Tokenizer Distributed Executor:  
User Guide

Contents

[Overview 2](#_Toc386201506)

[Cassandra Column Families 2](#_Toc386201507)

[url\_records 2](#_Toc386201508)

[Key 2](#_Toc386201509)

[Columns 2](#_Toc386201510)

[url\_sitemap\_idx 3](#_Toc386201511)

[Key 3](#_Toc386201512)

[timestamp\_url\_idx 4](#_Toc386201513)

[Key 4](#_Toc386201514)

[Columns 4](#_Toc386201515)

[message\_url\_idx 4](#_Toc386201516)

[Key 4](#_Toc386201517)

[Columns 4](#_Toc386201518)

[webpage\_records 4](#_Toc386201519)

[Key 4](#_Toc386201520)

[Columns 4](#_Toc386201521)

[xml\_records 5](#_Toc386201522)

[Key 5](#_Toc386201523)

[Columns 5](#_Toc386201524)

[message\_records 6](#_Toc386201525)

[Key 6](#_Toc386201526)

[Columns 6](#_Toc386201527)

# Overview

Tokenizer Distributed Executor (or simply TDE) is an application platform and development framework to run Big Data solutions in a clustered multicomputing environment.

This document describes practical solutions (also called “Distributed Tasks”) provided on top of TDE. It is frequently updated; and most recent version is always available at GitHub repository.

# Cassandra Column Families

## url\_records

### Key

Type: BYTES

Description:

### Columns

#### fetchedUrl

#### fetchTime

Type: LONG

Description: timestamp of last fetch attempt

#### contentType

Type: ASCII

Desciption: HTTP Header in response corresponding to MIME content type

#### responseRate

Type: INTEGER

Description: reserved for future needs

#### headers

Type: BYTES

Description: all HTTP headers serialized

#### newBaseUrl

Type: ASCII

Description: final landing URL in case of redirects

#### numRedirects

Type: INTEGER

Description: number of redirects

#### hostAddress

Type: ASCII

Description: Not used. Supposed to be IP address to avoid harmful hits of same IP possibly serving thousands of hosts.

#### httpStatus

Type: INTEGER

Description: HTTP Response Code

#### reasonPhrase

Type: ASCII

Description: HTTP Response Message

#### host

Type: ASCII

Description: “host” part of the URL; indexed

#### host\_fetchAttemptCounter

Type: ASCII

Description: “host” field plus “attempt counter”; indexed; needed for quick finding of URLs to be fetched

#### webpageDigest

Type: BYTES

Description: this is foreign key to “webpage” tablespace. MD5 is calculated for successfully fetched web page, and it is primary key for “webpage” object.

## url\_sitemap\_idx

### Key

The “key” is just URL.

Type: ASCII

Description: URLs listed in the sitemaps. Right now we only insert field into this table without any subsequent reuse; reserved for future needs.

## timestamp\_url\_idx

### Key

“host”

### Columns

<timestamp, URL>

This is “wide” table; single row contains multiple pairs of <timestamp, URL>

## message\_url\_idx

Message -> URL inverted index.

### Key

MD5 Hash of the Message.

### Columns

URLs of pages containing this message. Many URLs can contain the same Message.

## webpage\_records

Fetched pages, including all HTTP response headers and extra post-processing data.

### Key

MD5 Hash of the raw content (bytearray), so that we avoid duplicate (same) pages from different URLs (such as error response, redirect to login screen, and etc.)

### Columns

#### host

“host” part of the URL

#### baseUrl

Initial URL

#### fetchedUrl

Final URL; could be different in case of redirects.

#### fetchTime

Timestamp of last fetch attempt

#### content

Raw array of bytes representing ‘content’ of HTTP response

#### contentType

“Content-Type” HTTP response header; it contains encoding to be applied to byte array in case of plain text / html, and it contains MIME type of the response.

#### responseRate

Reserved for future needs. May contain “speed” characteristics for specific site – so we can decide how frequent can we hit the same site. Compare with Googlebot which can dynamically adjust crawl speed + webmasters are allowed to tune it too via Webmaster Tools.

#### headers

Serialized “Metadata” object representing all HTTP response headers.

#### newBaseUrl

Final URL in case of permanent redirects.

#### numRedirects

Number of redirects

#### hostAddress

IP address of a host

#### httpStatus

HTTP response status such as 200, 404, etc.

#### reasonPhrase

HTTP response phrase such as “Ok”, “Error”, etc.

#### host\_extractOutlinksAttemptCounter

The index consisting of “host” and “counter” concatenated field, such as “kaypok.com\_0” “kaypok.com\_1”, and etc. It is used to quickly find pages to be processed by Outlink Extractor.

#### host\_splitAttemptCounter

The index consisting of “host” and “counter” concatenated field, such as “kaypok.com\_0” “kaypok.com\_1”, and etc. It is used to quickly find pages to be processed by Html Splitter.

#### xmlLinks

List of foreign keys pointing to extracted XML records in another table.

## xml\_records

### Key

MD5 Hash of the XML Snippet

### Columns

#### host

“host” part of the URL; index

#### mainSubject

The main page subject / title such as hotel name, book name, and etc.

#### timestamp

Last fetch timestamp

#### content

Serialized byte array representing XML; encoded in UTF-8.

#### host\_parseAttemptCounter

The index for quick finding records to be parsed/processed/reprocessed.

## message\_records

### Key

MD5 Hash of the message. If message was generated from XML-Record then it will be copied from XML-Record-Key. In other cases, such as Twitter Stream, it will be generated based on content.

### Columns

#### host

Host

#### topic

Not used

#### date

Date

#### author

Author

#### age

Age

#### sex

Male/Female

#### title

#### content

Content

#### userRating

Rating

#### location

Location

#### mainSubject

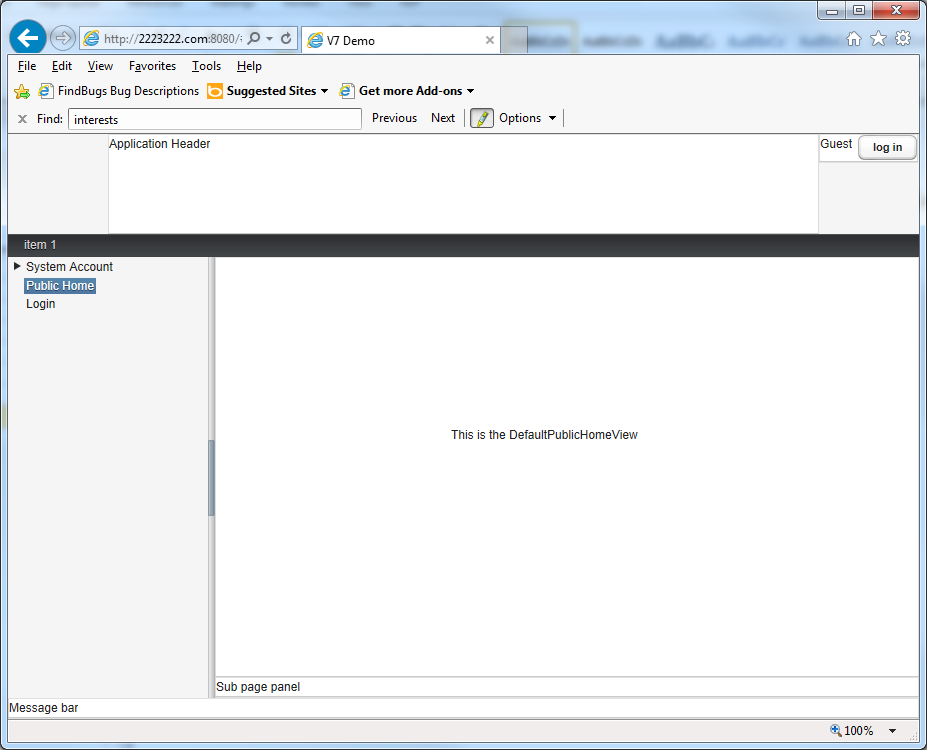
Subject (copied from corresponding XML Record)

#### reviewText

NLP Tools generate this “fat” object: it contains sentiment, features, and sentences. Serialized into byte array using standard Java serialization.

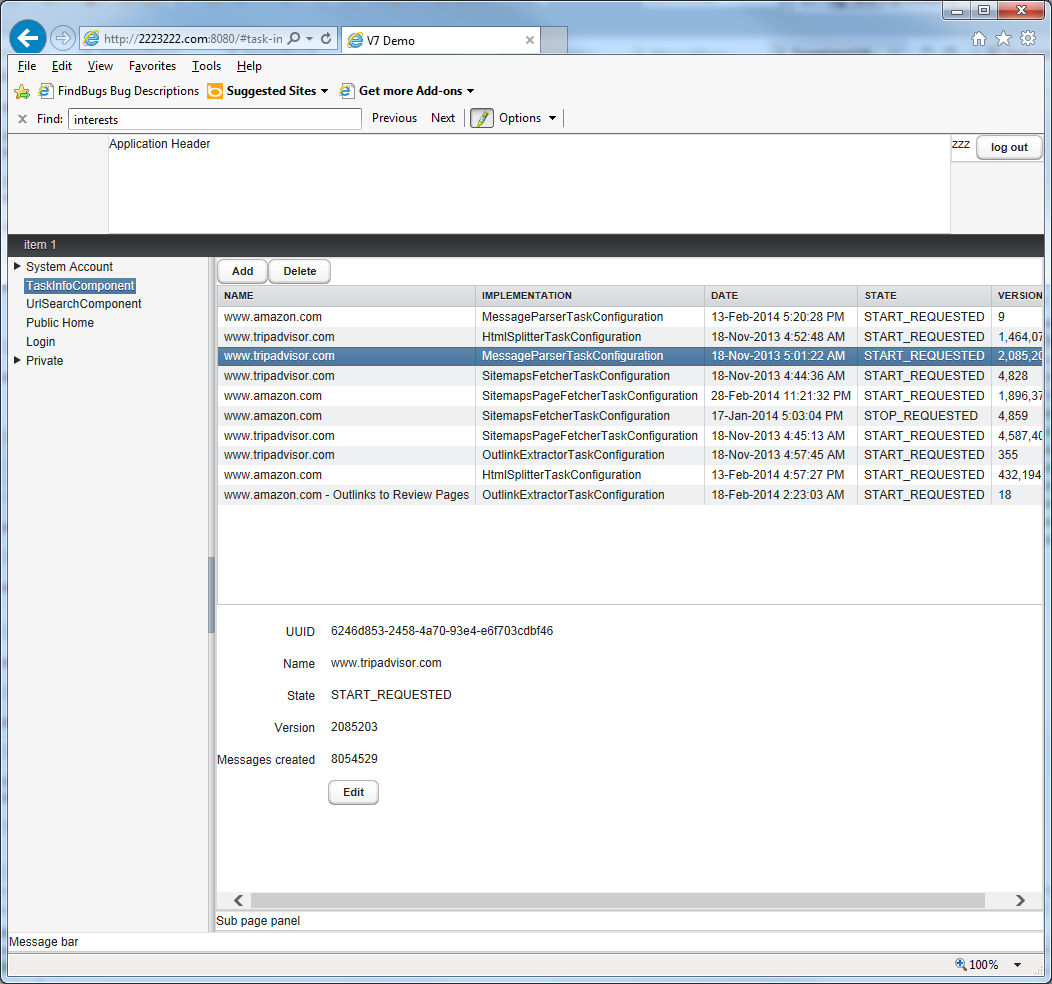
# Administrator GUI

This is default landing page as of today,



To login, as of today, username is anything, and password is just word “password”.

Upon successful login, left side menu panel will contain link to restricted pages:



On this screen, you can see statistics updates in real time, you can start/stop/delete tasks, create new tasks, reconfigure existing tasks. UI is intuitive, self-descriptive.

# Distributed Tasks

Using GUI you can create instance of a distributed task. Distributed task will run in a cluster of agents, and configuration of some tasks allows, for instance, to run 25 instances of the task in a cluster of 5 computers. In case of node failure (computer shutdown) remaining 4 nodes will run the same 25 task instances; and etc.

Clustered Singleton means that only single task instance can exist in a cluster of N nodes.

Clustered Multiton means that multiple (preconfigured) M task instances can exist in a cluster of N nodes.

Following are descriptions of existing tasks.

## Classic Robot Task

Clustered Singleton. Fetches pages from a specified host, and does not fetch from other hosts.

This task will fetch home page of the host, parse it, retrieve outlinks, filter “outgoing” outlinks, and fetch these filtered outlinks, and subsequently, in a loop, for all pages from the same host.

Configuration includes host address.

## Sitemaps Fetcher Task

Clustered Singleton. Retrieves sitemaps from the specified site, parses it, and stores URLs in corresponding tables. It does not fetch HTML pages; it only retrieves sitemaps.

It stores URL in url\_records and url\_sitemap\_idx.

## Sitemaps Page Fetcher Task

Clustered Singleton. The word “sitemaps” should disappear in future; this task simply scans UrlRecords and retrieves those not yet retrieved / expired for the specified host. It does not parse HTML nor create & insert new UrlRecords; it only fetches URLs. It inserts WebpageRecord, and updates UrlRecord.

## Sitemaps Linked Page Fetcher Task (Deprecated)

Deprecated; not used anymore. Initially, Sitemaps Fetcher Task retrieved only pages listed in sitemaps, such as Amazon product pages, and Sitemaps Linked Page Fetcher Task retrieved, for instance, linked pages not necessarily listed in a sitemap, such as user review pages.

## Outlink Extractor Task

It scans Webpage Records, parses HTML, retrieves outlinks, and inserts those into URL Records. Sample of scenario: Amazon sitemaps list only product pages. We parse product pages, filter URLs containing word “review” inside, and store those in URL Records for future fetch.

## Html Splitter Task

It scans Webpage Records, parses HTML, splits it on multiple XMLs, and inserts those into XML Records. Sample: product review pages contain repeated structural “snippets”, multiple reviews. Configuration is based upon single XPath.

## Message Parser Task

It scans XML Records, and retrieves fields using preconfigured XPath. Fileds include age, author, topic, text, and etc. It stores processed object into Massage Records.

## Common Configuration for Robot

This is important: our robot has a signature which includes Email address, robot name, and link to the webpage describing technical specifics of our robot. Webmasters can restrict access using “robots.txt” file. For a sample of robot description page, see <http://nutch.sourceforge.net/docs/en/bot.html> - our robot follows the same rules.

All users of Tokenizer framework encouraged to have their own unique “bot.html” page, robot name, and contact Email.

## Common Configuration for URL Filter

URL Filter is based upon Regular Expressions. This is copied from predefined (default) filter settings:

# The default url filter.

# Each non-comment, non-blank line contains a regular expression

# prefixed by '+' or '-'. The first matching pattern in the file

# determines whether a URL is included or ignored. If no pattern

# matches, the URL is ignored.

# skip file: ftp: and mailto: urls

-(file|ftp|mailto):.\*

# skip image and other suffixes

-.\*\.(gif|GIF|jpg|JPG|png|PNG|ico|ICO|css|CSS|sit|SIT|eps|EPS|wmf|WMF|zip|ZIP|ppt|PPT|mpg|MPG|xls|XLS|gz|GZ|rpm|RPM|tgz|TGZ|mov|MOV|exe|EXE|jpeg|JPEG|bmp|BMP|js|JS)

# skip URLs containing certain characters as probable queries, etc. (uncomment)

# -.\*[?\*!@=].\*

# accept anything else

+.\*