeds(.gz) add each URI found in file as a new seed (to be crawled if not already; to affect scope if appropriate). - (.s).recover(.gz) treat as traditional recovery log: consider all 'Fs'-tagged lines included, then try-rescheduling all 'F+'-tagged lines. (If ".s." present, try scoping URIs before including/scheduling.) - (.s).include(.gz) add each URI found in a recover-log like file (regardless of its tagging) to the frontier's alreadyIncluded filter, preventing them from being recrawled. ('.s.' indicates to apply scoping.) - (.s).schedule(.gz) add each URI found in a recover-log like file (regardless of its tagging) to the frontier's queues. ('.s.' indicates to apply scoping.)

Future support planned: - .robots: invalidate robots ASAP - (?) .block: block-all on named site(s) - .overlay: add new overlay settings - .js .rb .bsh .rb etc - execute arbitrary script (a la ScriptedProcessor)

```
<bean id="actionDirectory" class="org.archive.crawler.framework.ActionDirectory">
  <!-- <property name="initialDelaySeconds" value="10" /> -->
  <!-- <property name="delaySeconds" value="30" /> -->
  <!-- <property name="actionDir" value="" /> -->
  <!-- <property name="doneDir" value="" /> -->
  <!-- <property name="applicationContext" value="" /> -->
  <!-- <property name="seeds" value="" /> -->
  <!-- <property name="frontier" value="" /> -->
</bean>
```

initialDelaySeconds (int) how long after crawl start to first scan action directory

delaySeconds (int) delay between scans of actionDirectory for new files

actionDir (ConfigPath)

doneDir (ConfigPath)

applicationContext (ApplicationContext)

seeds (SeedModule)

frontier (Frontier) autowired frontier for actions

4.1.2 BdbCookieStore

Cookie store using bdb for storage. Cookies are stored in a SortedMap keyed by #sortableKey(Cookie), so they are grouped together by domain. #cookieStoreFor(String) returns a facade whose CookieStore#getCookies() returns a list of cookies limited to the supplied host and parent domains, if applicable.

```
<bean id="bdbCookieStore" class="org.archive.modules.fetcher.BdbCookieStore">
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

bdbModule (BdbModule)

recoveryCheckpoint (Checkpoint)

4.1.3 BdbFrontier

A Frontier using several BerkeleyDB JE Databases to hold its record of known hosts (queues), and pending URIs.

```
<bean id="bdbFrontier" class="org.archive.crawler.frontier.BdbFrontier">
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="beanName" value="" /> -->
  <!-- <property name="dumpPendingAtClose" value="false" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

bdbModule (BdbModule)

beanName (String)

dumpPendingAtClose (boolean)

recoveryCheckpoint (Checkpoint)

4.1.4 BdbModule

Utility module for managing a shared BerkeleyDB-JE environment

```
<bean id="bdbModule" class="org.archive.bdb.BdbModule">
  <!-- <property name="dir" value="" /> -->
  <!-- <property name="cachePercent" value="1" /> -->
  <!-- <property name="cacheSize" value="1" /> -->
  <!-- <property name="useSharedCache" value="true" /> -->
  <!-- <property name="maxLogFileSize" value="10000000" /> -->
  <!-- <property name="expectedConcurrency" value="64" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="cleanerThreads" value="" /> -->
  <!-- <property name="evictorCoreThreads" value="1" /> -->
  <!-- <property name="evictorMaxThreads" value="1" /> -->
  <!-- <property name="useHardLinkCheckpoints" value="true" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

dir (ConfigPath)

cachePercent (int)

cacheSize (int)

useSharedCache (boolean)

maxLogFileSize (long)

expectedConcurrency (int) Expected number of concurrent threads; used to tune nLockTables according to JE FAQ http://www.oracle.com/technology/products/berkeley-db/faq/je_faq.html#33

cleanerThreads (int)

evictorCoreThreads (int) Configure the number of evictor threads (-1 means use the default) https://docs.oracle.com/cd/E17277_02/html/java/com/sleepycat/je/EnvironmentConfig.html#EVICTOR_CORE_THREADS

evictorMaxThreads (int) Configure the maximum number of evictor threads (-1 means use the default) https://docs.oracle.com/cd/E17277_02/html/java/com/sleepycat/je/EnvironmentConfig.html#EVICTOR_MAX_THREADS

useHardLinkCheckpoints (boolean) Whether to use hard-links to log files to collect/retain the BDB log files needed for a checkpoint. Default is true. May not work on Windows (especially on pre-NTFS filesystems). If false, the BDB 'je.cleaner.expunge' value will be set to 'false', as well, meaning BDB will *not* delete obsolete JDB files, but only rename the '.DEL'. They will have to be manually deleted to free disk space, but .DEL files referenced in any checkpoint's 'jdbfiles.manifest' should be retained to keep the checkpoint valid.

recoveryCheckpoint (Checkpoint)

4.1.5 BdbServerCache

ServerCache backed by BDB big maps; the usual choice for crawls.

```
<bean id="bdbServerCache" class="org.archive.modules.net.BdbServerCache">
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

bdbModule (BdbModule)

recoveryCheckpoint (Checkpoint)

4.1.6 BdbUriUniqFilter

A BDB implementation of an AlreadySeen list. This implementation performs adequately without blowing out the heap. See AlreadySeen. Makes keys that have URIs from same server close to each other. Mercator and 2.3.5 'Elminating Already-Visited URLs' in 'Mining the Web' by Soumen Chakrabarti talk of a two-level key with the first 24 bits a hash of the host plus port and with the last 40 as a hash of the path. Testing showed adoption of such a scheme halving lookup times (Tutilhis implementation actually concatenates scheme + host in first 24 bits and path + query in trailing 40 bits).

```
<bean id="bdbUriUniqFilter" class="org.archive.crawler.util.BdbUriUniqFilter">
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="beanName" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

bdbModule (BdbModule)

beanName (String)

recoveryCheckpoint (Checkpoint)

4.1.7 CheckpointService

Executes checkpoints, and offers convenience methods for enumerating available Checkpoints and injecting a recovery-Checkpoint after build and before launch (setRecoveryCheckpointByName). Offers optional automatic checkpointing at a configurable interval in minutes.

```
<bean id="checkpointService" class="org.archive.crawler.framework.CheckpointService">
  <!-- <property name="checkpointsDir" value="" /> -->
  <!-- <property name="checkpointIntervalMinutes" value="1" /> -->
  <!-- <property name="forgetAllButLatest" value="false" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
  <!-- <property name="crawlController" value="" /> -->
  <!-- <property name="applicationContext" value="" /> -->
  <!-- <property name="recoveryCheckpointByName" value="" /> -->
</bean>
```

checkpointsDir (ConfigPath) Checkpoints directory

checkpointIntervalMinutes (long) Period at which to create automatic checkpoints; -1 means no auto checkpointing.

forgetAllButLatest (boolean) True to save only the latest checkpoint, false to save all of them. Default is false.

recoveryCheckpoint (Checkpoint)

crawlController (CrawlController)

applicationContext (ApplicationContext)

recoveryCheckpointByName (String) Given the name of a valid checkpoint subdirectory in the checkpoints directory, create a Checkpoint instance, and insert it into all Checkpointable beans.

4.1.8 CrawlController

CrawlController collects all the classes which cooperate to perform a crawl and provides a high-level interface to the running crawl. As the "global context" for a crawl, subcomponents will often reach each other through the CrawlController.

```
<bean id="crawlController" class="org.archive.crawler.framework.CrawlController">
  <!-- <property name="applicationContext" value="" /> -->
  <!-- <property name="metadata" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="frontier" value="" /> -->
  <!-- <property name="scratchDir" value="" /> -->
  <!-- <property name="statisticsTracker" value="" /> -->
  <!-- <property name="seeds" value="" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="fetchChain" value="" /> -->
  <!-- <property name="dispositionChain" value="" /> -->
  <!-- <property name="candidateChain" value="" /> -->
  <!-- <property name="maxToeThreads" value="" /> -->
  <!-- <property name="runWhileEmpty" value="false" /> -->
  <!-- <property name="pauseAtStart" value="true" /> -->
  <!-- <property name="recorderOutBufferBytes" value="" /> -->
  <!-- <property name="recorderInBufferBytes" value="" /> -->
  <!-- <property name="loggerModule" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

applicationContext (ApplicationContext)

metadata (CrawlMetadata)

serverCache (ServerCache)

frontier (Frontier) The frontier to use for the crawl.

scratchDir (ConfigPath) Scratch directory for temporary overflow-to-disk

statisticsTracker (StatisticsTracker) Statistics tracking modules. Any number of specialized statistics trackers that monitor a crawl and write logs, reports and/or provide information to the user interface.

seeds (SeedModule)

fetchChain (FetchChain) Fetch chain

dispositionChain (DispositionChain) Disposition chain

candidateChain (CandidateChain) Candidate chain

maxToeThreads (int) Maximum number of threads processing URIs at the same time.

runWhileEmpty (boolean) whether to keep running (without pause or finish) when frontier is empty

pauseAtStart (boolean) whether to pause at crawl start

recorderOutBufferBytes (int) Size in bytes of in-memory buffer to record outbound traffic. One such buffer is reserved for every ToeThread.

recorderInBufferBytes (int) Size in bytes of in-memory buffer to record inbound traffic. One such buffer is reserved for every ToeThread.

loggerModule (CrawlerLoggerModule)

recoveryCheckpoint (Checkpoint)

4.1.9 CrawlerLoggerModule

Module providing all expected whole-crawl logging facilities

```
<bean id="crawlerLoggerModule" class="org.archive.crawler.reporting.
↪CrawlerLoggerModule">
  <!-- <property name="path" value="" /> -->
  <!-- <property name="logExtraInfo" value="false" /> -->
  <!-- <property name="crawlLogPath" value="" /> -->
  <!-- <property name="alertsLogPath" value="" /> -->
  <!-- <property name="progressLogPath" value="" /> -->
  <!-- <property name="uriErrorsLogPath" value="" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="runtimeErrorsLogPath" value="" /> -->
  <!-- <property name="nonfatalErrorsLogPath" value="" /> -->
  <!-- <property name="upSimpleLog" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

path (ConfigPath)

logExtraInfo (boolean) Whether to include the "extra info" field for each entry in crawl.log. "Extra info" is arbitrary JSON. It is the last field of the log line.

crawlLogPath (ConfigPath)

alertsLogPath (ConfigPath)

progressLogPath (ConfigPath)

uriErrorsLogPath (ConfigPath)

runtimeErrorsLogPath (ConfigPath)

nonfatalErrorsLogPath (ConfigPath)

upSimpleLog (String)

recoveryCheckpoint (Checkpoint)

4.1.10 CrawlLimitEnforcer

Bean to enforce limits on the size of a crawl in URI count, byte count, or elapsed time. Fires off the StatSnapshotEvent, so only checks at the interval (configured in StatisticsTracker) of those events.

```
<bean id="crawlLimitEnforcer" class="org.archive.crawler.framework.CrawlLimitEnforcer
↪">
  <!-- <property name="maxBytesDownload" value="0" /> -->
  <!-- <property name="maxNovelBytes" value="0" /> -->
  <!-- <property name="maxNovelUrls" value="0" /> -->
  <!-- <property name="maxWarcNovelUrls" value="0" /> -->
  <!-- <property name="maxWarcNovelBytes" value="0" /> -->
  <!-- <property name="maxDocumentsDownload" value="0" /> -->
  <!-- <property name="maxTimeSeconds" value="0" /> -->
  <!-- <property name="crawlController" value="" /> -->
</bean>
```

maxBytesDownload (long) Maximum number of bytes to download. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxNovelBytes (long) Maximum number of uncompressed payload bytes to write to WARC response or resource records. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxNovelUrls (long) Maximum number of novel (not deduplicated) urls to download. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxWarcNovelUrls (long) Maximum number of urls to write to WARC response or resource records. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxWarcNovelBytes (long) Maximum number of novel (not deduplicated) bytes to write to WARC response or resource records. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxDocumentsDownload (long) Maximum number of documents to download. Once this number is exceeded the crawler will stop. A value of zero means no upper limit.

maxTimeSeconds (long) Maximum amount of time to crawl (in seconds). Once this much time has elapsed the crawler will stop. A value of zero means no upper limit.

crawlController (CrawlController)

4.1.11 CrawlMetadata

Basic crawl metadata, as consulted by functional modules and recorded in ARCs/WARCs.

```xml
<bean id="crawlMetadata" class="org.archive.modules.CrawlMetadata">
  <!-- <property name="robotsPolicyName" value="obey" /> -->
  <!-- <property name="availableRobotsPolicies" value="" /> -->
  <!-- <property name="operator" value="" /> -->
  <!-- <property name="description" value="" /> -->
  <!-- <property name="userAgentTemplate" value="Mozilla/5.0 (compatible; heritrix/
↪@VERSION@ +@OPERATOR_CONTACT_URL@)" /> -->
  <!-- <property name="operatorFrom" value="" /> -->
  <!-- <property name="operatorContactUrl" value="" /> -->
  <!-- <property name="audience" value="" /> -->
  <!-- <property name="organization" value="" /> -->
  <!-- <property name="jobName" value="" /> -->
</bean>
```

robotsPolicyName (String) Robots policy name

availableRobotsPolicies (Map) Map of all available RobotsPolicies, by name, to choose from. assembled from declared instances in configuration plus the standard ‘obey’ (aka ‘classic’) and ‘ignore’ policies.

operator (String)

description (String)

userAgentTemplate (String)

operatorFrom (String)

operatorContactUrl (String)

audience (String)

organization (String)

jobName (String)

4.1.12 CredentialStore

Front door to the credential store. Come here to get at credentials. See Credential Store Design.

```xml
<bean id="credentialStore" class="org.archive.modules.credential.CredentialStore">
  <!-- <property name="credentials" value="" /> -->
</bean>
```

credentials (Map) Credentials used by heritrix authenticating. See http://crawler.archive.org/proposals/auth/ for background.

4.1.13 DiskSpaceMonitor

Monitors the available space on the paths configured. If the available space drops below a specified threshold a crawl pause is requested.

Monitoring is done via the `java.io.File.getUsableSpace()` method. This method will sometimes fail on network attached storage, returning 0 bytes available even if that is not actually the case.

Paths that do not resolve to actual filesystem folders or files will not be evaluated (i.e. if `java.io.File.exists()` returns `false` no further processing is carried out on that File).

Paths are checked available space whenever a StatSnapshotEvent occurs.

```
<bean id="diskSpaceMonitor" class="org.archive.crawler.monitor.DiskSpaceMonitor">
  <!-- <property name="monitorPaths" value="" /> -->
  <!-- <property name="pauseThresholdMiB" value="500" /> -->
  <!-- <property name="monitorConfigPaths" value="true" /> -->
  <!-- <property name="crawlController" value="" /> -->
  <!-- <property name="configPathConfigurer" value="" /> -->
</bean>
```

monitorPaths (List)

pauseThresholdMiB (long) Set the minimum amount of space that must be available on all monitored paths. If the amount falls below this pause threshold on any path the crawl will be paused.

monitorConfigPaths (boolean) If enabled, all the paths returned by ConfigPathConfigurer#getAllConfigPaths() will be monitored in addition to any paths explicitly specified via #setMonitorPaths(List).

`true` by default.

Note: This is not guaranteed to contain all paths that Heritrix writes to. It is the responsibility of modules that write to disk to register their activity with the ConfigPathConfigurer and some may not do so.

crawlController (CrawlController) Autowire access to CrawlController *

configPathConfigurer (ConfigPathConfigurer) Autowire access to ConfigPathConfigurer *

4.1.14 RulesCanonicalizationPolicy

URI Canonicalizatioon Policy

```
<bean id="rulesCanonicalizationPolicy" class="org.archive.modules.canonicalize.
↪RulesCanonicalizationPolicy">
  <!-- <property name="rules" value="" /> -->
</bean>
```

rules (List)

4.1.15 SheetOverlaysManager

Manager which marks-up CrawlURIs with the names of all applicable Sheets, and returns overlay maps by name.

```
<bean id="sheetOverlaysManager" class="org.archive.crawler.spring.SheetOverlaysManager
↪">
  <!-- <property name="beanFactory" value="" /> -->
  <!-- <property name="sheetsByName" value="" /> -->
</bean>
```

beanFactory (BeanFactory)

sheetsByName (Map) Collect all Sheets, by beanName.

4.1.16 StatisticsTracker

This is an implementation of the AbstractTracker. It is designed to function with the WUI as well as performing various logging activity.

At the end of each snapshot a line is written to the 'progress-statistics.log' file.

The header of that file is as follows:

```
[timestamp] [discovered]    [queued] [downloaded] [doc/s(avg)]  [KB/s(avg)] [dl-
↪failures] [busy-thread] [mem-use-KB]
```

First there is a **timestamp**, accurate down to 1 second.

discovered, **queued**, **downloaded** and **dl-failures** are (respectively) the discovered URI count, pending URI count, successfully fetched count and failed fetch count from the frontier at the time of the snapshot.

KB/s(avg) is the bandwidth usage. We use the total bytes downloaded to calculate average bandwidth usage (KB/sec). Since we also note the value each time a snapshot is made we can calculate the average bandwidth usage during the last snapshot period to gain a "current" rate. The first number is the current and the average is in parenthesis.

doc/s(avg) works the same way as doc/s except it show the number of documents (URIs) rather then KB downloaded.

busy-threads is the total number of ToeThreads that are not available (and thus presumably busy processing a URI). This information is extracted from the crawl controller.

Finally mem-use-KB is extracted from the run time environment (`Runtime.getRuntime().totalMemory()`).

In addition to the data collected for the above logs, various other data is gathered and stored by this tracker.

Successfully downloaded documents per fetch status code Successfully downloaded documents per document mime type Amount of data per mime type Successfully downloaded documents per host Amount of data per host Disposition of all seeds (this is written to 'reports.log' at end of crawl) Successfully downloaded documents per host per source

```xml
<bean id="statisticsTracker" class="org.archive.crawler.reporting.StatisticsTracker">
  <!-- <property name="seeds" value="" /> -->
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="reportsDir" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="liveHostReportSize" value="20" /> -->
  <!-- <property name="applicationContext" value="" /> -->
  <!-- <property name="trackSeeds" value="true" /> -->
  <!-- <property name="trackSources" value="true" /> -->
  <!-- <property name="intervalSeconds" value="20" /> -->
  <!-- <property name="keepSnapshotsCount" value="5" /> -->
  <!-- <property name="crawlController" value="" /> -->
  <!-- <property name="reports" value="" /> -->
  <!-- <property name="beanName" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

seeds (SeedModule)

bdbModule (BdbModule)

reportsDir (ConfigPath)

serverCache (ServerCache)

liveHostReportSize (int)

applicationContext (ApplicationContext)

trackSeeds (boolean) Whether to maintain seed disposition records (expensive in crawls with millions of seeds)

trackSources (boolean) Whether to maintain hosts-per-source-tag records for; very expensive in crawls with large numbers of source-tags (seeds) or large crawls over many hosts

intervalSeconds (int) The interval between writing progress information to log.

keepSnapshotsCount (int) Number of crawl-stat sample snapshots to keep for calculation purposes.

crawlController (CrawlController)

reports (List)

beanName (String)

recoveryCheckpoint (Checkpoint)

4.1.17 TextSeedModule

Module that announces a list of seeds from a text source (such as a ConfigFile or ConfigString), and provides a mechanism for adding seeds after a crawl has begun.

```
<bean id="textSeedModule" class="org.archive.modules.seeds.TextSeedModule">
  <!-- <property name="textSource" value="null" /> -->
  <!-- <property name="blockAwaitingSeedLines" value="1" /> -->
</bean>
```

textSource (ReadSource) Text from which to extract seeds

blockAwaitingSeedLines (int) Number of lines of seeds-source to read on initial load before proceeding with crawl. Default is -1, meaning all. Any other value will cause that number of lines to be loaded before fetching begins, while all extra lines continue to be processed in the background. Generally, this should only be changed when working with very large seed lists, and scopes that do *not* depend on reading all seeds.

4.2 Decide Rules

4.2.1 AcceptDecideRule

```
<bean id="acceptDecideRule" class="org.archive.modules.deciderules.AcceptDecideRule">
</bean>
```

4.2.2 ClassKeyMatchesRegexDecideRule

Rule applies configured decision to any CrawlURI class key – i.e. CrawlURI#getClassKey() – matches matches supplied regex.

```
<bean id="classKeyMatchesRegexDecideRule" class="org.archive.crawler.deciderules.
↪ClassKeyMatchesRegexDecideRule">
  <!-- <property name="crawlController" value="" /> -->
</bean>
```

crawlController (CrawlController)

4.2.3 ContentLengthDecideRule

```
<bean id="contentLengthDecideRule" class="org.archive.modules.deciderules.
↪ContentLengthDecideRule">
  <!-- <property name="contentLengthThreshold" value="" /> -->
</bean>
```

contentLengthThreshold (long) Content-length threshold. The rule returns ACCEPT if the content-length is less than this threshold, or REJECT otherwise. The default is 2^63, meaning any document will be accepted.

4.2.4 ContentTypeMatchesRegexDecideRule

DecideRule whose decision is applied if the URI's content-type is present and matches the supplied regular expression.

```
<bean id="contentTypeMatchesRegexDecideRule" class="org.archive.modules.deciderules.
↪ContentTypeMatchesRegexDecideRule">
</bean>
```

4.2.5 ContentTypeNotMatchesRegexDecideRule

DecideRule whose decision is applied if the URI's content-type is present and does not match the supplied regular expression.

```
<bean id="contentTypeNotMatchesRegexDecideRule" class="org.archive.modules.
↪deciderules.ContentTypeNotMatchesRegexDecideRule">
</bean>
```

4.2.6 ExpressionDecideRule (contrib)

Example usage:

```
 <bean class=”org.archive.modules.deciderules.ExpressionDecideRule”>
    <property name=”groovyExpression” value=’curi.via == null &amp;&amp; curi␣
↪==~ “^https?://(?:www\.)?(facebook|vimeo|flickr)\.com/.*”’/>
</bean>
```

```
<bean id="expressionDecideRule" class="org.archive.modules.deciderules.
↪ExpressionDecideRule">
  <!-- <property name="groovyExpression" value="" /> -->
</bean>
```

groovyExpression (String)

4.2.7 ExternalGeoLocationDecideRule

A rule that can be configured to take alternate implementations of the ExternalGeoLocationInterface. If no implementation specified, or none found, returns configured decision. If host in URI has been resolved checks CrawlHost for the country code determination. If country code is not present, does country lookup, and saves the country code to `CrawlHost` for future consultation. If country code is present in `CrawlHost`, compares it against the configured code. Note that if a host's IP address changes during the crawl, we still consider the associated hostname to be in the country of its original IP address.

```
<bean id="externalGeoLocationDecideRule" class="org.archive.modules.deciderules.
↪ExternalGeoLocationDecideRule">
  <!-- <property name="lookup" value="null" /> -->
  <!-- <property name="countryCodes" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
</bean>
```

lookup (ExternalGeoLookupInterface)

countryCodes (List) Country code name.

serverCache (ServerCache)

4.2.8 FetchStatusDecideRule

Rule applies the configured decision for any URI which has a fetch status equal to the 'target-status' setting.

```
<bean id="fetchStatusDecideRule" class="org.archive.modules.deciderules.
↪FetchStatusDecideRule">
  <!-- <property name="statusCodes" value="" /> -->
</bean>
```

statusCodes (List)

4.2.9 FetchStatusMatchesRegexDecideRule

```
<bean id="fetchStatusMatchesRegexDecideRule" class="org.archive.modules.deciderules.
↪FetchStatusMatchesRegexDecideRule">
</bean>
```

4.2.10 FetchStatusNotMatchesRegexDecideRule

```
<bean id="fetchStatusNotMatchesRegexDecideRule" class="org.archive.modules.
↪deciderules.FetchStatusNotMatchesRegexDecideRule">
</bean>
```

4.2.11 HasViaDecideRule

Rule applies the configured decision for any URI which has a 'via' (essentially, any URI that was a seed or some kinds of mid-crawl adds).

```
<bean id="hasViaDecideRule" class="org.archive.modules.deciderules.HasViaDecideRule">
</bean>
```

4.2.12 HopCrossesAssignmentLevelDomainDecideRule

Applies its decision if the current URI differs in that portion of its hostname/domain that is assigned/sold by registrars, its 'assignment-level-domain' (ALD) (AKA 'public suffix' or in previous Heritrix versions, 'topmost assigned SURT')

```
<bean id="hopCrossesAssignmentLevelDomainDecideRule" class="org.archive.modules.
↪deciderules.HopCrossesAssignmentLevelDomainDecideRule">
</bean>
```

4.2.13 HopsPathMatchesRegexDecideRule

Rule applies configured decision to any CrawlURIs whose ‘hops-path’ (string like “LLXE” etc.) matches the supplied regex.

```
<bean id="hopsPathMatchesRegexDecideRule" class="org.archive.modules.deciderules.
↪HopsPathMatchesRegexDecideRule">
</bean>
```

4.2.14 IdenticalDigestDecideRule

Rule applies configured decision to any CrawlURIs whose revisit profile is set with a profile matching WARCConstants#PROFILE_REVISIT_IDENTICAL_DIGEST

```
<bean id="identicalDigestDecideRule" class="org.archive.modules.deciderules.recrawl.
↪IdenticalDigestDecideRule">
</bean>
```

4.2.15 IpAddressSetDecideRule

IpAddressSetDecideRule must be used with org.archive.crawler.prefetch.Preselector#setRecheckScope(boolean) set to true because it relies on Heritrix’ dns lookup to establish the ip address for a URI before it can run.

```
<bean class=”org.archive.modules.deciderules.IpAddressSetDecideRule”>
 <property name=”ipAddresses”>
  <set>
   <value>127.0.0.1</value>
   <value>69.89.27.209</value>
  </set>
 </property>
 <property name=’decision’ value=’REJECT’ />
</bean>
```

```
<bean id="ipAddressSetDecideRule" class="org.archive.modules.deciderules.
↪IpAddressSetDecideRule">
  <!-- <property name="ipAddresses" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
</bean>
```

ipAddresses (Set)

serverCache (ServerCache)

4.2.16 MatchesFilePatternDecideRule

Compares suffix of a passed CrawlURI, UURI, or String against a regular expression pattern, applying its configured decision to all matches. Several predefined patterns are available for convenience. Choosing 'custom' makes this the same as a regular MatchesRegexDecideRule.

```
<bean id="matchesFilePatternDecideRule" class="org.archive.modules.deciderules.
↪MatchesFilePatternDecideRule">
  <!-- <property name="usePreset" value="" /> -->
</bean>
```

usePreset (Preset)

4.2.17 MatchesListRegexDecideRule

Rule applies configured decision to any CrawlURIs whose String URI matches the supplied regexs.

The list of regular expressions can be considered logically AND or OR.

```
<bean id="matchesListRegexDecideRule" class="org.archive.modules.deciderules.
↪MatchesListRegexDecideRule">
  <!-- <property name="timeoutPerRegexSeconds" value="0" /> -->
  <!-- <property name="regexList" value="" /> -->
  <!-- <property name="listLogicalOr" value="true" /> -->
</bean>
```

timeoutPerRegexSeconds (long) The timeout for regular expression matching, in seconds. If set to 0 or negative then no timeout is specified and there is no upper limit to how long the matching may take. See the corresponding test class MatchesListRegexDecideRuleTest for a pathological example.

regexList (List) The list of regular expressions to evalute against the URI.

listLogicalOr (boolean) True if the list of regular expression should be considered as logically AND when matching. False if the list of regular expressions should be considered as logically OR when matching.

4.2.18 MatchesRegexDecideRule

Rule applies configured decision to any CrawlURIs whose String URI matches the supplied regex.

```
<bean id="matchesRegexDecideRule" class="org.archive.modules.deciderules.
↪MatchesRegexDecideRule">
  <!-- <property name="regex" value="" /> -->
</bean>
```

regex (Pattern)

4.2.19 MatchesStatusCodeDecideRule

Provides a rule that returns "true" for any CrawlURIs which have a fetch status code that falls within the provided inclusive range. For instance, to select only URIs with a "success" status code you must provide the range 200 to 299.

```
<bean id="matchesStatusCodeDecideRule" class="org.archive.modules.deciderules.
↪MatchesStatusCodeDecideRule">
  <!-- <property name="lowerBound" value="" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="upperBound" value="" /> -->
</bean>
```

lowerBound (Integer) Sets the lower bound on the range of acceptable status codes.

upperBound (Integer) Sets the upper bound on the range of acceptable status codes.

4.2.20 NotMatchesFilePatternDecideRule

Rule applies configured decision to any URIs which do *not* match the supplied (file-pattern) regex.

```
<bean id="notMatchesFilePatternDecideRule" class="org.archive.modules.deciderules.
↪NotMatchesFilePatternDecideRule">
</bean>
```

4.2.21 NotMatchesListRegexDecideRule

Rule applies configured decision to any URIs which do *not* match the supplied regex.

```
<bean id="notMatchesListRegexDecideRule" class="org.archive.modules.deciderules.
↪NotMatchesListRegexDecideRule">
</bean>
```

4.2.22 NotMatchesRegexDecideRule

Rule applies configured decision to any URIs which do *not* match the supplied regex.

```
<bean id="notMatchesRegexDecideRule" class="org.archive.modules.deciderules.
↪NotMatchesRegexDecideRule">
</bean>
```

4.2.23 NotMatchesStatusCodeDecideRule

Provides a rule that returns “true” for any CrawlURIs which has a fetch status code that does not fall within the provided inclusive range. For instance, to reject any URIs with a “client error” status code you must provide the range 400 to 499.

```
<bean id="notMatchesStatusCodeDecideRule" class="org.archive.modules.deciderules.
↪NotMatchesStatusCodeDecideRule">
  <!-- <property name="upperBound" value="" /> -->
</bean>
```

upperBound (Integer) Sets the upper bound on the range of acceptable status codes.

4.2.24 NotOnDomainsDecideRule

Rule applies configured decision to any URIs that are *not* in one of the domains in the configured set of domains, filled from the seed set.

```
<bean id="notOnDomainsDecideRule" class="org.archive.modules.deciderules.surt.
↪NotOnDomainsDecideRule">
</bean>
```

4.2.25 NotOnHostsDecideRule

Rule applies configured decision to any URIs that are *not* on one of the hosts in the configured set of hosts, filled from the seed set.

```
<bean id="notOnHostsDecideRule" class="org.archive.modules.deciderules.surt.
↪NotOnHostsDecideRule">
</bean>
```

4.2.26 NotSurtPrefixedDecideRule

Rule applies configured decision to any URIs that, when expressed in SURT form, do *not* begin with one of the prefixes in the configured set. The set can be filled with SURT prefixes implied or listed in the seeds file, or another external file.

```
<bean id="notSurtPrefixedDecideRule" class="org.archive.modules.deciderules.surt.
↪NotSurtPrefixedDecideRule">
</bean>
```

4.2.27 OnDomainsDecideRule

Rule applies configured decision to any URIs that are on one of the domains in the configured set of domains, filled from the seed set.

```
<bean id="onDomainsDecideRule" class="org.archive.modules.deciderules.surt.
↪OnDomainsDecideRule">
</bean>
```

4.2.28 OnHostsDecideRule

Rule applies configured decision to any URIs that are on one of the hosts in the configured set of hosts, filled from the seed set.

```
<bean id="onHostsDecideRule" class="org.archive.modules.deciderules.surt.
↪OnHostsDecideRule">
</bean>
```

4.2.29 PathologicalPathDecideRule

Rule REJECTs any URI which contains an excessive number of identical, consecutive path-segments (eg http://example.com/a/a/a/boo.html == 3 '/a' segments)

```
<bean id="pathologicalPathDecideRule" class="org.archive.modules.deciderules.
↪PathologicalPathDecideRule">
  <!-- <property name="maxRepetitions" value="2" /> -->
</bean>
```

maxRepetitions **(int)** Number of times the pattern should be allowed to occur. This rule returns its decision (usually REJECT) if a path-segment is repeated more than number of times.

4.2.30 PredicatedDecideRule

Rule which applies the configured decision only if a test evaluates to true. Subclasses override evaluate() to establish the test.

```
<bean id="predicatedDecideRule" class="org.archive.modules.deciderules.
↪PredicatedDecideRule">
  <!-- <property name="decision" value="" /> -->
</bean>
```

decision **(DecideResult)**

4.2.31 PrerequisiteAcceptDecideRule

Rule which ACCEPTs all 'prerequisite' URIs (those with a 'P' in the last hopsPath position). Good in a late position to ensure other scope settings don't lock out necessary prerequisites.

```
<bean id="prerequisiteAcceptDecideRule" class="org.archive.modules.deciderules.
↪PrerequisiteAcceptDecideRule">
</bean>
```

4.2.32 RejectDecideRule

```
<bean id="rejectDecideRule" class="org.archive.modules.deciderules.RejectDecideRule">
</bean>
```

4.2.33 ResourceLongerThanDecideRule

Applies configured decision for URIs with content length greater than a given threshold length value. Examines either HTTP header Content-Length or actual downloaded content length (based on the useHeaderLength property), and has no effect on resources shorter than or equal to the given threshold value. Note that because neither the Content-Length header nor the actual size are available at URI-scoping time, this rule is unusable in crawl scopes. Instead, the earliest it can be used is as a mid-fetch rule (in FetchHTTP), when the headers are available but not yet the body. It can also be used to affect processing after the URI is fully fetched.

```
<bean id="resourceLongerThanDecideRule" class="org.archive.modules.deciderules.
↪ResourceLongerThanDecideRule">
</bean>
```

4.2.34 ResourceNoLongerThanDecideRule

Applies configured decision for URIs with content length less than or equal to a given threshold length value. Examines either HTTP header Content-Length or actual downloaded content length (based on the useHeaderLength property), and has no effect on resources longer than the given threshold value. Note that because neither the Content-Length header nor the actual size are available at URI-scoping time, this rule is unusable in crawl scopes. Instead, the earliest it can be used is as a mid-fetch rule (in FetchHTTP), when the headers are available but not yet the body. It can also be used to affect processing after the URI is fully fetched.

```
<bean id="resourceNoLongerThanDecideRule" class="org.archive.modules.deciderules.
↪ResourceNoLongerThanDecideRule">
  <!-- <property name="useHeaderLength" value="true" /> -->
  <!-- <property name="contentLengthThreshold" value="1" /> -->
</bean>
```

useHeaderLength (boolean) Shall this rule be used as a midfetch rule? If true, this rule will determine content length based on HTTP header information, otherwise the size of the already downloaded content will be used.

contentLengthThreshold (long) Max content-length this filter will allow to pass through. If -1, then no limit.

4.2.35 ResponseContentLengthDecideRule

Decide rule that will ACCEPT or REJECT a uri, depending on the “decision” property, after it’s fetched, if the content body is within a specified size range, specified in bytes.

```
<bean id="responseContentLengthDecideRule" class="org.archive.modules.deciderules.
↪ResponseContentLengthDecideRule">
  <!-- <property name="lowerBound" value="0" /> -->
  <!-- <property name="upperBound" value="" /> -->
</bean>
```

lowerBound (long) The rule will apply if the url has been fetched and content body length is greater than or equal to this number of bytes. Default is 0, meaning everything will match.

upperBound (long) The rule will apply if the url has been fetched and content body length is less than or equal to this number of bytes. Default is `Long.MAX_VALUE`, meaning everything will match.

4.2.36 SchemeNotInSetDecideRule

Rule applies the configured decision (default REJECT) for any URI which has a URI-scheme NOT contained in the configured Set.

```
<bean id="schemeNotInSetDecideRule" class="org.archive.modules.deciderules.
↪SchemeNotInSetDecideRule">
  <!-- <property name="schemes" value="" /> -->
</bean>
```

schemes (Set) set of schemes to test URI scheme

4.2.37 ScriptedDecideRule

Rule which runs a JSR-223 script to make its decision. Script source may be provided via a file local to the crawler or an inline configuration string. The source must include a one-argument function “decisionFor” which returns the appropriate DecideResult. Variables available to the script include ‘object’ (the object to be evaluated, typically a

CrawlURI), 'self' (this ScriptedDecideRule instance), and 'context' (the crawl's ApplicationContext, from which all named crawl beans are easily reachable).

```
<bean id="scriptedDecideRule" class="org.archive.modules.deciderules.
↪ScriptedDecideRule">
  <!-- <property name="engineName" value="beanshell" /> -->
  <!-- <property name="scriptSource" value="null" /> -->
  <!-- <property name="isolateThreads" value="true" /> -->
  <!-- <property name="applicationContext" value="" /> -->
</bean>
```

engineName (String) engine name; default "beanshell"

scriptSource (ReadSource)

isolateThreads (boolean) Whether each ToeThread should get its own independent script engine, or they should share synchronized access to one engine. Default is true, meaning each thread gets its own isolated engine.

applicationContext (ApplicationContext)

4.2.38 SeedAcceptDecideRule

Rule which ACCEPTs all 'seed' URIs (those for which isSeed is true). Good in a late position to ensure other scope settings don't lock out explicitly added seeds.

```
<bean id="seedAcceptDecideRule" class="org.archive.modules.deciderules.
↪SeedAcceptDecideRule">
</bean>
```

4.2.39 SourceSeedDecideRule

Rule applies the configured decision for any URI with discovered from one of the seeds in `sourceSeeds`. SeedModule#getSourceTagSeeds() must be enabled or the rule will never apply.

```
<bean id="sourceSeedDecideRule" class="org.archive.modules.deciderules.
↪SourceSeedDecideRule">
  <!-- <property name="sourceSeeds" value="" /> -->
</bean>
```

sourceSeeds (Set)

4.2.40 SurtPrefixedDecideRule

Rule applies configured decision to any URIs that, when expressed in SURT form, begin with one of the prefixes in the configured set. The set can be filled with SURT prefixes implied or listed in the seeds file, or another external file. The "also-check-via" option to implement "one hop off" scoping derives from a contribution by Shifra Raffel of the California Digital Library.

```
<bean id="surtPrefixedDecideRule" class="org.archive.modules.deciderules.surt.
↪SurtPrefixedDecideRule">
  <!-- <property name="surtsSourceFile" value="" /> -->
  <!-- <property name="surtsSource" value="null" /> -->
  <!-- <property name="seedsAsSurtPrefixes" value="true" /> -->
  <!-- <property name="surtsDumpFile" value="" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="alsoCheckVia" value="false" /> -->
  <!-- <property name="seeds" value="" /> -->
  <!-- <property name="beanName" value="" /> -->
  <!-- <property name="recoveryCheckpoint" value="" /> -->
</bean>
```

surtsSourceFile (ConfigFile)

surtsSource (ReadSource) Text from which to infer SURT prefixes. Any URLs will be converted to the implied SURT prefix, and literal SURT prefixes may be listed on lines beginning with a '+' character.

seedsAsSurtPrefixes (boolean) Should seeds also be interpreted as SURT prefixes.

surtsDumpFile (ConfigFile) Dump file to save SURT prefixes actually used: Useful debugging SURTs.

alsoCheckVia (boolean) Whether to also make the configured decision if a URI's 'via' URI (the URI from which it was discovered) in SURT form begins with any of the established prefixes. For example, can be used to ACCEPT URIs that are 'one hop off' URIs fitting the SURT prefixes. Default is false.

seeds (SeedModule)

beanName (String)

recoveryCheckpoint (Checkpoint)

4.2.41 TooManyHopsDecideRule

Rule REJECTs any CrawlURIs whose total number of hops (length of the hopsPath string, traversed links of any type) is over a threshold. Otherwise returns PASS.

```
<bean id="tooManyHopsDecideRule" class="org.archive.modules.deciderules.
↪TooManyHopsDecideRule">
  <!-- <property name="maxHops" value="20" /> -->
</bean>
```

maxHops (int) Max path depth for which this filter will match.

4.2.42 TooManyPathSegmentsDecideRule

Rule REJECTs any CrawlURIs whose total number of path-segments (as indicated by the count of '/' characters not including the first '//') is over a given threshold.

```
<bean id="tooManyPathSegmentsDecideRule" class="org.archive.modules.deciderules.
↪TooManyPathSegmentsDecideRule">
  <!-- <property name="maxPathDepth" value="20" /> -->
</bean>
```

maxPathDepth (int) Number of path segments beyond which this rule will reject URIs.

4.2.43 TransclusionDecideRule

Rule ACCEPTs any CrawlURIs whose path-from-seed ('hopsPath' – see CrawlURI#getPathFromSeed() ends with at least one, but not more than, the given number of non-navlink ('L') hops. Otherwise, if the path-from-seed is empty or if a navlink ('L') occurs within max-trans-hops of the tail of the path-from-seed, this rule returns PASS.

Thus, it allows things like embedded resources (frames/images/media) and redirects to be transitively included ('transcluded') in a crawl, even if they otherwise would not, for some reasonable number of hops (usually 1-5).

```
<bean id="transclusionDecideRule" class="org.archive.modules.deciderules.
↪TransclusionDecideRule">
  <!-- <property name="maxTransHops" value="2" /> -->
  <!-- <property name="maxSpeculativeHops" value="1" /> -->
</bean>
```

maxTransHops (int) Maximum number of non-navlink (non-'L') hops to ACCEPT.

maxSpeculativeHops (int) Maximum number of speculative ('X') hops to ACCEPT.

4.2.44 ViaSurtPrefixedDecideRule

Rule applies the configured decision for any URI which has a 'via' whose surtform matches any surt specified in the surtPrefixes list

```
<bean id="viaSurtPrefixedDecideRule" class="org.archive.modules.deciderules.
↪ViaSurtPrefixedDecideRule">
  <!-- <property name="surtPrefixes" value="" /> -->
</bean>
```

surtPrefixes (List)

4.3 Candidate Processors

4.3.1 CandidateScoper

Simple single-URI scoper, considers passed-in URI as candidate; sets fetchstatus negative and skips to end of processing if out-of-scope.

```
<bean id="candidateScoper" class="org.archive.crawler.prefetch.CandidateScoper">
</bean>
```

4.3.2 FrontierPreparer

Processor to preload URI with as much precalculated policy-based info as possible before it reaches frontier critical sections. Frontiers also maintain a direct reference to this class, in case they need to perform remedial preparation for URIs that do not pass through this processor on the CandidateChain.

```
<bean id="frontierPreparer" class="org.archive.crawler.prefetch.FrontierPreparer">
  <!-- <property name="preferenceDepthHops" value="1" /> -->
  <!-- <property name="preferenceEmbedHops" value="1" /> -->
  <!-- <property name="canonicalizationPolicy" value="" /> -->
  <!-- <property name="queueAssignmentPolicy" value="" /> -->
  <!-- <property name="uriPrecedencePolicy" value="" /> -->
  <!-- <property name="costAssignmentPolicy" value="" /> -->
</bean>
```

preferenceDepthHops (int) Number of hops (of any sort) from a seed up to which a URI has higher priority scheduling than any remaining seed. For example, if set to 1 items one hop (link, embed, redirect, etc.) away from a seed will be scheduled with HIGH priority. If set to -1, no preferencing will occur, and a breadth-first search with seeds processed before discovered links will proceed. If set to zero, a purely depth-first search will proceed, with all discovered links processed before remaining seeds. Seed redirects are treated as one hop from a seed.

preferenceEmbedHops (int) number of hops of embeds (ERX) to bump to front of host queue

canonicalizationPolicy (UriCanonicalizationPolicy) Ordered list of url canonicalization rules. Rules are applied in the order listed from top to bottom.

queueAssignmentPolicy (QueueAssignmentPolicy) Defines how to assign URIs to queues. Can assign by host, by ip, by SURT-ordered authority, by SURT-ordered authority truncated to a topmost-assignable domain, and into one of a fixed set of buckets (1k).

uriPrecedencePolicy (UriPrecedencePolicy) URI precedence assignment policy to use.

costAssignmentPolicy (CostAssignmentPolicy) cost assignment policy to use.

4.4 Pre-Fetch Processors

4.4.1 PreconditionEnforcer

Ensures the preconditions for a fetch – such as DNS lookup or acquiring and respecting a robots.txt policy – are satisfied before a URI is passed to subsequent stages.

```
<bean id="preconditionEnforcer" class="org.archive.crawler.prefetch.
↪PreconditionEnforcer">
  <!-- <property name="ipValidityDurationSeconds" value="" /> -->
  <!-- <property name="robotsValidityDurationSeconds" value="" /> -->
  <!-- <property name="calculateRobotsOnly" value="false" /> -->
  <!-- <property name="metadata" value="" /> -->
  <!-- <property name="credentialStore" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="loggerModule" value="" /> -->
</bean>
```

ipValidityDurationSeconds (int) The minimum interval for which a dns-record will be considered valid (in seconds). If the record's DNS TTL is larger, that will be used instead.

robotsValidityDurationSeconds (int) The time in seconds that fetched robots.txt information is considered to be valid. If the value is set to '0', then the robots.txt information will never expire.

calculateRobotsOnly (boolean) Whether to only calculate the robots status of an URI, without actually applying any exclusions found. If true, exlcuded URIs will only be annotated in the crawl.log, but still fetched. Default is false.

metadata (CrawlMetadata) Auto-discovered module providing configured (or overridden) User-Agent value and RobotsHonoringPolicy

credentialStore (CredentialStore)

serverCache (ServerCache)

loggerModule (CrawlerLoggerModule)

4.4.2 Preselector

If set to recheck the crawl's scope, gives a yes/no on whether a CrawlURI should be processed at all. If not, its status will be marked OUT_OF_SCOPE and the URI will skip directly to the first "postprocessor".

```
<bean id="preselector" class="org.archive.crawler.prefetch.Preselector">
  <!-- <property name="recheckScope" value="false" /> -->
  <!-- <property name="blockAll" value="false" /> -->
  <!-- <property name="blockByRegex" value="" /> -->
  <!-- <property name="allowByRegex" value="" /> -->
</bean>
```

recheckScope (boolean) Recheck if uri is in scope. This is meaningful if the scope is altered during a crawl. URIs are checked against the scope when they are added to queues. Setting this value to true forces the URI to be checked against the scope when it is coming out of the queue, possibly after the scope is altered.

blockAll (boolean) Block all URIs from being processed. This is most likely to be used in overrides to easily reject certain hosts from being processed.

blockByRegex (String) Block all URIs matching the regular expression from being processed.

allowByRegex (String) Allow only URIs matching the regular expression to be processed.

4.5 Fetch Processors

4.5.1 FetchDNS

Processor to resolve 'dns:' URIs.

```
<bean id="fetchDNS" class="org.archive.modules.fetcher.FetchDNS">
  <!-- <property name="acceptNonDnsResolves" value="false" /> -->
  <!-- <property name="disableJavaDnsResolves" value="false" /> -->
  <!-- <property name="dnsOverHttpServer" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
</bean>
```

acceptNonDnsResolves (boolean) If a DNS lookup fails, whether or not to fall back to InetAddress resolution, which may use local 'hosts' files or other mechanisms.

disableJavaDnsResolves (boolean) Optionally, only allow InetAddress resolution, precisely because it may use local 'hosts' files or other mechanisms. This should not generally be used in production as it will prevent DNS lookups from being recorded properly.

dnsOverHttpServer (String) URL to the DNS-on-HTTP(S) server. If this not set or set to an empty string, no DNS-over-HTTP(S) will be used; otherwise if should contain the URL to the DNS-over-HTTPS server.

serverCache (ServerCache) Used to do DNS lookups.

digestContent (boolean) Whether or not to perform an on-the-fly digest hash of retrieved content-bodies.

digestAlgorithm (String) Which algorithm (for example MD5 or SHA-1) to use to perform an on-the-fly digest hash of retrieved content-bodies.

4.5.2 FetchFTP

Fetches documents and directory listings using FTP. This class will also try to extract FTP "links" from directory listings. For this class to archive a directory listing, the remote FTP server must support the NLIST command. Most modern FTP servers should.

```
<bean id="fetchFTP" class="org.archive.modules.fetcher.FetchFTP">
  <!-- <property name="username" value="anonymous" /> -->
  <!-- <property name="password" value="password" /> -->
  <!-- <property name="extractFromDirs" value="true" /> -->
  <!-- <property name="extractParent" value="true" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
  <!-- <property name="maxLengthBytes" value="0" /> -->
  <!-- <property name="maxFetchKBSec" value="0" /> -->
  <!-- <property name="timeoutSeconds" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
</bean>
```

username (String) The username to send to FTP servers. By convention, the default value of "anonymous" is used for publicly available FTP sites.

password (String) The password to send to FTP servers. By convention, anonymous users send their email address in this field.

extractFromDirs (boolean) Set to true to extract further URIs from FTP directories. Default is true.

extractParent (boolean) Set to true to extract the parent URI from all FTP URIs. Default is true.

digestContent (boolean) Whether or not to perform an on-the-fly digest hash of retrieved content-bodies.

digestAlgorithm (String) Which algorithm (for example MD5 or SHA-1) to use to perform an on-the-fly digest hash of retrieved content-bodies.

maxLengthBytes (long) Maximum length in bytes to fetch. Fetch is truncated at this length. A value of 0 means no limit.

maxFetchKBSec (int) The maximum KB/sec to use when fetching data from a server. The default of 0 means no maximum.

timeoutSeconds (int) If the fetch is not completed in this number of seconds, give up (and retry later).

soTimeoutMs (int) If the socket is unresponsive for this number of milliseconds, give up. Set to zero for no timeout (Not. recommended. Could hang a thread on an unresponsive server). This timeout is used timing out socket opens and for timing out each socket read. Make sure this value is < #TIMEOUT_SECONDS for optimal configuration: ensures at least one retry read.

4.5.3 FetchHTTP

HTTP fetcher that uses Apache HttpComponents.

```
<bean id="fetchHTTP" class="org.archive.modules.fetcher.FetchHTTP">
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
  <!-- <property name="userAgentProvider" value="" /> -->
  <!-- <property name="sendConnectionClose" value="true" /> -->
  <!-- <property name="defaultEncoding" value="ISO-8859-1" /> -->
  <!-- <property name="useHTTP11" value="false" /> -->
  <!-- <property name="ignoreCookies" value="false" /> -->
  <!-- <property name="sendReferer" value="true" /> -->
  <!-- <property name="acceptCompression" value="false" /> -->
  <!-- <property name="acceptHeaders" value="" /> -->
  <!-- <property name="cookieStore" value="" /> -->
  <!-- <property name="credentialStore" value="" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="httpBindAddress" value="" /> -->
  <!-- <property name="httpProxyHost" value="" /> -->
  <!-- <property name="httpProxyPort" value="" /> -->
  <!-- <property name="httpProxyUser" value="" /> -->
  <!-- <property name="httpProxyPassword" value="" /> -->
  <!-- <property name="maxFetchKBSec" value="0" /> -->
  <!-- <property name="timeoutSeconds" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
  <!-- <property name="maxLengthBytes" value="0" /> -->
  <!-- <property name="sendRange" value="false" /> -->
  <!-- <property name="sendIfModifiedSince" value="true" /> -->
  <!-- <property name="sendIfNoneMatch" value="true" /> -->
  <!-- <property name="shouldFetchBodyRule" value="" /> -->
  <!-- <property name="sslTrustLevel" value="" /> -->
  <!-- <property name="socksProxyHost" value="" /> -->
  <!-- <property name="socksProxyPort" value="" /> -->
</bean>
```

serverCache (ServerCache) Used to do DNS lookups.

digestContent (boolean) Whether or not to perform an on-the-fly digest hash of retrieved content-bodies.

digestAlgorithm (String) Which algorithm (for example MD5 or SHA-1) to use to perform an on-the-fly digest hash of retrieved content-bodies.

userAgentProvider (UserAgentProvider)

sendConnectionClose (boolean) Send ‘Connection: close’ header with every request.

defaultEncoding (String) The character encoding to use for files that do not have one specified in the HTTP response headers. Default: ISO-8859-1.

useHTTP11 (boolean) Use HTTP/1.1. Note: even when offering an HTTP/1.1 request, Heritrix may not properly handle persistent/keep-alive connections, so the sendConnectionClose parameter should remain ‘true’.

ignoreCookies (boolean) Disable cookie handling.

sendReferer (boolean) Send ‘Referer’ header with every request.

The ‘Referer’ header contans the location the crawler came from, the page the current URI was discovered in. The ‘Referer’ usually is logged on the remote server and can be of assistance to webmasters trying to figure how a crawler got to a particular area on a site.

acceptCompression (boolean) Set headers to accept compressed responses.

acceptHeaders (List) Accept Headers to include in each request. Each must be the complete header, e.g., ‘Accept-Language: en’. (Thus, this can also be used to other headers not beginning ‘Accept-’ as well.) By default heritrix sends an Accept header similar to what a typical browser would send (the value comes from Firefox 4.0).

cookieStore (AbstractCookieStore)

credentialStore (CredentialStore) Used to store credentials.

httpBindAddress (String) Local IP address or hostname to use when making connections (binding sockets). When not specified, uses default local address(es).

httpProxyHost (String) Proxy host IP (set only if needed).

httpProxyPort (Integer) Proxy port (set only if needed).

httpProxyUser (String) Proxy user (set only if needed).

httpProxyPassword (String) Proxy password (set only if needed).

maxFetchKBSec (int) The maximum KB/sec to use when fetching data from a server. The default of 0 means no maximum.

timeoutSeconds (int) If the fetch is not completed in this number of seconds, give up (and retry later).

soTimeoutMs (int) If the socket is unresponsive for this number of milliseconds, give up. Set to zero for no timeout (Not. recommended. Could hang a thread on an unresponsive server). This timeout is used timing out socket opens and for timing out each socket read. Make sure this value is < #getTimeoutSeconds() for optimal configuration: ensures at least one retry read.

maxLengthBytes (long) Maximum length in bytes to fetch. Fetch is truncated at this length. A value of 0 means no limit.

sendRange (boolean) Send 'Range' header when a limit (#MAX_LENGTH_BYTES) on document size.

Be polite to the HTTP servers and send the 'Range' header, stating that you are only interested in the first n bytes. Only pertinent if #MAX_LENGTH_BYTES > 0. Sending the 'Range' header results in a '206 Partial Content' status response, which is better than just cutting the response mid-download. On rare occasion, sending 'Range' will generate '416 Request Range Not Satisfiable' response.

sendIfModifiedSince (boolean) Send 'If-Modified-Since' header, if previous 'Last-Modified' fetch history information is available in URI history.

sendIfNoneMatch (boolean) Send 'If-None-Match' header, if previous 'Etag' fetch history information is available in URI history.

shouldFetchBodyRule (DecideRule) DecideRules applied after receipt of HTTP response headers but before we start to download the body. If any filter returns FALSE, the fetch is aborted. Prerequisites such as robots.txt by-pass filtering (i.e. they cannot be midfetch aborted.

sslTrustLevel (TrustLevel) SSL certificate trust level. Range is from the default 'open' (trust all certs including expired, selfsigned, and those for which we do not have a CA) through 'loose' (trust all valid certificates including selfsigned), 'normal' (all valid certificates not including selfsigned) to 'strict' (Cert is valid and DN must match servername).

socksProxyHost (String) Sets a SOCKS5 proxy host to use. This will override any set HTTP proxy.

socksProxyPort (Integer) Sets a SOCKS5 proxy port to use.

4.5.4 FetchSFTP

```
<bean id="fetchSFTP" class="org.archive.modules.fetcher.FetchSFTP">
  <!-- <property name="username" value="anonymous" /> -->
  <!-- <property name="password" value="password" /> -->
  <!-- <property name="extractFromDirs" value="true" /> -->
  <!-- <property name="extractParent" value="true" /> -->
  <!-- <property name="digestContent" value="true" /> -->
  <!-- <property name="digestAlgorithm" value="sha1" /> -->
  <!-- <property name="maxLengthBytes" value="0" /> -->
  <!-- <property name="maxFetchKBSec" value="0" /> -->
  <!-- <property name="timeoutSeconds" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
</bean>
```

username (String) The username to send to SFTP servers. By convention, the default value of "anonymous" is used for publicly available SFTP sites.

password (String) The password to send to SFTP servers. By convention, anonymous users send their email address in this field.

extractFromDirs (boolean) Set to true to extract further URIs from SFTP directories. Default is true.

extractParent (boolean) Set to true to extract the parent URI from all SFTP URIs. Default is true.

digestContent (boolean) Whether or not to perform an on-the-fly digest hash of retrieved content-bodies.

digestAlgorithm (String) Which algorithm (for example MD5 or SHA-1) to use to perform an on-the-fly digest hash of retrieved content-bodies.

maxLengthBytes (long) Maximum length in bytes to fetch. Fetch is truncated at this length. A value of 0 means no limit.

maxFetchKBSec (int) The maximum KB/sec to use when fetching data from a server. The default of 0 means no maximum.

timeoutSeconds (int) If the fetch is not completed in this number of seconds, give up (and retry later).

soTimeoutMs (int) If the socket is unresponsive for this number of milliseconds, give up. Set to zero for no timeout (Not. recommended. Could hang a thread on an unresponsive server). This timeout is used timing out socket opens and for timing out each socket read. Make sure this value is < #TIMEOUT_SECONDS for optimal configuration: ensures at least one retry read.

4.5.5 FetchWhois

WHOIS Fetcher (RFC 3912). If this fetcher is enabled, Heritrix will attempt WHOIS lookups on the topmost assigned domain and the IP address of each URL. WHOIS URIs

There is no pre-existing, canonical specification for WHOIS URIs. What follows is the the format that Heritrix uses, which we propose for general use.

Syntax in ABNF as used in RFC 3986 *Uniform Resource Identifier (URI): Generic Syntax*:

whoisurl = “whois:” [“//” host [“:” port] “/”] whoisquery

whoisquery is a url-encoded string. In ABNF, `whoisquery = 1*pchar` where pchar is defined in RFC 3986. **host** and **port** also as defined in RFC 3986.

To resolve a WHOIS URI which specifies host[:port], open a TCP connection to the host at the specified port (default 43), send the query (whoisquery, url-decoded) followed by CRLF, and read the response until the server closes the connection. For more details see RFC 3912.

Resolution of a “serverless” WHOIS URI, which does not specify host[:port], is implementation-dependent.

Serverless WHOIS URIs in Heritrix

For each non-WHOIS URI processed which has an authority, FetchWhois adds 1 or 2 serverless WHOIS URIs to the CrawlURI’s outlinks. These are “whois:{ipAddress}” and, if the authority includes a hostname, “whois:{topLevelDomain}”. See #addWhoisLinks(CrawlURI).

Heritrix resolves serverless WHOIS URIs by first querying an initial server, then following referrals to other servers. In pseudocode:

```
if query is an IPv4 address
    resolve whois://#DEFAULT_IP_WHOIS_SERVER/whoisquery
else
    let domainSuffix = part of query after the last '.' (or the whole query␣
 ↪if no '.'), url-encoded
    resolve whois://#ULTRA_SUFFIX_WHOIS_SERVER/domainSuffix
while last response refers to another server, i.e. matches regex #WHOIS_
 ↪SERVER_REGEX
```

```
    if we have a special query formatting rule for this whois server, apply␣
↪it - see #specialQueryTemplates
    resolve whois://referralServer/whoisquery
```

See #deferOrFinishGeneric(CrawlURI, String)

```
<bean id="fetchWhois" class="org.archive.modules.fetcher.FetchWhois">
  <!-- <property name="bdbModule" value="" /> -->
  <!-- <property name="specialQueryTemplates" value="" /> -->
  <!-- <property name="soTimeoutMs" value="" /> -->
  <!-- <property name="serverCache" value="" /> -->
</bean>
```

bdbModule (BdbModule)

specialQueryTemplates (Map)

soTimeoutMs (int) If the socket is unresponsive for this number of milliseconds, give up. Set to zero for no timeout (Not. recommended. Could hang a thread on an unresponsive server). This timeout is used timing out socket opens and for timing out each socket read. Make sure this value is < #TIMEOUT_SECONDS for optimal configuration: ensures at least one retry read.

serverCache (ServerCache)

4.6 Link Extractors

4.6.1 ExtractorChrome (contrib)

Extracts links using a web browser via the Chrome Devtools Protocol.

To use, first define this as a top-level bean:

```
<bean id=”extractorChrome” class=”org.archive.modules.extractor.
↪ExtractorChrome”>
  <!- <property name=”captureRequests” value=”true” /> ->
  <!- <property name=”devtoolsUrl” value=”ws://127.0.0.1:1234/devtools/
↪browser/2bc831e8-6c02-4c9b-affd-14c93b8579d7” /> ->
  <!- <property name=”executable” value=”chromium-browser” /> ->
  <!- <property name=”loadTimeoutSeconds” value=”30” /> ->
  <!- <property name=”maxOpenWindows” value=”16” /> ->
  <!- <property name=”windowWidth” value=”1366” /> ->
  <!- <property name=”windowWidth” value=”768” /> ->
</bean>
```

Then add `<ref bean=”extractorChrome”/>` to the fetch chain before `extractorHTML`.

By default an instance of the browser will be run as a subprocess for the duration of the crawl. Alternatively set `devtoolsUrl` to connect to an existing instance of the browser (run with `-headless -remote-debugging-port=1234`).

```
<bean id="extractorChrome" class="org.archive.modules.extractor.ExtractorChrome">
  <!-- <property name="executable" value="null" /> -->
```

(continues on next page)

(continued from previous page)

```
  <!-- <property name="maxOpenWindows" value="16" /> -->
  <!-- <property name="devtoolsUrl" value="null" /> -->
  <!-- <property name="windowWidth" value="1366" /> -->
  <!-- <property name="windowHeight" value="768" /> -->
  <!-- <property name="loadTimeoutSeconds" value="30" /> -->
  <!-- <property name="commandLineOptions" value="" /> -->
</bean>
```

executable (String) The name or path to the browser executable. If null common locations will be searched. Not used if devtoolsUrl is set.

maxOpenWindows (int) The maximum number of browser windows that are allowed to be opened simultaneously. Feel free to increase this if you have lots of RAM available.

devtoolsUrl (String) URL of the devtools server to connect. If null a new browser process will be launched.

windowWidth (int) Width of the browser window.

windowHeight (int) Height of the browser window.

loadTimeoutSeconds (int) Number of seconds to wait for the page to load.

commandLineOptions (List) Extra command-line options passed to the browser process. Not used if devtoolsUrl is null.

4.6.2 ExtractorCSS

This extractor is parsing URIs from CSS type files. The format of a CSS URL value is 'url(' followed by optional white space followed by an optional single quote (') or double quote (") character followed by the URL itself followed by an optional single quote (') or double quote (") character followed by optional white space followed by ')'. Parentheses, commas, white space characters, single quotes (') and double quotes (") appearing in a URL must be escaped with a backslash: '\(', '\)', '\,'. Partial URLs are interpreted relative to the source of the style sheet, not relative to the document. Source: www.w3.org

Note: This processor may open a ReplayCharSequence from the CrawlURI's Recorder, without closing that ReplayCharSequence, to allow reuse by later processors in sequence. In the usual (Heritrix) case, a call after all processing to the Recorder's endReplays() method ensures timely close of any reused ReplayCharSequences. Reuse of this processor elsewhere should ensure a similar cleanup call to Recorder.endReplays() occurs.

```
<bean id="extractorCSS" class="org.archive.modules.extractor.ExtractorCSS">
</bean>
```

4.6.3 ExtractorDOC

This class allows the caller to extract href style links from word97-format word documents.

```
<bean id="extractorDOC" class="org.archive.modules.extractor.ExtractorDOC">
</bean>
```

4.6.4 ExtractorHTML

Basic link-extraction, from an HTML content-body, using regular expressions.

Note: This processor may open a ReplayCharSequence from the CrawlURI's Recorder, without closing that ReplayCharSequence, to allow reuse by later processors in sequence. In the usual (Heritrix) case, a call after all processing to the Recorder's endReplays() method ensures timely close of any reused ReplayCharSequences. Reuse of this processor elsewhere should ensure a similar cleanup call to Recorder.endReplays() occurs.

```
<bean id="extractorHTML" class="org.archive.modules.extractor.ExtractorHTML">
  <!-- <property name="maxElementLength" value="64" /> -->
  <!-- <property name="maxAttributeNameLength" value="64" /> -->
  <!-- <property name="maxAttributeValLength" value="2048" /> -->
  <!-- <property name="treatFramesAsEmbedLinks" value="true" /> -->
  <!-- <property name="ignoreFormActionUrls" value="false" /> -->
  <!-- <property name="extractOnlyFormGets" value="true" /> -->
  <!-- <property name="extractJavascript" value="true" /> -->
  <!-- <property name="extractValueAttributes" value="true" /> -->
  <!-- <property name="ignoreUnexpectedHtml" value="true" /> -->
  <!-- <property name="metadata" value="" /> -->
  <!-- <property name="extractorJS" value="" /> -->
</bean>
```

maxElementLength (int)

maxAttributeNameLength (int)

maxAttributeValLength (int)

treatFramesAsEmbedLinks (boolean) If true, FRAME/IFRAME SRC-links are treated as embedded resources (like IMG, 'E' hop-type), otherwise they are treated as navigational links. Default is true.

ignoreFormActionUrls (boolean) If true, URIs appearing as the ACTION attribute in HTML FORMs are ignored. Default is false.

extractOnlyFormGets (boolean) If true, only ACTION URIs with a METHOD of GET (explicit or implied) are extracted. Default is true.

extractJavascript (boolean) If true, in-page Javascript is scanned for strings that appear likely to be URIs. This typically finds both valid and invalid URIs, and attempts to fetch the invalid URIs sometimes generates webmaster concerns over odd crawler behavior. Default is true.

extractValueAttributes (boolean) If true, strings that look like URIs found in unusual places (such as form VALUE attributes) will be extracted. This typically finds both valid and invalid URIs, and attempts to fetch the invalid URIs sometimes generate webmaster concerns over odd crawler behavior. Default is true.

ignoreUnexpectedHtml (boolean) If true, URIs which end in typical non-HTML extensions (such as .gif) will not be scanned as if it were HTML. Default is true.

metadata (CrawlMetadata) CrawlMetadata provides the robots honoring policy to use when considering a robots META tag.

extractorJS (ExtractorJS) Javascript extractor to use to process inline javascript. Autowired if available. If null, links will not be extracted from inline javascript.

4.6.5 AggressiveExtractorHTML

Extended version of ExtractorHTML with more aggressive javascript link extraction where javascript code is parsed first with general HTML tags regex, and than by javascript speculative link regex.

```
<bean id="aggressiveExtractorHTML" class="org.archive.modules.extractor.
↪AggressiveExtractorHTML">
</bean>
```

4.6.6 JerichoExtractorHTML

Improved link-extraction from an HTML content-body using jericho-html parser. This extractor extends ExtractorHTML and mimics its workflow - but has some substantial differences when it comes to internal implementation. Instead of heavily relying upon java regular expressions it uses a real html parser library - namely Jericho HTML Parser (http://jerichohtml.sourceforge.net). Using this parser it can better handle broken html (i.e. missing quotes) and also offer improved extraction of HTML form URLs (not only extract the action of a form, but also its default values). Unfortunately this parser also has one major drawback - it has to read the whole document into memory for parsing, thus has an inherent OOME risk. This OOME risk can be reduced/eleminated by limiting the size of documents to be parsed (i.e. using NotExceedsDocumentLengthTresholdDecideRule). Also note that this extractor seems to have a lower overall memory consumption compared to ExtractorHTML. (still to be confirmed on a larger scale crawl)

```
<bean id="jerichoExtractorHTML" class="org.archive.modules.extractor.
↪JerichoExtractorHTML">
</bean>
```

4.6.7 ExtractorHTMLForms

Extracts extra information about FORMs in HTML, loading this into the CrawlURI (for potential later use by FormLoginProcessor) and adding a small annotation to the crawl.log. Must come after ExtractorHTML, as it relies on information left in the CrawlURI's A_FORM_OFFSETS data key. By default (with 'extractAllForms' equal false), only saves-to-CrawlURI and annotates forms that appear to be login forms, by the test HTMLForm.seemsLoginForm(). Typical CXML configuration would be, first, as top-level named beans:

```
{@code

   <property name=”extractAllForms” value=”false” />


   generally these are overlaid with sheets rather than set directly
   <property name=”applicableSurtPrefix” value=”” />
   <property name=”loginUsername” value=”” />
   <property name=”loginPassword” value=”” />

}
```

Then, inside the fetch chain, after all other extractors:

```
{@code



   ...ALL USUAL PREPROCESSORS/FETCHERS/EXTRACTORS HERE, THEN...
```

}

Note: This processor may open a ReplayCharSequence from the CrawlURI's Recorder, without closing that ReplayCharSequence, to allow reuse by later processors in sequence. In the usual (Heritrix) case, a call after all processing to the Recorder's endReplays() method ensures timely close of any reused ReplayCharSequences. Reuse of this processor elsewhere should ensure a similar cleanup call to Recorder.endReplays() occurs.

```
<bean id="extractorHTMLForms" class="org.archive.modules.forms.ExtractorHTMLForms">
  <!-- <property name="extractAllForms" value="false" /> -->
</bean>
```

extractAllForms (boolean) If true, report all FORMs. If false, report only those that appear to be a login-enabling FORM. Default is false.

4.6.8 ExtractorHTTP

Extracts URIs from HTTP response headers.

```
<bean id="extractorHTTP" class="org.archive.modules.extractor.ExtractorHTTP">
  <!-- <property name="inferRootPage" value="false" /> -->
</bean>
```

inferRootPage (boolean) should all HTTP URIs be used to infer a link to the site's root?

4.6.9 ExtractorImpliedURI

An extractor for finding 'implied' URIs inside other URIs. If the 'trigger' regex is matched, a new URI will be constructed from the 'build' replacement pattern. Unlike most other extractors, this works on URIs discovered by previous extractors. Thus it should appear near the end of any set of extractors. Initially, only finds absolute HTTP(S) URIs in query-string or its parameters.

```
<bean id="extractorImpliedURI" class="org.archive.modules.extractor.
↪ExtractorImpliedURI">
  <!-- <property name="regex" value="" /> -->
  <!-- <property name="format" value="" /> -->
  <!-- <property name="removeTriggerUris" value="false" /> -->
</bean>
```

regex (Pattern) Triggering regular expression. When a discovered URI matches this pattern, the 'implied' URI will be built. The capturing groups of this expression are available for the build replacement pattern.

format (String) Replacement pattern to build 'implied' URI, using captured groups of trigger expression.

removeTriggerUris (boolean) If true, all URIs that match trigger regular expression are removed from the list of extracted URIs. Default is false.

4.6.10 ExtractorJS

Processes Javascript files for strings that are likely to be crawlable URIs.

Note: This processor may open a ReplayCharSequence from the CrawlURI's Recorder, without closing that ReplayCharSequence, to allow reuse by later processors in sequence. In the usual (Heritrix) case, a call after all processing to the Recorder's endReplays() method ensures timely close of any reused ReplayCharSequences. Reuse of this processor elsewhere should ensure a similar cleanup call to Recorder.endReplays() occurs.

```
<bean id="extractorJS" class="org.archive.modules.extractor.ExtractorJS">
</bean>
```

4.6.11 KnowledgableExtractorJS (contrib)

A subclass of ExtractorJS that has some customized behavior for specific kinds of web pages. As of April 2015, the one special behavior it has is for drupal generated pages. See https://webarchive.jira.com/browse/ARI-4190

```
<bean id="knowledgableExtractorJS" class="org.archive.modules.extractor.
↪KnowledgableExtractorJS">
</bean>
```

4.6.12 ExtractorMultipleRegex

An extractor that uses regular expressions to find strings in the fetched content of a URI, and constructs outlink URIs from those strings.

The crawl operator configures these parameters:

`uriRegex`: a regular expression to match against the url `contentRegexes` a map of named regular expressions { name => regex } to run against the content `template`: the template for constructing the outlinks

The URI is checked against `uriRegex`. The match is done using Matcher#matches(), so the full URI string must match, not just a substring. If it does match, then the matching groups are available to the URI-building template as `${uriRegex[n]}`. If it does not match, processing of the URI is finished and no outlinks are extracted.

Then the extractor looks for matches for each of the `contentRegexes` in the fetched content. If any of the regular expressions produce no matches, processing of the URI is finished and no outlinks are extracted. If at least one match is found for each regular expression, then an outlink is constructed, using the URI-building template, for every combination of matches. The matching groups are available to the template as `${name[n]}`.

Outlinks are constructed using the URI-building `template`. Variable interpolation using the familiar ${. . . } syntax is supported. The template is evaluated for each combination of regular expression matches found, and the matching groups are available to the template as `${regexName[n]}`. An example template might look like: `http://example.org/${uriRegex[1]}/foo?bar=${myContentRegex[0]}`.

The template is evaluated as a Groovy Template, so further capabilities beyond simple variable interpolation are available.

```
<bean id="extractorMultipleRegex" class="org.archive.modules.extractor.
↪ExtractorMultipleRegex">
  <!-- <property name="uriRegex" value="" /> -->
  <!-- <property name="contentRegexes" value="" /> -->
  <!-- <property name="template" value="" /> -->
</bean>
```

uriRegex (String) Regular expression against which to match the URI. If the URI matches, then the matching groups are available to the URI-building template as `${uriRegex[n]}`. If it does not match, processing of this URI is finished and no outlinks are extracted.

contentRegexes (Map) A map of { name => regex }. The extractor looks for matches for each regular expression in the content of the URI being processed. If any of the regular expressions produce no matches, processing of the URI is finished and no outlinks are extracted. If at least one match is found for each regular expression, then an outlink is constructed for every combination of matches. The matching groups are available to the URI-building template as `${name[n]}`.

template (String) URI-building template. Provides variable interpolation using the familiar ${...} syntax. The template is evaluated for each combination of regular expression matches found, and the matching groups are available to the template as `${regexName[n]}`. An example template might look like: `http://example.org/${uriRegex[1]}/foo?bar=${myContentRegex[0]}`.

The template is evaluated as a Groovy Template, so further capabilities beyond simple variable interpolation are available.

4.6.13 ExtractorPDF

Allows the caller to process a CrawlURI representing a PDF for the purpose of extracting URIs

```
<bean id="extractorPDF" class="org.archive.modules.extractor.ExtractorPDF">
  <!-- <property name="maxSizeToParse" value="" /> -->
</bean>
```

maxSizeToParse (long) The maximum size of PDF files to consider. PDFs larger than this maximum will not be searched for links.

4.6.14 ExtractorPDFContent (contrib)

PDF Content Extractor. This will parse the text content of a PDF and apply a regex to search for links within the body of the text. Requires itextpdf jar: http://repo1.maven.org/maven2/com/itextpdf/itextpdf/5.5.0/itextpdf-5.5.0.jar

```
<bean id="extractorPDFContent" class="org.archive.modules.extractor.
↪ExtractorPDFContent">
  <!-- <property name="maxSizeToParse" value="" /> -->
</bean>
```

maxSizeToParse (long) The maximum size of PDF files to consider. PDFs larger than this maximum will not be searched for links.

4.6.15 ExtractorRobotsTxt

```
<bean id="extractorRobotsTxt" class="org.archive.modules.extractor.ExtractorRobotsTxt
↪">
</bean>
```

4.6.16 ExtractorSitemap

```
<bean id="extractorSitemap" class="org.archive.modules.extractor.ExtractorSitemap">
  <!-- <property name="urlPattern" value="null" /> -->
  <!-- <property name="enableLenientExtraction" value="false" /> -->
</bean>
```

urlPattern (String) If urlPattern is not null then any url marked as a sitemap and matching the pattern is assumed to be a sitemap. Otherwise the mime-type is checked (must be “text/xml” or “application/xml”) and the file is “sniffed” for the expected start of a sitemap file.

enableLenientExtraction (boolean) If true, all urls in the sitemap file are extracted, regardless of whether or not they obey the scoping rules specified in the sitemap protocol (https://www.sitemaps.org/protocol.html).

4.6.17 ExtractorSWF

Extracts URIs from SWF (flash/shockwave) files. To test, here is a link to an swf that has links embedded inside of it: http://www.hitspring.com/index.swf.

```
<bean id="extractorSWF" class="org.archive.modules.extractor.ExtractorSWF">
  <!-- <property name="extractorJS" value="" /> -->
</bean>
```

extractorJS (ExtractorJS) Javascript extractor to use to process inline javascript. Autowired if available. If null, links will not be extracted from inline javascript.

4.6.18 ExtractorUniversal

A last ditch extractor that will look at the raw byte code and try to extract anything that *looks* like a link. If used, it should always be specified as the last link extractor in the order file.

To accomplish this it will scan through the bytecode and try and build up strings of consecutive bytes that all represent characters that are valid in a URL (see #isURLableChar(int) for details). Once it hits the end of such a string (i.e. finds a character that should not be in a URL) it will try to determine if it has found a URL. This is done be seeing if the string is an IP address prefixed with http(s):// or contains a dot followed by a Top Level Domain and end of string or a slash.

```
<bean id="extractorUniversal" class="org.archive.modules.extractor.ExtractorUniversal
↪">
  <!-- <property name="maxSizeToParse" value="" /> -->
</bean>
```

maxSizeToParse (long) How deep to look into files for URI strings, in bytes.

4.6.19 ExtractorURI

An extractor for finding URIs inside other URIs. Unlike most other extractors, this works on URIs discovered by previous extractors. Thus it should appear near the end of any set of extractors. Initially, only finds absolute HTTP(S) URIs in query-string or its parameters.

```
<bean id="extractorURI" class="org.archive.modules.extractor.ExtractorURI">
</bean>
```

4.6.20 ExtractorXML

A simple extractor which finds HTTP URIs inside XML/RSS files, inside attribute values and simple elements (those with only whitespace + HTTP URI + whitespace as contents).

Note: This processor may open a ReplayCharSequence from the CrawlURI's Recorder, without closing that ReplayCharSequence, to allow reuse by later processors in sequence. In the usual (Heritrix) case, a call after all processing to the Recorder's endReplays() method ensures timely close of any reused ReplayCharSequences. Reuse of this processor elsewhere should ensure a similar cleanup call to Recorder.endReplays() occurs.

```
<bean id="extractorXML" class="org.archive.modules.extractor.ExtractorXML">
</bean>
```

4.6.21 ExtractorYoutubeDL (contrib)

Extracts links to media by running youtube-dl in a subprocess. Runs only on html.

Also implements WARCRecordBuilder to write youtube-dl json to the warc.

To use `ExtractorYoutubeDL`, add this top-level bean:

```
<bean id=”extractorYoutubeDL” class=”org.archive.modules.extractor.
↪ExtractorYoutubeDL”/>
```

Then add `<ref bean=”extractorYoutubeDL”/>` to end of the fetch chain, and to the end of the warc writer chain.

Keeps a log of containing pages and media captured as a result of youtube-dl extraction. The format of the log is as follows:

```
[timestamp] [media-http-status] [media-length] [media-mimetype] [media-digest] [media-
↪timestamp] [media-url] [annotation] [containing-page-digest] [containing-page-
↪timestamp] [containing-page-url] [seed-url]
```

For containing pages, all of the `media-*` fields have the value “`-`”, and the annotation field looks like “`youtube-dl:3`”, meaning that ExtractorYoutubeDL extracted 3 media links from the page.

For media, the annotation field looks like “`youtube-dl:1/3`”, meaning this is the first of three media links extracted from the containing page. The intention is to use this for playback. The rest of the fields included in the log were also chosen to support creation of an index of media by containing page, to be used for playback.

```
<bean id="extractorYoutubeDL" class="org.archive.modules.extractor.ExtractorYoutubeDL
↪">
  <!-- <property name="crawlerLoggerModule" value="" /> -->
</bean>
```

crawlerLoggerModule (CrawlerLoggerModule)

4.6.22 ExtractorYoutubeFormatStream (contrib)

Youtube stream URI extractor. This will check the content of the youtube watch page looking for the url_encoded_fmt_stream_map json value. The json object is decoded and the stream URIs are constructed and queued.

```
{@code

   enable via sheet for http://(com,youtube,

```

(continues on next page)

(continued from previous page)

```
    38  MP4 3072p (Discontinued)
    37  MP4 1080p (Discontinued)
    22  MP4 720p
    18  MP4 270p/360p
    35  FLV 480p (Discontinued)
    34  FLV 360p (Discontinued)
    36  3GP 240p
    5  FLV 240p
    17  3GP 144p



}
```

```
<bean id="extractorYoutubeFormatStream" class="org.archive.modules.extractor.
↪ExtractorYoutubeFormatStream">
  <!-- <property name="extractLimit" value="1" /> -->
  <!-- <property name="itagPriority" value="" /> -->
</bean>
```

extractLimit (Integer) Maximum number of video urls to extract. A value of 0 means extract all discovered video urls. Default is 1.

itagPriority (List) Itag priority list. Youtube itag parameter specifies the video and audio format and quality. The default is an empty list, which tells the extractor to extract up to extractLimit video urls. When the list is not empty, only video urls with itag values in the list are extracted.

4.6.23 ExtractorYoutubeChannelFormatStream (contrib)

```
<bean id="extractorYoutubeChannelFormatStream" class="org.archive.modules.extractor.
↪ExtractorYoutubeChannelFormatStream">
</bean>
```

4.6.24 TrapSuppressExtractor

Pseudo-extractor that suppresses link-extraction of likely trap pages, by noticing when content's digest is identical to that of its 'via'.

```
<bean id="trapSuppressExtractor" class="org.archive.modules.extractor.
↪TrapSuppressExtractor">
</bean>
```

4.7 Post-Processors

4.7.1 CandidatesProcessor

Processor which sends all candidate outlinks through the CandidateChain, scheduling those with non-negative status codes to the frontier. Also performs special handling for 'discovered seeds' – URIs, as with redirects from seeds, that may deserve special treatment to expand the scope.

```
<bean id="candidatesProcessor" class="org.archive.crawler.postprocessor.
↪CandidatesProcessor">
  <!-- <property name="candidateChain" value="" /> -->
  <!-- <property name="frontier" value="" /> -->
  <!-- <property name="loggerModule" value="" /> -->
  <!-- <property name="seedsRedirectNewSeeds" value="true" /> -->
  <!-- <property name="seedsRedirectNewSeedsAllowTLDs" value="true" /> -->
  <!-- <property name="processErrorOutlinks" value="false" /> -->
  <!-- <property name="seeds" value="" /> -->
  <!-- <property name="sheetOverlaysManager" value="" /> -->
</bean>
```

candidateChain (CandidateChain) Candidate chain

frontier (Frontier) The frontier to use.

loggerModule (CrawlerLoggerModule)

seedsRedirectNewSeeds (boolean) If enabled, any URL found because a seed redirected to it (original seed returned 301 or 302), will also be treated as a seed, as long as the hop count is less than {@value #SEEDS_REDIRECT_NEW_SEEDS_MAX_HOPS}.

seedsRedirectNewSeedsAllowTLDs (boolean) If enabled, any URL found because a seed redirected to it (original seed returned 301 or 302), will also be treated as a seed, as long as the hop count is less than {@value #SEEDS_REDIRECT_NEW_SEEDS_MAX_HOPS}.

processErrorOutlinks (boolean) If true, outlinks from status codes <200 and >=400 will be sent through candidates processing. Default is false.

seeds (SeedModule)

sheetOverlaysManager (SheetOverlaysManager)

4.7.2 DispositionProcessor

A step, late in the processing of a CrawlURI, for marking-up the CrawlURI with values to affect frontier disposition, and updating information that may have been affected by the fetch. This includes robots info and other stats. (Formerly called CrawlStateUpdater, when it did less.)

```
<bean id="dispositionProcessor" class="org.archive.crawler.postprocessor.
↪DispositionProcessor">
  <!-- <property name="serverCache" value="" /> -->
  <!-- <property name="delayFactor" value="5.0f" /> -->
  <!-- <property name="minDelayMs" value="3000" /> -->
  <!-- <property name="respectCrawlDelayUpToSeconds" value="300" /> -->
  <!-- <property name="maxDelayMs" value="30000" /> -->
  <!-- <property name="maxPerHostBandwidthUsageKbSec" value="0" /> -->
  <!-- <property name="forceRetire" value="false" /> -->
  <!-- <property name="metadata" value="" /> -->
</bean>
```

serverCache (ServerCache)

delayFactor (float) How many multiples of last fetch elapsed time to wait before recontacting same server.

minDelayMs (int) always wait this long after one completion before recontacting same server, regardless of multiple

respectCrawlDelayUpToSeconds (int) Whether to respect a 'Crawl-Delay' (in seconds) given in a site's robots.txt

maxDelayMs (int) never wait more than this long, regardless of multiple

maxPerHostBandwidthUsageKbSec (int) maximum per-host bandwidth usage

forceRetire (boolean) Whether to set a CrawlURI's force-retired directive, retiring its queue when it finishes. Mainly intended for URI-specific overlay settings; setting true globally will just retire all queues after they offer one URI, rapidly ending a crawl.

metadata (CrawlMetadata) Auto-discovered module providing configured (or overridden) User-Agent value and RobotsHonoringPolicy

4.7.3 ReschedulingProcessor

The most simple forced-rescheduling step possible: use a local setting (perhaps overlaid to vary based on the URI) to set an exact future reschedule time, as a delay from now. Unless the reschedulDelaySeconds value is changed from its default, URIs are not rescheduled.

```
<bean id="reschedulingProcessor" class="org.archive.crawler.postprocessor.
↪ReschedulingProcessor">
  <!-- <property name="rescheduleDelaySeconds" value="1" /> -->
</bean>
```

rescheduleDelaySeconds (long) amount of time to wait before forcing a URI to be rescheduled default of -1 means "don't reschedule"

4.7.4 WARCWriterChainProcessor

WARC writer processor. The types of records that to be written can be configured by including or excluding WARCRecordBuilder implementations (see #setChain(List)). This is the default chain:

```
  <property name=”chain”>
   <list>
    <bean class=”org.archive.modules.warc.DnsResponseRecordBuilder”/>
    <bean class=”org.archive.modules.warc.HttpResponseRecordBuilder”/>
    <bean class=”org.archive.modules.warc.WhoisResponseRecordBuilder”/>
    <bean class=”org.archive.modules.warc.FtpControlConversationRecordBuilder”/
↪>
    <bean class=”org.archive.modules.warc.FtpResponseRecordBuilder”/>
    <bean class=”org.archive.modules.warc.RevisitRecordBuilder”/>
    <bean class=”org.archive.modules.warc.HttpRequestRecordBuilder”/>
    <bean class=”org.archive.modules.warc.MetadataRecordBuilder”/>
   </list>
  </property>
```

Replaces WARCWriterProcessor.

```
<bean id="wARCWriterChainProcessor" class="org.archive.modules.writer.
↪WARCWriterChainProcessor">
  <!-- <property name="chain" value="" /> -->
</bean>
```

chain (List)

CHAPTER 5

REST API

This manual describes the REST application programming interface (API) of the Heritrix Web crawler.

This document is intended for application developers and administrators interested in controlling the Heritrix Web crawler through its REST API.

Any client that supports HTTPS can be used to invoke the Heritrix API. The examples in this document use the command line tool curl which is typically found in most unix environments. Curl is available for many systems including Windows.

5.1 Get Engine Status

GET https://*(heritrixhost)***:8443/engine**

Returns information about this instance of Heritrix such as version number, memory usage and the list of crawl jobs.

XML Example:

```
curl -v -k -u admin:admin --anyauth --location -H "Accept: application/xml" https://localhost:8443/engine
```

Response:

```
<engine>
  <heritrixVersion>3.3.0-SNAPSHOT-2017-07-12T04:17:56Z</heritrixVersion>
  <heapReport>
    <usedBytes>69529904</usedBytes>
    <totalBytes>589824000</totalBytes>
    <maxBytes>2885681152</maxBytes>
  </heapReport>
  <jobsDir>/heritrix/jobs</jobsDir>
  <jobsDirUrl>https://localhost:8443/engine/jobsdir/</jobsDirUrl>
  <availableActions>
    <value>rescan</value>
```

(continues on next page)

(continued from previous page)

```
    <value>add</value>
    <value>create</value>
  </availableActions>
  <jobs>
    <value>
      <shortName>myjob</shortName>
      <url>https://localhost:8443/engine/job/myjob</url>
      <isProfile>false</isProfile>
      <launchCount>0</launchCount>
      <lastLaunch/>
      <hasApplicationContext>false</hasApplicationContext>
      <statusDescription>Unbuilt</statusDescription>
      <isLaunchInfoPartial>false</isLaunchInfoPartial>
      <primaryConfig>/heritrix/jobs/myjob/crawler-beans.cxml</primaryConfig>
      <primaryConfigUrl>https://localhost:8443/engine/jobdir/crawler-beans.cxml</
↪primaryConfigUrl>
      <key>myjob</key>
    </value>
  </jobs>
</engine>
```

5.2 Create New Job

`POST https://` *(heritrixhost)* **`:8443/engine [action=create]`**

Creates a new crawl job. It uses the default configuration provided by the profile-defaults profile.

Form Parameters

- **`action`** – must be `create`
- **`createpath`** – the name of the new job

HTML Example:

```
curl -v -d "createpath=myjob&action=create" -k -u admin:admin --anyauth --
↪location \
  https://localhost:8443/engine
```

XML Example:

```
curl -v -d "createpath=myjob&action=create" -k -u admin:admin --anyauth --
↪location \
  -H "Accept: application/xml" https://localhost:8443/engine
```

5.3 Add Job Directory

`POST https://` *(heritrixhost)* **`:8443/engine [action=add]`**

Adds a new job directory to the Heritrix configuration. The directory must contain a cxml configuration file.

Form Parameters

- **`action`** – must be `add`
- **`addpath`** – the job directory to add

HTML Example:

```
curl -v -d "action=add&addpath=/Users/hstern/job" -k -u admin:admin --anyauth --
↪location https://localhost:8443/engine
```

XML Example:

```
curl -v -d "action=add&addpath=/Users/hstern/job" -k -u admin:admin --anyauth --
↪location -H "Accept: application/xml" https://localhost:8443/engine
```

5.4 Get Job Status

GET https://*(heritrixhost)***:8443/engine/job/**
jobname Returns status information and statistics about the chosen job.

XML Example:

```
curl -v -k -u admin:admin --anyauth --location -H "Accept: application/xml"␣
↪https://localhost:8443/engine/job/myjob
```

Response:

```
<job>
  <shortName>myjob</shortName>
  <crawlControllerState>FINISHED</crawlControllerState>
  <crawlExitStatus>FINISHED</crawlExitStatus>
  <statusDescription>Finished: FINISHED</statusDescription>
  <availableActions>
    <value>teardown</value>
  </availableActions>
  <launchCount>1</launchCount>
  <lastLaunch>2020-04-01T02:07:42.531Z</lastLaunch>
  <isProfile>false</isProfile>
  <primaryConfig>/heritrix/jobs/myjob/crawler-beans.cxml</primaryConfig>
  <primaryConfigUrl>https://localhost:8443/engine/job/myjob/jobdir/crawler-beans.
↪cxml</primaryConfigUrl>
  <url>https://localhost:8443/engine/job/myjob/job/myjob</url>
  <jobLogTail>
    <value>2020-04-01T03:50:44.708Z INFO FINISHED 20200401020744</value>
    <value>2020-04-01T03:50:42.670Z INFO EMPTY 20200401020744</value>
    <value>2020-04-01T03:50:42.669Z INFO STOPPING 20200401020744</value>
  </jobLogTail>
  <uriTotalsReport>
    <downloadedUriCount>3920</downloadedUriCount>
    <queuedUriCount>0</queuedUriCount>
    <totalUriCount>3920</totalUriCount>
    <futureUriCount>0</futureUriCount>
  </uriTotalsReport>
  <sizeTotalsReport>
    <dupByHash>0</dupByHash>
    <dupByHashCount>0</dupByHashCount>
    <notModified>0</notModified>
    <notModifiedCount>0</notModifiedCount>
    <novel>2177235508</novel>
    <novelCount>3920</novelCount>
    <total>2177235508</total>
```

(continues on next page)

(continued from previous page)

```
    <totalCount>3920</totalCount>
    <warcNovelContentBytes>2177235508</warcNovelContentBytes>
    <warcNovelUrls>3920</warcNovelUrls>
  </sizeTotalsReport>
  <rateReport>
    <currentDocsPerSecond>0.0</currentDocsPerSecond>
    <averageDocsPerSecond>0.6354171124312226</averageDocsPerSecond>
    <currentKiBPerSec>0</currentKiBPerSec>
    <averageKiBPerSec>344</averageKiBPerSec>
  </rateReport>
  <loadReport>
    <busyThreads>0</busyThreads>
    <totalThreads>0</totalThreads>
    <congestionRatio>NaN</congestionRatio>
    <averageQueueDepth>0</averageQueueDepth>
    <deepestQueueDepth>0</deepestQueueDepth>
  </loadReport>
  <elapsedReport>
    <elapsedMilliseconds>6169176</elapsedMilliseconds>
    <elapsedPretty>1h42m49s176ms</elapsedPretty>
  </elapsedReport>
  <threadReport/>
  <frontierReport>
    <totalQueues>1</totalQueues>
    <inProcessQueues>0</inProcessQueues>
    <readyQueues>0</readyQueues>
    <snoozedQueues>0</snoozedQueues>
    <activeQueues>0</activeQueues>
    <inactiveQueues>0</inactiveQueues>
    <ineligibleQueues>0</ineligibleQueues>
    <retiredQueues>0</retiredQueues>
    <exhaustedQueues>1</exhaustedQueues>
    <lastReachedState>FINISH</lastReachedState>
  </frontierReport>
  <crawlLogTail>
    ...
  </crawlLogTail>
  <configFiles>
    ...
  </configFiles>
  <isLaunchInfoPartial>false</isLaunchInfoPartial>
  <isRunning>false</isRunning>
  <isLaunchable>false</isLaunchable>
  <hasApplicationContext>true</hasApplicationContext>
  <alertCount>549</alertCount>
  <checkpointFiles></checkpointFiles>
  <alertLogFilePath>/heritrix/jobs/myjob/20200401020744/logs/alerts.log</
↪alertLogFilePath>
  <crawlLogFilePath>/heritrix/jobs/myjob/20200401020744/logs/crawl.log</
↪crawlLogFilePath>
  <reports>
    <value>
      <className>CrawlSummaryReport</className>
      <shortName>CrawlSummary</shortName>
    </value>
    ...
  </reports>
```

(continues on next page)

(continued from previous page)

```
  <heapReport>
    <usedBytes>66893400</usedBytes>
    <totalBytes>589824000</totalBytes>
    <maxBytes>2885681152</maxBytes>
  </heapReport>
</job>
```

5.5 Build Job Configuration

POST https://(*heritrixhost*)**:8443/engine/job/**

jobname **[action=build]** Builds the job configuration for the chosen job. It reads an XML descriptor file and uses Spring to build the Java objects that are necessary for running the crawl. Before a crawl can be run it must be built.

Form Parameters

- **action** – must be `build`

HTML Example:

```
curl -v -d "action=build" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=build" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.6 Launch Job

POST https://(*heritrixhost*)**:8443/engine/job/**

jobname **[action=launch]** Launches a crawl job. The job can be launched in the “paused” state or the “unpaused” state. If launched in the “unpaused” state the job will immediately begin crawling.

Form Parameters

- **action** – must be `launch`
- **checkpoint** – optional field: If supplied, Heritrix will attempt to launch from a checkpoint. Should be the name of a checkpoint (e.g. `cp00001-20180102121229`) or (since version 3.3) the special value `latest`, which will automatically select the most recent checkpoint. If no `checkpoint` is specified (or if the `latest` checkpoint is requested and there are no valid checkpoints) a new crawl will be launched.

HTML Example:

```
curl -v -d "action=launch" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=launch" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.7 Rescan Job Directory

POST https://(*heritrixhost*)**:8443/engine [action=rescan]**
Rescans the main job directory and returns an HTML page containing all the job names. It also returns information about the jobs, such as the location of the job configuration file and the number of job launches.

Form Parameters

- **action** – must be `rescan`

HTML Example:

```
curl -v -d "action=rescan" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine
```

XML Example:

```
curl -v -d "action=rescan" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine
```

5.8 Pause Job

POST https://(*heritrixhost*)**:8443/engine/job/**
jobname **[action=pause]** Pauses an unpaused job. No crawling will occur while a job is paused.

Form Parameters

- **action** – must be `pause`

HTML Example:

```
curl -v -d "action=pause" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=pause" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.9 Unpause Job

POST https://(*heritrixhost*)**:8443/engine/job/**
jobname **[action=unpause]** This API unpauses a paused job. Crawling will resume (or begin, in the case of a job launched in the paused state) if possible.

Form Parameters

- **action** – must be `unpause`

HTML Example:

```
curl -v -d "action=unpause" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=unpause" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.10 Terminate Job

POST https://(*heritrixhost*)**:8443/engine/job/***jobname* **[action=terminate]** Terminates a running job.

Form Parameters

- **action** – must be `terminate`

HTML Example:

```
curl -v -d "action=terminate" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=terminate" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.11 Teardown Job

POST https://(*heritrixhost*)**:8443/engine/job/***jobname* **[action=teardown]** Removes the Spring code that is used to run the job. Once a job is torn down it must be rebuilt in order to run.

Form Parameters

- **action** – must be `teardown`

HTML Example:

```
curl -v -d "action=teardown" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=teardown" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.12 Copy Job

POST https://(*heritrixhost*)**:8443/engine/job/***jobname* **[copyTo]** Copies an existing job configuration to a new job configuration. If the "as profile" checkbox is selected, than the job configuration is copied as a non-runnable profile configuration.

Form Parameters

- **copyTo** – the name of the new job or profile configuration

- **asProfile** – whether to copy the job as a runnable configuration or as a non-runnable profile. The value `on` means the job will be copied as a profile. If omitted the job will be copied as a runnable configuration.

HTML Example:

```
curl -v -d "copyTo=mycopy&asProfile=on" -k -u admin:admin --anyauth --location␣
↪https://localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "copyTo=mycopy&asProfile=on" -k -u admin:admin --anyauth --location -H
↪"Accept: application/xml" https://localhost:8443/engine/job/myjob
```

5.13 Checkpoint Job

POST https://(*heritrixhost*)**:8443/engine/job/**
jobname **[action=checkpoint]** This API checkpoints the chosen job. Checkpointing writes the current state of a crawl to the file system so that the crawl can be recovered if it fails.

Form Parameters

- **action** – must be `checkpoint`

HTML Example:

```
curl -v -d "action=checkpoint" -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob
```

XML Example:

```
curl -v -d "action=checkpoint" -k -u admin:admin --anyauth --location -H "Accept:␣
↪application/xml" https://localhost:8443/engine/job/myjob
```

5.14 Execute Script in Job

POST https://(*heritrixhost*)**:8443/engine/job/**
*jobname***/script** Executes a script. The script can be written as Beanshell, ECMAScript, Groovy, or AppleScript.

Form Parameters

- **engine** – the script engine to use. One of `beanshell`, `js`, `groovy` or `AppleScriptEngine`.
- **script** – the script code to execute

HTML Example:

```
curl -v -d "engine=beanshell&script=System.out.println%28%22test%22%29%3B" -k -u␣
↪admin:admin --anyauth --location https://localhost:8443/engine/job/myjob/script
```

XML Example:

```
curl -v -d "engine=beanshell&script=System.out.println%28%22test%22%29%3B" -k -u␣
↪admin:admin --anyauth --location -H "Accept: application/xml" https://
↪localhost:8443/engine/job/myjob/script
```

(continues on next page)

(continued from previous page)

5.15 Submitting a CXML Job Configuration File

PUT https://*⟨heritrixhost⟩***:8443/engine/job/***jobname***/jobdir/crawler-beans.cxml** Submits the contents of a CXML file for a chosen job. CXML files are the configuration files used to control a crawl job. Each job has a single CXML file.

Example:

```
curl -v -T my-crawler-beans.cxml -k -u admin:admin --anyauth --location https://
↪localhost:8443/engine/job/myjob/jobdir/crawler-beans.cxml
```

Status Codes

- 200 OK – On success, the Heritrix REST API will return a HTTP 200 with no body.

5.16 Conventions and Assumptions

The following curl parameters are used when invoking the API.

curl Parameter	Description
-v	Verbose. Output a detailed account of the curl command to standard out.
-d	Data. These are the name/value pairs that are send in the body of a POST.
-k	Insecure. Allows connections to SSL sites without certificates.
-u	User. Allows the submission of a username and password to authenticate the HTTP request.
–anyauth	Any authentication type. Allows authentication of the request based on any type of authentication method.
–location	Follows HTTP redirects. This option is used so that API calls that return data (such as HTML) will not halt upon receipt of a redirect code (such as an HTTP 303).
-H	Set the value of an HTTP header. For example, “Accept: application/xml”.

It is assumed that the reader has a working knowledge of the HTTP protocol and Heritrix functionality. Also, the examples assume that Heritrix is run with an administrative username and password of “admin.”

5.17 About the REST implementation

Representational State Transfer (REST) is a software architecture for distributed hypermedia systems such as the World Wide Web (WWW). REST is built on the concept of representations of resources. Resources can be any coherent and meaningful concept that may be addressed. A URI is an example of a resource. The representation of the resource is typically a document that captures the current or intended state of the resource. An example of a representation of a resource is an HTML page.

Heritrix uses REST to expose its functionality. The REST implementation used by Heritrix is Restlet. Restlet implements the concepts defined by REST, including resources and representations. It also provides a REST container that processes RESTful requests. The container is the Noelios Restlet Engine. For detailed information on Restlet, visit http://www.restlet.org/.

Heritrix exposes its REST functionality through HTTPS. The HTTPS protocol is used to send requests to retrieve or modify configuration settings and manage crawl jobs.

CHAPTER 6

Glossary

Bytes, KB and statistics Heritrix adheres to the following conventions for displaying byte and bit amounts:

Legend	Type	
B	Bytes	
KB	Kilobytes	1 KB = 1024 B
MB	Megabytes	1 MB = 1024 KB
GB	Gigabytes	1 GB = 1024 MB
b	bits	
Kb	Kilobits	1 Kb = 1000 b
Mb	Megabits	1 Mb = 1000 Kb
Gb	Gigabits	1 Gb = 1000 Mb

Checkpointing Heritrix checkpointing is heavily influenced by Mercator checkpointing. In one of the papers on Mercator, checkpointing is described this way: “Checkpointing is an important part of any long-running process such as a web crawl. By checkpointing we mean writing a representation of the crawler’s state to stable storage that, in the event of a failure, is sufficient to allow the crawler to recover its state by reading the checkpoint and to resume crawling from the exact state it was in at the time of the checkpoint. By this definition, in the event of a failure, any work performed after the most recent checkpoint is lost, but none of the work up to the most recent checkpoint. In Mercator, the frequency with which the background thread performs a checkpoint is user-configurable; we typically checkpoint anywhere from 1 to 4 times per day.”

See Checkpointing for a discussion of the Heritrix implementation.

CrawlURI A URI and its associated data such as the parent URI and number of links.

Dates and Times All times in Heritrix are GMT, assuming the clock and timezone on the local system are correct. This means that all dates/times in logs are GMT, all dates and times shown in the WUI are GMT, and any times or dates entered by the user must be in GMT.

Discovered URIs A discovered URI is any URI that has been confirmed to be within “scope.” This includes URIs that have been processed, are being processed, and have finished processing. It does not include URIs that have been “forgotten.” Forgotten URIs are URIs deemed out of scope during fetch. This is most likely due to the operator changing the scope definition.

Note: Since the same URI can be fetched multiple times (at least in most Frontiers), the number of discovered URIs may be somewhat lower then the combined queued, in process, and finished items. This is due to duplicate URIs being queued and processed. The variance is likely to be especially high in Frontiers implementing "revisit" strategies.

Discovery Path Each URI has a discovery path. The path contains one character for each link or embed followed from the seed, for example "LLLE" might be an image on a page that's 3 links away from a seed.

The character legend is as follows:

- R - Redirect
- E - Embedded links necessary to render the page (such as `<img src=...>`)
- X - Speculative embed (aggressive JavaScript link extraction)
- L - Link (normal navigation links like `<a href=...>`)
- P - Prerequisite (such as DNS lookup or robots.txt)
- I - Inferred/implied links. Not necessarily in the source material, but deduced by convention (such as /favicon.ico)
- M - Manifest (such as links discovered from a sitemap file)
- S - Synthesized form-submit

The discovery path of a seed is an empty string.

The discovery path can be used to configure scope of a crawl using the HopsPathMatchesRegexDecideRule. It also appears in the crawl logs and in the WARC metadata records as the `hopsFromSeed` field.

Frontier A Frontier is a pluggable module in Heritrix that maintains the internal state of the crawl. See Frontier.

Host A host can serve multiple domains or a domain can be served by multiple hosts. For our purposes, a host is the same as the hostname in a URI. DNS is not considered because it is volatile and may be unavailable. For example, if multiple URIs point to the same ip address, they are considered three different logical hosts (at the same level of the URI/HTTP protocol).

Conforming HTTP proxies behave similarly. They would not consider a URI and a IP address interchangeable.

This is not ideal for politeness because it applies politeness rules to the physical host rather than the logical host.

Crawl Job In order to run a crawl, a configuration must be created. In Heritrix such a configuration is called a **crawl job**. A crawl job is based on the Spring framework. The job uses Spring beans as configuration objects that define the crawl.

Link Hop Count This is the number of links followed from the seed to reach a URI. Seeds have a link hop count of zero. Link hop count is equal to the count of `L's` in a URIs discovery path.

Pending URIs This is the number of URIs that are waiting for detailed processing. It is also the number of discovered URIs that have not been inspected for scope or duplicates. Depending on the implementation of the Frontier this might always be zero. It may also be an adjusted number that accounts for duplicates.

Profile A profile is a template for a crawl job. It contains all the configurations in a crawl job, but it is not considered "crawlable." Heritrix will not allow you to directly crawl a profile. Only jobs based on profiles can be crawled.

A common example of a profile configuration is leaving the `metadata.operatorContactUrl` property undefined to force the operator to input a valid value. This applies to the default profile that comes with Heritrix. Other examples would be to leave the seed list empty or not specify a mandatory processor.

Profiles can be used as templates by leaving their configuration settings in an invalid state. In this way, an operator is forced to choose his or her settings when creating a job from a profile. This can be advantageous when an administrator must configure many different crawl jobs to accommodate his or her crawling policy.

Politeness Politeness refers to attempts by the crawler software to limit the load on a site it is crawling. Without politeness restrictions the crawler might overwhelm smaller sites and even cause moderately sized sites to slow down significantly. Unless you have express permission to crawl a site aggressively, you should apply strict politeness rules to any crawl.

Queue States

State	Meaning
ready	Queues ready to emit a URL now.
in-process	Queues that have emitted a URL that is currently being processed.
snoozed	Due to the crawl delay, or waiting before retries.
active	Total in-process + ready + snoozed
inactive	Queues currently not being considered (due to queue rotation).
ineligible	Inactive queues where the queue precedence exceeds the precedence floor.
retired	Disabled for some reason, e.g. that queue has hit it's allocated quota.
exhausted	Queues that are now empty.

Queued URIs The number of URIs queued and waiting for processing. Queued URIs include any URIs that failed to be fetched but will be retried.

Regular Expressions All regular expressions in Heritrix are Java regular expressions.

Java regular expressions differ from those used in other programming languages, like Perl. For detailed information on Java regular expressions see the Java API description of the `java.util.regex.Pattern` class.

SHA1 The Secure Hash Algorithm (SHA) used by Heritrix to encrypt files.

Server A server is a service on a host. There may be more than one service on a host. Different services are usually differentiated by port number.

Spring Spring is a Java application framework used by Heritrix. Crawl jobs are based on Spring components, known as "beans." In order to view the Spring beans of a crawl configuration, use the Browse Beans functionality.

SURT SURT stands for Sort-friendly URI Reordering Transform. It is a transformation applied to URIs that makes their left-to-right representation better match the natural hierarchy of domain names.

A URI `scheme://domain.tld/path?query` has a SURT form of `scheme://(tld,domain,)/path?query`.

Conversion to SURT form also involves making all characters lowercase and changing the https scheme to http. Further, the "/" character after a URI authority component will only appear in SURT form if it appears in plain URI form. An example of a URI authority component is the third slash in a regular HTTP URI. This convention proves important when using real URIs as a shorthand for SURT prefixes.

SURT form URIs are typically not used to specify exact URIs for fetching. Rather, SURT form is useful when comparing or sorting URIs. URIs in SURT format sort into natural groups. For example, all "archive.org" URIs will be adjacent, regardless of subdomains such as "books.archive.org" or "movies.archive.org."

Most importantly, a SURT form URI, or a truncated version of a SURT form URI can be used as a SURT prefix. A SURT prefix will often correspond to all URIs within a common area of interest. For example, the prefix `http://(is,` will be shared by all URIs in the `.is` top-level domain.

SURT Prefix A URI in SURT form, especially if truncated, may be of use as a "SURT prefix," a shared prefix string of all SURT form URIs in the same area of interest. For example, the prefix `http://(is.,` will be shared by all SURT form URIs in the `.is` top-level domain. The prefix `http://(org,archive.www,)/movies` will be shared by all URIs at www.archive.org with a path beginning with /movies. `http://(org,archive.www,)/movies` is also a valid full SURT form URI.

A collection of sorted SURT prefixes is an efficient way to specify a desired crawl scope. For example, any URI whose SURT form starts with any of the prefixes should be included.

A small set of conventions can be used to calculate an "implied SURT prefix" from a regular URI, such as a URI supplied as a crawl seed. These conventions are:

1. Convert the URI to its SURT form.
2. If there are at least three slashes ("/") in the SURT form, remove everything after the last slash. For example, `http://(org,example,www,)/main/subsection/` is unchanged. `http://(org,example,www,)/main/subsection` is truncated to `http://(org,example,www,)/main/`. `http://(org.example,www,)/` is unchanged and `http://(org,example,www)` is unchanged.
3. If the resulting form ends in an off-parenthesis (")"), remove the off-parenthesis. Each of the above examples except the last one is unchanged. The last one `http://(org,example,www,)` becomes `http://(org,example,www,`.

This allows many seed URIs, in their usual form, to imply the most useful SURT prefixes for crawling related URIs. The presence or absence of a trailing "/" on URIs without further path-info is a subtle indicator as to whether subdomains of the supplied domain should be included.

For example, seed `http://www.archive.org/` will become SURT form and supplied SURT prefix `http://(org,archive,www,)/`, and is the prefix of all SURT form URIs on www.archive.org. However, any subdomain URI like `http://homepages.www.archive.org/directory` would be ruled out because its SURT form `http://(org,archive,www,homepages,)/directory` does not begin with the full SURT prefix, including the ")" deduced from the seed.

Toe Threads When crawling, Heritrix employs a configurable number of Toe Threads to process URIs. Each of these threads will request a URI from the Frontier, apply the set of Processors to it, and finally report it as completed to the Frontier.

6.1 Status codes

Each crawled URI gets a status code. This code (or number) indicates the result of a URI fetch in Heritrix.

Codes ranging from 200 to 599 are standard HTTP response codes and information about their meanings is available at the World Wide Web consortium's Web page.

Other Heritrix status codes are listed below.

1	Successful DNS lookup
0	Fetch never tried (perhaps protocol unsupported or illegal URI)
-1	DNS lookup failed
-2	HTTP connect failed
-3	HTTP connect broken
-4	HTTP timeout
-5	Unexpected runtime exception. See runtime-errors.log.
-6	Prerequisite domain-lookup failed, precluding fetch attempt. (the main pre-requisite is WHOIS lookup. If you see this it's likely the domain doesn't exist anymore)
-7	URI recognized as unsupported or illegal.
-8	Multiple retries failed, retry limit reached.
-50	Temporary status assigned to URIs awaiting preconditions. Appearance in logs may be a bug.
-60	URIs assigned a failure status. They could not be queued by the Frontier and may be unfetchable.
-61	Prerequisite robots.txt fetch failed, precluding a fetch attempt.
-62	Some other prerequisite failed, precluding a fetch attempt.
-63	A prerequisite (of any type) could not be scheduled, precluding a fetch attempt.
-404	Empty HTTP response interpreted as a 404.
-3000	Severe Java Error condition occured such as OutOfMemoryError or StackOverflowError during URI processing.
-4000	"Chaff" detection of traps/content with negligible value applied.
-4001	The URI is too many link hops away from the seed.
-4002	The URI is too many embed/transitive hops away from the last URI in scope.
-5000	The URI is out of scope upon reexamination. This only happens if the scope changes during the crawl.
-5001	Blocked from fetch by user setting.
-5002	Blocked by a custom processor, which could include the hash mapper (for multi-node crawling) if enabled.
-5003	Blocked due to exceeding an established quota.
-5004	Blocked due to exceeding an established runtime
-6000	Deleted from Frontier by user.
-7000	Processing thread was killed by the operator. This could happen if a thread is an a non-responsive condition.
-9998	Robots.txt rules precluded fetch.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

HTTP Routing Table