ЗАО «АЙТИ»

УТВЕРЖДАЮ

Генеральный директор
ЗАО «АйТи».

_____ Бакиев О.Р.

"19" декабря 2011 г.

Создание высокотехнологичного производства комплексных решений в области предметно-ориентированных облачных вычислений для нужд науки, промышленности, бизнеса и социальной сферы

ПРОГРАММНАЯ БИБЛИОТЕКА ДЛЯ ВСТРАИВАНИЯ ПАКЕТОВ ДЛИТЕЛЬНОГО ИСПОЛНЕНИЯ CLAVIRE/LRWFLIB

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 60

2011

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

ЗАО «АЙТИ»

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 60-ЛУ

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНАЯ БИБЛИОТЕКА ДЛЯ ВСТРАИВАНИЯ ПАКЕТОВ ДЛИТЕЛЬНОГО ИСПОЛНЕНИЯ CLAVIRE/LRWFLIB

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 60

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

ЛИСТОВ 15

2011

АННОТАЦИЯ

Документ содержит описание программной библиотеки CLAVIRE/LRWFLib RU.СНАБ.80066-06 01 60, используемой при встраивании вычислительных предметных пакетов (сервисов) длительного исполнения в платформу CLAVIRE. Программная библиотека разработана в ходе выполнения проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» в рамках реализации постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства».

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Программная библиотека для встраивания пакетов длительного исполнения CLAVIRE/LRWFLib RU.СНАБ.80066-06 01 60 предназначена для использования при интеграции прикладных вычислительных пакетов (сервисов) в многопрофильную инструментально-технологической платформу (МИТП) CLAVIRE, работающих согласно модели LRWF. Библиотека разработана на языке программирования Python (версия 2.7) и использует библиотеку pyzmq (версия 2.1.10) для взаимодействия и библиотеку сериализации в формат BSON, входящую в состав библиотеки pymongo (версия 2.1). Соответственно для функционирования библиотеки необходимо наличие интерпретатора Python с установленными библиотеками pymongo и pyzmq.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программная библиотека разработана для интеграции вычислительных пакетов с инфраструктурой CLAVIRE для функционирования заданий длительного исполнения платформы. Он позволяет организовать взаимодействие с компонентами подсистемы исполнения WF и другими пакетами длительного исполнения. Данная библиотека предназначена для встраивания в приложения, написанные на языке Python, либо для написания приложений-оберток для пакетов длительного исполнения. В структуру библиотеки заложены общие принципы для пакетов длительного исполнения и для обеспечения поддержки других языков программирования, она может быть портирована под любую платформу исполнения и язык программирования.

К функциональному назначению программной библиотеки относится:

1) конфигурирование пакета длительного исполнения для работы в среде CLAVIRE;
2) предоставление программного каркаса для приложения длительного исполнения;
3) предоставление программного интерфейса для взаимодействия с другими пакетами длительного исполнения;
4) реализация принципа реактивного программирования поверх схемы оповещения/подписки ZMQ.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Принципы LRWF

Для преодоления ограничений, связанных с пакетным исполнением WF, в платформе CLAVIRE используется специальное расширение модели WF – WF длительного исполнения (Long running WF), которое выражается в возможности работы WF в интерактивном виде, долгое время (или бесконечно) с возможностью управления и динамического изменения WF, а также возможностью обмена информацией между шагами во время исполнения. Используемая модель основана на нескольких ключевых позициях, которые отличают ее от пакетного WF.

1) Поддержка бесконечно исполняющихся WF. Узлы, время жизни которых не ограничено внутри WF – узлы длительного исполнения.
2) Поддержка коммуникации между узлами WF во время исполнения. Одновременная работа нескольких (возможно, всех) узлов WF, обменивающихся информацией по каналам связи.
3) Возможность изменения WF во время исполнения за счет сценария WF, а также за счет внешнего управления. Узлы и дуги такого WF могут быть добавлены или удалены во время исполнения.

Стоит обратить внимание на то, что модель не определяет метода реализации и вида коммуникаций между узлами. Не важно, каким конкретным образом реализован обмен: будь то передача сообщений, использование потоков данных или удаленный вызов процедур – все они инвариантны и каждый может быть реализован на базе любого другого. Метод передачи управляющих воздействий также может быть изменен. В качестве метода коммуникации в CLAVIRE был выбран механизм обмена сообщениями между узлами WF по сети. Управление узлами организовано в виде удаленного вызова процедур (далее – выполнения команд) поверх механизма передачи сообщений. Команда, передаваемая узлу, характеризуется именем для ее идентификации и параметром. Результат выполнения команды запаковывается и передается как ответ на входное сообщение.

В качестве абстракции для коммуникационной связи узлов LRWF применяется парадигма реактивного программирования, ориентированная на потоки данных и распространение изменений. Каждый узел WF имеет входные и выходные параметры (длительного исполнения). Связи можно устанавливать между выходным параметром одного узла и входным – другого. При этом, когда изменяется значение выходного

параметра, все завязанные на него входные параметры других узлов изменяют свое значение. В результате этого процесс, входной параметр которого изменился, учитывает новое значение в своей работе и может пересчитать значения своих выходных параметров. Таким образом, происходит распространение изменений по всем связанным узлам WF.

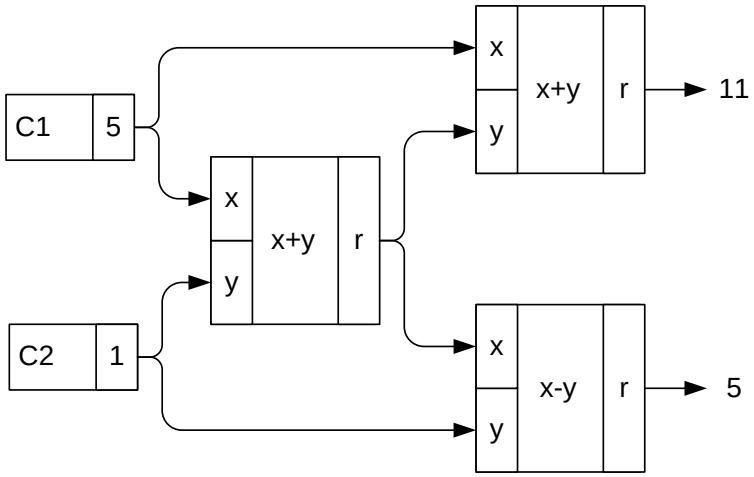

Рисунок 3.1 – Пример приложения, построенного на концепции реактивного программирования

Для иллюстрации данного принципа на рис. 3.1 представлена схема WF с тремя узлами, представляющими бинарные арифметические операции. Если во время работы такого WF будет изменено значение C1 с 5 на 1, то на выходах значения изменятся на 3 и 1 (сверху вниз соответственно). Реализация принципа распространения изменений значений параметров построена на шаблоне проектирования – подписка/оповещение (pub/sub).

В качестве среды коммуникации и передачи команд в CLAVIRE используется программный сетевой транспортный уровень ZMQ (zero message queue), реализуемый посредством соответствующей программной библиотеки ZMQ. Данная библиотека позволяет использовать различные транспортные протоколы. Процессы могут быть соединены связями с различными отношениями: один к одному, многие ко многим, один ко многим, многие к одному. Помимо этого в библиотеке поддерживаются различные схемы коммуникации, реализованные поверх механизма передачи сообщений: вызов процедур, оповещение/подписка, распределение задач. Данная библиотека построена согласно асинхронной модели ввода-вывода.

Вычислительный пакет, встраиваемый в платформу в режиме длительного исполнения, должен технически поддерживать механизмы управления и коммуникации. Поэтому одним из требований при выборе технологий для реализации LRWF была кроссплатформенность и возможность объединять пакеты, разработанные на различных

языках программирования. Поддержка механизмов LRWF для конкретного пакета может быть обеспечена несколькими путями: за счет доработки самого пакета (если имеется возможность развития его исходного кода) либо разработки обертки, которая позволяет установить сопряжение между механизмами управления, поддерживаемыми пакетом и механизмами, предоставляемыми платформой.

Далее для встраивания пакета длительного исполнения необходимо описать его на языке EasyPackage. С точки зрения подсистемы исполнения вычислительный пакет длительного исполнения – это обычный узел пакетного запуска. Поэтому для такого пакета сохраняются все возможности по описанию пакетного режима запуска, но и приобретаются дополнительные. В таком описании пакет помечается ключевым словом, которое свидетельствует о том, что данный пакет поддерживает режим длительного исполнения.

Для задания WF длительного исполнения в виде скрипта в язык EasyFlow были внедрены синтаксические конструкции, которые позволяют: помечать узел как узел длительного исполнения; задавать коммуникационные зависимости путем связи выходных и входных параметров.

```
1 ~step Camera runs crowdg (
2    delay = 30,
3    agent_count = 10000,
4    agent_r = 0.4,
5    tracing = true
6 );
7 ~step Model runs crowdm (
8    isLr = true,
9    needPress = false,
10   time_step = 0.2,
11   agent_count = 10000,
12   tracing = true,
13   init_agent_pos <- Camera.init_agent_pos
14 );
```

Рисунок 3.2 – Фрагмент описания WF длительного исполнения на EasyFlow

На рис. 3.2 представлен пример скрипта на языке EasyFlow, описывающий WF моделирования поведения толпы людей в условиях паники, состоящий из двух узлов. Первый узел периодически генерирует исходное положение людей на карте, а второй моделирует перемещение людей, начиная с полученного исходного состояния. В отличие от узлов пакетного запуска данные узлы помечены ключевым словом ~step, которое говорит о том, что узлы являются узлами длительного исполнения (строки 1 и 7). В скрипте задана коммуникационная связь между узлами (строка 13): выходной параметр *init_agent_pos* узла *Camera* связан с одноименным входным параметром узла Model.

Рассмотрим процесс исполнения LRWF в рамках платформы CLAVIRE. На рис. 3.3 представлена схема взаимодействия компонентов платформы при исполнении задач длительного исполнения.

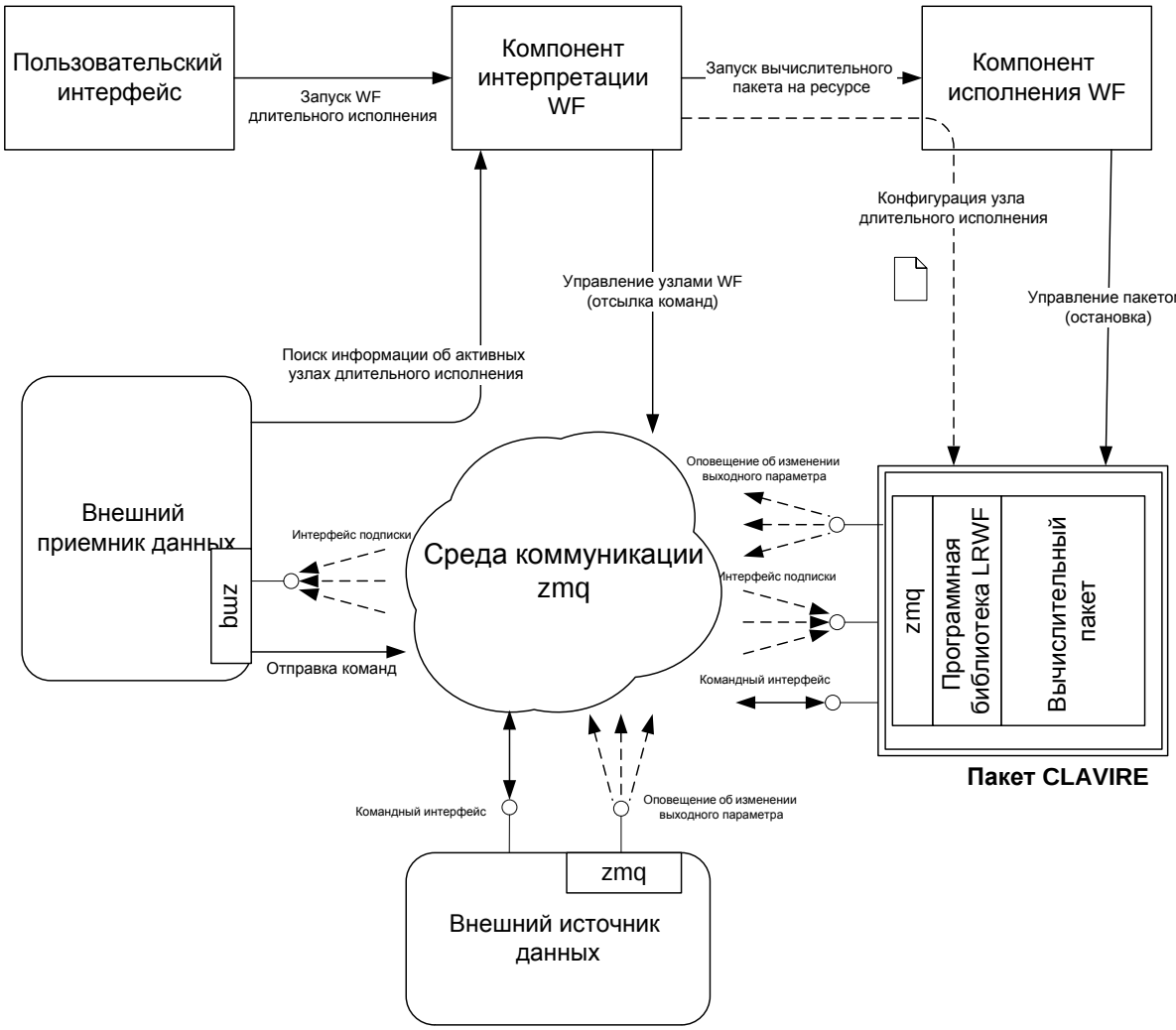

Рисунок 3.3 – Принципиальная схема функционирования LRWF

Скрипт на языке EasyFlow поступает на обработку в модуль интерпретации, где после разбора скрипта WF строится внутреннее представление WF в виде дерева связанных узлов. Узел запускается, только когда все условия, указанные для него, удовлетворены. К условиям, для примера, относится то, что все узлы, от которых он зависит по данным, завершились, или то, что все узлы, от которых он зависит по коммуникациям, уже запущены на ресурсах. Запуск узла длительного исполнения производится через компонент исполнения WF. В результате процесса планирования, согласно указанным для узла условиям подбора ресурса, находится активный вычислительный ресурс с системой управления, поддерживающей режим длительного исполнения. Для компонента исполнения WF узел длительного исполнения не отличается

от обычного узла, запускаемого в пакетном режиме. Конфигурация запуска узла длительного исполнения помимо стандартного набора содержит данные о связях между параметрами, где указано, какие подписки необходимо установить. Информация о подписках формируется модулем интерпретации WF и передается в пакет в виде конфигурационного файла, пример которого представлен на рис. 3.4.

```
#---- DEFAULTS ----
ID = 1
#---- SUBSCRIPTIONS ----
0 tcp://192.168.129.135:5000 init_agent_pos -> init_agent_pos
```

Рисунок 3.4 – Пример конфигурационного файла пакета длительного исполнения

Подписка устанавливается пакетом, входной параметр которого связан с выходным параметром другого пакета. После установки подписки данные об изменении выходного параметра пересылаются подписанному пакету.

Интеграция с внешними источниками данных или клиентами осуществляется за счет описанных выше механизма передачи команд и подписки/оповещения. Для внешних приемников данных в модуле интерпретации реализована специальная интерфейсная операция запроса информации об активных пакетах длительного исполнения. Используя эту операцию, потребитель данных, генерируемых WF длительного исполнения, например визуализатор, может в режиме реального времени получать данные от пакета.

С точки зрения ресурсного уровня платформы представленная реализация не задает определенных требований к ресурсам и их системам управления помимо возможности доступа к запущенному пакету по сети. Если есть необходимость отправки команд либо получения данных извне, то необходимо настроить доступ к внешнему ресурсу с вычислительных ресурсов платформы.

3.2. Программная архитектура

Центральной частью программной библиотеки является класс LrInterface. Он содержит интерфейс взаимодействия с другими пакетами длительного исполнения, выполняет интеграционные задачи с инфраструктурой LRWF, а также несет в себе программный каркас, который может быть использован при разработке приложения длительного исполнения.

На рис. 3.5 представлена принципиальная схема встраивания библиотеки при разработке пакета длительного исполнения. Для взаимодействия с другими пакетами длительного исполнения библиотека предоставляет два способа: посылка команд, взаимодействие через подписку/оповещение. Для этого в библиотеке предоставляется три

интерфейсных метода: SendCommand – метод отсылки команд, UpdateOutput – метод оповещения об изменении выходного параметра и CheckInbox – метод приема входящих сообщений. Рассмотрим подробнее последний метод – CheckInbox. Он реализован для обработки входящих сообщений: извлекает сначала все сообщения из очереди входящих сообщений командного интерфейса и вызывает для каждого сообщения метод OnCommand (или OnStop, если пришла команда остановки), затем – извлекает все сообщения из очереди интерфейса подписки, для которых вызывается метод OnInputParamChanged.

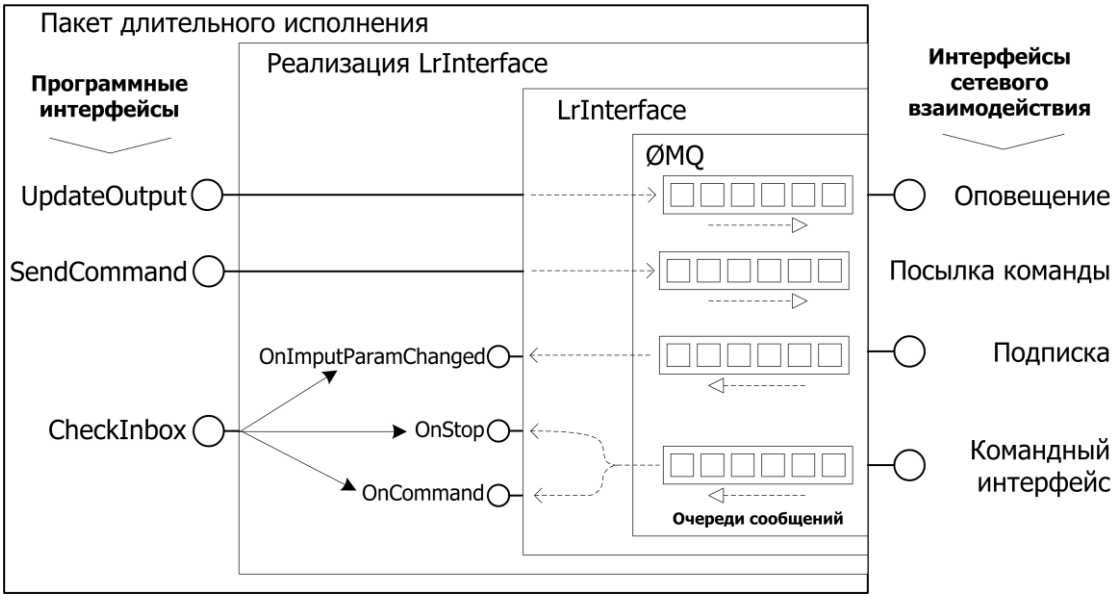

Рисунок 3.5 – Схема встраивания программной библиотеки

Существует два способа использования данного класса при встраивании нового пакета: доработка кода пакета для интеграции в инфраструктуру LRWF либо разработка на основе данного класса программы-обертки. Второй способ может быть использован в ситуации, когда исходный код пакета недоступен для модификации. При использовании библиотеки из кода пакета необходимо создать новый класс-реализацию LrInterface. В данном классе необходимо реализовать методы OnInputParamChanged, OnStop, OnCommand.

Библиотека разработана на языке Python и может быть встроена только в приложения на Python. Важная особенность библиотеки – она несет в себе основные принципы проектирования пакетов длительного исполнения для работы в среде CLAVIRE. Данные принципы и архитектура библиотеки могут быть использованы для других языков программирования. Разработка подобной библиотеки под другие языки программирования и среды исполнения (или портирование) является несложной задачей

(библиотеки для работы с BSON и ZMQ доступны для большинства популярных языков программирования).

3.3. Основные классы

Ниже приводятся сокращенные описания структуры и методов основных классов программной библиотеки для встраивания пакетов длительного исполнения.

3.3.1. Класс LrInterface

Основной класс библиотеки, являющийся каркасом приложения длительного исполнения. При встраивании пакета абстрактные методы данного класса должны быть реализованы.

Методы

- UpdateOutput – обновление значения выходного параметра. Возвращаемое значение отсутствует.
- SendCommand – отправка команды другому пакету. Аргументы: адрес, имя команды (строка), параметр команды (произвольный хэш).
- CheckInbox – метод проверки внешних сообщений. Возвращаемое значение отсутствует.
- OnInputParamChanged – метод обработки изменения входного параметра. Должен быть переопределен в наследнике. Аргументы: имя параметра (строка), значение параметра (произвольный хэш). Возвращаемое значение отсутствует.
- OnStop – метод обработки команды остановки. Должен быть переопределен в наследнике.
- OnCommand – метод обработки полученных команд. Должен быть переопределен в наследнике. Аргументы: имя команды (строка), параметр команды (произвольный хэш). Возвращает хэш вида {“result”: <результат выполнения команды>}.
- _GetConfiguration – считывает из конфигурационного файла, переданного CLAVIRE, адреса интерфейсов для инициализации, список подписок на изменения в других пакетах. Возвращает кортеж из словаря считанных значений, подписок.
- _InitCommunications – использует считанные из конфигурационного файла адреса для инициализации интерфейсов: оповещения, командного интерфейса. Затем устанавливает подписки.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Программный компонент предназначен для функционирования на аппаратных системах с характеристиками:

- архитектура процессора – любая, поддерживаемая интерпретатором Python и библиотеками pyzmq, pymongo;
- минимальный объем оперативной памяти – 1 ГБ;
- минимальная тактовая частота процессора – 1 ГГц;
- минимальная пропускная способность сетевого интерфейса – 100 Мбит/с.

5. ВЫЗОВ И ЗАГРУЗКА

Компонент реализован в виде библиотеки языка Python «lr.py». Подключение производится стандартным способом включения в программный код внешнего модуля. После подключения компонента необходимо разработать класс-наследник от LrInterface (см. раздел 3.3.1). На рис. 5.1 представлен пример реализации класса.

```
import time
import lr
import sys
class MyLr(lr.LrInterface):
    def __init__(s, conf):
        super(MyLr, s).__init__(conf)
        s.tick0 = 0
        s.tick1 = 0
    def OnInputParamChanged(s, pname, pval):
        if(pname == "tick0"):
            s.tick0 = pval
        elif(pname == "tick1"):
            s.tick1 = pval
def main(argv):
    Lr = MyLr(argv[1])
    count = 0
    while(1):
        time.sleep(1)
        Lr.CheckInbox()
        Lr.UpdateOutput("counter", Lr.tick0 + Lr.tick1 + count)
        count += 1
        print ".",
if __name__ == "__main__":
    main(sys.argv)
```

Рисунок. 5.1 – Пример реализации абстрактного класса LrInterface

Загрузка полученной программы зависит от типа загрузки модифицируемого пакета. По умолчанию пакет загружается путем запуска из командной строки через интерпретатор языка Python.

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для программной библиотеки является конфигурационный файл, передаваемый через инфраструктуру CLAVIRE. Файл передается в пакет под именем «lr.config». Структура файла состоит из двух секций: указания конфигурационных свойств, списка подписок. К свойствам относятся: идентификатор процесса длительного исполнения, адреса командного интерфейса и интерфейса оповещения. На рис. 6.1 представлен пример конфигурационного файла с тремя свойствами и двумя подписками.

```
#---- DEFAULTS ----
PUBLISH_ENDPOINT = tcp://*:5200
COMMAND_ENDPOINT = tcp://*:5201
ID = 111

#---- SUBSCRIPTIONS ----
000 tcp://localhost:5100 counter -> tick0
222 tcp://localhost:5300 counter -> tick1
```

Рисунок 6.1 – Пример конфигурационного файла системных подписок

7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными являются отсылаемые во время работы пакета длительного исполнения данные по сети в формате BSON. При отсылке сообщений используется простой протокол. При обновлении значения выходного параметра рассылается структура вида {"param": ParamName, "value": ParamValue}, где ParamName – имя параметра, ParamValue – значение параметра. При отправке команд используется структура вида {"command": CommandName, "param": CommandData}, где CommandName – имя команды, CommandData – хэш или произвольный объект.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД	База данных
WF	Workflow, поток заданий
LRWF	WF длительного исполнения
МИТП	Многопрофильная инструментально-технологическая платформа

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	**Номера листов (страниц)**				**Всего листов (страниц) в докум.**	**№ документа**	**Входящий № сопроводительного докум. и дата**	**Подп.**	**Дата**
	измененных	**замененных**	**новых**	**Аннулированных**					