Сервис CLI. Описание

CLI сервис — это backend часть системы CLI. Предназначен для работы в качестве демона и обслуживания одного или нескольких клиентов (одним экземпляром CLI сервиса).

Функциональность:

* Позволяет одновременно работать нескольким пользователям.
* Поддерживает возможность по аутентификации и авторизации пользователей.
* Связывает с каждым пользователем, подключившимся к CLI сервису, его клиентскую сессию: обработчик клиентских запросов и клиентское состояние. Клиентские сессии двух разных пользователей не зависят друг от друга.
* Поддерживает автоматический выход пользователя по таймауту (и очистку его клиентской сессии).
* Поддерживает динамическую конфигурацию при запуске: информация о наборе команд, конфигурации машины состояний CLI, уровнях доступа загружается из соответствующих конфигурационных файлов при запуске.
* Содержит команды, режимы работы CLI и переходы между ними согласно спецификации CLI.
* Каждое клиентское состояние содержит свою машину состояний (конечный автомат) CLI, используемую для управления режимами работы CLI.
* Информация о наборе команд содержит следующую информацию: команды для работы с реальным железом, команды, реализующие некоторую логику на стороне CLI сервиса и команды, реализующие некоторую логику на стороне терминального клиента. Информация о командах терминального клиента нужна исключительно для формирования помощи и данных автодополнения.
* Принимает ввод от пользователя, разбирает его и выполняет соответствующую цепочку команд в случае успешности разбора. Ввод пользователя может быть разобран как цепочка из одной команды, так и из нескольких команд.
* Перед выполнением команды происходит динамическая проверка возможности выполнить команду в данный момент времени. Результат динамической проверки зависит от следующих факторов: наличие у пользователя прав для выполнения команды, доступность команды в данном режиме работы CLI.
* Во время выполнения, команда может изменить состояние клиента, взаимодействовать с железом, а также передать данные в потоки вывода и выводы ошибок. Результатом выполнения каждой команды является ее код возврата: код возврата 0 означает, что команды выполнилась успешно, в противном случае значение кода возврата будет отлично от 0. Обработчик клиентских запросов может преобразовать этот код возврата в сообщение и отправить это сообщение клиенту.
* Реализует протокол взаимодействия с терминальным клиентом и создает точку подключения для терминального клиента через сокеты.
* Каждая команда CLI сервиса содержит поток ввода, поток вывода и поток вывода ошибок. Команды в цепочке команд соединяются через потоки ввода, вывода и вывода ошибок. Обработчик клиентских запросов является итоговым приемником вывода цепочки команд.
* Каждая команда CLI сервиса содержит справочную информацию о самой себе. Также каждая команда содержит свойства, упрощающие разбор пользовательского ввода и построения цепочки команд. Эти свойства содержат следующую информацию: принимает ли команда ввод через поток ввода, взаимодействует ли команда с реальным железом, является ли команда заглушкой для представления команды терминального клиента и т. д.

Поведение

* При запуске CLI сервиса для взаимодействия с клиентами создается только слушающий сокет (listen сокет). При подключении клиента (при запуске терминального клиента) к этому сокету, создается клиентский сокет, через который и происходит взаимодействие клиента с CLI сервисом; также создается обработчик запросов для взаимодействия только с этим клиентом и клиентское состояние.
* Перед тем как начать работу пользователь должен войти в систему послав запрос о входе в систему с корректными именем пользователя и паролем. При попытке работы до успешного входа пользователя в систему выполнение обработчика клиентских запросов будет завершено с ошибкой (в будущем, возможно, это поведение будет изменено на более user friendly).
* После того как пользователь успешно вошел в систему (в случае успешного входа пользователя в систему) в его клиентское состояние помещаются его реквизиты (по крайней мере имя пользователя и уровень его доступа).
* При получении запроса от клиента на выполнение некоторой CLI команды (при получении ввода пользователя) происходит ее разбор. В результате разбора создается цепочка из одной или нескольких команд.
* После разбора запроса от клиента в цепочку команд, все команды из этой цепочки выполняются последовательно. Перед выполнением команды происходит проверка, может ли данная команда быть выполнена в данный момент времени. Доступность команды для выполнения зависит от следующих факторов: достаточно ли у текущего пользователя прав для выполнения данной команды, доступна ли команда в данном режиме работы CLI. Если команда для выполнения в данный момент времени не доступна, она не будет выполнена и выполнение все цепочки команд прервется. При этом результат работы предыдущих команд из цепочки останется (транзакционность выполнения цепочки команд не предусмотрена).
* Если выполнение некоторой команды из цепочки команд завершилось успешно, то эта команда (имя команды) используется для возможного изменения текущего состояния машины состояний CLI (не всякая команда приводит к изменению состояния) и, соответственно, режима работы CLI. Машина состояний CLI для данного клиента хранится в его клиентском состоянии.
* Если выполнение некоторой команды из цепочки команд закончилось с ошибкой, то выполнение цепочки команд прерывается; код завершения выполнения этой команды будет кодом завершения выполнения всей цепочки.
* Если выполнение всех команд из цепочки команд завершилось успешно, то мы считаем, что выполнение всей цепочки команд завершилось успешно; кодом завершения выполнения всей цепочки команд будет 0.
* Если команда не является последней командой в цепочке команд, то вывод из потока вывода этой команды направляется в поток ввода следующей за ней команды. Если команда является последней командой в цепочке команд, то вывод из потока вывода этой команды направляется обработчику клиентских запросов. Если перенаправления потока вывода ошибок (в поток вывода) нет, то данные из потока вывода ошибок направляются обработчику клиентских запросов. Если перенаправления потока вывода ошибок (в поток вывода) есть, то данные из потока вывода ошибок направляются в поток вывода этой же команды.
* В случае выхода пользователя из системы (в случае успешного вызова команды **logout**) взаимодействие с клиентом прекращается. Перед завершением работы клиенту шлется уведомление о завершении работы клиентской сессии После этого обработчик клиентских запросов завершает свою работу, клиентский сокет закрывается на стороне CLI сервиса.
* При отключении клиента от CLI сервиса обработчик клиентских запросов завершает свою работу и закрывает клиентский сокет на стороне CLI сервиса.
* Если пользователь бездействует в течении некоторого времени, то обработчик клиентских запросов на CLI сервисе завершает свою работу, отсылает об этом уведомление пользователю и закрывает клиентский сокет на стороне CLI сервиса.

Детали реализации

* Все конфигурационные файлы будут хранить данные в виде объектов (term) языка Erlang. С одной стороны это сильно облегчит и ускорит разбор таких файлов, а с другой стороны особо не усложнит их синтаксис.
* Каждой команде CLI сервиса соответствует некоторый модуль, реализующий поведение команды.
* В качестве потока вывода для команды мы будем использовать «буфер обмена»: специальный процесс, накапливающий вывод пользователя. В этот же «буфер обмена» мы будем накапливать вывод ошибок в случае его перенаправления. Когда команда закончит свое выполнение весь вывод, накопленный в этом «буфере обмена», становится потоком ввода для следующей команды. Весь вывод последней команды в цепочке команд, а также весь вывод ошибок, которые не были перенаправлены в поток вывода, отправляются на обработчик клиентских запросов. Поэтому каждой команде необходимо передавать три идентификатора процесса в качестве потоков ввода, вывода и вывода ошибок.
* Первая команда в цепочке в качестве потока ввода будет использовать пустой «буфер обмена», т.к. на данный момент у нас нет команд, получающих ввод пользователя.
* Если запрос от клиента корректен, то будет создана цепочка команд, соответствующая запросу от клиента. Помимо этого, будет создан контекст выполнения этой цепочки, связывающий команды в цепочке команд друг с другом. Контекст выполнения команд передает каждой команде соответствующие идентификаторы процессов (в качестве потоков ввода, вывода и вывода ошибок). Также контекст выполнения после выполнения каждой команды посылает «буферу обмена» сообщение о том, что весь накопленный вывод можно считать вводом.
* После создания цепочки команд и контекста выполнения, мы запускаем эту цепочку команд на выполнение; для этого мы создаем отдельный процесс и выполняем в нем некоторую функцию из контекста выполнения.
* Если выполнение цепочки команд завершилось с ошибкой (если какая-либо команда из цепочки команд вернула код возврата отличный от 0), то обработчик клиентских запросов вернет клиенту сообщение об этом через стандартный поток вывода ошибок.

Машина состояний (конечный автомат) CLI

Клиентское состояние, помимо всего прочего содержит машину состояний для управления режимами работы CLI. На данный момент конечный автомат имеет следующий вид:

*1) Состояния (режимы работы CLI)*

* Фундаментальный режим работы
* Режим глобальной настройки
* Режим настройки интерфейса
* Режим настройки группы интерфейсов
* Режим настройки VLAN

2) События, приводящие к смене состояния (команды)

* Успешный вход в систему: фундаментальный режим работы становится текущим состоянием.
* Команда ***configure terminal***: фундаментальный режим работы → режим глобальной настройки.
* Команда ***interface***: режим глобальной настройки → режим настройки интерфейса.
* Команда ***interface range***: режим глобальной настройки → режим настройки группы интерфейсов.
* Команда ***vlan***: режим глобальной настройки → режим настройки VLAN.
* Команда ***end***: режим глобальной настройки, режим настройки интерфейса, режим настройки группы интерфейсов, режим настройки VLAN → фундаментальный режим работы.
* Команда ***exit***: режим настройки интерфейса, режим настройки группы интерфейсов, режим настройки VLAN → режим глобальной настройки; режим глобальной настройки → фундаментальный режим работы.
* Команда **logout**: завершение работы с данным клиентом.

Все остальные команды состояние машины состояний CLI не меняют. В машину состояний CLI уведомление о выполнении команды, попадает только после ее успешного выполнения. Если команда выполняется не успешно, то машина состояний CLI уведомления не получает.

Структура (состояния и переходы) машины состояний будет меняться. Так, например, настройка некоторых протоколов (BGP, OSPF, MSTP) требуют добавления одного или более режима CLI. Поэтому мы будем загружать конфигурации конечного автомата из внешнего источника, вместо жесткой реализации в теле конечного автомата.

Команды CLI сервиса

Ниже приведены все команды, а также ряд свойств этих команд, которые известны CLI сервису на данный момент. Следует отметить следующее: режимы CLI определяют в каких режимах работы можно использовать ту или иную команду. Это свойство не является свойством команды: доступность той или иной команды в том или ином режиме работы устанавливается извне (при создании машины состояний CLI).

|  |
| --- |
| **logout** |
| Определение: выход пользователя из системы  Использование: **logout**  Режимы CLI: фундаментальный режим работы, режим глобальной настройки, режим настройки интерфейсов, режим настройки группы интерфейсов, режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: нет |
| **configure terminal** |
| Определение: переход в режим глобальной настройки  Использование: см. спецификацию  Режимы CLI: фундаментальный режим работы  Использует поток ввода: нет  Взаимодействие с реальным железом: возможно? |
| **ping** |
| Определение: отправка ICMP эхо-запросов  Использование: см. спецификацию  Режимы CLI: фундаментальный режим работы  Использует поток ввода: нет  Взаимодействие с реальным железом: нет |
| **interface** |
| Определение: переход в режим настройки интерфейса  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки  Использует поток ввода: нет  Взаимодействие с реальным железом: возможно? |
| **interface range** |
| Определение: переход в режим настройки группы интерфейсов  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки  Использует поток ввода: нет  Взаимодействие с реальным железом: возможно? |
| **vlan** |
| Определение: создание VLAN и переход в режим настройки VLAN  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки, режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **no vlan** |
| Определение: удаление VLAN  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки, режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **switchport access vlan** |
| Определение: настройка номера VLAN, который будет использоваться при работе интерфейса в режиме «Access»  Использование: см. спецификацию  Режимы CLI: режим настройки интерфейса, режим настройки группы интерфейсов  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **no switchport access vlan** |
| Определение: установка значения по умолчанию для номера VLAN, который будет использоваться при работе интерфейса в режиме «Access»  Использование: см. спецификацию  Режимы CLI: режим настройки интерфейса, режим настройки группы интерфейсов  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **name** |
| Определение: настройка имени VLAN  Использование: см. спецификацию  Режимы CLI: режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **no name** |
| Определение: удаление имени VLAN  Использование: см. спецификацию  Режимы CLI: режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: да |
| **end** |
| Определение: возврат в фундаментальный режим  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки, режим настройки интерфейса, режим настройки группы интерфейсов, режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: возможно? |
| **exit** |
| Определение: возврат в предыдущий по уровню режим  Использование: см. спецификацию  Режимы CLI: режим глобальной настройки, режим настройки интерфейса, режим настройки группы интерфейсов, режим настройки VLAN  Использует поток ввода: нет  Взаимодействие с реальным железом: возможно? |
| **show vlan** |
| Определение: вывод информации о VLAN, созданных на устройстве  Использование: см. спецификацию  Режимы CLI: фундаментальный режим работы  Использует поток ввода: нет  Взаимодействие с реальным железом: да |

Конфигурирование

Для конфигурирования CLI сервиса необходимо наличие нескольких конфигурационных файлов. Все конфигурационные файлы содержат данные в виде объектов (term) языка Erlang; такой подход позволяет сразу же получать объекты языка Erlang при чтении содержимого этих файлов при помощи, например, функции **io:read/2**. Следующие файлы используются на данный момент для конфигурации CLI сервиса:

* Основной конфигурационный файл.
* Конфигурационный файл с описанием всех используемых команд.
* Конфигурационный файл с описанием машины состояний CLI.
* Конфигурационный файл с определением всех зарегистрированных пользователей в системе.
* Конфигурационный файл с определением разрешений на использование команд.

Расположение основного конфигурационного файла определяется в app файле CLI сервиса. Расположение остальных конфигурационных файлов определяется в основном конфигурационном файле. При этом все относительные пути рассматриваются относительно директории, в которой располагается основной конфигурационный файл. Обычно все конфигурационные файлы располагаются вместе — в одной директории.

Основной конфигурационный файл по умолчанию имеет следующее имя: *cli\_service.conf* (по умолчанию полное имя */etc/cli\_service/cli\_service.conf*). Формат этого файла следующий:

**[{'cli\_terminal', CliTerminalConfig,**

**{'commands', DataSourceConfig},**

**{'cli\_fsm', DataSourceConfig},**

**{'authentication\_service', DataSourceConfig},**

**{'authorization\_service', DataSourceConfig}].**

Здесь запись с ключом **'cli\_terminal'** содержит конфигурацию взаимодействия с терминальным клиентом, запись с ключом **'commands'** содержит информацию о расположении конфигурационного файла с описанием всех используемых команд, запись с ключом **'cli\_fsm'** содержит информацию о расположении конфигурационного файла с описанием машины состояний CLI, запись с ключом **'authentication\_service'** содержит информацию о расположении конфигурационного файла с определением всех зарегистрированных пользователей в системе, запись с ключом **'authorization\_service'** содержит информацию о расположении конфигурационного файла с определением разрешений на использование команд. Конфигурация взаимодействия с терминальным клиентом **CliTerminalConfig** имеет следующий формат:

**[{'port\_number', PortNumber}, {'downtime', Downtime}]**.

Здесь **PortNumber** — номер listen порта для подключения терминального клиента, **Downtime** — время простоя терминального клиента в секундах. Информация о расположении конфигурационного файла **DataSourceConfig** имеет следующий формат:

**[{'data\_source', DataSourceLocation}]**.

Здесь DataSourceLocation — путь (относительный или абсолютный) до конфигурационного файла.

Конфигурационный файл с описанием всех используемых команд по умолчанию имеет следующее имя: *command\_data*. Формат этого файла следующий:

**[{CommandName :: atom(), CommandModule :: atom()}].**

Здесь, **CommandName** - имя команды, **CommandModule** - имя модуля, в котором эта команда определена. Для всех команд, определенных в системе должна быть соответствующая запись в этом файле.

Конфигурационный файл с описанием машины состояний CLI по умолчанию имеет следующее имя: *cli\_fsm\_data*. Формат этого файла следующий:

**[{'states', StateTable}, {'transitions', TransitionTable}, {'initial\_state', Statename :: atom()}].**

Здесь запись с ключом **'states'** содержит таблицу всех состояний в системе, запись с ключом **'transitions'** — таблицу всех переходов в системе, а запись с ключом **'initial\_state'** — имя начального состояния. Таблица всех состояний в системе имеет следующий вид:

**[{Statename :: atom(), Representation :: string(), AllowedCommands :: [atom()]}].**

Здесь **Statename** — имя состояния, **Representation** — строковое представление состояния (для отображения состояния в строке подсказки), **AllowedCommands** — список имен команд, доступных в данном состоянии. Таблица всех переходов в системе имеет следующий вид:

**[{FromState :: atom(), ToState :: atom(), CommandName :: atom()}].**

Здесь **FromState** — имя исходного состояния, **ToState** — имя состояния, в которое будет осуществлен переход, **CommandName** — имя команды, в результате успешного выполнения которой будет осуществлен переход.

Конфигурационный файл с определением всех зарегистрированных пользователей в системе по умолчанию имеет следующее имя: *authentication\_data*. Формат этого файла следующий:

**[{Uid :: non\_neg\_integer(), Username :: string(), PasswordHash :: binary(), AccessLevel :: non\_neg\_integer()}].**

Здесь **Uid** — идентификатор пользователя, **Username** — имя пользователя, **PasswordHash** — хэш пароля пользователя (об алгоритме его формирования см. ...), **AccessLevel** — уровень доступа пользователя (пользователь может запускать команды, уровень доступа которых не выше его уровня доступа).

Конфигурационный файл с определением разрешений на использование команд по умолчанию имеет следующее имя: *authorization\_data*. Формат этого файла следующий:

**[{CommandName :: atom(), AccessLevel :: non\_neg\_integer()}].**

Здесь **CommandName** — имя команды, **AccessLevel** — уровень доступа к команде (команду могут запускать на выполнение пользователи, уровень доступа которых не меньше ее уровня). Для всех команд, определенных в системе должна быть соответствующая запись в этом файле.