

Содержание

CCU Конфигурация	3
Формат YAML	3
Соглашения при описании параметров	4
Параметры	4
Рабочие каталоги	4
События	4
Журналирование	5
Сбор статистики	5
Категории дней и классы времени	6
Задачи	7
Хранилище	14
SSD-Кэширование	19
Начальная конфигурация	20

CCU Конфигурация

Основным файлом конфигурации является файл `/etc/ccu/ccu.conf`. Конфигурация в этом файле задается в формате [YAML](#).

Формат YAML

Базовыми элементами формата [YAML](#) являются:

- пара "Ключ-Значение", например:
`default: offpeak`
- последовательность значений, например:
 - `"00:00:00 - 01:00:00"`
 - `"17:00:00 - 23:59:59"`

На основе базовых элементов возможно создавать сложные структуры данных.

Одним из принципов для описания сложных структур заложенных в формат [YAML](#) является правило форматирования вложенных элементов при помощи отступов от начала строки. Элементы лежащие на одном уровне иерархии должны иметь одинаковое количество ведущих пробелов, например, следующий текст:

```
#
# Time classes
time_classes:           # Defines the classes of time
    default:    offpeak # Specifies the default value for time classes
                    # All others items define the names of time
                    # classes and rules for them definition
                    # Each rule is started from the name of day
                    # category and contains
                    # the list of ranges of times in form "HH:MI:SS-
                    HH:MI:SS"

    peak:
        workdays:
            - "00:00:00 - 01:00:00"
            - "17:00:00 - 23:59:59"
        weekend_eve:
            - "00:00:00 - 02:00:00"
            - "16:00:00 - 23:59:59"
        weekend:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"
        holidays:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"
```

можно трактовать как структуру данных `time_classes`, у которой есть поля - `default`, `peak`.

Поле `peak`, в свою очередь, также является структурой с полями `workdays`, `weekend_eve`, `weekend`, `holidays`, которые представляют собой именованные последовательности строк.

Учитывая выше сказанное, главным правилом при редактировании документов в формате [YAML](#) является **использование определенного количества пробельных символов для формирования отступов вместо символов табуляции**.

Соглашения при описании параметров

- Для задания имени конкретного поля в сложной вложенной структуре используется полное имя поля, которое представляет собой перечисление всех имен полей, разделенных косой чертой, например: `time_classes/peak/workdays`
- Для задания имени элемента, который определяется пользователем или ссылается на другой элемент, используется символическое определение этого элемента, заданное между знаками меньше и больше, например: `time_classes/peak/<категория дня>`
- Если не оговаривается особым образом, полное имя задается относительно всего файла конфигурации, например: `time_classes/peak/workdays`, либо при описании полного имени можно указать, что является базой для него, например: описание классов времени (`time_classes`) задаются в параметрах вида `<класс времени>/<категория дня>`, которые содержат перечисление интервалов времени

Параметры

Рабочие каталоги

Следующие два параметра задают каталоги, где будут храниться файлы с идентификаторами запущенных процессов - `pid_files_path`, и рабочие файлы утилиты управления КЭШем - `work_files_path`. К числу рабочих файлов, в частности, относятся файлы, хранящие статистику использования КЭШа.

```
#
# PID files path, default: /var/run/ccu
pid_files_path:    /var/run/ccu

#
# Work files directory, default: /var/lib/ccu
work_files_path:   /var/lib/ccu
```

События

По возникновению определенных событий **ccu** может выполнять действия заданные пользователем. Действия для выполнения необходимо задавать в соответствии с правилами интерпретатора командной строки.

В настоящее время поддерживается только одно событие - создание текстового файла, содержащего информацию об объектах, хранимых в КЭШах. Для задания реакции на это событие необходимо определить параметр `events/on_after_enumeration_creation`. В

шаблоне конфигурации для этого события задана команда создание бинарного представления содержимого КЭШей.

```
#
# Events
events:
    # Command which should be executed after caches' files enumeration
    on_after_enumeration_creation: "cut -d' ' -f2
/var/cache/ccu/data/enumerated.cs | url2dic
/var/cache/ccu/data/enumerated.bin"
```

Журналирование

Параметры журналирования включают в себя путь к файлам журналов, а также уровни сообщений, которые будут записываться в файл. Самый подробный уровень журналирования - *debug*, при его включении будут записываться все возможные сообщения. При установке уровня журналирования в *error*, в файл будут попадать только ошибки. Уровни журналирования задаются отдельно для каждой команды **ccu**.

```
#
# Logging parameters
logging:
    path: /var/log/ccu
    levels: # Enabled log levels for
particular command
# Possible levels are:
# error, warning, info,
diagnostic, debug
# default logging level is
"info"
    load: "info"
    purge: "info"
    remove: "warning"
    online: "info"
    monitor: "info"
```

Сбор статистики

Поля параметра `statistics/collectors/<коллектор статистики>` задают адрес узла и порт, на который будут посылаются данные об использовании КЭШей. В дальнейшем имя коллектора статистики может использоваться при [описании КЭШей](#).

```
#
# Statistics parameters
statistics:
    collectors: # list of collectors'
descriptions
    local: # name of collector
```

description

host: localhost
port: 1600

listening address
listening port

Категории дней и классы времени

Категории дней и классы времени используются для задания правил ограничения по загрузке данных в КЭШ. Для того, чтобы задать категорию дня необходимо создать параметр `day_categories/<категория дня>`, значением которого является список дней, подпадающих в создаваемую категорию дня. Элементом списка дней может быть название дня недели, а также частично или полностью заданная дата. При определении категории дня для конкретной даты происходит просмотр в первую очередь более частных определений, а затем более общих.

```
#
# Day categories
day_categories:      # Defines day categories which could be used in time
class definitions    # Items under "day_categories" define names of
categories;          # To define the day categorie it is necessary to specify
list of values,      # value can be one of following:
                     #   - week day
                     #   - partially or fully defined date in form
DD.MM.YYYY           #
                     #   for example:
                     #   "01"           - the first day of each month
                     #   "07.01"        - the 7th of January of each year
                     #   "09.05.2015"   - 2015, the 9th of May

workdays:           [Mon, Tue, Wed, Thu]
weekend_eve:         [Fri]
weekend:             [Sat, Sun]

holidays:
- "01.01"
- "07.01"
- "23.02"
- "08.03"
- "01.05"
- "09.05"
- "12.06"
- "04.11"
```

Классы времени делят день определенной категории на интервалы времени. Для задания класса времени необходимо создать параметр `time_classes/<класс времени>`, а также для класса времени набор параметров, задающих категорию дня, значением которых является список интервалов времени. Особым образом задается класс времени по умолчанию - значение

параметра `time_classes/default`. Если класс времени не удастся определить по заданным параметрам, то именно класс времени по умолчанию будет использован.

В приведенном ниже примере задан класс времени `peak` и определены интервалы времени для каждой категории дней, которые считаются пиковыми. Если же не одно из правил, задающих пиковый класс времени не может быть использовано, то определяется класс времени по умолчанию `offpeak`.

```
#
# Time classes
time_classes:                # Defines the classes of time
    default:    offpeak      # Specifies the default value for time classes
                                # All others items define the names of time
                                # classes and rules for them definition
                                # Each rule is started from the name of day
                                # categorie and contains
                                # the list of ranges of times in form "HH:MI:SS-
                                HH:MI:SS"

    peak:
        workdays:
            - "00:00:00 - 01:00:00"
            - "17:00:00 - 23:59:59"

        weekend_eve:
            - "00:00:00 - 02:00:00"
            - "16:00:00 - 23:59:59"

        weekend:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"

        holidays:
            - "00:00:00 - 02:00:00"
            - "09:00:00 - 23:59:59"
```

Задачи

Параметры задач задаются как вложенные структуры параметра `jobs`. Существует следующие задачи:

- `jobs/monitor` - мониторинг использования сетевых интерфейсов и использования данных КЭШа;
- `jobs/load` - загрузка/проверка корректности данных КЭШа;
- `jobs/scan` - вспомогательная задача предназначенная для анализа данных [хранилища](#);

Далее, при описании, полное имя параметров будет базироваться на параметре `jobs`.

Мониторинг

Для задачи мониторинга необходимо задать параметры для входящего соединения (`monitor/listener/host` и `monitor/listener/port`), которое будет использовано для приема сообщений от других процессов. Если используется один компьютер для всех процессов, то эти параметры должны совпадать с параметрами, определенными при [описании коллекторов статистики](#).

Другими словами параметр `monitor/listener` задает серверную сторону IP-соединения, а параметры заданные при [описании коллекторов статистики](#) - клиентскую.

Задача мониторинга помимо сбора статистики процессов работающих с КЭШем, также осуществляет сбор системной информации по сетевым интерфейсам. Для того, чтобы ограничить список сетевых интерфейсов, по которым происходит накопление статистики, необходимо задать их список в параметре `monitor/network_interfaces`.

```
monitor:
  listener:                                # listener collects
information from different processes about  # incoming/outgoing traffic
      host:      localhost
      port:      1600

  network_interfaces:                      # list of network interfaces
which is used for getting                  # incoming/outgoing traffic
statistics                                # if list is empty, all
interfaces except "lo" will be used
      -
```

Загрузка

Параметры загрузки разделяются на общие, которые задаются как поля параметра `load`; используемые при offline-загрузке (`load/offline`); используемые при online-загрузке (`load/online`);

Общие параметры загрузки:

```
#
# Load jobs parameters
load:
  ip_binding:                            # optional, list of local IP-adresses
which could be used                        # to binding while loading
      -

  ignored_clients:                       # list of CIDRs for source IP addresses
which should be ignored;
```


Параметры online-загрузки можно разделить на следующие логические группы:

- Описания источников IPFIX потока
- Описание процессов анализа входного IPFIX потока
- Описание коллекторов запрашиваемых объектов
- Описание процессов загрузки данных

Описания источников IPFIX потока задается в структуре load/online/exporters:

```
#
# Load jobs parameters
load:
    # ... skipping ...

    online:
        exporters:                # list of IPFIX exporters
            main:
                queue_size: 1000    # optional, max number of
messages from exporter which is sent to analyzing queue,
                                     # must be between 1 and
100000, default: 1000

                host:      localhost    # host name or IP-address
                port:      1500         # listening port
                protocol:   udp          # one of possible protocols:
UDP or TCP

                information_elements:    # list of used
information elements;
                                     # names are
reserved, values must be in form "PEN/NUM"
                timestamp:      "43823/1001"    # mandatory,
timestamp of event, must be IPFIX dateTimeSeconds
                host:            "43823/1005"    # mandatory,
host name, must be IPFIX string
                path:            "43823/1006"    # mandatory,
path to resource, must be IPFIX string
                login:           "43823/1002"    # optional,
login, must be IPFIX string
                source_ip4:      "43823/1003"    # optional,
source IP-address
                destination_ip4: "43823/1004"    # optional,
destination IP-address
                referral:        "43823/1007"    # optional,
referral, must be IPFIX string
                user_agent:      "43823/1008"    # optional, user
agent, must be IPFIX string
                cookie:          "43823/1009"    # optional,
cookie, must be IPFIX string

    # ... skipping ...
```

Для создания описания источника IPFIX потока необходимо создать структуру `load/online/exporters/<IPFIX-источник>` в соответствии с приведенным выше примером. Следует обратить внимание на задание следующих параметров:

- `load/online/exporters/<IPFIX-источник>/host` и `load/online/exporters/<IPFIX-источник>/port` - определение параметров серверного сокета (на этот адрес должен посылаться IPFIX-поток);
- `load/online/exporters/<IPFIX-источник>/protocol` - определение протокола, который используется для передачи сообщений;

Описание процессов анализа входного IPFIX потока задается в структуре `load/online/analyzing`:

```
#
# Load jobs parameters
load:
    # ... skipping ...

    online:
        # ... skipping ...

        analyzing:                                # the analyzing processes
perform URL rewriting, filtering                    # and binding URLs getting
from exporters to cache's definition                # each analyzzng process
sends valid URLs to collection queue
        parallel_workers: 1                        # optional, number of
parallel analyzing jobs, default 1
        queue_size: 1000                           # optional, max number of
messages from each analyzing process                # which is sent to
collecting queue, must be between 1 and 100000,    # default: 1000

    # ... skipping ...
```

При задании параметров анализаторов следует в первую очередь обратить внимание на параметр `load/online/analyzing/parallel_workers`. При большом количестве сообщений от процессов приема IPFIX-потока потребуется увеличение числа процессов-анализаторов. Признаком необходимости увеличить число процессов-анализаторов является сообщение о переполнении очереди сообщений в журнальных файлах процессов приема IPFIX-потока.

Описание коллекторов запрашиваемых объектов задается в структуре `load/online/collectors`:

```
#
# Load jobs parameters
load:
    # ... skipping ...
```

```

online:
    # ... skipping ...

    collectors:                                # list of collectors
        default:                              # the collector is used for
aggregation of identical URLs                # and distributes them into
time windows                                # each event could hit no
more than 1 window according to event's timestamp
                                             # the time window is a
period of time which defines interval of time in the past
                                             # it is possible to have
many time windows, in this case each window is
                                             # connected to previous one

        slots:                                24          # optional, number of
windows, could not be greater than 100, default 24
        window:                              3600         # optional, window size in
seconds, could not be less than 60 secs, default 3600

    week_by_4_hours:
        slots:                                42
        window:                              14400

    # ... skipping ...

```

Коллектор запрашиваемых объектов является основным фактором для принятия решения о старте загрузки объекта. Логически коллектор представляет собой набор счетчиков в скользящих интервалах времени в прошлом от настоящего момента. При запросе объекта, увеличивается счетчик в определенном интервале, и как только суммарное значение всех счетчиков для объекта достигает порогового значения, задаваемого при [описании КЭШа](#), разрешается загрузка этого объекта.

Для того, чтобы создать описание коллектора, необходима создать параметры:

- load/online/collectors/<коллектор объектов>/slots - количество интервалов;
- load/online/collectors/<коллектор объектов>/window - размер каждого интервала;

В приведенном выше примере описано два коллектора: default и week_by_4_hours. Коллектор default имеет 24 интервала по 1 часу, т.е. позволяет анализировать количество запросов объекта происходящие в течении суток.

Описание процессов загрузки данных задается в структуре load/online/loading:

```

#
# Load jobs parameters
load:
    # ... skipping ...

online:

```

```

# ... skipping ...

loading:
    parallel_workers:      1      # optional, number of
parallel loading jobs, default 1
    queue_size:           100     # optional, max number of
messages from collecting process
                                # which is sent to loading
                                # queue, must be between "parallel_workers" and 10000,
                                # default: 100
    unbuffered_queue_size: 10     # optional, if current queue
size less or equal this value, the messages from
                                # collecting process will
put into loading queue immediately. The messages will be buffered
                                # and ordered before putting
into loading queue according to URLs' weight;
                                # must be between 0 and
"queue_size" - 2 * "parallel_workers", default 2 * "parallel_workers"

# ... skipping ...

```

При задании параметров процессов загрузки следует обратить внимание на количество параллельных процессов. Не следует значительно увеличивать размер очереди (параметр `load/online/loading/queue_size`), т.к. при большом значении этого параметра при старте, очередь будет заполнена объектами, которые удовлетворяют условиям старта загрузки, но во время обработки очереди, возможно появление объектов, которые запрашиваются более часто.

Сканирование

Сканирование - это вспомогательная задача, запускается при операциях загрузки и удаления данных из КЭШей.

```

#
# Scannig jobs parameters
scan:
    workers:                # the scanning processes perform
scan cache directories on demand;
                                # each scanning process sends
                                # information about found items to scanning queue
    parallel_workers:      4      # optional, number of parallel
scanning jobs, default 4
    job_queue_size:        5000    # optional, max number of
messages in job queue, must be between 1000 and 10000
                                # default: 5000
    result_queue_size:     100000  # optional, max number of
messages from each scanner process
                                # which are sent to result
scanning queue, must be between 1000 and 1000000,

```

```
# default: 100000
```

Среди функций, которые выполняются задачей сканирования, можно отметить следующие:

- начальное сканирование каталогов при старте процесса загрузки или удаления старых объектов;
- проверка времени жизни объекта в КЭШе и при необходимости удаление его;
- удаление объектов по запросу от управляющего процесса;
- проверка соответствия хранимого в КЭШе объекта его оригиналу;

Хранилище

Параметры хранилища задаются как поля параметра `storage_parameters`. Все параметры хранилища делятся на следующие группы:

- общие характеристики хранилища - структура `storage_parameters/general`
- описание параметров КЭШей - структура `storage_parameters/caches`

Общие параметры задают путь к корню хранилища, а также максимальный размер, который допустимо занять суммарно под все объекты КЭШей:

```
storage_parameters:
  general:
    path:      "/var/cache/ccu/data"  # optional, base path for caches,
default: /var/cache/ccu/data;
    max_size:  "1T"                  # optional ("unlimited", 0 or
missed - unlimited),                  # maximum data size for all
                                     caches, possible suffixes are:
                                     # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
```

Параметры определенного КЭШа задаются как структура `storage_parameters/caches/<описание КЭШа>`, например:

```
storage_parameters:
  # ... skipping ...

  caches:
    youtube.com:
      is_enabled: yes  # optional (default yes), if
value is "no" the cache definition      # will not be used for loading
data
  # ... skipping ...
```

задает [описание КЭШа](#) с именем `youtube.com`.

Описание КЭШа

При описании параметров КЭШа в этом разделе имена параметров будут базироваться на параметре `storage_parameters/caches/<описание КЭШа>`.

Параметр `is_enabled` определяет будут ли объекты загружаться в КЭШ или нет.

Статистика

Параметры статистики для КЭШа задаются в структуре `statistics`:

```
statistics:
  group: "youtube.com" # name of statistic's
group                                     # all statistics in the
same group is aggregated together
  collector: local # name of statistics
collector
```

Параметр `statistics/group` определяет группу, в которой будут учитываться операции с КЭШем. Параметр `statistics/collector` задает [коллектор статистики](#), который будет производить учет операций с КЭШем.

Online-работа

Для online-загрузки необходимо задать следующие параметры:

```
online:
  collector: "week_by_4_hours" # name of collector which
should be used                # for online processing
  validating:
    interval: 24h # revalidating interval
for particular cache item;    # could be used only for
caches with "general" algorithm
```

Параметр `online/collector` задает имя коллектора запрашиваемых объектов. Параметр `online/validating/interval` может быть использован только КЭШей с *general* алгоритмом загрузки и задает интервал времени, определяющий как часто объект хранимый в КЭШе будет сверяться с его оригиналом.

Загрузка

Для задания правил загрузки используются следующие параметры:

```

loading:
    algorithm:      "youtube.com"      # internal algorithm name
    required_weight: 3                  # min URL's weight which is
required to start loading, default 3

    urls:
        matching:    # list of parameters for URL matching and
rewriting;
                        # - the "key" item is used for unique
identification of object;
                        # - the "target" item specifies target value for
URL; usually target is used as
                        # as additional URL for loading
                        # - the "weight" item is used as weight addition,
default 1;
                        # all items under "sources" are used as a source
masks;
                        # the "key" and "target" can be omitted in this
case
                        # the sources URLs will not be rewritten.
                        # it is necessary to have at least one URL in
sources.
                        # It is possible to use in the key and target's
expression variables from sources;
                        # See RE syntax on
https://docs.python.org/2/library/re.html

        - key:      '\1'
weight:      1
sources:
    - '^www\.youtube\.com/watch?v=([a-zA-Z0-9_-]+)'
    - '^www\.youtube\.com/embed/([a-zA-Z0-9_-]+)'

    ignoring:    # list of regular expr. for ignoring urls
        -        # default: all matched with matching parameters
URLs will be processed

        loadable_rejecting:    # list of regular expr. for rejecting
"real" urls
                                # which is obtained according to
particular URL
                                # default: all "real" URLs will be
processed
                                # See RE syntax on
https://docs.python.org/2/library/re.html

        - "itag=(?! (18|22|140) (?! [0-9]))"

```

- Параметр loading/algorithm определяет алгоритм, который используется при загрузке объекта. Существуют следующие алгоритмы:
 - youtube.com - загрузка видео файлов с youtube.com;

- *rutube.ru* - загрузка видео файлов с *rutube.ru*;
- *vk.com* - загрузка видео файлов с *vk.com*;
- *general* - общий алгоритм загрузки по исходному URL;
- Параметр *loading/required_weight* - задает требуемый вес для коллектора запрошенных объектов, при котором начинается загрузка объекта;
- Параметр *loading/urls/matching* - определяет правила обработки/перезаписи исходного URL; Данный параметр представляет собой список из структур элементов с полями:
 - *key* - ключ объекта, необходимый работы алгоритма загрузки, например для алгоритма *youtube.com* - это идентификатор видео;
 - *weight* - добавляемый вес при учете запрашиваемого объекта в коллекторе;
 - *target* - переписанный исходный URL;
 - *sources* - список регулярных выражений для анализа исходного URL-а;

При задании правил сопоставления можно использовать ссылки на элементы исходного URL-а. Так в приведенном выше примере в качестве значения параметра `key` используется идентификатор видео, заданный как 1-ая группа в регулярных выражениях.

- Параметр `loading/urls/ignoring` - задает список регулярных выражений для URL-ов прошедших стадию сопоставления (`loading/urls/matching`), которые необходимо исключить из загрузки;
- Параметр `loading/urls/loadable_rejecting` - задает список регулярных выражений для загружаемых URL-ов, которые необходимо исключить из загрузки; загружаемые URL-ы - это ссылки на реальные файлы, которые получены при исполнении алгоритма загрузки (`loading/algorithm`) для исходного URL-а, например, для при исполнении алгоритма *youtube.com* для URL-а определяющего WEB-страницу просмотра видео, будет получен список URL-ов на видео и аудио файлы во всех доступных форматах.

Хранение

Для задания правил хранения объектов используются следующие параметры:

```

storage:
    path:      "sites/youtube.com"      # regard to
"storage_parameters/general/path"
    levels:    "2:2"                    # optional, store files in
subfolders created by leading symbols

# of md5 sum of file name
# possible values:
#   "1"       - creating
subfolder by the last symbol of md5 sum
#   "2"       - creating
subfolder by the last two symbols of md5 sum
#   "1:2"     - creating
subfolder by the last symbol of md5 sum
#               and creating
subfolder by 2th and 3th symbols
#               from the end of
md5 sum inside

```

```

# "2:2" - creating
subfolder by the last two symbols of md5 sum
#
# and creating
subfolder by 3th and 4th symbols
#
# from the end of
md5 sum inside
max_size: "unlimited" # optional ("unlimited", 0 or
missed - general limit will be used), # maximum data size, possible
suffixes are: # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
expiry_time: "30d" # optional (0 or missed -
unlimited), expiry time, # default in seconds (without
suffix) # possible suffixes are:
# m - minutes
# h - hours
# d - days

```

- Параметр storage/path задает путь для хранения объектов относительно значения параметра storage_parameters/general/path;
- Параметр storage/levels определяет количество дополнительных каталогов создаваемых на основе суммы md5 от имени файла;
- Параметр storage/max_size задает максимальный суммарный объем всех объектов, хранимых в КЭШе. Если данное ограничение не задано, то используется ограничение на все хранилище;
- Параметр storage/expiry_time определяет время жизни объекта в КЭШе с момента его загрузки;

Ограничения

Для объектов КЭШа возможно задать ограничения, например:

```

constraints: # optional, specifies
constraints for loaded files
min_file_size: 128k # optional (default 0), min file
size, possible suffixes are: # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
max_file_size: "unlimited" # optional (default unlimited),
max file size, possible suffixes are: # Kb, Mb, Gb, Tb, Pb or without
trailing "b" just K, M, G, T, P
# optional, command which should be executed after file loading;
# if command returns non-zero result, the loaded file will be
assumed as invalid and will be removed;
# the next variables could be used in the command:

```

```

# {cache_name} - cache name
# {full_file_name} - fully qualified file name
post_load_validation: "ft=`file -b {full_file_name}`; echo $ft;
echo $ft | grep -E \"WebM|ISO Media|MPEG\" 2>/dev/null 1>&2"

```

SSD-Кэширование

Существует возможность организовать КЭШ-ирование наиболее часто запрашиваемых объектов хранилища на SSD, для этого необходимо определить параметры:

```

#
# SSD caching parameters
ssd_caching:
    is_enabled:          no                # enable or not SSD caching,
by default SSD caching is disabled
    path:                "/var/cache/ccu/ssd" # optional, base path for
SSD caching area, default: /var/cache/ccu/ssd
# NOTES:
# - to SSD caching you
have to mount SSD on specified path (or create link /var/cache/ccu/ssd to
mount point)
# - changes this parameter
should be performed in cooperation with changes in web server configuration
# (it is recommended do
not change this parameter)
    max_size:            128G              # maximum data size for
storing on SSD, possible suffixes are:
# Kb, Mb, Gb, Tb, Pb or
without trailing "b" just K, M, G, T, P
    required_weight:     10                # optional, min URL's weight
which is required to start caching, default 10

    uri_prefixes:        # URI prefixes which are
used in HTTP requests to access data
    ssd_cache_requests:   "/ssd"           # optional, prefix for
requests to SSD cache, default: "/ssd"
    main_storage_requests: "/cache"        # optional, prefix for
requests to main storage, default: "/cache"

    frozen_time:         180               # optional, interval of time
in seconds after putting file into cache in which that
# file cannot be replaced by
another one, default: 3 * collecting window
# frozen time cannot be less
than collecting window

    collector:            # the collector is used for aggregation
of identical requests
# and distributes them into time windows
    slots:                60              # optional, number of windows, could not

```

```

be greater than 120, default 60
    window:          60          # optional, window size in seconds,
could not be less than 60 secs, default 60

    workers:          # the worker processes perform scan
cache directories on demand;

                                # each process sends information about
found items to result queue
    parallel_workers:  2          # optional, number of parallel jobs,
default 2
    job_queue_size:    5000       # optional, max number of messages in
job queue, must be between 1000 and 10000
                                # default: 5000
    result_queue_size: 100000    # optional, max number of messages from
each process
                                # which is sent to result queue, must be
between 1000 and 1000000,
                                # default: 100000

```

При задании параметров следует обратить внимание на количество рабочих процессов и параметры коллектора.

Начальная конфигурация

При установке **ccu** создается файл шаблона конфигурации - /etc/ccu/ccu.conf.default. Для получения рабочей конфигурации необходимо скопировать файл шаблона в /etc/ccu/ccu.conf и отредактировать/проверить как минимум значение следующих параметров:

- [jobs/monitor/network_interfaces](#)
- [jobs/load/ip_binding](#)
- [jobs/load/online/exporters/main/host](#)
- [jobs/load/online/exporters/main/port](#)
- [jobs/load/online/exporters/main/protocol](#)
- [storage_parameters/general/max_size](#)