Общество с ограниченной ответственностью

"Технологии системного анализа"

**РУКОВОДСТВО**

**Администратора**

**TextAppliance версия 1.7.2**



ОГЛАВЛЕНИЕ

[1 Введение 4](#_Toc519608096)

[2 Системные требования 5](#_Toc519608097)

[3 Основные сведения о системе 6](#_Toc519608098)

[3.1 Список используемых переменных 6](#_Toc519608099)

[3.2 Основные компоненты системы 6](#_Toc519608100)

[3.3 Общее расположение файлов в системе 7](#_Toc519608101)

[3.4 Основные служебные утилиты 7](#_Toc519608102)

[3.5 Управление системой 8](#_Toc519608103)

[3.5.1 Включение\Выключение системы 8](#_Toc519608104)

[3.5.2 Мониторинг состояния системы 8](#_Toc519608105)

[3.5.3 Управление экземплярами модулей 8](#_Toc519608106)

[3.5.4 Диагностика работы модулей системы 9](#_Toc519608107)

[3.5.5 Диагностика работы сервисов системы 9](#_Toc519608108)

[3.5.6 Управление бэкапами 10](#_Toc519608109)

[3.5.7 Восстановление после сбоев 12](#_Toc519608110)

[3.5.8 Восстановление системы после потери данных 12](#_Toc519608111)

[3.5.9 Добавление новых форматов и метаполей 13](#_Toc519608112)

[3.5.10 Синхронизация конфигурационных файлов 14](#_Toc519608113)

[3.6 Расширение системы 14](#_Toc519608114)

[3.6.1 Изменения состава серверов 15](#_Toc519608115)

[3.6.2 Изменение состава модулей 15](#_Toc519608116)

[3.7 Мониторинг системы 15](#_Toc519608117)

[3.7.1 Сбор и хранение метрик 16](#_Toc519608118)

[3.8 Оповещения системы 16](#_Toc519608119)

[3.8.1 Изменение списка почтовых адресов для оповещений 17](#_Toc519608120)

[4 Индексация документов 17](#_Toc519608121)

[4.1 Подготовка к индексации 17](#_Toc519608122)

[4.1.1 Создание новой коллекции 17](#_Toc519608123)

[4.2 Мониторинг процесса индексации 21](#_Toc519608124)

[4.3 Завершение индексации 22](#_Toc519608125)

[4.4 Откат после неудачной индексации 22](#_Toc519608126)

[4.5 Объем индексных баз 22](#_Toc519608127)

[4.6 Удаление коллекции 22](#_Toc519608128)

[5 Добавление документов из внешних ресурсов в очередь 23](#_Toc519608129)

[5.1 Составление задания 23](#_Toc519608130)

[5.2 Тестирование задания 26](#_Toc519608131)

[5.3 Подготовка к обходу 26](#_Toc519608132)

[5.4 Запуск обхода 27](#_Toc519608133)

[5.5 Мониторинг обхода 27](#_Toc519608134)

[5.6 Завершение обхода 28](#_Toc519608135)

[5.7 Регулярные обновления 28](#_Toc519608136)

[6 Добавление локальных документов в очередь 29](#_Toc519608137)

[6.1 Описание локального краулера 29](#_Toc519608138)

[6.2 Дополнительные опции локального краулера 29](#_Toc519608139)

[6.3 Формат метаописания для локального краулера 31](#_Toc519608140)

[6.3.1 Описание формата 31](#_Toc519608141)

[6.4 Определение формата загружаемого документа 32](#_Toc519608142)

[6.5 Формирование урла загружаемого документа 32](#_Toc519608143)

[6.6 Выводимые сообщения 32](#_Toc519608144)

[6.7 Регулярные обновления 33](#_Toc519608145)

[7 Тестирование индексации в виртуальной среде 33](#_Toc519608146)

[7.1 Значения основных переменных в виртуальной машине 33](#_Toc519608147)

[7.2 Доступ к виртуальной машине 34](#_Toc519608148)

[7.3 Основные операции с виртуальной машиной 34](#_Toc519608149)

[7.4 Разделяемые папки 34](#_Toc519608150)

[7.5 Тестирование индексации в виртуальной среде 34](#_Toc519608151)

[7.6 Проверка проиндексированных данных 35](#_Toc519608152)

[7.6.1 Проверка с помощью скриптов 35](#_Toc519608153)

[7.6.2 Проверка с помощью веб-интерфейса 37](#_Toc519608154)

[7.7 Решение проблем с виртуальной машиной 37](#_Toc519608155)

[7.7.1 Доступ по ssh не работает 37](#_Toc519608156)

# Введение

Настоящее руководство содержит общие сведения необходимые для администрирования системы, добавления новых документов в индекс.

# Системные требования

Минимальные требования к серверу:

* Процессор: Архитектура: amd64; Количество ядер: не менее 4; тактовая частота процессора не менее 2,4 Ггц.
* Оперативная память не менее 16 Гбайт.
* 2 дисковых накопителя HDD (объединенных в RAID 1); объемом не менее 3000 Гбайт.

Рекомендуемые требования к серверу:

* Процессор: Архитектура: amd64; Количество ядер: 16; тактовая частота процессора 2,6 Ггц или более.
* Оперативная память 64 Гбайт или более.
* 4 дисковых накопителя HDD (объединенных в RAID 10); скорость дисков 10000 rpm; суммарный объем не менее 6000 Гбайт, 2 SSD диска (объединенных в RAID 1), объемом 1000Гб, не менее.

Требования к операционной системе и программному обеспечению:

* ОС: Debian 8 amd64

#  Основные сведения о системе

##  Список используемых переменных

|  |  |
| --- | --- |
| **Название** | **Пример значения** |
| user\_name | puser |
| app\_name | Pak |
| data\_dir | /data/<app\_name> |
| config\_dir | /compiled/config/ |
| node\_name | pn |
| database\_name | metastorage |
| monitoring\_node | 192.168.1.1 |
| monitoring\_user | admin |
| monitoring\_pass | zxcvbnm |

Актуальные значения переменных нужно уточнить в приложении. Переменные заключаются в угловые скобки, например <data\_dir>. Вместо переменной в каждом случае требуется подставить соответствующее ей значение.

В этом документе используются стандартные обозначения при указании команд. Команды, начинающиеся со знака доллара ($), должны быть выполнены пользователем <user\_name>. Команды, начинающиеся со знака хеша (#), должны быть выполнены от пользователя root.

## Основные компоненты системы

В систему входят модули, программы, реализующие сервисы (CGI-скрипты) и база данных, хранящая метаинформацию.

**Модули**

В системе присутствуют следующие модули:

1. Index (Индексатор) – модуль индексирования и поиска документов.
2. IP (Интеллектуальный анализатор документов) – модуль анализа документов. Используется только CGI-скриптами.
3. DocAnalyzer (Анализатор) – модуль анализа документов. Проводит анализ документов при индексировании.
4. TextAnalyzer (Анализатор запросов) – модуль анализа текстов. Используется для анализа поисковых запросов.
5. Storage (Хранилище) – модуль, реализующий хранение HTML-копий документов.
6. Queue (Очередь) – модуль, производящий хранение документов, подлежащих индексации.
7. Crawler (Краулер) – модуль для обхода внешних источников документов.
8. Frontier (Фронтир) – модуль, управляющий обходом внешних источников.
9. Logger (Логгер) – модуль централизованного логирования.
10. LocalCrawler – модуль для добавления локальных документов (документов, представленных в файловой системе).

Модули могут быть запущены в нескольких экземплярах. Описание процесса регулирования составом модулей и количеством запущенных экземпляров представлено в п. . Каждый экземпляр модуля должен иметь уникальное имя. Имя формируется следующим образом: <base\_name>\_<num>, где <base\_name> название модуля (допустимые значения перечислены в списке выше), <num> порядковый номер экземпляра в системе. **Примеры**: IP\_1, IP\_2, Index\_1, Storage\_1. Далее в этом руководстве под модулем будет пониматься один из его экземпляров, если не оговорено иное.

***CGI-скрипты*** – набор исполняемых файлов, запускаемых под управлением Веб-сервера, и предназначенных для взаимодействия с системой по протоколу JSON-RPC/HTTP. Они взаимодействуют с модулями системы по внутреннему программному интерфейсу. Подробный список скриптов и их зависимостей от модулей представлен в п. 3.5.5.

***база данных*** используется для хранения метаинформации документов. Она необходима для корректной работы некоторых CGI-скриптов. Имя базы данных – <database\_name>.

## Общее расположение файлов в системе

Все исполняемые файлы и разделяемые библиотеки находятся в каталогах /compiled/bin и /compiled/lib соответственно. Рабочие директории экземпляров модулей, запущенных в системе создаются в директории <data\_dir>. Название рабочей директории экземпляра совпадает с его именем (Index\_1, IP\_2, IP\_3). Все данные хранятся в этой директории: индексные базы – .<data\_dir>/Index\*, хранилище документов – <data\_dir>/Storage\*.

Конфигурационные файлы находятся в каталоге <config\_dir>.

Таблица 1основные конфигурационные файлы

|  |  |
| --- | --- |
| **Имя конфигурационного файла** | **Краткое описание** |
| collections.xml | Описание доступных коллекций документов |
| formats.xml | Описание поддерживаемых форматов документов |
| newpool.xml | Конфигурационный файл, задающий команды для преобразования документов разных форматов в html |
| descriptors.xml | Описание метаполей подлежащих индексации.  |
| search\_tags.xml | Описание метаполей, по которым разрешается осуществлять поиск |
| metastorage.conf | Настройки доступа к базе данных |
| cgi\_scripts.cfg | Конфигурационный файл для всех cgi-скриптов, запускаемых на этом сервере |
| colls.d, idx.cfg | Конфигурационные файлы индекса |
| stor\_colls.d | Конфигурационные файлы хранилища |

## Основные служебные утилиты

Таблица 2 основные служебные утилиты

|  |  |  |
| --- | --- | --- |
| **Название** | **Основные подкоманды** | **Краткое описание** |
| iceadmin.sh | list, disable, enable, stop, start, state, describe | Утилита для управления модулями  |
| iceapp.sh | list, add, remove, describe, diff, update | Утилита управления составом модулей в системе |
| backup.sh | - | Утилита для создания резервных копий данных, используемых модулями |
| crawlers\_stat.sh | - | Утилита для составления краткого отчета о ходе обхода |
| StatMonitor | - | Выполняет сбор статистики модулей |
| Printer | - | Осуществляет печать документов во внутреннем представлении (например, документов, хранящихся в очереди) |

Обязательным входным параметром этих утилит является имя модуля. Вместо имени может быть передано регулярное выражение, определяющее имя одного или нескольких модулей. Регулярное выражение должно быть совместимо с Posix Extended Regular Expressions. **Примеры:**

$ iceadmin.sh state Index\_1

$ iceadmin.sh state IP.\*

## Управление системой

### Включение\Выключение системы

Для запуска системы на узле выполните команду:

$ sudo service exactus start

Для остановки системы на узле выполните команду:

$

Этот скрипт выполнятся автоматически при включении и выключении системы. Соответственно при плановых операциях, связанных с перезагрузкой или выключением узла система сможет корректно остановиться.

### Мониторинг состояния системы

Для проверки состояния системы выполните:

$ iceadmin.sh state all

В результате будет выведен список модулей, входящих в состав системы. Большинство модулей должно находиться в состоянии active. **Пример вывода**:

Server - Pak.Index\_1

**active** (pid = 7737, enabled).

Основные модули, которые должны быть в состоянии active для корректного функционирования системы: IP, TextAnalyzer, Index, Logger.

Для получения статистики по отдельному модулю можно указать его имя. **Пример:**

$ iceadmin.sh state Index\_2.

Для получения информации о всех экземплярах данного модуля, можно использовать регулярное выражение. **Пример:**

$ iceadmin.sh state In.\*

В случае неактивности какого-либо модуля необходимо изучить его логи для диагностирования ошибки. Подробности описаны в п. 3.5.4.

### Управление экземплярами модулей

Для включения экземпляра модуля или группы экземпляров необходимо выполнить следующую команду:

$ start.sh < instance\_regexp> <instance\_regexp>…

**Примеры:**

$ start.sh Index.\*

$ start.sh Queue\_1

$ start.sh Index.\* Storage.\*

Для выключения отдельного модуля используйте следующую команду:

$ halt.sh <instance\_regexp> <instance\_regexp>…

**Пример:**

$ halt.sh Index.\*

$ halt.sh Index.\* Storage.\*

Для перезагрузки отдельного модуля используйте следующую команду:

$ restart.sh <instance\_regexp> <instance\_regexp>…

**Пример:**

$ restart.sh Index.\*

$ restart.sh Index.\* Storage.\*

### Диагностика работы модулей системы

Локальные логи каждого запущенного модуля хранятся в директории <data\_dir>/<instance\_name>/. В этой директории присутствует 4 вида логов.

Таблица 3Типы логов каждого модуля

|  |  |
| --- | --- |
| **Путь к логу** | **Тип информации в логе** |
| <data\_dir>/<instance\_name>/<instance\_name>.log | информация, связанная с логикой работы модуля |
| <data\_dir>/<instance\_name>/<instance\_name>.icelog | Информация, связанная с коммуникацией с другими модулями системы |
| <data\_dir>/<instance\_name>/<instance\_name>.stderr | Информация, которая пишется в обход логгера модуля в стандартный поток ошибок |
| <data\_dir>/<instance\_name>/<instance\_name>.stdout | Информация, которая пишется в обход логгера модуля в стандартный поток вывода |

При возникновении проблем с запуском модулей полезная информация также может содержаться в файлах /data/<node\_name>/<node\_name>.stderr и /data/<node\_name>/<node\_name>.stdout.

Также логи всех экземпляров записываются модулем Logger в формате json. Чтобы определить сервер, на котором расположен централизованный логгер (для инсталляций развернутых на нескольких серверах), нужно выполнить:

$ exsystemctl locate\_node -i Logger.\*

На сервере логгера:

$ cd <data\_dir>/Logger\_1

В папке Logger\_1 расположены директории для каждого модуля.

**Пример**:

$ cd IntelProc

$ tail IntelProc.json | jq .

При диагностировании проблем с группой модулей имеет смысл оставить включенным только один экземпляр модуля.

### Диагностика работы сервисов системы

Для тестирования работы сервисов, предоставляемых системой, выполните следующую команду:

$ test\_services.py

*Стоит отметить, что сразу после включения системы большая часть сервисов может не работать, так как индексу нужно время, чтобы загрузить все коллекции в память. Следить за прогрессом загрузки можно в логе модуля индексатора. Сообщение, свидетельствующее об успешной загрузке: End of initialization of metrics.*

В случае возникновения ошибок диагностику сервисов системы имеет смысл начать с изучения логов веб-сервера apache. Ошибки выполнения можно обнаружить в логе /var/log/apache2/error.log

Кроме того каждая программа пишет в лог <data\_dir>/Logger\_1/<program\_name>/<program\_name>.json

Каждая программа, реализующая сервис, взаимодействует с одним или несколькими модулями системы. Поэтому в случае возникновения неполадок нужно удостовериться, что необходимые сервису модули запущены и работают без ошибок (логирование модулей описано в п. 3.5.4). В следующей таблице представлены зависимости сервисов системы. Программе могут требоваться все экземпляры модуля (в таблице выделено жирным), либо только один экземпляр, выбираемый динамически во время выполнения (выделено курсивом).

Таблица 4 Зависимости программ, реализующих сервисы.

|  |  |  |
| --- | --- | --- |
| **Название сервиса** | **Имя программы, реализующей сервис** | **Модули, используемые программой** |
| сервис семантического поиска | search.cgi | **Index**, *TextAnalyzer,* Logger |
| сервис поиска похожих документов | sim-doc-search.cgi | **Index**,Logger |
| сервис поиска текстовых заимствований | plagsearch.cgi | **Index**, *IP*, Logger |
| сервис поиска похожих документов для загружаемого документа | doc\_analyzer.cgi | *IP*, **Index,** Logger |
| сервис получения ключевых слов | doc-terms.cgi | **Index**,Logger |
| сервис построения расширенной аннотации | annot.cgi | **Index**, *TextAnalyzer,* Logger |
| сервис подсветки слов запроса в найденных документах | text-marking.cgi | **Storage**, *TextAnalyzer, IP,* Logger |
| сервис формирования и обработки пользовательских коллекций документов | local\_terms.cgi, local\_clustering.cgi, collectioncompare.cgi, collectiondefinitions.cgi | **Index**,Logger |

### Управление бэкапами

***Бэкап данных***

Бэкапы данных хранятся в директории <data\_dir>/backup . Там поддерживаются бэкапы модулей Index и Storage. Бэкап данных влечет за собой отключение вышеуказанных модулей, что делает недоступным часть сервисов (см. табл. 4). По умолчанию бэкапы делаются каждую неделю в ночь с субботы на воскресенье. Изменить это поведение можно с помощью планировщика cron. Для этого нужно выполнить команду:

$ crontab -e

Откроется текстовый файл со списком команд, выполняющих бэкап и их расписанием. Формат расписания может быть уточнен в руководстве утилиты cron, доступном в Интернете. Для отключения автоматического выполнения бэкапа, строку с командой нужно закомментировать, добавив символ # в начале строки.

Бэкап выполняется с помощью утилиты backup.sh. По умолчанию эта утилита выполняет бэкап индексов. Бэкап индексов настраивается через конфигурационный файл (<config\_dir>/post\_indexing.cfg). Для настройки поведение утилиты существует несколько опций:

Таблица 5Опции скрипта backup.sh

|  |  |  |  |
| --- | --- | --- | --- |
| **Ключ командной строки** | **Опция в конфигурационном файле** | **Описание** | **Значение по умолчанию** |
| -s | index\_names | Название модуля данные, которого нужно забэкапить, например Storage\_1. В этом примере будет скопирована директория <data\_dir>/Storage\_1. |  |
| -d | destination\_dir | Директория, в которую будут скопированы данные |  |
| -a | destination\_host | Адрес удаленного сервера, на который нужно скопировать данные, в формате [USER@]HOST. Между узлами должен быть настроен беспарольный ssh доступ. На удаленном сервере должен быть установлен пакет rsync. Если это опция не указано данные будут копированы на этот же узел. |  |
| -p | ssh\_port | Порт для подключения по ssh к удаленной машине. | 22 |
| -e | exclude\_log\_files | Исключить лог-файлы при копировании | False |
| -t | times\_backup | Количество старых бэкапов, которые надо хранить. Cтарые версии бэкапов именуются следующим образом <module\_name>\_old\_data.<num>, где чем меньше <num>, тем свежее бэкап. | 0 |
|  **Опции только для бэкапа индексов (т.е. задаются в <config\_dir>/post\_indexing.cfg)** |
|  | sync\_ssd\_to\_hdd | Если True, то копировать все индексы из <fast\_storage> в <hdd\_storage>. Это может быть полезно, чтобы держать индексы на hdd и ssd были синхронизированы. | False |
|  | fast\_storage | Путь к файлам индекса на ssd диске | /data/fast\_storage/index\_data |
|  | hdd\_storage | Путь к файлам индекса на hdd диске | /data/hdd\_storage/index\_data |
|  | calc\_index\_stat | Посчитать и напечатать статистику индексации с момента последнего бэкапа. | False |
|  | reset\_metrics | Пересчитать метрики индекса перед бэкапом. | False |
|  | find\_overloaded\_index\_stor | Напечатать предупреждение о потенциально переполненных индексах в конце бэкапа. | False |
|  | mail\_subject | Тема письма, которое будет отправлено в конце бэкапа со всей сводной информацией. Чтобы отправление писем работало, на хосте должен быть настроен postfix. Набор получателей и пр. опции задаются через конфигурационный файл <config\_dir>/mail.cfg |  |

***Бэкап системы***

Для того чтобы иметь возможность восстановить систему после потери части данных, нужно делать бэкапы следующих директорий на внешний носитель:

* /compiled, содержащая основной дистрибутив системы
* /data/<node\_name>
* /data/registry
* /data/<app\_name>/backup

### Восстановление после сбоев

В случае аппаратных сбоев или некорректном выключений сервера необходимо произвести копирование резервных копий модулей Index.

***Восстановление индексных баз***

Для восстановления всех коллекций нужно выполнить:

$ restore\_from\_backup.sh -i Index\_1

Восстановление коллекций 2000 и 2010:

$ restore\_from\_backup.sh -i Index\_1 -c "2000 2010"

Таким же образом происходит восстановление из более поздней версии бэкапа. **Пример**:

$ restore\_from\_backup.sh -i Index\_1 -b Index\_1\_old\_data.1

Во время восстановления коллекции будут поочередно отключаться, т.е. они будут недоступны для поиска и прочих сервисов.

***Восстановление индексации***

В случае если была запущена индексация, то необходимо также переиндексировать документы, которые были добавлены в индекс с момента последнего бэкапа. Перед этим необходимо остановить индексацию:

$ halt.sh DocAn.\*

После этого необходимо собрать идентификаторы документов, проиндексированных с момента последнего бэкапа и добавить эти документы в очередь:

$ cd <data\_dir>/Index\_1

$ exsystemctl collect\_ids -i Index\_1.log > index\_1\_ids

$ fill\_queue\_from\_storage --input\_path=index\_1\_ids --storage\_searcher\_obj=Storage\_1-searcher --queue\_obj\_name=Queue\_1

Команда fill\_queue\_from\_storage может занять долгое время, поэтому ее желательно запускать в сессии tmux или screen.

Если количество документов равно нулю, т.е. файл index\_1\_ids пустой, то команду fill\_queue\_from\_storage выполнять не следует.

Такую же операцию нужно провести для всех индексаторов на сервере.

После того как индексные базы восстановлены из бэкапа индексацию следует возобновить командой:

$ start.sh DocAn.\*

### Восстановление системы после потери данных

Для восстановления после потери данных необходим сервер, с настроенным TextApp.

***Если потеряно содержимое папки /srv***, то для экстренного восстановления работоспособности системы нужно провести следующие операции.

В первую очередь нужно восстановить системную директорию /compiled. Для этого нужно создать символьную ссылку

$ sudo mkdir /srv/compiled && ln -s /srv/compiled /compiled

$ sudo chown -R <user\_name>:<user\_name> /srv/compiled

После этого скопируйте в /compiled/ содержимое системной директории, сохраненной заранее на внешнем носителе.

Те же действия необходимо проделать с директорией /data.

$ sudo mkdir /srv/data && ln -s /srv/data /data

$ sudo chown -R <user\_name>:<user\_name> /srv/data

В этой директории в первую очередь должны находиться следующие каталоги: <node\_name>, registry, <app\_name>, tmp.

***Каталоги <node\_name> и registry*** являются служебными и должны быть восстановлены из внешних бэкапов.

***Каталог tmp*** должен быть просто создан и ему должны быть выставлены соответствующие права:

$ mkdir /data/tmp && chmod 777 /data/tmp

***Каталог <app\_name>*** содержит основные данные (индексы) необходимые для работы системы, а также логи и другую служебную информацию. Данные бэкапятся в /data/<app\_name>/backup, процесс бэкапа этих данных описан в . Чтобы восстановить эти данные требуется наличие бэкапа /data/<app\_name>/backup на внешнем носителе. Процесс восстановления индексов описан в . Требуется восстановить все индексаторы, т.е. директории с префиксом Index, но исключая директории с суффиксом \_old\_data.1. Модуль Storage нужно восстановить копированием из папки бэкап директории Storage\_1 в директорию <data\_dir>.

***База данных***

Также для функционирования системы необходимо восстановить базу данных. Сделать это можно, выполнив команду.

$ restore\_metastorage.sh

***Восстановление docker-образов***

Сначала нужно убедиться, что докер запущен:

$ sudo service docker restart

$ sudo service docker status˚

Статус должен быть active (running).

После этого необходимо пересобрать docker-образы.

$ cd ~/pretty-plag/all2pretty/docker\_all2pretty

$ ./build.sh

$ cd ~/pretty-plag/all2pretty/docker\_unoconv

$ ./build.sh

$ cd ~/pretty-plag/cgi-pretty-plagsearch/docker\_pretty\_plagsearch

$ ./build.sh

После этого требуется перезагрузить систему.

$ sudo service exactus restart

### Добавление новых форматов и метаполей

***формат***

Для добавления нового формата необходимо отредактировать файл <config\_dir>/formats.xml. Каждый формат описывается следующим образом:

<format name=" " id="" description="" mime=" "></format>

**Пример**:

<format name="ODT" id="101" description="" mime="application/vnd.oasis.opendocument.text">

</format>

Id – это номер формата. При добавлении новых форматов необходимо использовать идентификаторы из **интервала (100, 255)**.

Если необходимо добавить, например, несколько mime-типов для одного формата, то нужно добавить несколько таких записей, различающихся только mime-типом.

При добавлении формата также необходимо указать команду для конвертации документов этого формата в html. Это делается добавлением записи следующего вида в <config\_dir>/newpool.xml:

<elem type="common" from="odt" to="html" command="<comm>"/>

<comm> должна содержать параметр $input, вместо которого будет подставлен путь к входному файлу. Результат конвертации должен быть напечатан в стандартный поток вывода. **Пример команды**: /compiled/bin/doc2text.sh -I $input -O - 2> /dev/null

Если система развернута на нескольких узлах, необходимо синхронизировать конфигурационные файлы между ними (см. 3.5.10).

После добавления нового формата нужно перезагрузить краулеры: restart.sh Crawler.\*

После добавления новой команды нужно перезагрузить анализаторы: restart.sh IP.\*

***метаполя***

Для добавления метаполей необходимо отредактировать файлы, указанные в следующей таблице.

|  |  |
| --- | --- |
| **Путь к файлу** | **Назначение** |
| <config\_dir>/descriptors.xml | Используется при индексации документов |
| <config\_dir>/search\_tags.xml | Используется при поиске по проиндексированным коллекциям |

 Любое метаполе записывается в виде

<tag id="2" name="mf\_example" type="1" before="!" after="!" weight="0.85" hidden="1" mode="2"/>

Для определения метаполя используются следующие атрибуты:

|  |  |
| --- | --- |
| **Имя** | **Принимаемые значения** |
| id | id должен быть уникальным идентификатором среди всех метаполей. Разрешается использовать незанятые значения с 2 по 90 и с 206 по 255. Идентификаторы с 91 по 205 зарезервированы для внутреннего использования. |
| name | Название метаполя |
| hidden | 0 – значение метаполя учитывается при построении аннотации к поисковому запросу |
| 1 – значение метаполя не выводится в аннотации к поисковому запросу |
| weight | Вес метаполя, учитываемый при расчете релевантности |
| type | 0 – неизвестный тип |
| 1 – блочный тип. После значения метаполя будет добавлен перенос строки |
| 2 – строковой тип. В отличие от блочного типа перенос строки не добавляется |
| before | Символ, который должен быть добавлен после значения метаполя. В случае если тип метаполя блочный, то символ будет добавлен перед добавлением переноса строки. |
| after | Символ, который должен быть добавлен перед значением метаполя. |
| mode | 0 – метаполе не предназначено для индексации. Полезно для вспомогательных метаполей, например url, format. |
| 1 – стандартный html или xml тэг, например h1, a, b, i |
| 2 – метаполе, которое должно быть проиндексировано |

Если система развернута на нескольких узлах, необходимо синхронизировать конфигурационные файлы между ними (см. 3.5.10).

После добавления нового метаполя нужно перезагрузить анализаторы: restart.sh DocAnalyzer.\*

### Синхронизация конфигурационных файлов

Для синхронизации конфигурационных файлов между несколькими серверами системы нужно выполнить следующий набор команд.

$ cd /compiled/config

$ sync\_config.py -i descriptors.xml search\_tags.xml

Где descriptors.xml search\_tags.xml надо заменить набором файлов, которые требуется синхронизировать.

## Расширение системы

Конфигурация системы находится в файле <config\_dir>/<app\_name>.xml. Он содержит информацию о серверах, на которых запущена система (тэг node). Для каждого сервера перечисляется набор экземпляров модулей, которые должны быть на нем запущены.

### Изменения состава серверов

Для добавления нового сервера необходимо развернуть на новом сервере систему и добавить тэг node в файл <config\_dir>/<app\_name>.xml

<node name="*hostname*" load-factor="1.0">

 <variable name="node\_addr" value="*ip*"/>

</node>

заменив hostname на имя сервера, а ip на внешний адрес этого сервера (который доступен другим серверам из этой системы). После этого надо выполнить:

$ iceapp.sh update <config\_dir>/<app\_name>.xml

Для удаления сервера необходимо удалить соответствующий ему тэг node и выполнить:

 $ iceapp.sh update <config\_dir>/<app\_name>.xml

### Изменение состава модулей

Состав модулей задается в конфигурационном файле в тэге node. Каждый экземпляр модуля создается с помощью тега server-instance. **Пример:**

<server-instance template="DocAnalyzerTemplate" index="\_1" \>

В атрибуте template указывается имя шаблона модуля. Для создания очередного экземпляра нужно скопировать любой существующий server-instance с нужным модулем и изменить значение атрибута index таким образом, чтобы оно было уникальным среди всех значение index, использованных для этого модуля. Для удаления экземпляра модуля необходимо удалить любой server-instance.

После модификации конфигурационного файла необходимо выполнить команду:

$ iceapp.sh update <config\_dir>/<app\_name>.xml

## Мониторинг системы

TextApp поставляется с системой мониторинга и оповещения, которая мониторит потребляемые ресурсы, процесс индексации, и работу некоторых сервисов. Для доступа к web-интерфейсу системы мониторинга необходимо перейти по адресу http://<monitoring\_node>:3000. Логин и пароль указаны в переменных <monitoring\_user> и <monitoring\_pass>. В следующей таблице представлен список основных дашбордов, которые настроены по умолчанию.

Таблица 6 список основных дашбордов

|  |  |
| --- | --- |
| **Название** | **Краткое описание** |
| Main Servers | Основные метрики серверов, на которых развернута система: потребление RAM, активность CPU, load average, свободное место на дисках. |
| Software Raid mdadm | Информация о состоянии программного рейд массива. |
| Indexing | Основные метрики индексаторов: количество документов, количество документов, добавленное за последние 5 минут, количество документов на каждую коллекцию и пр. |
| Main Plag Info | Основные метрики сервиса поиска текстовых заимствований: кол-во проверок за последние пять минут, размер проверяемых файлов, кол-во файлов с сокрытиями и пр. |
| Plag Search Speed | Метрики сервиса поиска текстовых заимствований, отражающие скорость работы сервиса на разных стадиях. |

Для отображения метрик используется Grafana. Grafana обращается к prometheus для получения метрик (Prometheus описан в следующем пункте). Grafana установлена через docker, поэтому управление осуществляется через командную строку с помощью утилиты docker

### Сбор и хранение метрик

Prometheus осуществляет сбор и хранение метрик серверов и системы TextApp, обработку правил для отправки уведомлений. Конфигурацию можно посмотреть по адресу http://<monitoring\_node>:9090/config.

Файлом конфигурации для prometheus является /etc/prometheus/prometheus.yml.

В случае внесения изменений в файл конфигурации необходимо перезагрузить модуль с помощью команды:

# service prometheus restart

Данный модуль с некоторой периодичностью (по умолчанию 30c) обращается к target-ам. Список target-ов можно посмотреть адресу http://<monitoring\_node>:9090/targets.

Список target-ов можно изменить в файлах, расположенных в папке /etc/prometheus/targets. Изменения в существующих файлах подтянутся автоматически. При добавлении нового файла с target-ами необходимо убедиться в том, что данный файл указан в конфигурации prometheus-а.

Список правил для оповещений можно посмотреть по адресу http://<monitoring\_node>:9090/rules.

Список правил для оповещений можно изменить в файлах, расположенных в папке /etc/prometheus/alerts. При добавлении нового файла с правилами необходимо убедиться в том, что данный файл указан в конфигурации prometheus-а. Если срабатывает некоторое правило для уведомлений, то prometheus обращается к alertmanager для отправки сообщения, который непосредственно отправляет уведомления, например, на электронную почту. Prometheus поставляется с модулем node\_exporter, который осуществляет сбор метрик серверов: RAM, CPU, диски, сеть, и т.д. Предоставляет аппаратные метрики серверов. Основные из этих метрик можно посмотреть в дашборде «Main Servers».

## Оповещения системы

В системе существуют следующие типы оповещений:

* **экстренные оповещения**. Срабатывают в том случае, если место на диске в указанных директориях становится меньше, чем указано в секции watcher в конфигурации <config\_dir>/post\_indexing.cfg. В опции disk\_req через запятую перечислены директории с требуемым свободным пространством по формату <путь к папке>:<размер свободного пространства в байтах>. Проверка наличия свободного места запускается ежеминутно как cron-задача от пользователя root. В случае если место закончилось во время индексации, то индексация будет остановлена и будет запущен процесс дефрагментации индексных баз с целью увеличения свободного пространства. В случае, если это место было освобождено, то будет выслано повторное оповещение с сообщением, что свободное место появилось и индексация возобновлена. В противном случае оповещения будут периодически приходить на почту с напоминаем, о том что необходимо освободить место на диске.

Также в файле конфигурации <config\_dir>/post\_indexing.cfg указана тема email-сообщения (параметр mail\_subject).

* **backup оповещения.** Срабатывают после совершения бэкапа. Письмо с соответствующей темой содержит следующую информацию о совершенном бэкапе:
	+ время бэкапа;
	+ информацию о проиндексированных документах с разбиением по коллекциям с момента предыдущего бэкапа;
	+ информацию о количестве запусков и выключений индексатора, запусков дефрагментации индексных структур, уменьшение объема хранилища слов и пр.
* **prometheus оповещения.** Prometheus отправляет уведомление, если срабатывает правило из http://<monitoring\_node>:9090/rules. Prometheus отправляет уведомление alertmanager-у, а alertmanager группирует однотипные сообщения за небольшой интервал времени и рассылает по списку почтовых адресов. Уведомления могут быть о сильной нагрузке на сервере, о нехватке RAM, о частых ошибках при индексации, о частых ошибках сервиса поиска текстовых заимствований и пр.

### Изменение списка почтовых адресов для оповещений

По умолчанию настроены уведомления на электронные адреса. Для изменения списка почтовых адресов необходимо изменить конфигурации в двух местах:

* Для экстренных и backup-оповещений. Изменить раздел mail\_addrs в файле <config\_dir>/mail.cfg. Изменения начнут действовать сразу после сохранения файла.
* Для aletmanager-а. Отредактировать раздел receivers в файле /etc/prometheus/alertmanager.yml. После внесения изменений необходимо перезагрузить alertmanager командой:

# service alertmanager restart

# Индексация документов

Для добавления документов в индекс необходимо создать новую коллекцию, либо «доиндексировать» новые документы в существующие коллекции. Процесс индексации начинается, когда в очередь (модуль Queue) попадают документы. Процессу добавления документов в очередь посвящены пункты 5 и 6. Из очереди документы забираются экземплярами модуля DocAnalyzer. После проведения всех видов анализа они попадают в Индекс (модуль Index). Схематичное изображение процесса индексации представлено ниже.



## Подготовка к индексации

### Создание новой коллекции

При создании новой коллекции необходимо учесть, что нужно использовать идентификаторы из интервала **[3000,10000)**. Эти идентификаторы зарезервированы за клиентами, что позволит избежать коллизий идентификаторов с коллекциями, поставляемыми вместе с TextApp.

Для генерации конфигурационных файлов для новой коллекции используется следующая команда:

$ manage\_coll -c <coll> create -n "<name>" --est\_docs\_count=<docs\_count>

Где <coll> – идентификатор создаваемой коллекции, <name> – короткое название этой коллекции (будет добавлено в файл <config\_dir>/collections.xml), <docs\_count> – предполагаемое количество документов в этой коллекции (точное значение не требуется, главное передать порядок, т.е. будут индексироваться миллионы документов, сотни тысяч или тысячи).

***Создание коллекции: система на одном узле***

Для простых инсталляций системы (развернутых на одном узле), достаточно просто выполнить следующую команду.

**Пример:**

$ manage\_coll -c 3000 create -n "Test Collection" --est\_docs\_count=5000

Команда выведет названия модулей, для которых будут созданы коллекции (надо набрать y и нажать Enter). После этого модули будут перезагружены и коллекция готова к работе.

***Создание коллекции: инсталляция на нескольких узлах***

Если система распределена на несколько узлов (и как следствие имеет несколько индексаторов), необходимо указать конкретные экземпляры индексатора и хранилища документов, для которых нужно создать коллекцию.

**Примеры:**

$ manage\_coll -c 3000 create -n "Test Collection" --est\_docs\_count=5000 -m Index\_1\_1 Storage\_1\_1

Можно задавать название модулей через регулярное выражение:

$ manage\_coll -c 3000 create -n "Test Collection" --est\_docs\_count=5000 -m Index\_.\* Storage\_.\*

Команда выведет названия модулей, для которых будут созданы коллекции, если это то, что нужно, то надо набрать ‘y’ и нажать Enter. После этого модули будут перезагружены и коллекция готова к работе.

***Создание коллекции для объемных документов***

Если планируется индексации объемных документов (книг, учебников, монографий), то имеет смысл также указать среднее количество слов в документе из этой коллекции. Значение по умолчанию для параметра <words\_count> равно 4000. Этого достаточно для большинства документов (статья, студенческий реферат, студенческая курсовая). Однако, для книг и монографий средний размер документа значительно больше, поэтому следует его изменить. Сделать это можно с помощью опции --est\_words\_count <words\_count>.

**Пример:**

manage\_coll -c 3000 create -n "Test Collection" --est\_docs\_count=5000 --est\_words\_count=60100

***Дополнительные опции созданной коллекции***

* Фильтрация дубликатов при индексации **(Экспериментальная опция!)**. Для коллекции можно включить автоматическую фильтрацию дубликатов. Сделать это можно, добавив в конец конфигурационного файла коллекции (например, /compiled/config/colls.d/2000.col): [dup\_tester]

enable\_dup\_tester=1

По умолчанию дубликаты будут искаться только в этой коллекции. Если добавить опцию colls=0, то дубликаты будут искаться по всем коллекциям, которые обнаружены в индексе. Также можно передать список коллекций, в которых нужно искать дубликаты, например, colls=2000 2010 31.

***Создание коллекции на сервере с ssd диском***

Если на сервере установлен ssd диск и он настроен для работы с системой, т.е. существует символьная ссылка /data/fast\_storage, ведущая на одну из директорий на ssd диске, то часть индексных баз будет перенесена на ssd диск после создания новой коллекции автоматически. Распределение индексных баз по дискам осуществляется согласно файлу <config\_dir>/index\_storage\_schema.json. В этом файле могут быть заданы настройки для конкретного индекса (Index\_5\_2 как в примере) или для конкретной коллекции (2000 как в примере). Также там присутствуют значения по умолчанию – ключ со значением “DEFAULT”.

{

 "DEFAULT": {

 "DEFAULT": {

 "ri": "fast",

 "rsi": "fast",

 "flat\_idx": "fast",

 "direct\_idx": "hdd",

 "docsinfo": "hdd",

 "doc\_store": "hdd",

 "spidx": "hdd",

 "glob\_cnt": "hdd",

 "ribuf": "hdd",

 "rsibuf": "hdd"

 },

 "2000": {

 "direct\_idx": "fast"

 }

 },

 "Index\_5\_2": {

 "DEFAULT": {

 "flat\_idx": "hdd"

 },

 "2000": {

 "ri": "hdd"

 }

 }

}

В случае если схема размещения индексных баз была изменена для индекса Index\_1 и коллекции 2000 следует выполнить скрипт:

$ switch\_index\_storage.sh -i Index\_1 -c 2000

Если необходимо переместить файлы всех коллекций индекса Index\_1, то нужно выполнить:

$ switch\_index\_storage.sh -i Index\_1

При перемещении индексных баз, коллекция становится недоступной. Следует это учесть, так как копирование больших индексных баз может занять длительное время.

Ниже представлено описание индексных баз, и в каких сервисах они используются.

| **Название файла** | **Описание** |
| --- | --- |
| ri | Файл хранит инвертированный индекс всех текстов, проиндексированных в этой коллекции. Для каждого слова, встретившегося в коллекции, в соответствующем списке, перечислена информация о вхождении слова в документ. Информация включает идентификатор документа, в котором оно встретилось, позиции этого слова в документе и в предложении, синтаксические отношения, в которых находится это слово в тексте, семантическое значение этого слова (если есть) и пр.  |
| rsi | Файл хранит обратный спектральный индекс всех текстов, проиндексированных в этой коллекции. Отличие от обычного обратного индекса заключается в том, что в данном индексе хранится только ltf вес словоупотребления для каждого документа, в котором оно встретилось. Вторым важным отличием является вхождение в индекс словосочетаний, помимо обычных слов, т.е. с помощью этого индекса можно быстро получить вес слов и словосочетаний. |
| ri\_buf | Временное хранилище для обратного индекса. В этот файл записываются данные, если они не успели добавиться в файл ri, до выключения индекса. |
| rsi\_buf | Временное хранилище для обратного спектрального индекса. В этот файл записываются данные, если они не успели добавиться в файл rsi, до выключения индекса. |
| direct\_idx | Файл хранит текстовое представление предложений всех документов коллекции. Для каждого документа коллекции хранится набор предложений, содержащихся в этом документе. Для каждого предложения хранится дополнительная информация: отступ предложения от начала текста в байтах, вес предложения, является ли предложение скрытым и пр. |
| flat\_idx | Файл хранит прямой индекс всех текстов коллекции. Для каждого документа коллекции хранится информация о входящих в него словах. Для слова хранится информация, почти совпадающая с той, что используется в обратном индексе. Не хранится только информация о документе, в котором встречается слово, так как это является избыточным для этого индекса. |
| spidx | Файл хранит прямой спектральный индекс всех текстов коллекции. Для каждого документа хранится информация о tf-весе содержащихся в нем слов и словосочетаний. |
| doc\_store | В этом файле хранится метаинформация каждого документа входящего в коллекцию. |
| docsinfo | Этот файл хранит отображение между глобальным идентификатором документов, который является уникальным для всех коллекций, и внутренними идентификаторами документов, который является уникальным только в рамках этой коллекции. Docsinfo загружается целиком в память. |

| **Название** | **Список индексных файлов** | **Пояснение** |
| --- | --- | --- |
| индексация | Все индексные файлы | Все файлы обновляются при добавлении нового документа |
| сервис семантического поиска | ri | Используется для получения списка документов, в которых встречаются слова из поискового запроса. Дополнительная информация, сохраненная в обратном индексе, используется для расчета релевантности документов. |
| direct\_idx | Используется для построения поисковых сниппетов для найденных релевантных документов. |
| doc\_store | Используется для возвращения метаинформации, связанной с найденными релевантными документами. |
| docsinfo | Используется для сопоставления глобальных и коллекционных идентификаторов |
| сервис поиска текстовых заимствований | rsi | Используется для получения списка документов-кандидатов, т.е. тех документов, которые имеют некоторое пересечение по словам. |
| flat\_idx | Используется для детального сравнения предложений загруженного на проверку документа со всеми предложениями, входящими в документы-кандидаты. |
| direct\_idx | Используется для представления в текстовом виде исходных предложений, которые послужили источником заимствований. |
| doc\_store | Используется для возвращения метаинформации, связанной с найденными документами. |
| ri | Используется только в том случае, если заданы дополнительные критерии фильтрации документов-кандидатов по метаполям. |
| docsinfo | Используется для сопоставления глобальных и коллекционных идентификаторов |
| сервис поиска похожих документов | rsi | Используется для подсчета меры тематического сходства между эталонным документом и остальными документами коллекции. |
| spidx | Используется для получения весов слов и словосочетаний, входящих в эталонный документ, в случае если он проиндексирован в системе. |
| doc\_store | Используется для возвращения метаинформации, связанной с найденными релевантными документами. |
| docsinfo | Используется для сопоставления глобальных и коллекционных идентификаторов |
| сервис получения ключевых слов | spidx | Используется для получения весов слов и словосочетаний, входящих в эталонный документ, в случае если он проиндексирован в системе |
| сервис построения расширенной аннотации | spidx | Используется для получения ключевых слов и словосочетаний документа |
| direct\_idx | Используется для представления в текстовом виде предложений составляющих аннотацию |
| сервис формирования и обработки пользовательских коллекций документов | rsi | Используется для подсчета меры тематического сходства между эталонным документом и остальными документами коллекции. |
| spidx | Используется для получения весов слов и словосочетаний, входящих в эталонный документ, в случае если он проиндексирован в системе. |
| doc\_store | Используется для возвращения метаинформации, связанной с документами коллекции. |

## Мониторинг процесса индексации

Для мониторинга процесса индексации можно воспользоваться утилитой StatMonitor.

$ StatMonitor -um -t Queue.\* Index.\*

Она предоставляет статистику по количеству документов, которые находятся в индексе (docCount в статистике для Index) и по количеству документов, ожидающих в очереди (docsCount в статистике для Queue). Для оценки динамики поступления новых документов можно воспользоваться дашбордом Indexing в системе мониторинга (см. п. 3.7). Также можно использовать утилиту StatMonitor, для этого необходимо добавить параметр -d, тогда будет предоставлена статистика за последние 60 секунд.

Например, если требуется проверить, что в очередь на индексацию добавляются документы. Может пройти некоторое время, прежде чем документы начнут добавляться в очередь (это зависит от способа краулинга):

$ StatMonitor -t Q.\* -d

Значение receivedDocsTotal должно быть больше 0

Например, чтобы проверить что в индекс документы действительно попадают:

$ StatMonitor -um -t Index.\* -d | jq .basicMetrics.docMetrics.docCount

Покажет кол-во добавленных документов в индекс за минуту (число должно быть > 0).

Для просмотра расширенной статистике по каждой коллекции, ожидающей в очереди, выполните:

$ StatMonitor -e -t Q.\*

Для просмотра расширенной статистики по каждой коллекции, выполните:

$ StatMonitor -ume -t Index.\* | less

Для просмотра расширенной статистики по конкретной коллекции, например с номером 2000, выполните:

$ StatMonitor -ume -t Index.\* | jq '.collMetrics."2000"' |less

Процесс индексации считается завершенным, если процесс добавления документов в очередь завершен (пункт 5.5) и в очереди не осталось документов, т.е. docsCount == 0.

## Завершение индексации

Приведенные ниже операции являются **необязательными**!

***Загрузка коллекций в память***

Для ускорения операций поиска можно загрузить обратные индексы в оперативную память. Для этого необходимо обновить конфигурационные файлы новых коллекций.

$ manage\_configs.py -c <coll> manage\_mem\_conf -w <index\_work\_dir>

В результате работы команды будет выведен объем памяти, который будут занимать обратные индексы всех коллекций, например: total size of indices in memory: 15.5GB.

Необходимо убедиться, что на узле останется как минимум 12 Гб свободной памяти.

Чтобы выгрузить коллекцию из памяти нужно выполнить команду с добавленным ключом -u:

$ manage\_configs.py -c <coll> manage\_mem\_conf -u -w <index\_work\_dir>

После обновления конфигов необходимо перезапустить индекс.

**Пример:**

$ manage\_configs.py -c 3000 manage\_mem\_conf -w /data/Pak/Index\_1

$ manage\_configs.py -c 2000 manage\_mem\_conf -w /data/Pak/Index\_1 -u

$ restart.sh Index.\*

Для просмотра, сколько в памяти занимает каждая коллекция, есть флаг -s. **Пример:**

manage\_configs.py -c 2000 manage\_mem\_conf -w /data/Pak/Index\_1 -s

## Откат после неудачной индексации

В случае если нужно переиндексировать новые документы, по причине ошибки выделения метаполей или других ошибках связанных с содержимым документов, необходимо удалить новую коллекцию (см. 4.6). После этого произвести заново обход и переиндексацию, предварительно создав коллекцию заново.

Если производилась «доиндексация», то коллекцию необходимо восстановить из бэкапа.

$ restore\_from\_backup.sh -i Index\_1 -c <coll\_num>

## Объем индексных баз

Объем индексов примерно в три раза больше, чем объем исходных данных. Под объемом исходных данных понимается объем чистого текста, а не объем документов в их формате представления. Однако в процессе индексации объем индексных баз может значительно «распухать». Это вызвано особенностью используемой на низком уровне key-value БД kyotocabinet. Размер базы автоматически сжимается еженедельно в процессе бэкапа данных.

## Удаление коллекции

При необходимости полностью удалить коллекцию из индекса и хранилища нужно выполнить следующую команду:

$ manage\_coll -c 2022 delete

Где 2022 заменить номером коллекции, которую нужно удалить. Команда выведет названия модулей, для которых будет удалена коллекция, если это то, что нужно, то надо набрать ‘y’ и нажать Enter. Далее будет нужно набрать ‘y’, если требуется также удалить все индексы этой коллекции. Если данные нужно сохранить, то нужно набрать ‘n’ и нажать Enter. После этого модули будут перезагружены.

# Добавление документов из внешних ресурсов в очередь

Для обхода внешних ресурсов требуется, чтобы узел обладал доступом во внешнюю сеть. Для каждого внешнего ресурса необходимо написать задание. В задании указывается вся необходимая для обхода информация. Краулер посетит все ссылки, которые удовлетворяют заданным ограничениям, начиная со стартового адреса. Документы, тип которых был указан в теге indexed\_formats, будут добавлены в очередь.

## Составление задания

Задание представляет собой xml-файл, корневым элементом которого является:

<crawling\_task task\_id="1" coll\_id="31">

</crawling\_task>

Атрибут task\_id должен быть уникальным среди всех заданий, а coll\_id задает номер коллекции для обходящихся документов.

***info***

В тэге info заполняется основная информация обхода:

<info>

 <indexed\_formats>pdf,bin</indexed\_formats>

 <crawl\_delay\_ms>600</crawl\_delay\_ms>

 <max\_url\_depth>0</max\_url\_depth>

 <max\_jumps\_depth>0</max\_jumps\_depth>

 <save\_transformed\_doc>yes</save\_transformed\_doc>

 <use\_proxies>no</use\_proxies>

 <change\_site\_kind>disallow</change\_site\_kind>

 <ref\_tags>a,span</ref\_tags>

 <ref\_attribs>href,data-link</ref\_attribs>

</info>

* Форматы документов, подлежащих индексированию, задаются в тэге indexed\_formats. Все поддерживаемые форматы описаны в файле <config\_dir>/formats.xml. Если нужного формата там нет, то его необходимо добавить. Эта процедура описана в п. 3.5.9. Определения формата происходит на основе mime-типа, возвращаемого сайтом при скачивании документа. Для проверки MIME-формата документа, можно скачать его с помощью утилиты wget: wget '<url>'. Утилита выведет много сообщений, среди, которых должны быть сообщения следующего вида:

HTTP request sent, awaiting response... 200 OKLength: 32564 (32K) [**application/octet-stream**]

Жирным выделен mime-тип документа.

* crawl\_delay\_ms задает задержку в миллисекундах между двумя последовательными скачиваниями.
* max\_url\_depth и max\_jumps\_depth определяют максимальную глубину обхода и максимальное количество переходов по ссылкам соответственно. Значения равные нулю означают отсутствие ограничений.
* save\_transformed\_doc должен быть равен yes, если необходимо сохранять в хранилище html-страницу, полученную после преобразования из целевого формата (pdf, doc, bin).
* use\_proxies если равен yes, то для обхода будут использованы прокси. Поддерживаются http (формат: 194.158.204.194:1080 ) и socks прокси (формат: socks5://103.14.112.93:1080). Каждая прокси должна находиться на отдельной строке. При обходе прокси будут браться из <config\_dir>/crawler\_proxies.lst. Для обновления списка прокси нужно перезапустить модуль Crawler.
* change\_site\_kind может принимать следующие значения: disallow/subdomains/any. Disallow запрещает переход по ссылкам, имеющим домен отличный домена страницы, из которой они были выделены; subdomains разрешает переход только на поддомены; any разрешает переход на любые сайты.
* Тэги ref\_attribs и ref\_tags могут быть использованы для указания тэгов и атрибутов, из которых должны извлекаться ссылки. По умолчанию ссылки извлекаются из атрибута *href* тэга *a*. Множественные значения следует передавать через запятую.

***metafields***

<metafields>

 <docmeta overwrite="yes">

 </docmeta>

</metafields>

Базовая метаинформация для всех документов. В тэге docmeta задается список предустановленных метаполей. Список доступных метаполей представлен в <config\_dir>/descriptors.xml. В этот тэг можно добавить, например, язык, на котором написан документ, если он заранее известен:

<docmeta overwrite="yes">

 <metafield name="content-language">rus</metafield>

</docmeta>

Если не указывать язык явно, то он будет определен автоматически для каждого документа. Если нет необходимости задавать метаинформацию для всех документов, то тэг metafields, можно не включать в задание.

***seeds***

<seeds>

 <seed>[http://www.hindawi.com/journals/</seed](http://www.hindawi.com/journals/%3C/seed)>

 <seed>http://www.hindawi.com/articles/</seed>

</seeds>

С ссылок указанных в этом тэге начинается обход внешнего ресурса. Указанные страницы в тэгах seed, скачиваются, затем из них выделяются ссылки. К этим выделенным ссылкам применяются ограничения, задаваемые в тэге constraints. Для дальнейшего скачивания отбираются только ссылки, к которым удалось применить какое-нибудь правило allow, и не было перед этим применено правило disallow. Если формат скачанной странице содержится в тэге //info/indexed\_formats, то скачанная страница будет также отправлена на индексацию.

***constraints***

<constraints filters="e">

 <noindex>.\*/journals/\w+/</noindex >

 <allow src=".\*/journals/">.\*/journals/\w+/</allow>

 <allow>.\*/journals/\w+/contents/\d\*/?</allow>

 <allow src=".\*/journals/\w+/contents/\d\*/?">.\*/journals/\w+/contents/[a-z.]+/</allow>

 <allow content="//div[@class='article']" >.\*/journals3/\w+/</allow>

 <disallow/>

</constraints>

Ограничения обхода. Правила применяются в указанном в ограничениях порядке. Переход по любой ссылке будет выполнен только в том случае, если к ней удается применить правило из какого-нибудь тэга allow. В теле тэга allow указывается регулярное выражение, которое будет сопоставляться с ссылкой. Атрибут src задает дополнительное ограничение на адрес страницы, из которой была извлечена ссылка. Тэг <disallow/> запрещает переход по всем ссылкам, к которым не удалось применить правила из тэгов allow.

Когда в indexed\_formats указан тип html, то все страницы формата html будут индексироваться. Это не всегда желательно. Тэг noindex позволяет исключить индексацию ненужных html-страниц. В нем нужно прописать регулярное выражение адреса страницы, которая не должна индексироваться.

По умолчанию краулером обрабатывается вся html-страница целиком. Чтобы он выделял ссылки или текст из конкретной области существует атрибут content. В атрибуте content задается xpath, который определяет фрагмент страницы, который будет обрабатываться краулером. Если по xpath-выражению выделено несколько областей, они будут объединены между собой. Если xpath выражение применить не удалось, страница не будет обработана и в логе будет записано сообщение об ошибке. Дополнительный атрибут extract\_meta\_from\_fragment принимает значения yes/no. В случае если он равен yes, то метаполя будут извлекаться из фрагмента, заданного тэгом content, иначе они будут извлечены из всей страницы.

***metarules***

<metarules>

 <rule url=".\*/journals/\w+/\d{4}/\d+/" goaldoc="no" format="HTML">

 <get\_date tag="date" path="//\*[@id='content']/div[2]/div[2]/pre" regexp=".\*\((\d{4})\).\*" format="%Y" />

 <get\_text tag="mf\_doi" path="//a[contains(text(), 'doi')]" regexp=".\*http://dx.doi.org/(.\*)" />

 <get\_text tag="mf\_journal" path="//p[@class='banner\_title']" />

 <get\_text tag="mf\_doc\_type" path="//div[@class='article\_type']" />

 <get\_text tag="title" path="//\*[@id='content']//h2" />

 <get\_text tag="author" path="//div[@class='author\_gp']" />

 <get\_text tag="abstract" path="//h4[contains(text(), 'Abstract')]/following-sibling::p" />

 <get\_block/>

 </rule>

 </metarules>

В этой секции задаются правила для выделения метаполей. В тэге rule описываются правила для выделения метаполей со страницы <url>, описываемой регулярным выражением. Атрибут goaldoc должен быть равным yes, если страница подлежит индексированию. Общими элементами для всех правил являются атрибуты: tag, path, regexp.

* Атрибут tag определяет метаполе, для которого будет извлекаться текст. Описание доступных метаполей представлено в <config\_dir>/descriptors.xml. Инструкция по добавлению новых метаполей находится в п. 3.5.9.
* В атрибуте path указывается xpath-выражение до элемента html-страницы, в котором находится желаемый текст. Xpath может быть не только путем к тексту элемента, но и путем к группе вложенных элементов. В этом случае текст извлекается из всех элементов, и объединяется через строку, переданную в атрибуте delimiter (по умолчанию пробел). В случае если xpath-выражение указывает на несколько элементов, то текст выделяться не будет.
* Атрибут regexp служит для постобработки выделенного текста. Допустима только одна группа захвата.

В специфическом элементе get\_date добавляется один атрибут:

* format задает формат даты. Все возможные значения описываются на странице <http://www.boost.org/doc/libs/1_56_0/doc/html/date_time/date_time_io.html> в разделе Format Flags.

В случае если одна html-страница содержит метаполя для нескольких документов необходимо использовать тэг get\_block.

<get\_block path="//dt" pathend="//dd/div[@class='meta']/div[@class='list-subjects']">

 <get\_text tag="title" path="//div[@class='list-title']" regexp=".\*Title.(.\*)"/>

 <get\_text tag="author" path="//div[@class='list-authors']" regexp=".\*Authors.(.\*)"/>

</get\_block>

Элемент get\_block имеет атрибуты path и pathend, которые указывают на начальный и конечный элементы блока. Блок должен содержать метаинформацию для одного документа и ссылку, к которой нужно привязать извлеченную метаинформацию.

В случае если метаинформация находится за несколько ссылок до целевого документа, то ее можно «протащить» до него создавая, при необходимости, пустые правила для промежуточных страниц.

<rule url="<mid\_url>" goaldoc="no" format="HTML">

<get\_block/>

</rule>

***Дополнительные примеры***

Пример задания с дополнительными комментариями и со всеми возможными параметрами расположен на сервере /data/tasks/examples/sample.xml. Дополнительные примеры находятся в директории /data/tasks/examples.

## Тестирование задания

Для тестирования задания нужно использовать утилиту taskschecker:

$ taskschecker -t <task.xml> -u '<url>' --out-urls > out

Утилита принимает на вход задание и ссылку на страницу. Она печатает результат работы в файл out. Если со страницы были выделены ссылки, они будут записаны в этот файл в порядке появления на веб-странице. Для каждой ссылки приводятся список метаполей, извлеченных по соответствующим правилам. Если <url> ведет на документ, подлежащий индексации, то утилита напечатает преобразованный документ. Если не было извлечено ни одной ссылки, и <url> не вел на документ, то утилита будет висеть, пока ее не убьют. Это обычно обозначает ошибочную ситуацию. Параметр -s позволяет смоделировать обход.

## Подготовка к обходу

***Если планируется обход нового сайта,***

То ничего делать не нужно и можно переходить к следующему пункту.

***Если планируется обновление ранее «накрауленных» сайтов***

То обратитесь к п. 5.7. Возможно, достаточно выставить частоту переобхода в тэге <recrawl\_time> и обновить задание в модуле Frontier, выполнив команду:

$ CrawlersManager -f Frontier\_1 <task1>

***Если планируется полный переобход ранее «накрауленных» сайтов***

Такая потребность может возникнуть, в случае, если часть метаинформации при первичном обходе была упущена. Тогда нужно очистить старые данные, связанные с этим заданием. Для этого нужно выполнить следующую команду:

$ exfrontierctl deleteTask -t <task\_id>

Где task\_id это идентификатор задания. Список всех идентификаторов заданий, которые находятся во Frontier можно получить с помощью команды:

$ exfrontierctl getTaskIds

*Возможные проблемы.* После выполнения команды exfrontierctl deleteTask данные, связанные с сидами задания могут остаться в модуле Frontier. Это обычно связано с тем, что в сидах указана неточная ссылка, например, указана ссылка <http://ya.ru>, которая при скачивании перенаправляется на <https://ya.ru>. В этом случае все данные связанные с настоящим адресом (<https://ya.ru>) останутся нетронутыми. Это может помешать старту следующего переобхода, при этом в логе модуля Crawler будут возникать сообщения о том эта ссылка является дубликатом, например: id: 2031:a0102c3cad52c9f0:49947ba63bd4 url: <http://www.hindawi.com/journals/> msg: Out analyzing duplicate.

Чтобы избежать этого, следует указывать в сидах реальные ссылки. Кроме того можно удалить конкретные ссылки из модуля Frontier с помощью команды:

$ exfrontierctl deleteUrl -t <task\_id> -c <coll\_id> -u <url>

Где task\_id – идентификатор задания, coll\_id – идентификатор коллекции, url – ссылка которую надо удалить.

**Пример:**

$ exfrontierctl deleteUrl -t 11 -c 2031 -u https://ya.ru

## Запуск обхода

Параллельно с обходом начнется процесс индексации, поэтому нужно также подготовиться к этому процессу как описано в п. 4.1.

Для старта обхода нужно включить модули Crawler и Frontier и добавить файл с новым задание во Frontier.

$ start.sh Fron.\* Crawler.\*

$ CrawlersManager -f Frontier\_1 <task1>

Где <task1> это путь к файлу с заданием. Может быть передано сразу несколько заданий за один раз.

**Пример:**

$ cd /data/tasks/examples

$ CrawlersManager -f Frontier\_1 journals/international\_journals/hindawi/2031\_hindawi.com.xml

## Мониторинг обхода

Для мониторинга обхода есть несколько команд:

$ StatMonitor -m -t Frontier.\* Crawler\_.\*

Эта команда предоставит статистику по обходу: общее количество извлеченных ссылок (refsTotal), количество скачанных целевых документов (targetDocsTotal), количество скачанных дубликатов (downloadedDuplicatesTotal) и общее количество скачанных урлов (downloadedUrlsTotal). Также предоставляется информация о кодах, которые были возвращены веб-сервером (respCodeInfos). Большое количество 400-ых ошибок может свидетельствовать о том, что сайт банит обходчика. Для предотвращения этого можно увеличить время задержки, либо использовать прокси для обхода. Также возвращается информация о форматах, которые были скачаны (formatInfos), если скачивается большое количество документов с форматами, которые непригодны для индексации (jpeg, png) надо проверить задание на краулинг.

В логах Краулера доступна информация о каждой скачанной html-странице (чтобы определить местоположение логов краулера см. п. 3.5.4). Например, чтобы распечатать все записи для урла <http://www.dialog-21.ru/digest/2006/> нужно выполнить:

$ jq 'select(.url == "http://www.dialog-21.ru/digest/2006/")' Crawler.json

Команда, которая распечатает урлы, скачивание которых завершилось ошибкой:

$ jq 'select(.code and .code != 200) | .url, .code' Crawler.json

Следующая команда распечатает скачанные html-страницы, размер которых не превышает 1Кб.

$ jq 'select(.msg == "Downloaded!" and .size < 1000)' Crawler.json

Большое количество таких html-страниц может означать, что вместо страниц скачиваются сообщения о бане. Общее количество таких страниц можно получить, выполнив:

$ jq -c 'select(.msg == "Downloaded!" and .size < 1000)|.size' Crawler.json | wc -l

## Завершение обхода

Обход можно считать завершенным, если выполнены все условия:

* В модуле Frontier и Crawler не осталось ссылок для обхода. Это можно узнать, запустив команду

$ StatMonitor -t Frontier.\* Crawler\_.\* 2>/dev/null | jq '.refsCount // empty, .[keys[] | select(startswith("current"))]'

Если все значения равны 0, то ссылок на обход не осталось.

* Если на узле отсутствуют запущенные процессы tesseract (отвечающие за распознавание текста).

## Регулярные обновления

По умолчанию обход будет производиться один раз. Это можно исправить, изменив файл с заданием для сайта, добавив в тэг info опцию recrawl\_time.

<info>

 ….

 <recrawl\_time>24:00:00</ recrawl\_time>

 ….

</info>

Формат значения указывается согласно “%H:%M:%S”, где %H – количество часов, %M – количество минут, %S – количество секунд.

В этой опции указывается временной интервал, по истечении которого будет произведен повторный обход сайта. Отсчет времени начинается с момента поступления файла с заданием в краулер, т.е. после выполнения команды:

$ CrawlersManager -f Frontier\_1 < task1.xml>

Когда перезагружается модуль Frontier, то отсчет сбрасывается и начинается заново после повторного включения модуля. Переобход будет выполняться каждый интервал времени указанный в recrawl\_time. Переобход начинается с добавления ссылок из раздела <seed> в очередь ссылок на обход. При этом не учитывается завершен ли текущий обход сайта, связанного с этим заданием. Чтобы отключить переобход ресурса, нужно либо убрать эту опцию из задания и обновить его в системе, с помощью команды

$ CrawlersManager -f Frontier\_1 <task1.xml>

либо изменить его значение на 00:00:00 и также обновить задание в системе.

При переобходе, страницы, которые являются дубликатами, будут игнорироваться. Дубликатом считается страница, которая ранее была закраулена и которая не была изменена со времени последнего краулинга. Для определения дубликатов считается CRC32 сумма скачанных страниц и целевых документов. Если необходимо всегда выделять ссылки, которые могут привести к новым документам, даже из дубликатов, то можно указать регулярные выражения для этих ссылок в разделе constraints.

<constraints filters="e">

 ….

 <forced\_follow>.\*/page=\d+</forced\_follow>

 ….

</constraints>

Имеет смысл указывать в тэге forced\_follow ссылки на страницы, содержащие целевые документы. Это нужно, чтобы иметь возможность добраться до последних страниц, которые возможно содержат новые документы, через страницы, которые не были изменены.

# Добавление локальных документов в очередь

Документы, расположенные непосредственно на сервере, могут быть добавлены в очередь с помощью сервиса пополнения коллекций текстов. Для этого необходимо написать скрипт, вызывающий этот сервис для каждого документа. Помимо этого доступен модуль LocalCrawler, выполняющий такую же задачу.

## Описание локального краулера

Модуль принимает на вход список документов, либо директорию для обхода, либо файл в формате json с дополнительной метаинформацией для каждого документа. Для запуска краулера достаточно указать источник документов и номер коллекции. **Примеры:**

$ LocalCrawler --log\_file=crawler.log -d/data/files --coll\_id=3456

$ LocalCrawler --log\_file=crawler.log -l/data/doc\_list.txt --coll\_id=3456

$ LocalCrawler --log\_file=crawler.log -j/data/files.json --coll\_id=3456

Так как процесс краулинга может занять долгое время, желательно запускать представленные выше команды в сессии tmux или screen. При этом все логи пишутся в файл crawler.log. Параллельно с обходом начнется процесс индексации, поэтому нужно также подготовиться к этому процессу, как описано в п. 4.1.

## Дополнительные опции локального краулера

Таблица 8 Опции модуля LocalCrawler

|  |  |  |  |
| --- | --- | --- | --- |
| **Название опции** | **ключ** | **Описание** | **Значение по умолчанию** |
| ***Основные опции*** |
| --coll\_id |  | Номер коллекции, в которую будут добавлены документы |  |
| --list | -l | Путь к файлу, содержащему пути до локальных файлов, подлежащих добавлению в индекс. Пути должны быть разделены символом переноса строки. |  |
| --json\_list | -j | Путь к файлу, содержащему информацию о документах, подлежащих индексации. Информация включает в себя путь к документу и дополнительную метаинформацию. Описание формата этого файла приводится в пункте 6.3. |  |
| --directory | -d | Путь к директории содержимое, которой необходимо проиндексировать. Индексироваться будут файлы, формат которых известен системе (файл <config\_dir>/formats.xml). |  |
| --recurs | -r | Обходить директорию рекурсивно. Имеет смысл только в сочетании с опцией -d. | false |
| --filter | -f | Регулярное выражение (pcre) для фильтрации загружаемых документов.  |  |
| --docs\_cnt |  | Максимальное количество документов, которые должны быть загружены в индекс. Если значение равно 0, то количество неограниченно. | 0 |
| --thrds | -t | Количество потоков, в которых будут одновременно загружаться документы в очередь. | 1 |
| --config\_file |  | Опции, передаваемые локальному краулеру, могут быть записаны в конфигурационный файл. При этом нужно передать путь к файлу с помощью этой опции. Формат конфигурационного файла обычный: каждая опция с новой строки; в виде <opt\_name>=<opt\_value>.Пример:verbose=10thrds=1skip\_storage\_save=1 |  |
| ***Опции дополнительной метаинформации*** |
| --default\_format |  | Задает формат по умолчанию для загружаемого файла (строка с названием формата). Используется в том случае, если не удалось определить формат по расширению. Список поддерживаемых форматов представлен в файле <config\_dir>/formats.xml. | txt |
| --default\_enc |  | задает кодировку по умолчанию для входного текста (строка с названием кодировки). Если опция не передана, то кодировка устанавливается автоматически. |  |
| --base\_url |  | базовая часть URL, которая будет конкатенироваться с названием файла, для получения полного адреса документа. Пример: если "BASE\_URL" == <http://ya.ru>, а путь к файлу, переданному в краулер, "/mnt/data/4287.doc", то полный URL документа будет равен <http://ya.ru/4287.doc> |  |
| ***Опции документов*** |
| --save\_transformed\_doc |  | Сохранять ли текст, полученный после извлечения из документа, вместо самого документа (число 1 – режим включен, 0 – режим выключен). Рекомендуется всегда оставлять этот режим включенным. | true |
| --compressor |  | Обозначение компрессора, который будет использован при сохранении документов в очередь. Допустимые значения bz2|gz|zip. no используется для сохранения без использования компрессора. | no |
| --compress\_docs |  | Сжимать документы, отправляемые в хранилище. | false |
| --compress\_mark\_docs |  | Сжимать документы, отправляемые в очередь | false |
| --skip\_storage\_save |  | Не сохранять документы в хранилище | false |
| --skip\_queue\_save |  | Не сохранять документы в очередь | false |
| ***Прочие опции*** |
| --glob\_log |  | Если передано значение 1, то будет использоваться модуль централизованного логирования (иначе лог пишется в файл). См. п. 3.5.4 |  |
| --verbose |  | Степень детальности логов. Значение может варьироваться от 0 до 15. Чем больше значение, тем больше логов. | 4 |

## Формат метаописания для локального краулера

Локальный краулер принимает список документов в формате json. В этом файле следует указать значения метаполей для загружаемых документов. Список доступных метаполей находится в конфигурационном файле <config\_dir>/descriptors.xml. Для добавления новых метаполей см. п. 3.5.9.

Кодировка json файла должна быть utf-8. Для перекодирования из cp1251 (или другой кодировки) на сервере нужно выполнить команду:

$ enca -xutf8 -Lrussian <file>.json.

### Описание формата

Вся информация содержится в массиве "docs". В этот массив входят объекты для всех документов, подлежащих индексации. Каждый имеет следующие поля:

* "path" – обязательное поле, содержащее путь к документу. Путь может быть относительным. В этом случае обнаружение файла будет проводиться относительно рабочей директории краулера (по умолчанию директория, в которой он был запущен). Расширение должно соответствовать формату файла, либо нужно другим способ указать формат документа. Подробное описание в п. 6.4.
* "metafields" – опциональное поле, содержащее набор метаполей, которые будут привязаны к этому документу. В качестве ключа должно выступать имя метаполя, определенное в файле <config\_dir>/descriptors.xml. Почти все метаполя являются произвольными строками. Исключение составляет поле "date", оно должно быть заполнено согласно следующему формату: "%Y-%m-%d".

**Пример:**

{

 "docs": [

 {

 "path":"/mnt/data/871.html"

 },

 {

 "path":"/mnt/data/4287.html",

 "metafields": {

 "author":"РАН",

 "title":"отчет",

 "date":"2010-01-01",

 "mf\_url" : "http://ru.rationalwiki.org/wiki/Отчет" }

 },

 {

 "path":"/mnt/data/3061.html",

 "metafields": {

 "author":"кузьмич",

 "title":"отчет2",

 "date":"2010-09-24"}

 }

 ]

}

Перед запуском краулера желательно проверить json-файл на валидность, выполнив команду jq <file>.json. Если команада распечатает json, то все правильно, иначе будет выведено сообщение об ошибке с указанием номера строки.

## Определение формата загружаемого документа

На основе определенного системой формата выбирается преобразователь для документа, поэтому важно, чтобы формат был выбран правильно. Формат определяется на основе следующих факторов:

* Значение метаполя "mf\_format". Передача метаинформации описывается в п. 6.3.1. В качестве значение должно быть передано имя формата, поддерживаемого системой. Полный список приводится в <config\_dir>/formats.xml.
* Значение метаполя "mf\_mime". Передача метаинформации описывается в п. 6.3.1. В качестве значение должно быть передано название mime-типа, поддерживаемого системой. Полный список приводится в <config\_dir>/formats.xml.
* Значение расширения файла. Это значение воспринимается как имя формата, т.е. должно содержаться в файле <config\_dir>/formats.xml, чтобы система его восприняла.
* В случае если не удалось определить формат на основе вышеперечисленных факторов, используется опция default\_format.

Факторы имеют приоритет, который совпадает с порядком их следования в списке.

## Формирование урла загружаемого документа

На основе урла, присвоенного документу, вычисляется его уникальный идентификатор, поэтому важно, чтобы у каждому документу был присвоен уникальный урл. На формирование урла документа влияет несколько факторов.

* Значение метаполя "mf\_url". Если с документом передано непустое значение метаполя "mf\_url", то оно присваивается в качестве урла документа. Передача метаинформации описана в п. 6.3.
* Если передана опция "base\_url", то ее значение будет конкатенировано с названием файла, для получения полного урла документа. Пример: если "BASE\_URL" == <http://ya.ru>, а путь к файлу, переданному в краулер, "/mnt/data/4287.doc", то полный URL документа будет равен <http://ya.ru/4287.doc>
* Урл будет образован из пути файла, переданного в краулер. Например, если путь "/mnt/data/4287.doc", то результирующий урл будет равен <file://localhost/mnt/data/4287.doc>

## Выводимые сообщения

Модуль будет писать лог в файл указанный в опции --log\_file. Сообщения разделяются по уровням:

INFO, WARNING, ERROR, DEBUG.

|  |  |  |
| --- | --- | --- |
| **Уровень** | **Сообщение** | **Описание** |
| ERROR | no proxies are available for <app\_name>::icemodules::I<module\_name>. Waiting for turning on the module | Сообщение выводится, в случае если отключен модуль необходимый для работы краулера. Это может быть либо очередь, либо хранилище (значение <module\_name> Queue или Storage соответственно) |
| ERROR | no suitable endpoint available for proxy `<app\_name>.Queue -t @ <app\_name>.Queue.Adapter' trying to recreate proxies | Во время работы был отключен модуль очереди. Если очередь не запущена, то требуется ее запуск start.sh Q.\*. Краулер восстановит соединение с очередью.  |
| ERROR | LocalCrawler cant proceed working. Exception from Storage\_1: collection 2000 wasn`t found | При сохранении в хранилище не была найдена нужная коллекция. Возможно, она не создана. Создание новой коллекции – п. 4.1.1. Краулер будет выключен. |
| ERROR | during sending bulk of documents for one of them ( or for all ) was caught exception from Storage\_1: <err\_msg> | see Storage log for details | Возникла ошибка при сохранение в хранилище. Подробный текст ошибки содержится в сообщении <err\_msg>. Подробности также могут находится в логах модуля хранилища. Краулер будет выключен. |
| ERROR | LocalCrawler cant proceed working. Error need to be fixed: ConnectionI.cpp:1296: Ice::MemoryLimitException: protocol error: memory limit exceeded | Возникла ошибка в работе краулера. В данном примере был превышен максимальный размер передаваемого сообщения между модулями (по умолчанию он составляет 200 Мб). Краулер будет выключен. |
| WARNING | (file://localhost/text/AC/wiki\_32) failed to detect web format. Fall back to default one 2 | Сообщение сигнализирует, что не удалось опознать формат загружаемого файла, поэтому используется значение по умолчанию, т.е. выбирается формат txt для этого файла. |

## Регулярные обновления

Для настройки регулярных обновлений можно добавить вызов локального краулера в стандартный планировщик Linux. Для доступа к настройке расписания нужно выполнить команду

$ crontab -e

Для выполнения обхода ежедневно в час ночи можно добавить в открывшийся файл следующую запись.

0 1 \* \* \* run-one LocalCrawler --log\_file=crawling/local\_crawler.log -jcrawling/files.json --coll\_id=3456 >> crawling /local\_crawler.stdout 2>&1

Команда run-one не допустит запуска модуля, в случае если еще работает экземпляр, запущенный ранее. Рабочей директорией будет выбран домашний каталог пользователя, поэтому большинство путей указано относительно него. Все логи будут перенаправляться в файл /home/<user\_name>/ crawling/local\_crawler.log. Чтобы краулер подхватывал изменения необходимо обновлять файл /home/<user\_name>/crawling/files.json до момента запуска. Важно, чтобы этот файл содержал только новые документы, потому что краулер кладет все документы в очередь, даже проиндексированные ранее.

# Тестирование индексации в виртуальной среде

Для тестирования процесса индексации новых сайтов, на сервере развернута виртуальная машина, в которой запущена копия системы. Основные переменные немного отличаются для виртуальной машины.

## Значения основных переменных в виртуальной машине

|  |  |
| --- | --- |
| **Название** | **Значение** |
| user\_name | vagrant |
| user\_password | vagrant |
| app\_name | Vagrant |
| data\_dir | /data/<app\_name> |
| config\_dir | /compiled/config/ |
| node\_name | textappvm1 |

## Доступ к виртуальной машине

Чтобы зайти в виртуальную машину нужно выполнить:

$ ssh vagrant@textappvm1

Если в терминале слева указано “**vagrant@textappvm1:~$**” , то это означает, что пользователь находится в виртуальной машине.

## Основные операции с виртуальной машиной

Операции производятся на хосте, если не написано иное.

Чтобы **включить** виртуальную машину и систему, нужно выполнить:

$ virsh -c qemu:///system start vagrant\_textappvm1

$ ssh vagrant@textappvm1

**textappvm1** $ ./start\_system.sh

Чтобы **выключить** виртуальную машину, нужно выполнить:

$ virsh -c qemu:///system shutdown vagrant\_textappvm1

Чтобы **откатить** виртуальную машину к первоначальному «чистому» состоянию, нужно выполнить:

$ virsh -c qemu:///system snapshot-revert vagrant\_textappvm1 textapp-clean-sys

После этого нужно включить машину и систему. Эта операция может быть полезна, чтобы протестировать индексацию какого-нибудь сайта с чистого листа.

## Разделяемые папки

Некоторые папки хоста примонтированы в виртуальной машине.

**/data/tasks**

/data/tasks содержит задания для краулинга внешних ресурсов. Соответственно, редактировать и тестировать файл с новым заданием для обхода можно как на хосте, так и в виртуальной машине.

Если возникают проблемы (permission denied, read only file etc)при записи в файл, находящийся в разделяемой папке, нужно выставить файлу необходимые права:

$ chmod o+rw <file>

## Тестирование индексации в виртуальной среде

Ниже приведен список шагов с отсылками к подробным пунктам, который необходим, чтобы запустить процесс индексации:

1. Зайти в виртуальную машину:

$ ssh vagrant@textappvm1

1. Перейти в смонтированную папку:

$ cd /data/tasks

1. Составить задание на обход, п. 5.1.

В качестве примера будем использовать готовые задания из папки examples (перед этим на хосте надо скопировать тестовое задание:

$ cp examples/conf/603\_dialog.xml . ).

1. Тестировать задание, п. 5.2.

$ taskschecker -t 603\_dialog.xml -u 'http://www.dialog-21.ru/digest/' --out-urls > out

$ taskschecker -t 603\_dialog.xml -u 'http://www.dialog-21.ru/digest/2013/' --out-urls > out

1. Создать коллекцию в индексе, п. 4.1.1.

$ manage\_coll -c 2021 create -n "Test name" --est\_docs\_count=9000

1. Запустить краулинг и индексацию, п. 5.4.

$ CrawlersManager -f Frontier\_1 603\_dialog.xml

$ start.sh Crawler.\*

1. Мониторить обход и индексацию, пп. 5.5 и 4.3.

Проверить, что скачиваются html-страницы и документы (downloadedUrlsTotal должен быть больше нуля):

$ StatMonitor -e -t Crawler\_.\* -d30 | jq .basicMetrics.downloadedUrlsTotal

Проверить, что в очередь на индексацию добавляются документы. Может пройти некоторое время, прежде чем документы начнут добавляться в очередь (это зависит от структуры сайта):

$ StatMonitor -t Q.\* -d

Значение docsCount должно быть больше 0

Проверить, что в индекс документы действительно попадают:

$ StatMonitor -u -t Index.\* -d | jq .basicMetrics.docMetrics.docCount

Покажет кол-во добавленных документов в индекс за минуту (число должно быть > 0).

1. Проверить результаты индексации, п. 7.6.

$ /data/scripts/search.sh -c "2021" -O search

$ /data/scripts/sim-doc-search.sh -i "2021:a63e9ec0dc580152:3ce30fe92c5d" -c "2021" -O sim

$ /data/scripts/download\_doc.sh 2021:a63e9ec0dc580152:3ce30fe92c5d

$ /data/scripts/plag\_search.py -i 2021:a63e9ec0dc580152:3ce30fe92c5d.html -c 2021

1. Прекратить индексацию и краулинг:

$ halt.sh Crawl.\* DocAnal.\*

1. Поправить ошибки и повторить процедуру, либо откатив виртуальную машину к чистому состоянию, как написано в п. 7.3. Либо почистив индекс п. 4.7 (manage\_coll -c 2021 delete) и остальные модули:

$ halt.sh Que.\* Fr.\* Craw.\* DocAn.\*

$ cb w && rm -rf Que\* Fr\* Craw\* DocAn\*

$ start.sh Que.\* Fr.\* DocAn.\*

## Проверка проиндексированных данных

Проверка может проводиться двумя способами: через стандартный веб-интерфейс или через скрипты, вызывающие сервисы системы и выводящие результат в текстовом виде.

### Проверка с помощью скриптов

**Поиск в индексе**

Для поиска проиндексированных документов можно использовать скрипт search.sh

$ /data/scripts/search.sh -c "<COLL\_ID1> <COLL\_ID2>" -q "<QUERY>" -h <HOST> -O <outfilename>

-c (COLLECTIONS) - номера коллекций, необходимо указать хотя бы один

-q (QUERY) - текст запроса (по умолчанию QUERY="поиск")

-h (HOST) - хост, к которому будет направлен запрос (по умолчанию "localhost")

-O (OUT) - название для результата запроса, который будет записан в <outfilename>.content. Логи записываются в <outfilename>.logs. Если не указывать этот параметр, то выводы и результата запроса, и логов будут направлены в stdout.

-E (EXTRA) - дополнительные параметры поиска

Полный список параметров для -E (EXTRA) запроса смотреть в документации к API в пункте 2.1.4. Пример использования параметра -E:

$ /data/scripts/search.sh -c "2021" -q поиск -O tmp -E "YEAR\_FROM=2006&YEAR\_TO=2016"

При тестировании поиска нужно обратить внимание на метаполя, которые возвращаются с документами. Метаполе имеет вид (Список после DOC\_CONTENT: <число> : ЗНАЧЕНИЕ). Если для некоторых документов метаполя возвращаются пустые, то необходимо вернуться к шагу тестирования задания для этого документа.

 Если перед индексацией были добавлены новые метаполя, имеет смысл проверить поиск по ним. Это возможно передав их через параметр -E:

$ /data/scripts/search.sh -c "2021" -q "" -O tmp -E "200=Книга"

Если результаты фильтруется по заданным метаполям, т.е. возвращаются только документы, у которых в заголовке встречается слово "Книга", то тогда метаполя настроены правильно. Возможно также комбинировать несколько метаполей в одном запросе:

$ /data/scripts/search.sh -c "2021" -q "" -O tmp -E "200=Губин&202=поиск"

Помимо метаполей возвращается поисковый сниппет, построенный для документа (Abstract:text:<<TEXT>>). Нужно проверить, что в сниппете возвращается не пустой текст, а текст который построен вокруг поискового запроса (QUERY). Если сниппеты пустые или неадекватные, то имеет смысл посмотреть какой именно текст был проиндексирован для этого документа. Сделать это можно с помощью команды:

$ /data/scripts/download\_doc.sh <COLL\_RES\_ID>

<COLL\_RES\_ID> приводится для каждого документа в поисковой выдаче (ID: <COLL\_RES\_ID>).

Чтобы увеличить количество возвращаемых документов нужно передать параметр DOCS\_PER\_PAGE:

$ /data/scripts/search.sh -c "2021" -q поиск -O tmp -E "DOCS\_PER\_PAGE=50"

**Поиск похожих документов**

Для поиска похожих документов присутствует обязательный параметр COLL\_RES\_ID – уникальный идентификатор документа, для которого выполняется поиск похожих. COLL\_RES\_ID состоит из частей: <COLL\_ID>:<SITE\_ID>:<DOC\_ID>, где <COLL\_ID> - уникальный номер коллекции,  <SITE\_ID> - уникальный идентификатор ресурса внутри коллекции,  <DOC\_ID> - уникальный идентификатор документа внутри ресурса. COLL\_RES\_ID можно, например, взять из поисковой выдачи скрипта search.sh (ID: <COLL\_ID>:<SITE\_ID>:<DOC\_ID>), или посмотреть в логах модуля Crawler или Index. Так же COLL\_RES\_ID без COLL\_ID можно сформировать с помощью утилиты url2res\_id:

$ url2res\_id "<http://example.com/doc/id/12345>"

Пример COLL\_RES\_ID: 2021:a63e9ec0dc580152:3ce30fe92c5d.

Для поиска похожих документов можно использовать скрипт sim-doc-search.sh

$ /data/scripts/sim-doc-search.sh -i "<COLL\_RES\_ID>" -c "<COLL\_ID1> <COLL\_ID2>" -h <HOST> -O <outfilename>

-c (COLLECTIONS) - номера коллекций, необходимо указать хотя бы один

-i (COLL\_RES\_ID) - уникальный идентификатор документа

-h (HOST) - хост, к которому будет направлен запрос (по умолчанию localhost)

-O (OUT) - название для результата запроса, который будет записан в <outfilename>.content. Логи записываются в <outfilename>.logs.  Если не указывать этот параметр, то выводы и результата запроса, и логов будут направлены в stdout.

-E (EXTRA) - дополнительные параметры поиска

Полный список параметров для -E(EXTRA) запроса смотреть в документации к API в пункте **2.2.3.**  Пример использования параметра -E:

$ /data/scripts/sim-doc-search.sh -i "2021:a63e9ec0dc580152:3ce30fe92c5d" -c "2021" -O tmp -E "YEAR\_FROM=2006&YEAR\_TO=2016"

При тестировании поиска похожих документов, нужно обратить внимание, что на первом месте должен быть сам этот документ с рейтингом сходства равным 1.

**Поиск заимствований**

Чтобы проверить поиск заимствований нужно сначала скачать какой-нибудь файл из проиндексированной коллекции. Сделать это можно с помощью следующей команды:

$ /data/scripts/download\_doc.sh 2039:e25f65853d6c6abe:87887e84cf63

Нужно заменить 2039:e25f65853d6c6abe:87887e84cf63 на какой-нибудь реальный идентификатор. Описание того, как его получить есть в разделе поиска похожих документов. Эта команда создаст файл с названием 2039:e25f65853d6c6abe:87887e84cf63.html. Этот файл далее используется для тестирования поиска заимствований:

$ /data/scripts/plag\_search.py -i 2039:e25f65853d6c6abe:87887e84cf63.html -c 2039

где -с это набор коллекций в которых искать заимствования.

Это команда напечатает общий процент оригинальности (Originality rating) и список найденных документов, из которых был заимствован текст. Для этого случая значение оригинальности должно стремится к нулю (т.к. документ содержится в индексе), а список должен содержать документ с таким же идентификатором (2039:e25f65853d6c6abe:87887e84cf63) и рейтинг этого документа (Rating) должен быть близок к 1.Если это не так, то значит, поиск заимствований работает некорректно. Причиной этого может быть неправильный язык документов, заданный на этапе краулинга. Информация об этом представлена в п. , при описании тэга <metafields>.

### Проверка с помощью веб-интерфейса

Веб-интерфейс системы, развернутой на виртуальной машине, доступен по адресу <server-ip>:53123, где server-ip это ip-адрес сервера, на котором развернута виртуальная машина. Это стандартный демо-интерфейс системы (логин/пароль – guest/isa60let) с помощью которого можно протестировать основную функциональность системы. В связи с тем, что это типовой веб-интерфейс, там скорее всего нет коллекции, которую нужно тестировать. Чтобы это исправить, необходимо в виртуальной машине выполнить следующую команду:

$ cd ~/wui/textapp\_demo && ./change\_test\_collection.sh 2021

где 2021 заменить на номер коллекции. В веб-интерфейсе коллекция будет доступна под названием "Тестовая коллекция".

## Решение проблем с виртуальной машиной

### Доступ по ssh не работает

* + Перейти в режим virsh: $virsh -c qemu:///system
	+ Далее все команды для использования в среде virsh
	+ Получить список виртуальных машин: # list -all
	+ Если state == shut off, то пробуем запустить:# start vagrant\_textappvm1
	+ Если state == running,то переходим в режим консоли: # console vagrant\_textappvm1
	+ **Режим консоли ttyS0 в virsh**
		- Если после перехода в режим консоли с помощью команды «# console vagrant\_textappvm1» появилось сообщение “Connected to domain vagrant\_textappvm1 Escape character is ^]”, то необходимонажать на Enter**,** затем при необходимости ввести данные user/password**: vagrant/vagrant**
		- Для выхода из консольного режима использовать сочетание клавиш: ctrl+]
		- Если удалось залогиниться под пользователем vagrant, то можно попробовать узнать ip адрес виртуальной машины:

$ ip addr

Далее нужно попробовать подключиться к машине по ssh, используя этот адрес.

* Если это не работает нужно перезагрузить машину выполнив:

$ sudo reboot

! Ошибка “error: operation failed: Active console session exists for this domain” при попытке подключиться в виртуальной машине через virsh c помощью команды console означает, что уже существует такое подключение. Для решения этой проблемы нужно выйти из сессии, открытой, возможно, в другом терминале.