**ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ**

**«СКАДИ»**

**(ООО «СКАДИ»)**

**СОГЛАСОВАНО** **УТВЕРЖДАЮ**

# СИСТЕМА КОМПЛЕКСНОГО ОБУЧЕНИЯ ЯДЕРНЫМ ТЕХНОЛОГИЯМ

## Руководство системного программиста

## 34185873.425510.005.32.М

**(На 70 листах)**

**2025**

**РАЗРАБОТАЛ**

**СОДЕРЖАНИЕ**

1 ОБЩИЕ СВЕДЕНИЯ О ПО ..... 5
1.1 Наименование ..... 5
1.2 Назначение и функции ..... 5
1.3 Необходимые технические и программные средства ..... 5
2 СТРУКТУРА ПО ..... 6
2.1 Состав серверного ПО ..... 6
2.2 Состав ПО контейнеров ..... 6
2.3 Скрипты на стороне сервера ..... 7
3 НАСТРОЙКА ПРОГРАММЫ ..... 8
3.1 Инсталляция ..... 8
3.1.1 Инсталляция для ОС Astra Linux, ОС Linux ..... 8
3.1.2 Инсталляция ПО nginx ..... 8
3.1.3 Инсталляция контейнеров с моделями процессов ..... 8
3.2 Настройка проекта ..... 8
3.2.1 Базовые файлы проекта ..... 8
3.2.2 Образы моделей в контейнерах ..... 10
3.3 Настройка переменных окружения ..... 10
3.4 Установка лицензии проекта ..... 11
3.5 Конфигурация задач и ресурсов ..... 12
3.5.1 Выбор состава задач ..... 12
3.5.2 Конфигурация глобалов общей памяти ..... 12
3.5.3 Настройка веб-сокет сервера ..... 14
3.5.4 Файлы начальных условий ..... 14
3.6 Создание и запуск контейнера ..... 15
3.6.1 Образы отладочного режима ..... 15
3.6.2 Образы рабочего режима ..... 15
3.6.3 Контейнер отладочного режима ..... 15
3.6.4 Контейнер рабочего режима ..... 16
3.7 Конфигурация связи СКОЯТ - тренажер ..... 16
3.7.1 Сервер обмена на стороне СКОЯТ ..... 16
3.7.2 Агент обмена на стороне тренажера ..... 17
3.8 Конфигурация СУБД ..... 17
3.8.1 Встроенная СУБД Sqlite ..... 17
3.8.2 Внешняя СУБД PostgreSQL ..... 18
3.9 Конфигурация nginx ..... 18
3.9.1 Проксирование соединения с WoAccess ..... 18
3.9.2 Проксирование соединения с WoData ..... 18
3.9.3 Проксирование соединения с WoMark ..... 19
3.9.4 Проксирование соединения с WsShmCntrl внутри контейнеров ..... 19
4 ПРОВЕРКА ПРОГРАММЫ ..... 20
4.1 Проверка наличия ПО и загрузки ..... 20
4.2 Проверка загрузки контейнеров ..... 20
4.3 Проверка загрузки глобалов общей памяти ..... 22
4.4 Контроль статуса задач ..... 22

4.5 Контроль состояния сети ..... 23
4.6 Контроль взаимодействия с сервисом WoMark ..... 24
5 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ ..... 25
5.1 Рестартуемые сервисы СКОЯТ ..... 25
6 СООБЩЕНИЯ СИСТЕМНОМУ ОПЕРАТОРУ ..... 26
6.1 Лог системных сообщений ..... 26
7 ИНСТРУКЦИЯ ПО УСТАНОВКЕ ..... 28
ПРИЛОЖЕНИЯ ..... 29
Приложение 1 – Описание конфигурации СКОЯТ WoDocker.json ..... 29
Приложение 2 – Состав задач и утилит СКАДИ-МОДЕЛЬ ..... 32
Приложение 3 – Динамически загружаемые модули ..... 34
Приложение 4 – Список переменных окружения ..... 35
Приложение 5 – Опции командной строки ..... 36
ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ ..... 37
ЛИСТ ИЗМЕНЕНИЙ ..... 38

# 1 ОБЩИЕ СВЕДЕНИЯ О ПО

## 1.1 НАИМЕНОВАНИЕ

Наименование изделия — Система комплексного обучения ядерным технологиям «СКОЯТ 1.0».

Сокращенное наименование: «СКОЯТ».

Программа «СКОЯТ» («SCONT») зарегистрирована в реестре программ для ЭВМ № 2025617289 от 25.03.2025.

Наименование документа 34185873.425510.005.32.М, обозначение по ГОСТ 34.201-89.

«СКОЯТ 1.0» разрабатывается на основании документа 34185873.425510.005.ТЗ.М, Техническое задание, утвержденного 2 декабря 2024г.

## 1.2 НАЗНАЧЕНИЕ И ФУНКЦИИ

Программное обеспечение СКОЯТ предназначено для обучения персонала ядерным технологиям в локальных и глобальных сетях.

Программное обеспечение СКАДИ-МОДЕЛЬ предназначено для разработки моделей технологических процессов. ПО СКАДИ-МОДЕЛЬ входит в состав программного обеспечения СКОЯТ.

Программное обеспечение СКОЯТ реализует следующие функции:

- регистрация и управление доступом и ресурсами СКОЯТ;
- учет используемых ресурсов;
- управление технологическим процессом посредством системы верхнего уровня;
- осуществление связи с моделями технологических процессов;
- архивирование действий обучаемого и контрольных точек данных технологического процесса;
- представления обучающих программ;
- оценивания действий обучаемого.

## 1.3 НЕОБХОДИМЫЕ ТЕХНИЧЕСКИЕ И ПРОГРАММНЫЕ СРЕДСТВА

Клиентский доступ к ПО СКОЯТ требует только наличие браузера. Все требования относятся к необходимым серверным техническим и программным средствам.

Для функционирования «СКОЯТ 1.0» требуется наличие операционной системы семейств Linux или Windows. Операционное системное программное обеспечение устанавливается на все узлы, где должно работать ПО СКОЯТ.

ПО «СКОЯТ 1.0» предназначено для работы с является операционным программным обеспечением, требующим для своей работы 64-разрядных процессоров AMD/Intel с архитектурой x86_64.

Для использования средств защиты информации операционных систем рекомендуется использование ОС Astra Linux, начиная с версии 1.6.

# 2 СТРУКТУРА ПО

## 2.1 СОСТАВ СЕРВЕРНОГО ПО

Программное обеспечение СКОЯТ состоит из набора исполняемых модулей, расположенных в каталоге инсталляции scont. Имена исполняемых модулей СКОЯТ имеют шаблон Wo* для Linux-систем. Исполняемые модули СКОЯТ разбиваются на сервисы.

| Имя | Наименование | Система |
|---|---|---|
| WoAccess | Сервер управления доступом | Регистрации и управления доступом |
| WoData | Сервер доступа к данным об использовании ресурсов | Система учета использованных ресурсов |
| WoMark | Сервер контроля действий обучаемого | Оценивание действий обучаемого |

Таблица 1 - Состав программных сервисов СКОЯТ

Сервисы запускаются на серверных аппаратных средствах. Ниже приведен состав типовой загрузки.

```
user@osco:~/osco$ ps x | grep Wo
 128577 pts/0    Sl     0:01 WoAccess -db sqlite
 128578 pts/0    Sl     6:23 WoData -trace_level -1 -jcfg WoDocker.json
 128579 pts/0    Sl     6:39 WoMark -trace_level -1 -jcfg WoDocker.json
 146853 pts/2    S+     0:00 grep Wo
user@osco:~/osco$
```

Рисунок 1 – Типовая загрузка СКОЯТ

## 2.2 СОСТАВ ПО КОНТЕЙНЕРОВ

При работе СКОЯТ создаются контейнеры, формирующиеся из образов, содержащих модели. Образы должны быть заранее подготовлены и зарегистрированы.

```
user@osco:~/osco$ docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
model-rn80   v1.5d     578c6f27adbd   12 days ago    257MB
model-rb60   v1.1d     0099db11c7d4   12 days ago    257MB
model-yalk   v1.0d     3a5a25396dc3   12 days ago    257MB
model-tk70   v1.0d     2fb81bbacdee   12 days ago    257MB
model-oisc   v6d       457f4b0c9f9b   2 weeks ago    115MB
model-rb60   v1.2d     345b65ed129b   5 weeks ago    290MB
debian       11-slim   b507fb61aa87   5 months ago   80.7MB
debian       11        c6410bc184e7   5 months ago   124MB
user@osco:~/osco$
```

Рисунок 2 – Типовой список образов

Сервис WoMark создает и запускает контейнеры из состава имеющихся образов. Список возможных образов добавлен в конфигурацию. Конфигурационной файл, в котором описаны возможные образы для работы с ПО docker, имеет имя по умолчанию WoDocker.json. На основании такого файла сервис WoMark формирует возможный список контейнеров для загрузки. Конкретный файл может быть задан опцией командной строки –jcfg в следующем виде.

```
WoMark –jcfg WoDocker-release.json &
```

В приведенном примере используется конфигурационный файл WoDocker-release.json. Описание конфигурационного файла WoDocker приведено в приложении 1.

Список запускаемых задач контейнеров соответствует записям массива TASK[] конфигурационный файлов DsmConfig контейнеров. Перечень задач СКАДИ-МОДЕЛЬ, размещенных внутри контейнеров, представлен в таблице 2-1 приложения 2.

Задачи СКАДИ-МОДЕЛЬ запускаются менеджером моделей DsMM в каждом контейнере.

```
user@osco:~/osco$ docker ps --all
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS         PORTS
                  NAMES
68c95e98f069   model-rb60:v1.1d   "/bin/sh -c 'DsMM -c…"   10 seconds ago   Up 9 seconds   0.0.0
.0:12021->12020/tcp   model-rb60-dock0_1
user@osco:~/osco$
```

Рисунок 3 – Контейнер с командой DsMM

Внутри контейнера должен быть обеспечен доступ к задачам СКАДИ-МОДЕЛЬ. Конкретный вид конфигурационного файла определяется установленной функциональностью контейнера.

Утилиты сервиса и отладки (таблица 2-2 приложения 2) используются пользователями и разработчиками СКАДИ-МОДЕЛЬ при необходимости доступа внутри контейнера.

Динамические загружаемые проектные модули реализуют функциональность отдельных моделей. Они относятся к конкретному проекту. ПО СКАДИ-МОДЕЛЬ реализует интерфейс для каждого типа модулей, позволяющий подключать конкретную реализацию, определяемую конфигурацией. Динамически загружаемые проектные модули располагаются в корневом каталоге проекта, имена их каталогов соответствуют шаблону Dp*.

На рисунке 4 представлен каталог для контейнера rb60 проекта model.

```
drwxr-xr-x 1 root root 4096 May 21 11:58 ..
-rw-r--r-- 1 root root 4512 May 21 11:58 DsmConfig.cfg
-rw-r--r-- 1 root root 4519 May 20 16:45 DsmConfig.tfg
drwxr-xr-x 4 1001 1001 4096 May 12 17:01 web
-rw-r--r-- 1 root root  180 Apr 10 10:55 Oml.cfg
drwxr-xr-x 5 1001 1001 4096 Feb  6 16:00 Dpgc
drwxr-xr-x 7 1001 1001 4096 Feb  6 16:00 Dpmodel
-rw-r--r-- 1 root root  300 Jan 22 16:19 Dsm_rb60-image.lic
-rw-r--r-- 1 root root  291 Jan 22 16:19 Wss_rb60-image.lic
drwxr-xr-x 4 1001 1001 4096 Jan 14 15:33 Dsmodel
root@model-rb60-dock0_1:/home/osco#
```

Рисунок 4 – Проектный каталог с динамическими модулями

В приведенном на рисунке проектном каталоге отмечены подкаталоги Dpgc, Dpmodel. Каждый из приведенных модулей на рисунке является реализацией по умолчанию. Соответствие между именем модуля и интерфейса приведено в таблице 3-1 приложения 3.

## 2.3 СКРИПТЫ НА СТОРОНЕ СЕРВЕРА

Для получения оценок результатов выполнения лабораторных работ используются скрипт на стороне сервера WoEval и скрипты оценки Estimate, устанавливаемые для каждой конкретной задачи.

Динамически загружаемые проектные модули загружаются задачами на этапе старта или по запросу, в зависимости от назначения модуля.

# 3 НАСТРОЙКА ПРОГРАММЫ

## 3.1 ИНСТАЛЛЯЦИЯ

### 3.1.1 Инсталляция для ОС Astra Linux, ОС Linux

Установить пакет ПО СКАДИ средствами ОС:
Добавить репозиторий scadi-debian
В файл /etc/apt/sources.list
deb [trusted=yes] http://plg3-srv3.corp.diaprom.ru/repo/scadi-debian scadi main
Выполнить команду sudo apt update
Выполнить команду sudo apt install osco
Пакет osco содержит все нужные зависимости для ПО scont

### 3.1.2 Инсталляция ПО nginx

Выполнить команду sudo apt install nginx

### 3.1.3 Инсталляция ПО docker

Выполнить команду sudo apt install docker

## 3.2 НАСТРОЙКА ПРОЕКТА

### 3.2.1 Базовые файлы проекта

Поскольку СКОЯТ предназначена для построения обучающих на основе базовой платформы, для ее использования необходимо создание проекта обучающей системы. Для проекта обучающей системы создается корневой каталог проекта. При дальнейшей работе все данные и конфигурация, относящиеся к проекту, должны находиться в корневом каталоге проекта.

Установить корневой каталог проекта (вместо названия osco подставляется конкретное имя разрабатываемого проекта):

```
mkdir osco
cd osco
```

Корневой каталог проекта используется для размещения проектных данных и программного обеспечения, относящихся к конкретному проекту. Он должен иметь определенную структуру для обеспечения доступа к указанным данным.

Проектный каталог для каждого проекта должен представлять из себя структуру:

osco> - место для конфигурационных файлов;

osco/doc> - место для файлов раздела Документация;

osco/images> - место для графических данных;

osco/labs - место для файлов описаний заданий на лабораторные работы;

osco/results - место для результатов выполнения лабораторных работ;

osco/web - место для веб-контента системы и SVG-файлов;

osco/web/images> - каталог изображений графических форматов;

osco/web/vk - каталог видеокадров проекта;

osco/Dp* - плагины подключаемых программных модулей (используются только для установки отладочной Debug-версии).

Наличие общего конфигурационного файла должно обеспечивать старт системы командами:

```
osco> WoAccess -db sqlite&
osco> WoData -jcfg WoDocker.json &
osco> WoMark -jcfg WoDocker.json &
```

В качестве аргументов команды задается имя конфигурационного файла. Если имя конфигурационного файла не задано, то берется по умолчанию WoDocker.json. Описание содержимого конфигурационного файла приведено в приложении 1.

Возможно создание нескольких конфигураций (например, для отладочного и рабочего режимов).

Конфигурация отладочного режима содержит каталоги базы данных проекта, рабочих файлов проекта и плагинов подключаемых модулей. Конфигурация рабочего режима имеет release-версии контейнеров с установленными в каждый контейнер каталогами.

Каталог базы данных проекта project/db содержит набор файлов для базы данных с расширением .tsd:

- types.tsd - проектно-зависимые метаданные;
- global_<globname>.tsd – набор параметров базы данных реального времени для globname;
- osco_*.xxx – файлы мэппинга базы данных реального времени.

Каталог рабочих файлов проекта project/wrk содержит набор файлов групп параметров с расширением *.pgg.

Плагины подключаемых программных модулей образует каталоги с именами систем модулей для каждого. Имя системы для модуля представляет собой префикс модуля до первой заглавной буквы, начиная со второй. Например, плагины DpgcValves и DpgcPumps будут находиться внутри одной системы модулей с именем Dpgc. Система модулей представляет собой каталог, в нашем примере Dpgc. Внутри системы модулей производится разделение на подкаталоги с именами:

- Clur – содержит динамически загружаемые коды программных модулей в виде объектных файлов *.o, скомпилированных из LLVM;
- Slur – содержит символьные файлы интерфейса программных модулей;
- Mod – содержит исходные тексты программных модулей (опционально, для конфигурации отладочного режима).

Файл внутри подкаталогов имеют имена за вычетом имени системы модулей. Для нашего примера DpgcValves представляется файлом кода Dpgc/Clur/DpgcValves.o и символьного файла Dpgc/Slur/Valves.osf. Кроме этого в каталоге содержатся модули, импортируемые из плагинов. При наличии исходного текста в каталоге может содержаться файл Dpgc/Mod/Valves.mob.

### 3.2.2 Образы моделей в контейнерах

Перечень доступных моделей представляет собой список образов, из которых может создаваться контейнер по запросу пользователя. Список образов, доступный на данном узле, может быть запрошен средствами docker.

```
user@osco:~/osco$
user@osco:~/osco$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
model-rn80   v1.5d     0b25bf2d315e   6 days ago      290MB
model-oisc   v6d       9ed0825d9631   6 days ago      283MB
model-rb60   v1.1d     8c20ac10b896   6 days ago      290MB
model-rn80   v1.4d     b7a49f03904f   5 weeks ago     291MB
debian       11        c6410bc184e7   2 months ago    124MB
debian       11-slim   b507fb61aa87   2 months ago    80.7MB
user@osco:~/osco$
```

Рисунок 5 – Список доступных образов

Имя образа для моделей в репозитории (REPOSITORY) состоит из префикса model- и имени секции, например, model-rn80. Имя образа является также началом имени создаваемого из образа контейнера. Тэг образа содержит версию реализации. Последняя буква d в версии означает отладочную debug-версию.

## 3.3 НАСТРОЙКА ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

При регистрации приложения СКОЯТ требуется задание конфигурации для работы Яндекса.

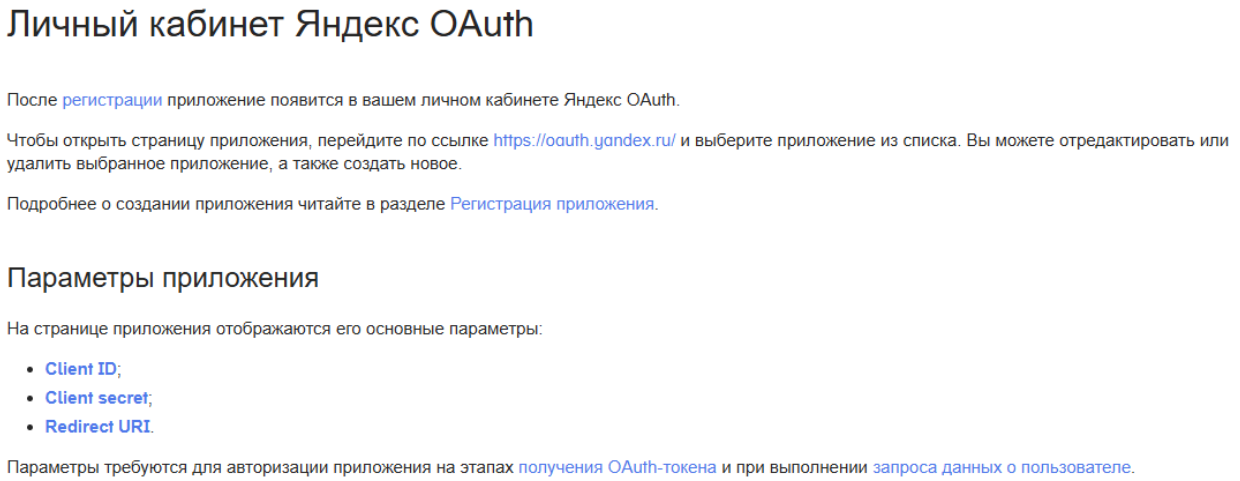


Рисунок 6 – Регистрация приложения в Яндекс-системе

Зарегистрированное приложение СКОЯТ имеет сгенеренные параметры Client ID, Client Secret, Redirect URI. Необходима установка конфигурационных параметров согласно полученным значениям.

```
export YANDEX_CLIENT_ID=qwerty1234567890
export YANDEX_CLIENT_SECRET=1234567890qwerty
export REDIRECT_URL=http://osco/callback
```

Необходимо задать уникальное имя на сервере.

```
export SESSION_SECRET=my_session_secret_123
```

Далее необходимы настройки переходов на модули обработки при авторизации.

```
export LOCAL_REDIRECT_URL=http://osco/data/indexw.html
```

```
export AUTH_REDIRECT=http://osco/account
```

Также необходимы настройки скриптов выполнения программ оценки и вне-контейнерного запуска.

```
export WOMARK_SHELL=/usr/bin/python3
export WOMARK_SCRIPT=WoEval
export WOMARK_UDSHELL=/usr/bin/python3
export WOMARK_UDSCRIPT=WoManage.py
```

Далее необходима настройка версии API при работе с docker.

```
export DOCKER_API_VERSION=1.41
```

Полный список переменных окружения приведен в приложении 4.

### 3.4 УСТАНОВКА ЛИЦЕНЗИИ ПРОЕКТА

Приложения СКОЯТ-сервисов WoAccess, WoData, WoMark используют файл лицензии Wos_<hostname>.lic как необходимое условие запуска для стандартного сервера данного проекта. Файл лицензии должен находиться в каталоге <project>. Файл лицензии должен иметь формат Wos_<hostname>.lic, где hostname является именем хоста стандартного сервера. Ниже пример для хоста osco-net-local.

```
user@osco-net-local:~/osco$ ls *.lic
Wos_osco-net-local.lic
```

При отсутствии валидной лицензии приложения-сервисы будут аварийно завершаться с выводом сообщения: “SCONT License error file=Wos_osco.lic”. Работа ПО СКОЯТ возможна только при наличии валидной лицензии.

При просроченной лицензии приложениями-сервисами будет выведено сообщение: “SCONT License file= Wos_osco.lic expired=14/10/2022”. Работа ПО СКОЯТ возможна только при наличии актуальной лицензии.

Лицензия вводит ограничения на максимальное количество одновременно запущенных ресурсов. Каждый ресурс представляет собой пользовательский контейнер. При превышении числа контейнеров над максимальным значением, заданным в лицензии выводится сообщение: “SCONT License resources requested=200 > max=100”. Работа ПО СКОЯТ возможна только при отсутствии выхода количества ресурсов за границы лицензии.

Приложение менеджера моделей DsMM использует файл лицензии Dsm_<hostname>.lic в контейнерах и автономных загрузках. Файл лицензии должен находиться в каталоге <project>. Файл лицензии должен иметь формат Dsm_<hostname>.lic, где hostname является именем хоста стандартного сервера. Для контейнеров, полученных из образов, имя хоста задается позже при генерации контейнера. Поэтому в конфигурационном файле DsmConfig.cfg задается имя, интерпретируемое как имя хоста для лицензии.

```
lic_hostname=rb60-image
```

В таком случае ищется файл лицензии Dsm_<lic_hostname>.lic, в нашем примере Dsm_rb60-image.lic.

При отсутствии валидной лицензии приложением DsMM будет выведено сообщение:

```
“SCADI license error file=Dsm_xxxxx.lic”
```

При просроченной лицензии приложением DsRM будет выведено сообщение:

```
“SCADI License file=Dsm_xxxxx.lic expired=DD/MM/YYYY”
```

Лицензия вводит ограничения на максимальное количество тэгов – аналоговых, дискретных, целочисленных или иных пользовательских объектов. При превышении числа объектов над максимальным значением, заданным в лицензии выводится сообщение:

```
“SCADI License amount of tags=140000 exceeds maximum=100000”
```

Задача веб-сокет сервера WsShmCntrl использует файл лицензии Wss_<hostname>.lic в контейнерах. Файл лицензии должен находиться в каталоге <project>. Файл лицензии должен иметь формат Wss_<lic_hostname>.lic, где lic_hostname является именем хоста в конфигурационном файле DsmConfig.cfg.

Все валидные лицензии проекта должны устанавливаться при инсталляции проекта.

## 3.5 КОНФИГУРАЦИЯ ЗАДАЧ И РЕСУРСОВ

### 3.5.1 Выбор состава задач

Состав задач СКАДИ-МОДЕЛЬ описывается в элементах массива TASK[] конфигурационного файла (таблица 2-1 приложения 2). Рантайм-менеджер DsMM всегда является первой задачей.

```
TASK[0].name=DsMM
```

Остальные задачи должны иметь имя в виде <prefix><suffix>, где prefix представляет конкретное имя из таблицы 2-1 приложения 2, а suffix указывает на экземпляр задачи. Например, DsModelGC подразумевает единственный экземпляр приложения DsModel, а задачи DsModelGC и DsModevKav представляют собой 2 процесса приложения DsModel, запущенные с разными именами и разными параметрами.

```
TASK[1].name=DsPersnap
TASK[1].exe_path=DsPersnap
TASK[1].watch_limit=600
TASK[1].restart_limit=-1
TASK[2].name=DsModelGC
TASK[2].exe_path=DsModel
TASK[2].command_line=-hls DpgcValves,DpgcCNValves,DpgcPumps,DpgcBlks,DpgcAvrs
TASK[2].watch_limit=60
TASK[2].restart_limit=-1
TASK[2].update_ms=1000
TASK[2].init_trace_level=64
TASK[2].run_at=STDSERVER
```

Запуск приложения с суффиксом означает передачу дополнительного параметра task_name при старте приложения. Например, запуск производится DsModel –task_name DsModelGC. Сами приложения, в зависимости от параметров командной строки, выбирают, с какой частью конфигурации они работают.

Параметр TASK[].run_at определяет способ запуска задачи. Серверные приложения STDSERVER предполагают прием и распространение данных, CLIENT обещает наличие клиентской поддержки, MANUAL предполагает ручной запуск. Отсутствие Task.run_at означает автоматический запуск везде.

Автоматический запуск со стороны менеджера моделей означает также наличие контроля за работой приложения. Параметр TASK[].watch_limit задает предел счетчика исправности в секундах. При достижении счетчика в 0 происходит рестарт приложения. Число рестартов определяется параметром TASK[].restart_limit. Отрицательное значение TASK[].restart_limit устанавливает бесконечное число рестартов.

### 3.5.2 Конфигурация глобалов общей памяти

Массив данных конфигурации должен заполняться в зависимости от типа глобала. Список конфигурируемых типов глобалов общей памяти приведен в таблице 2.

| Тип | Описание | Мета/Данные | Структура |
|---|---|---|---|
| n | Пользовательские реляционные таблицы | Данные | Таблица |
| i | Индексы реляционных таблиц | Индексы | Таблица |
| q | Очереди сообщений | Очереди | ЦиклБуфер |

| | | | |
|---|---|---|---|
| a | Архивные данные | Записи | ЦиклБуфер |
| c | Заголовочные записи архива | Заголовки | ЦиклБуфер |

Таблица 2. Список типов конфигурируемых глобалов

Глобалы n пользовательских реляционных таблиц должны содержать необходимое число записей для размещения всех тэгов разрабатываемой системы. Для глобала n необходимо указывать тип объекта и размер глобала в записях.

```
GLOBAL[0].globtype=n
GLOBAL[0].globname=AIs
GLOBAL[0].otype=AI
GLOBAL[0].max_count=15000
GLOBAL[0].max_count=$nAIs
GLOBAL[0].in_groups=
GLOBAL[0].persist_interval=-1
GLOBAL[0].flags=
```

Параметр globtype означает тип глобала (n в нашем случае), а параметр otype означает тип объекта (AI для приведенного примера). Параметр max_count задает размер глобала в записях. Это означает, что в базе данных реального времени размещается 15000 объектов AI. При оптимизации базы данных под размер задачи создается файл шаблона DsmConfig.tfg, в котором вместо характеристик размеров используются подстановки. Например, подстановка max_count=$nAIs означает, что в сгенеренном из данного шаблона DsmConfig.cfg будет подставлено число реальных объектов AI в глобале. Параметр persist_interval задает сброс на диск 1 раз в 30 секунд.

Содержательная часть файлов глобала n загружается из позиций db/global_<globname>.tsd. После загрузки разделяемая памяти маппируется на db/<project>_<globname>.xxx. Например, аналоговые данные AI загружаются из db/global_AIs.tsd, а файл разделяемой памяти создается в виде db/osco_AIs.tsd для проекта osco. Файлы xxx сохраняются на протяжении всей загрузки.

Индексные глобалы i задают индексные таблицы для ускорения доступа. Используются только для больших глобалов.

```
GLOBAL[12].globtype=i
GLOBAL[12].globname=AIi
GLOBAL[12].refglobal=AIs
GLOBAL[12].flags=C
```

Параметр refglobal должен указывать на глобал n, на который ссылается индексный глобал.

Глобалы q очередей сообщений сопутствуют ряду задач СКАДИ, которые формируют выходные данные. Например, для сервера обмена с КСО понадобится очередь сообщений, приведенная ниже.

```
GLOBAL[2].globtype=q
GLOBAL[2].globname=GCq
GLOBAL[2].max_count=2000
GLOBAL[2].reftask=DsModelGC
GLOBAL[2].run_at=STDSERVER
```

Для очереди сообщений необходимо указывать размер в записях max_count. Параметр reftask должен указывать на задачу, которая отвечает за формирование данных очереди. Обработчики работают с очередью на предмет проверки поступления очередных данных.

### 3.5.3 Настройка веб-сокет сервера

В системе СКОЯТ применяется задача WsShmCntrl в качестве веб-сокет сервера. Ниже приведен пример конфигурации задачи WsShmCntrl.

```
TASK[4].name=WsShmCntrl
TASK[4].exe_path=WsShmCntrl
TASK[4].command_line=-port 12020 -add_port username -interval 100ms
TASK[4].run_at=STDSERVER
TASK[4].watch_limit=60
TASK[4].restart_limit=-1
TASK[4].update_ms=1000
```

В аргументы командной строки command_line заносятся параметры конфигурации задач. В примере задан порт веб-сервера 12020, интервал 100миллисекунд и флаг инкрементирования номера порта по имени пользователя. Перечень всех опций командной строки приведен в приложении 5.

Глобалы q очередей сообщений для веб-сокет сервера должны содержать пользовательские запросы на управление данными.

```
GLOBAL[13].globtype=q
GLOBAL[13].globname=Controlq
GLOBAL[13].max_count=1000
GLOBAL[13].in_groups=VALS DVP
GLOBAL[13].in_xgroups=SIMULATOR
GLOBAL[13].reftask=WsShmCntrl
```

Для таких глобалов можно использовать небольшой max_count=1000.

Веб-сокет сервер записывает команды только в очереди сообщений и не записывает их в глобалы n общей памяти. Для целей синхронизации очередей с глобалами n в список добавляется задача DsDVP.

```
TASK[5].name=DsDVP
TASK[5].exe_path=DsDVP
TASK[5].run_at=NOTAUXSERVER
TASK[5].watch_limit=60
TASK[5].restart_limit=-1
TASK[5].update_ms=1000
```

Синхронизатор очередей DsDVP также требует при инициализации свою выходную очередь, используемую в справочно-диагностических целях.

```
GLOBAL[12].globtype=q
GLOBAL[12].globname=DVPq
GLOBAL[12].max_count=1000
GLOBAL[12].reftask=DsDVP
GLOBAL[12].in_groups=DVP
GLOBAL[12].in_xgroups=SIMULATOR
```

### 3.5.4 Файлы начальных условий

Старт системы после загрузки начинается с восстановления сохраненного состояния. Сохраненное состояние фиксируется в файлах начальных условий. Файл начальных условий имеет имя db/<project>_<globname>.t<ic>, где ic представляет собой номер начального условия. Например, условие 1 сохраняется в виде списка глобалов:

```
root@model-rb60-dock0_1:/home/osco# ls -al db/*.t01
-rw-r--r-- 1 root root  239 May 21 11:58 db/osco_AIs.t01
-rw-r--r-- 1 root root  253 May 21 11:58 db/osco_CNVs.t01
-rw-r--r-- 1 root root   39 May 21 11:58 db/osco_CPLs.t01
```

```
-rw-r--r-- 1 root root  902 May 21 11:58 db/osco_DIs.t01
-rw-r--r-- 1 root root   39 May 21 11:58 db/osco_HEAs.t01
-rw-r--r-- 1 root root 8759 May 21 11:58 db/osco_NODs.t01
-rw-r--r-- 1 root root   40 May 21 11:58 db/osco_NPMs.t01
-rw-r--r-- 1 root root 5058 May 21 11:58 db/osco_PIPs.t01
-rw-r--r-- 1 root root  193 May 21 11:58 db/osco_PMPs.t01
-rw-r--r-- 1 root root  201 May 21 11:58 db/osco_VLVs.t01
```

Использование начальных условий аналогично командам reset из отладчика DsISD. Чтобы стартовать с начального условия 1, надо набрать в отладчике reset 1, а затем run.

## 3.6 СОЗДАНИЕ И ЗАПУСК КОНТЕЙНЕРА

### 3.6.1 Образы отладочного режима

Для создания образов отладочного режима для каждого контейнера разрабатывается свой Dockerfile с каталогами привязки внешних пакетов и модулей.

```
#------------------------------------ SCADI ------------------------------
-------------
RUN mkdir -p /usr/local/scadi
RUN mkdir -p /usr/local/scadiweb
RUN mkdir -p /usr/local/osco
```

В частности, пакеты scadi, scadiweb и проектный каталог osco создаются путем монтирования соответствующих директорий. При необходимости происходит линковка каталогов плагинов.

```
RUN ln -sf /usr/local/scadi/Dpxch /home/osco/Dpxch
```

Устанавливается необходимое окружение перед стартом основного сервиса.

```
ENV LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/libtime64/lib
ENV
PATH=/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/scadi/sbinlur:/usr/local/scadiw
eb/sbinur
ENV SCADIWEB_PATH=/usr/local/scadiweb/sbinur
```

Основной сервис запускается включается в работу.

```
CMD  DsMM -cd /home/osco
```

### 3.6.2 Образы рабочего режима

Образы рабочего режима содержат установленное ПО в каталогах /usr/local/scadi, /usr/local/scadiweb.

Образы рабочего режима содержат установленное ПО плагинов в каталогах osco/Dp*. Версии плагинов соответствуют версиям типов в базе данных реального времени.

### 3.6.3 Контейнер отладочного режима

Создание контейнера из образа и запуск контейнера осуществляется программой WoMark. Создание контейнера реализуется через docker Api с учетом данных конфигурационного файла WoDocker.json (приложение 1).

Создание контейнера в виде командной строки будет выглядеть следующим образом.

```
docker stop model-rb60-grd1_0
docker rm model-rb60-grd1_0
docker                                  create                                  --mount
type=bind,source=/usr/local/scadi,target=/usr/local/scadi                       --mount
type=bind,source=/usr/local/scadiweb,target=/usr/local/scadiweb                 --mount
type=bind,source=/home/user/osco/mweb,target=/home/osco/web                     --mount
```

```
type=bind,source=/home/user/osco/Dsmodel,target=/home/osco/Dsmodel     --mount
type=bind,source=/home/user/osco/Dpmodel,target=/home/osco/Dpmodel     --mount
type=bind,source=/home/user/osco/Dpgc,target=/home/osco/Dpgc -p 12021:12020 --
tty --name model-rb60-grd1_0 model-rb60:v1.2d
docker start model-rb60-grd1_0
```

Доступ к запущенному контейнеру осуществляется командой docker exec –it model-<имя-контейнера> bash.

Контейнеры отладочного режима идентифицируются по последней букве d в имени (от слоба Debug англ отладка). В нашем примере model-rb60:v1.2d.

### 3.6.4 Контейнер рабочего режима

Так, как образы рабочего режима содержат установленное ПО в каталогах /usr/local/scadi, /usr/local/scadiweb, специальное монтирование указанных каталогов не требуется.

Так, как образы рабочего режима содержат установленное ПО плагинов в каталогах osco/Dp*, специальное монтирование каталога osco не требуется.

## 3.7 КОНФИГУРАЦИЯ СВЯЗИ СКОЯТ - ТРЕНАЖЕР

### 3.7.1 Сервер обмена на стороне СКОЯТ

СКОЯТ может соединяться с внешними моделями технологического процесса вплоть до полномасштабного тренажера по сетевому соединению. Такая объединенная система предполагает наличие двух контейнеров на вызываемые ресурс с тренажером. Первый контейнер будет содержать контейнер модели с системой верхнего уровня диспетчерского управления. Второй контейнер будет содержать экземпляр тренажера с системой управления задачами тренажера, множеством процессов расчетных задач, базу данных реального времени и ПО обмена на основе средств коммуникации зонной архитектуры с библиотекой доступа к данным тренажера

Программа DsXServ представляет собой сервер обмена. Сервер обмена осуществляет прием/передачу данных, внешних по отношению к системе СКАДИ-МОДЕЛЬ. Для использования сервера обмена необходима конфигурация в наборе задач.

```
TASK[1].name=DsXServSim
TASK[1].exe_path=DsXServ
TASK[1].command_line=-echo_cmd TRUE
TASK[1].watch_limit=60
TASK[1].restart_limit=5
TASK[1].update_ms=1000
TASK[1].run_at=STDSERVER
TASK[1].init_trace_level=0
```

Экземпляр программы DsXServ с именем DsXServSim запускается для обмена с внешним источником по сети.

Описание конфигурации производится для одного агента и одного сервера обмена. Со стороны сервера обмена нужно описание серверной группы.

```
XGROUP[0].name=SIMULATOR
XGROUP[0].task_name=DsXServSim
XGROUP[0].protocol=TCP
XGROUP[0].pl_protocol=DpxchUnixTCPChannel
XGROUP[0].cu_protocol=UNTCP
XGROUP[0].enable_ping=TRUE
XGROUP[0].tp_ping_timeout=1000000
XGROUP[0].with_crc_ref=TRUE
```

```
XGROUP[0].characteristics=UV.CC
XGROUP[0].area_id=101
XGROUP[0].pl_usercom=DpxchHndNImit
XGROUP[0].cu_usercom=HI
XGROUP[0].params_usercom=. -cycle_max 60 -update_ms 1000 -access B
XGROUP[0].tp_nconn_tmo=5000000
XGROUP[0].tp_lknown_sec=600
```

Также необходимо описать 2 точки обмена: сервер с ‘serv_host=.’ и агент обмена.

```
XCH[0].xgroup=SIMULATOR
XCH[0].serv_host=.
XCH[0].station_id=0
XCH[0].name=the_hostname
XCH[0].rtdb_port=10300

XCH[1].xgroup=SIMULATOR
XCH[1].station_id=1
XCH[1].params_usercom=. -data_file xch_simulator
XCH[1].name=my_hostname
XCH[1].rtdb_port=10300
```

Если агент обмена располагается на другой машине, необходимо прописывать имя хоста, с учетом доступности через DNS-сервер.

### 3.7.2 Агент обмена на стороне тренажера

Конфигурация подключения со стороны агента обмена описывается в файле Xch.cfg. Пример такого подключения выглядит следующим образом.

```
XGROUP.name=SIMULATOR
XGROUP.area_id=101
XGROUP.host_sid=2
XGROUP.protocol=TCP
XGROUP.with_crc_ref=TRUE
```

Каждый конфигурационный файл Xch.cfg (Xch_<group>.cfg) содержит описание связей одной коммуникационной группы. Поэтому в нем используется один экземпляр структуры XGROUP. Описанная группа охватывает оба узла обмена.

```
XCH[0].station_id=0
XCH[0].name=the_hostname
XCH[0].rtdb_port=10300
XCH[1].name=my_hostname
XCH[1].station_id=1
XCH[1].rtdb_port=10300
```

Группа XGROUP из Xch.cfg должна соответствовать группе XGROUP[0] из DsmConfig.cfg.

Конкретный протокол и конфигурация устанавливается по требованиям соответствующего тренажера.

## 3.8 КОНФИГУРАЦИЯ СУБД

### 3.8.1 Встроенная СУБД Sqlite

Инсталлированный экземпляр СУБД Sqlite представляет собой файл, расположенный в проектном каталоге. Имя файла задается в опции dsn командной строки.

```
user@osco:~/osco$ ls -al sdb
-rw-r--r-- 1 user scadi 8192 июн 17 11:14 sdb
```

Инсталляция и начальное заполнение необходимых таблиц осуществляется скриптом mkdb_users.sh с настройками –db sqlite –dsn sdb. Для начального ввода таблицы пользователей используйте команду следующего вида.

```
#!/bin/bash
/usr/local/scont/sbinur/WoDbsh -ef "users.sql" -q "select * from sc_users;" -
delim "|" -db_file "sdb"
```

### 3.8.2 Внешняя СУБД PostgreSQL

СУБД PostgreSQL запускается в отдельном контейнере и функционирует в течение времени работы СКОЯТ.

Инсталляция и начальное заполнение необходимых таблиц осуществляется скриптом mkdb_users.sh с настройками –db postgres.Для начального ввода таблицы пользователей используйте команду следующего вида.

```
#!/bin/bash
~/WoDbsh -ef "users.sql"  -q "select * from sc_users;" -delim "|" -db postgres
-db_dsn "postgres://postgres:qwerty@localhost:5432/users?sslmode=disable"
```

## 3.9 КОНФИГУРАЦИЯ NGINX

### 3.9.1 Проксирование соединения с WoAccess

Вход в систему СКОЯТ осуществляется через сервис WoAccess. Номер порта по умолчанию – 3000. Доступ к сервису осуществляется не напрямую, а через прокси nginx. Проксирование такого соединения требует записи в конфигурации nginx следующего вида.

```
location / {
      proxy_pass http://osco:3000;
}
```

Указанная запись переключает соединение с адреса / на локальный сервис WoAccess с портом № 3000.

### 3.9.2 Проксирование соединения с WoData

Сервис доступа к данным WoAccess. Номер порта по умолчанию – 3100. Кроме этого, данный сервис поддерживает протокол веб сокет. Проксирование такого соединения требует записи в конфигурации nginx следующего вида.

```
location /data/ {
      proxy_pass http://osco:3100/;

      proxy_http_version 1.1;
      proxy_set_header Host $host;
      proxy_set_header Upgrade $http_upgrade;
      proxy_set_header Connection "Upgrade";
      proxy_read_timeout 86400;
}
```

Согласно документации на nginx, предусмотрен особый режим работы, который позволяет установить туннель между клиентом и проксируемым сервером, если проксируемый сервер вернул ответ с кодом 101 (Switching Protocols), и клиент попросил сменить протокол с помощью заголовка “Upgrade” в запросе.

Указанная запись необходима для конфигурирования веб-сокет сервера, соединение переключается с адреса /data на локальные сервис WoData с портом № 3100.

### 3.9.3 Проксирование соединения с WoMark

Сервис контроля действий обучаемого WoMark. Номер порта по умолчанию – 3200. Кроме этого, данный сервис поддерживает протокол веб сокет. Проксирование такого соединения требует записи в конфигурации nginx следующего вида.

```
location /mark/ {
	proxy_pass http://osco:3200/;

	proxy_http_version 1.1;
	proxy_set_header Host $host;
	proxy_set_header Upgrade $http_upgrade;
	proxy_set_header Connection "Upgrade";
	proxy_read_timeout 86400;
}
```

Указанная запись необходима для конфигурирования веб-сокет сервера, соединение переключается с адреса /data на локальные сервис WoMark с портом № 3200.

### 3.9.4 Проксирование соединения с WsShmCntrl внутри контейнеров

Запуск контейнера осуществляется для данного номера ресурса resource_id, определенного для пользователя в базе данных. При создании и запуске контейнера ему присваивается определенный номер порта по правилу:

```
port := container_post + resource_id
```

Значение номера порта контейнера задается в конфигурационном файле в секции binding.

Необходимым условием корректной работы является проксирование относительного адреса cntrl/resource_id в адрес запущенного в контейнере сервиса WsShmCntrl на порте port. Проксирование такого соединения при установленном значении container_port=12020 требует записи в конфигурации nginx следующего вида.

```
	location /cntrl/1/ {
		proxy_pass http://osco:12021/;

		proxy_http_version 1.1;
		proxy_set_header Host $host;
		proxy_set_header Upgrade $http_upgrade;
		proxy_set_header Connection "Upgrade";
		proxy_read_timeout 86400;
	}
```

Запись параметра container_port предполагает следующий фрагмент конфигурационного файла.

```
	"binding": {
		"host_ip": "0.0.0.0",
		"host_port": 0,
		"container_protocol": "tcp",
		"container_port": 12020
	},
```

# 4 ПРОВЕРКА ПРОГРАММЫ

## 4.1 ПРОВЕРКА НАЛИЧИЯ ПО И ЗАГРУЗКИ

Установка проекта СКАДИ проверяется через наличие корневого каталога проекта project(вместо названия project подставляется конкретное имя разрабатываемого проекта, например, osco).

```
cd osco
```

В корневом каталоге проекта должны находиться каталоги и конфигурационные файлы, описанные в разделе 3.2.1.

Наличие инсталлированного ПО СКОЯТ проверяют путем наличия путей к бинарным исполняемым приложениям. Для Unix-систем можно использовать запрос пути.

```
which WoAccess
```

Загрузка программного обеспечения СКОЯТ осуществляется стартовым скриптом, запускающим сервисы WoAccess, WoMark, WoData.

```
cat up
for SERVICE in $SERVICES ; do
      echo -e "STATUS SERVICE=[$SERVICE] \c"
      sudo systemctl start $SERVICE || { echo "[ERROR !!!!]" ; break ; }
      echo "[DONE]"
done
```

Проверку наличия образов контейнеров и СУБД PostgreSQL осуществляется следующей командой.

```
user@osco:~/osco$ docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
model-ya1k   v1.0      b0b0e7c9f29f   32 minutes ago   285MB
model-tk70   v1.0      3742aed5c910   5 days ago       285MB
model-rn80   v1.6      74004d4f1fea   5 days ago       284MB
model-rb60   v1.1      949b7f790345   5 days ago       283MB
model-oisc   v6        28e45850c27d   5 days ago       306MB
model-kav1   v1.0      580830be3d00   6 days ago       283MB
model-ya1k   v1.0d     278eff1c1b91   7 days ago       257MB
model-kav1   v1.0d     7a8b1f579d86   7 days ago       257MB
postgres     latest    c0311cbf0603   2 weeks ago      438MB
```

Наличие образов контейнеров и их версии проверяем согласно списку в конфигурационном файле WoDocker.cfg. Наличие СУБД PostgreSQL проверяем по контейнеру postgres.

## 4.2 ПРОВЕРКА ЗАГРУЗКИ КОНТЕЙНЕРОВ

Консольная загрузка контейнера возможна для версии контейнера рабочего режима – release. Для загрузки контейнеров необходимо выбрать идентификатор ресурса и ресурсной группы. Для ресурса=1 и группы=0 загрузка контейнера rb60 осуществляется следующей командой.

```
docker create -p 12021:12020 --tty --name model-rb60-dock0_1 model-rb60:v1.1
docker start model-rb60-dock0_1
```

Консольная выгрузка контейнера реализуется следующими командами.

```
docker stop model-rb60-dock0_1
docker rm model-rb60-dock0_1
```

Определение ресурса=1 и группы=0 производится из записей базы данных для определенного пользователя. При наличии пользователя с Яндекс ID = sidorov см ниже.

```
create table sc_users(register_id int, name text, resource_id int, resource_grp
int, category int, activity text);
insert into sc_users values(1, 'sidorov', 1, 0, 15, 'basic');
```

При работе системы загрузка контейнеров производится из меню Режимы/Лабораторные работы. Сервис WoData работающей системы загружает контейнер, согласно значениям resource_id пользователя.

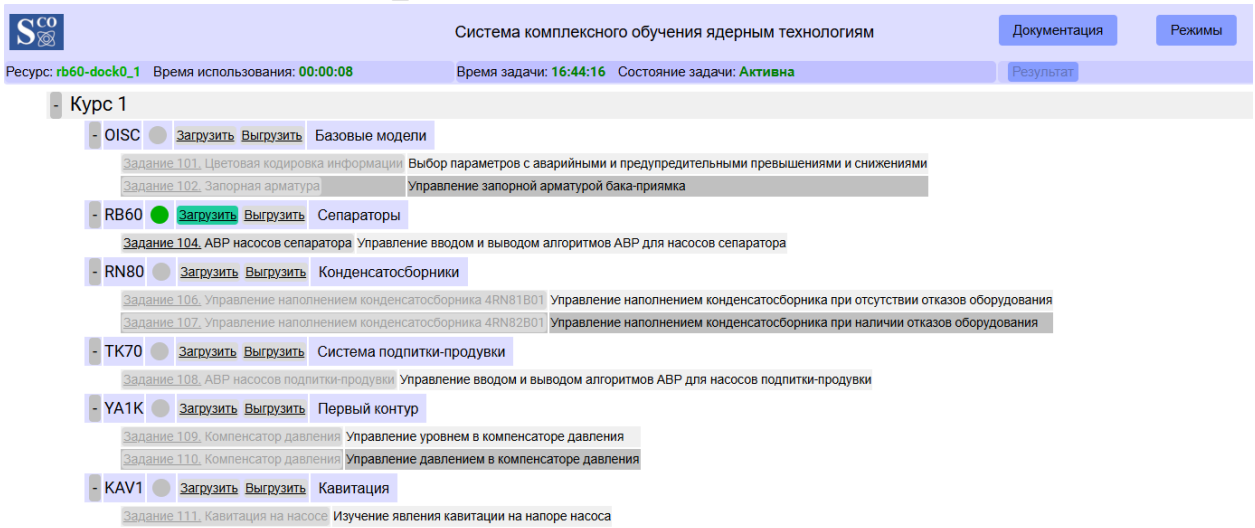


Рисунок 7 – Загрузка контейнера из браузера

При обновлении экрана проверяется наличие загруженного контейнера. При необходимости контейнер выгружается и загружается новый.

Для консольного доступа к загруженному контейнеру используется следующая команда.

```
docker exec –it model-rb60-dock0_1 bash
```

Команда обеспечивает консольный доступ внутрь контейнера.

## 4.3 ПРОВЕРКА ЗАГРУЗКИ ГЛОБАЛОВ ОБЩЕЙ ПАМЯТИ

Наличие и состав глобалов общей памяти проверяется командой.

```
DsView –globtype .
```

Команда DsView –globtype . выводит на консоль список имен, типов и заголовков объектов общей памяти. Пример консольного вывода приведен ниже.

```
root@model-rb60-dock0_1:/home/osco# DsView -globtype .
G 0 Globals n_total_objs= 512, n_used_objs= 42, sz_used_mem= 45056, usr_data= 0
E 1 Enums n_total_objs= 120, n_used_objs= 119, sz_used_mem= 51360, usr_data= 500
T 2 Types n_total_objs= 80, n_used_objs= 57, sz_used_mem= 11520, usr_data= 1
A 3 Attrs n_total_objs= 900, n_used_objs= 776, sz_used_mem= 187200, usr_data= 10
S 4 Syslogs n_total_objs= 3000, n_used_objs= 6, sz_used_mem= 648000, usr_data= 60
s 5 Consts n_total_objs= 404, n_used_objs= 194, sz_used_mem= 25856, usr_data= 17
s 6 Characteristics n_total_objs= 32, n_used_objs= 12, sz_used_mem= 5248, usr_data= 18
s 7 Usergroups n_total_objs= 16, n_used_objs= 5, sz_used_mem= 2688, usr_data= 19
s 8 Tasks n_total_objs= 6, n_used_objs= 6, sz_used_mem= 3168, usr_data= 20
s 9 HGroups n_total_objs= 2, n_used_objs= 2, sz_used_mem= 1184, usr_data= 21
s 10 Hosts n_total_objs= 2, n_used_objs= 2, sz_used_mem= 608, usr_data= 22
s 11 XGroups n_total_objs= 1, n_used_objs= 1, sz_used_mem= 1064, usr_data= 23
s 12 Xchs n_total_objs= 2, n_used_objs= 2, sz_used_mem= 1008, usr_data= 24
s 13 CSets n_total_objs= 5, n_used_objs= 5, sz_used_mem= 11040, usr_data= 27
s 14 Cells n_total_objs= 50, n_used_objs= 50, sz_used_mem= 54600, usr_data= 26
s 15 Reports n_total_objs= 5, n_used_objs= 5, sz_used_mem= 3280, usr_data= 29
s 16 Repdatas n_total_objs= 5, n_used_objs= 5, sz_used_mem= 1680, usr_data= 28
s 17 WSets n_total_objs= 5, n_used_objs= 5, sz_used_mem= 6920, usr_data= 31
s 18 WDatas n_total_objs= 5, n_used_objs= 5, sz_used_mem= 16260, usr_data= 30
s 19 VFiles n_total_objs= 10, n_used_objs= 10, sz_used_mem= 1720, usr_data= 32
s 20 FVars n_total_objs= 100, n_used_objs= 100, sz_used_mem= 32400, usr_data= 33
s 21 Alarms n_total_objs= 60, n_used_objs= 60, sz_used_mem= 6240, usr_data= 34
s 22 TFiles n_total_objs= 1, n_used_objs= 1, sz_used_mem= 172, usr_data= 38
s 23 TVars n_total_objs= 1, n_used_objs= 1, sz_used_mem= 324, usr_data= 39
s 24 TAlarms n_total_objs= 4, n_used_objs= 4, sz_used_mem= 1024, usr_data= 35
s 25 Archs n_total_objs= 1, n_used_objs= 1, sz_used_mem= 168, usr_data= 36
s 26 Jobs n_total_objs= 1, n_used_objs= 1, sz_used_mem= 728, usr_data= 37
s 27 Gstats n_total_objs= 512, n_used_objs= 42, sz_used_mem= 86016, usr_data= 25
n 28 AIs n_total_objs= 9, n_used_objs= 9, sz_used_mem= 3312, usr_data= 40
n 29 DIs n_total_objs= 83, n_used_objs= 83, sz_used_mem= 27888, usr_data= 41
q 30 GCq n_total_objs= 2000, n_used_objs= 146, sz_used_mem= 96000, usr_data= 146
n 31 VLVs n_total_objs= 100, n_used_objs= 11, sz_used_mem= 28000, usr_data= 46
n 32 CNVs n_total_objs= 100, n_used_objs= 3, sz_used_mem= 43200, usr_data= 47
n 33 PMPs n_total_objs= 100, n_used_objs= 2, sz_used_mem= 39200, usr_data= 48
n 34 NODs n_total_objs= 50, n_used_objs= 13, sz_used_mem= 61200, usr_data= 53
n 35 CPLs n_total_objs= 100, n_used_objs= 2, sz_used_mem= 20000, usr_data= 55
n 36 PIPs n_total_objs= 100, n_used_objs= 19, sz_used_mem= 72000, usr_data= 52
q 37 Modelq n_total_objs= 2000, n_used_objs= 374, sz_used_mem= 96000, usr_data= 374
n 38 NPMs n_total_objs= 1, n_used_objs= 1, sz_used_mem= 160, usr_data= 56
n 39 HEAs n_total_objs= 20, n_used_objs= 1, sz_used_mem= 4160, usr_data= 54
q 40 DVPq n_total_objs= 2000, n_used_objs= 0, sz_used_mem= 96000, usr_data= 0
q 41 Controlq n_total_objs= 1000, n_used_objs= 0, sz_used_mem= 48000, usr_data= 0
```

Для каждого глобала памяти выводится тип глобала, номер, имя, максимальное число объектов n_total_objs, текущее число объектов или номер n_used_objs, размер используемой памяти sz_used_mem, поле переменного значения usr_data.

Изменение состояния глобалов лучше всего контролируется по изменению значения последнего параметра usr_data при периодических вызовах.

## 4.4 КОНТРОЛЬ СТАТУСА ЗАДАЧ

Наличие и состав выполняющихся задач проверяется командой.

```
DsView –globname Tasks
```

Команда DsView –globname Tasks выводит на консоль список задач текущей загрузки, рисунок 8.

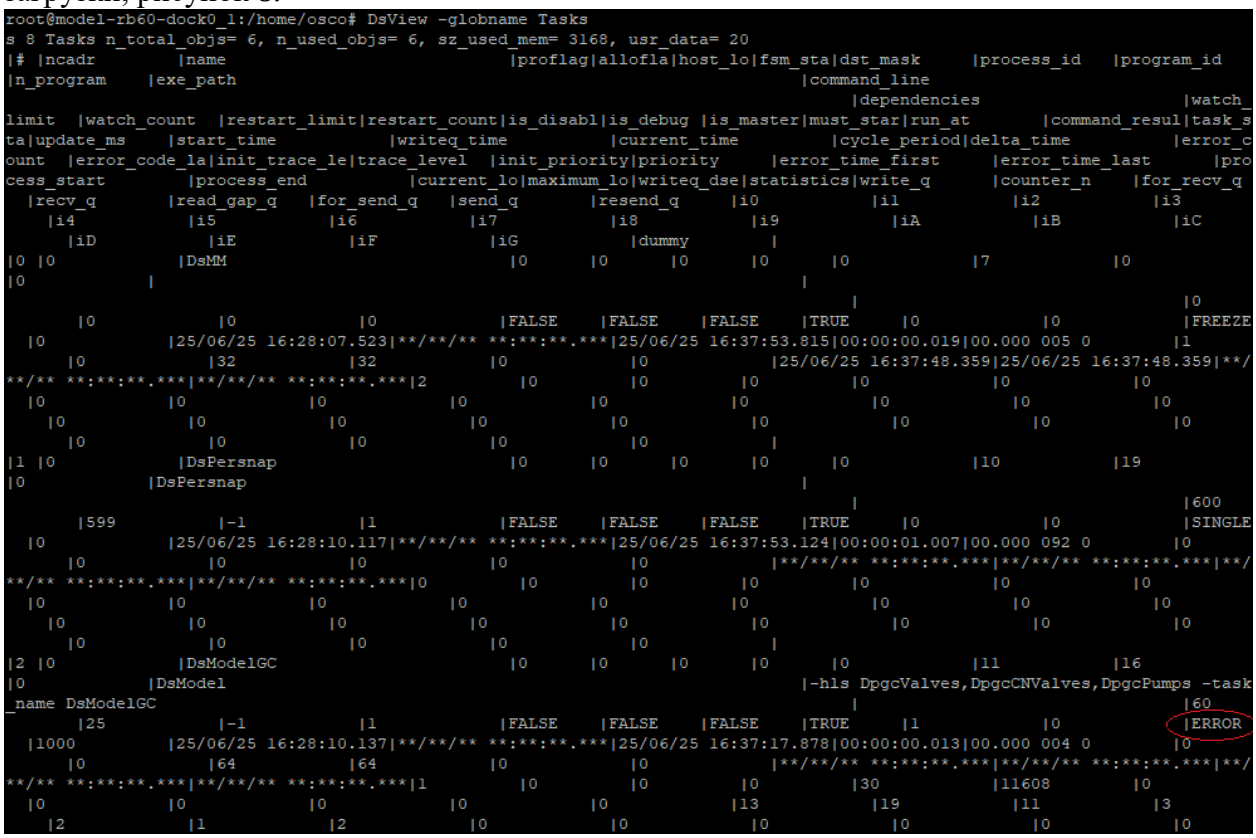


Рисунок 8 – Фрагмент списка задач

Состояние задачи контролирует параметр task_status. На приведенном рисунке отмечен статус задачи DsModelGC.task_status как ERROR. Это означает, что задача не стартовала корректно. Корректный рестарт оканчивается переходом в состояние с SINGLE|FREEZE. Отметим, что на приведенном рисунке задача DsMM имеет статус FREEZE. Такой статус означает, что выполнение задач лабораторной работы завершено и расчеты остановлены.

Печать системных сообщений Syslogs выдает все системные события. Их можно отфильтровать по коду важности (0-нет, 1-список, 2-доп, 3-инфо, 4-предупреждение, 5-ошибка, 6-фатально), начиная с 4 – оставив только ошибки и фатальные ошибки.

```
root@model-rb60-dock0_1:/home/osco# DsView -globname Syslogs -fld 4
S 4 Syslogs n_total_objs= 3000, n_used_objs= 6, sz_used_mem= 648000, usr_data=
68
25/06/25 16:37:48.359 ERROR 62# DsMM 1@0 DsrmMMServ.WatchTasks Error in TASK[2]
name=DsModel oldstat=FREEZE
```

Для приведенного примера задача DsModel не работает, ожидает рестарта системой.

### 4.5 КОНТРОЛЬ СОСТОЯНИЯ СЕТИ

Для проерки доступа по сети сначала определяется номер порта запущенного контейнера. Это можно сделать командой docker ps.

```
user@osco:~/osco$ docker ps --all
CONTAINER ID   IMAGE            COMMAND                  CREATED          STATUS
PORTS                      NAMES
154eee2f041a   model-rb60:v1.1   "/bin/sh -c 'DsMM -c…"   20 minutes ago   Up
20 minutes   0.0.0.0:12021->12020/tcp   model-rb60-dock0_1
```

Контроль наличия веб-сервера по данному порту (в нашем случае 12021) проверяется командой curl.

```
user@osco:~/osco$ curl localhost:12021
<pre>
<a href="Dpgc/">Dpgc/</a>
<a href="Dpmodel/">Dpmodel/</a>
<a href="Dpxch/">Dpxch/</a>
<a href="DsmConfig.cfg">DsmConfig.cfg</a>
<a href="DsmConfig.tfg">DsmConfig.tfg</a>
<a href="Dsm_rb60-image.lic">Dsm_rb60-image.lic</a>
<a href="Dsmodel/">Dsmodel/</a>
<a href="Host/">Host/</a>
<a href="Oml.cfg">Oml.cfg</a>
<a href="Wss_rb60-image.lic">Wss_rb60-image.lic</a>
<a href="db/">db/</a>
<a href="web/">web/</a>
</pre>
```

При корректной работе на выходе получается информация о данных проектного каталога (в приведенном примере – директории Dpgc, Dpmodel, файлы DsmConfig.cfg, DsmConfig.tfg и т.д.).

При некорректной работе ожидается сообщение об ошибке.

## 4.6 КОНТРОЛЬ ВЗАИМОДЕЙСТВИЯ С СЕРВИСОМ WOMARK

Контроль механизма взаимодействие с контейнером возможен при запущенном контейнере. Для проверки взаимодействия используется тестовая страница http://<project>/mark/imark.html.

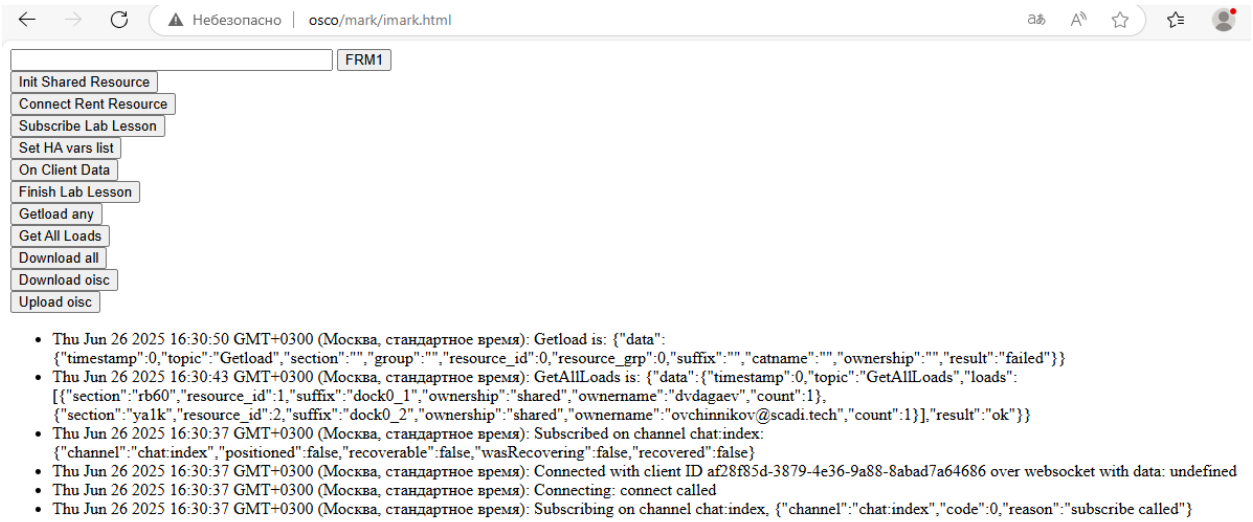


Рисунок 9 –Список и состояния коммуникационных узлов

Используя тестовую страницу, можно осуществлять загрузку, выгрузку и контроль загруженных контейнеров. Кроме этого, на странице реализован механизм запуска и управления подпиской при проведении лабораторной работы командами “Subscribe Lab Lesson”, “Finit Lab Lesson”.

# 5 ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

## 5.1 РЕСТАРТУЕМЫЕ СЕРВИСЫ СКОЯТ

Для сервисов WoAccess, WoMark, WoData реализованы механизмы автоматического старта/останова. Для каждого сервиса необходимо, как для woaccess, выполнить следующие действия.

Включить службу woaccess.service в автозагрузку ОС от имени пользователя root следующей командой:

```
sudo systemctl enable woaccess
```

Запустить службу woaccess.service от имени пользователя root следующей командой:

```
sudo systemctl start woaccess.service
```

# 6 СООБЩЕНИЯ СИСТЕМНОМУ ОПЕРАТОРУ

## 6.1 ЛОГ СИСТЕМНЫХ СООБЩЕНИЙ

Для записи системных сообщений для каждого из процессов СКАДИ-МОДЕЛЬ сделан специальный лог – глобал S. Лог системных сообщений представляет собой кольцевой буфер, в котором сохраняются сообщения даже после завершения работы модулей. Во время работы и по окончании лог сбрасывается в файл db/<project>_Syslogs.fla. При возобновлении работы общая память восстанавливается из файла с сохранением ранее сделанных сообщений. Удаление файла лога приведет к созданию нового, очищенного от предыдущих сообщений лога при очередной загрузке СКАДИ-МОДЕЛЬ.

Просмотр лога системных сообщений осуществляется командой DsView –global 4 (либо DsView –globname Syslogs), вывод на консоль приведен на рисунке 10.

```
root@model-yalk-dock0_2:/home/osco# DsView -globname Syslogs -limit 9
S 4 Syslogs n_total_objs= 3000, n_used_objs= 6, sz_used_mem= 648000, usr_data= 78
26/06/25 15:33:09.550 INFO 70# DsPersnap 1@0 DsrmPersnapInT.ReloadT CPLs global changes:0 file:db/osco_CPLs.t01 res=0()
26/06/25 15:33:09.553 INFO 71# DsPersnap 1@0 DsrmPersnapInT.ReloadT PIPs global changes:360 file:db/osco_PIPs.t01 res=0()
26/06/25 15:33:09.553 INFO 72# DsPersnap 1@0 DsrmPersnapInT.ReloadT NPMs global changes:0 file:db/osco_NPMs.t01 res=0()
26/06/25 15:33:09.554 INFO 73# DsPersnap 1@0 DsrmPersnapInT.ReloadT HEAs global changes:12 file:db/osco_HEAs.t01 res=0()
26/06/25 15:33:09.554 INFO 74# DsPersnap 1@0 DsrmPersnapInT.ReloadT BLKs global changes:18 file:db/osco_BLKs.t01 res=0()
26/06/25 15:33:09.554 INFO 75# DsPersnap 1@0 DsrmPersnapInT.ReloadT AVRs global changes:0 file:db/osco_AVRs.t01 res=0()
26/06/25 15:33:09.555 INFO 76# DsPersnap 1@0 DsrmPersnapInT.ReloadT TENs global changes:12 file:db/osco_TENs.t01 res=0()
26/06/25 15:33:09.555 INFO 77# DsPersnap 1@0 DsrmPersnapInT.SaveAll ResetT cmd:-1 reset_no:1
26/06/25 15:53:09.518 INFO 78# DsPersnap 1@0 DsrmPersnapInT.SaveAll ResetT cmd:-1000 reset_no:1000
root@model-yalk-dock0_2:/home/osco#
```

Рисунок 10 – Фрагмент лога системных сообщений глобала Syslogs

Сообщения системного лога представляют собой записи определенного формата Syslog. Например, можно разобрать по полям следующее сообщение.

```
24/06/25 13:47:56.758 WARNING 566# DsModelGC 1@0 DsmodelHandlers.CheckInitObj
init config type: VLV, obj:V11, rc:-1
```

Поле timestamp содержит время, в строковом представлении равное «`24/06/25 13:47:56.758`», поле важности severity – код WARNING, поле номера сообщения msg_id – номер 566, поле имени процесса module – имя программы DsModelGC, строка 1@0 означает user_id = 0 (сервер), host_ind = 0 (нулевой хост), поле имени действия operation содержит «`DsmodelHandlers.CheckInitObj`», поле сообщения message – шаблон `init config type: $, obj:$, rc:$`, а поле params – параметры «VLV V11 -1».

Все поля сообщений глобала S приведены в таблице 3.

| Поле | Наименование | Значения | Примечание |
|---|---|---|---|
| timestamp | Метка времени | Время | |
| severity | Код важности | NONE, LIST, AUX, INFO, WARNING, ERROR, FATAL | 0-нет, 1-список, 2-доп, 3-инфо, 4-предупреждение, 5-ошибка, 6-фатально данных |
| user_id | Идентификатор пользователя | 0 … | |
| host_ind | Номер хоста | 0 … | В таблице Hosts |
| msg_id | Номер сообщения | 0 … | |
| operation | Имя действия | <Имя> | |
| module | Имя процесса | <Имя> | В таблице Tasks |
| message | Шаблон сообщения | <Строка> | |
| params | Параметры к шаблону | <Строка> | |

Таблица 3. Поля сообщений глобала S

Глобал S системных сообщений разбит на непересекающиеся области из фиксированного количества сообщений для каждого процесса. Количество сообщений задается параметром DsmConfig.cfg:ldr_task_msgs=400 и не меняется после конфигурирования. Для каждого процесса сформирован кольцевой буфер из 400 сообщений, при заполнении которого перезаписываются наиболее сообщения данного процесса. Параметр user_data в заголовке глобала Syslogs определяет идентификатор следующего сообщения.

# 7 ИНСТРУКЦИЯ ПО УСТАНОВКЕ

Импортировать виртуальную машину osco-net.ova
Запустить виртуальную машину osco-net
Загрузить образы Docker контейнеров заданий
Выполнив следующие действия :
Скопировать образы контейнеров на машину osco-net с помощью scp
scp model-kav1 model-ya1k model-tk70 model-rn80 model-rb60 model-oisc vagrant@osco-net:~/dockerimage
Загрузить docker контейниры заданий
Выполнить комманду:
ssh vagrant@osco-net
cd ~:~/dockerimage
sudo docker load –input model-kav1.tar
sudo docker load –input model-ya1k.tar
sudo docker load –input model-tk70.tar
sudo docker load –input model-rn80.tar
sudo docker load –input model-rb60.tar
sudo docker load –input model-oisc.tar

# ПРИЛОЖЕНИЯ

## ПРИЛОЖЕНИЕ 1 – ОПИСАНИЕ КОНФИГУРАЦИИ СКОЯТ WODOCKER.JSON

Конфигурация представляет собой файл вида `Wo<conftype>.json`. Данный файл используется сервисами СКОЯТ WoMark и WoData. Конфигурация состоит из одиночных опций, структур опции и массивов опций. Файл должен соответствовать JSON-формату. Пример одиночной опции (имя: значение).

```
"wmcommand": "Estimate",
```

Структуры состоят из именованных полей и их значений. Пример структур опций.

```
"docker": {
	"project": "model",
	"cfrom": "dockerclient"
}
```

Массивы опций состоят из одиночных опций и/или структур. Пример массива опций.

```
"containers": [
	{
		"image": "model-oisc",
		"version": "v6d"
	},
	{
		"image": "model-rn80",
		"version": "v1.5d"
	}
],
```

Одиночные опции приведены в таблице 1-1.

| Опция | Тип | Описание | Примечание |
|---|---|---|---|
| wmshell | Str | Оболочка питона выполнения скриптов | /usr/bin/python3 |
| wmscript | Str | Имя скрипта управления оцениванием | WoEval |
| wmcommand | Str | Имя скрипта оценки задания | Estimate |
| rftype | Str | Тип управления ресурсами | docker\|shell |
| n_resources | I4 | Количество ресурсов контейнеров | 100 |

Таблица 1-1 – Список одиночных опций

Тип управления ресурсами rftype определяет механизм запуска ресурсами. Предполагается использование "rftype": "docker". В таком случае необходимо дополнительно описать данные структуры docker.

Структуры и массивы приведены в таблице 1-2.

| Опция | Родитель | Тип | Описание | Примечание |
|---|---|---|---|---|
| image | containers | Str | Имя контейнера | |
| version | containers | Str | Версия контейнера | v1.0d |
| host_ip | binding | Str | Привязка к хосту | 0.0.0.0 |
| host_port | binding | I4 | Привязка к порту | 0 |
| container_protocol | binding | Str | Привязка к протоколу | tcp |

| container_port | binding | I4 | Привязка к порту контейнера | 12020 |
|---|---|---|---|---|
| mounts | type | Str | Тип монтирования | bind |
| mounts | source | Str | Каталог в хост-системе | /home/user/osco/mweb |
| mounts | target | Str | Каталог в контейнере | /home/osco/web |
| project | docker | Str | Имя проекта с контейнерами | |
| cfrom | docker | Str | Механизм запуска контейнеров | dockerclient |
| containers | docker | Массив | Список используемых контейнеров | |
| binding | docker | Структура | Используемая привязка портов | |
| mounts | docker | Массив | Список используемых монтировок | |

Таблица 1-2 – Список структур и массивов из опций

Ниже приведен типовой конфигурационный файл WoDocker.json.

```
{
  "wmshell": "/usr/bin/python3",
  "wmscript": "WoEval",
  "wmcommand": "Estimate",
  "rftype": "docker",
  "docker": {
    "project": "model",
    "docker_from.0": "image",
    "cfrom": "dockerclient",
    "images":   "model-oisc:v6d   model-rn80:v1.5d   model-rb60:v1.1d   model-
tk70:v1.0d model-ya1k:v1.0d",
    "containers": [
      {
        "image": "model-oisc",
        "version": "v6d"
      },
      {
        "image": "model-rn80",
        "version": "v1.5d"
      },
      {
        "image": "model-rb60",
        "version": "v1.1d"
      },
      {
        "image": "model-tk70",
        "version": "v1.0d"
      },
      {
        "image": "model-ya1k",
        "version": "v1.0d"
      }
    ],
    "image_origin.1": "model-oisc:v4",
    "image_origin.0": "model-oisc",
    "container_origin.1": "/model-oisc-dock1",
    "binding": {
      "host_ip": "0.0.0.0",
      "host_port": 0,
```

```
        "container_protocol": "tcp",
        "container_port": 12020
      },
      "mounts": [
        {
          "type": "bind",
          "source": "/usr/local/scadi",
          "target": "/usr/local/scadi"
        },
        {
          "type": "bind",
          "source": "/usr/local/scadiweb",
          "target": "/usr/local/scadiweb"
        },
        {
          "type": "bind",
          "source": "/home/user/osco/mweb",
          "target": "/home/osco/web"
        },
        {
          "type": "bind",
          "source": "/home/user/osco/Dsmodel",
          "target": "/home/osco/Dsmodel"
        },
        {
          "type": "bind",
          "source": "/home/user/osco/Dpmodel",
          "target": "/home/osco/Dpmodel"
        },
        {
          "type": "bind",
          "source": "/home/user/osco/Dpgc",
          "target": "/home/osco/Dpgc"
        }
      ]
    },
    "n_resources": 100
  }
```

## ПРИЛОЖЕНИЕ 2 – СОСТАВ ЗАДАЧ И УТИЛИТ СКАДИ-МОДЕЛЬ

В таблице 2-1 приведен состав задач СКАДИ-МОДЕЛЬ, запускаемые менеджером моделей. Задачи, которые могут встраиваться в менеджер моделей, отмечены в правом столбце.

| Имя | Описание | Назначение | Встраивание |
|---|---|---|---|
| DsModel | Оболочка расчетной модели | Загрузка и выполнение плагинов прикладных расчетных моделей | да |
| DsSync | Синхронизация данных | обновление табличных данных узлов по очередям сообщений | нет |
| DsDVP | Очереди синхронизации | прием/передачу данных синхронизации узлов по сетевому интерфейсу | да |
| DsPersnap | Сохранение и восстановление данных | сброс пользовательских данных в номерные файлы; восстановление из номерных файлов | да |
| DsGroup | Групповое отображение | подготовка данных для группового отображения | да |
| DsAlarm | Сигнализации | подсистема регистрации и управления сигнализациями | нет |
| DsArchServ | Архивация | ПО для архивации текущих данных | нет |
| DsRepServ | Протоколирование | ПО для доступа к архивным и текущим данным | нет |
| DsXServ | Сервер обмена | сервер обмена данными | нет |
| WsShmCntrl | Веб-сокет сервер | Реализация веб-версии системы верхнего уровня | нет |

Таблица 2-1 – Состав запускаемых задач СКАДИ-МОДЕЛЬ

В таблице 2-2 приведен состав утилит сервиса и отладки, запускаемых как консольные приложения.

| Имя | Описание | Необходимое условие | Назначение |
|---|---|---|---|
| DsView | просмотр общей памяти | Общая память DsMM | Просмотр глобалов или локальных сегментов разделяемой памяти |
| DsISD | интерактивный отладчик значений | Общая память DsMM | Чтение/запись значений глобалов разделяемой памяти |
| DsQSD | интерактивный отладчик очередей | Очереди сообщений DsMM | Чтение данных из очередей глобалов или локальных сегментов разделяемой памяти |
| DsAccess | доступ к функциям управления | Общая память DsMM | Чтение, запись, управление, параметризация системных и пользовательских глобалов |
| DsArMan | менеджер архивных файлов | Архивные каталоги СКАДИ-МОДЕЛЬ | Администрирование и просмотр архивов |

| | | | |
|---|---|---|---|
| DsTrs | консольная печать протоколов | Общая память DsMM и таблицы групп и протоколов | Запуск генерации протоколов в виде текстовых файлов по запросу |

Таблица 2-2 – Состав утилит сервиса и отладки

Утилиты сервиса и отладки запускаются как консольные приложения. Они осуществляют вывод текстовых данных на консоль или перенаправляются в текстовый файл при использовании соответствующих команд оболочки операционной системы.

## ПРИЛОЖЕНИЕ 3 – ДИНАМИЧЕСКИ ЗАГРУЖАЕМЫЕ МОДУЛИ

Список динамически загружаемых модулей СКАДИ-МОДЕЛЬ приведен в таблице 3-1. Фабричные модули определяют интерфейс, который запрашивается у плагина с проверкой соответствия версии.

| Фабрика | Базовый тип | Реализация | Описание | Методы |
|---|---|---|---|---|
| DshmOVHandlersSingle: ObjectValueHandlers 112,1.0 | DsmodelHandlers.MOvh | DpgcValves | Управление запорной арматурой | InitObj(): INTEGER; UpdateOne(); |
| | | DpgcCNValves | Управление регулирующим клапаном | InitObj(): INTEGER; UpdateOne(); |
| | | DpgcPumps | Управление насосным агрегатом | InitObj(): INTEGER; UpdateOne(); |
| | | DpgcAvrs | Автоматическое включение резерва | InitObj(): INTEGER; UpdateOne(); |
| | | DpgcBlks | Технологические защиты и блокировки | InitObj(): INTEGER; UpdateOne(); |
| DshmOVHandlers: ObjectValueHandlersMulti 113,1.0 | DsmodelHandlers.Model | DpmodelMain | Расчетная модель теплогидравлики | InitAll(VAR res: INTEGER); UpdateAll(); |
| DshmFactory: InTask 114,10 | DshmFactory.InTask | DsrmModelInT | Встраиваемые задачи | SetStringOpt(); Init(); Update(); Finit(); |

Таблица 3-1 Загружаемые модули СКАДИ-МОДЕЛЬ

## ПРИЛОЖЕНИЕ 4 – СПИСОК ПЕРЕМЕННЫХ ОКРУЖЕНИЯ

Список переменных окружения приведен в таблице 4-1.

| Переменная | Описание | Значение |
|---|---|---|
| SESSION_SECRET | Ключ сессии при сохранении имен cookie | Уникально для каждой серверной инсталляции |
| YANDEX_CLIENT_ID | Идентификатор приложения Яндекс | Получается при регистрации в Яндексе |
| YANDEX_CLIENT_SECRET | Секретный ключ подписи токена | Получается при регистрации в Яндексе |
| REDIRECT_URL | Адрес перенаправления после авторизации | Фиксируется при регистрации в Яндексе |
| LOCAL_REDIRECT_URL | Адрес перехода после WoAccess | http://osco/data/indexw.html |
| AUTH_REDIRECT | Адрес перехода при необходимости авторизации | http:// osco/account |
| DOCKER_API_VERSION | Версия API связи с docker | 1.41 |
| PATH | Пути к сервисам СКОЯТ | /usr/local/scont/sbinur |

Таблица 4-1 – Список переменных окружения

## ПРИЛОЖЕНИЕ 5 – ОПЦИИ КОМАНДНОЙ СТРОКИ

Опции командной строки СКАДИВЕБ доступны вызовом приложения с флагом –help, рисунок 5-1.

```
C:\DVDagaev\lcc1>WsShm --help
Usage of WsShm:
  -ast_invalid string
        Symbols for invalid substitution
  -cset
        Use CSet booking for queries (default true)
  -hbinterval duration
        Time interval of heartbeat (default 1s)
  -interval duration
        Time interval of update (default 1s)
  -ndigits int
        Number of digits for analog conversion (default 4)
  -noempty
        Send only not empty messages (default true)
  -port int
        Port to bind app to (default 8010)
  -projpath string
        Project path to be connected via shm
  -sendall
        Send all values in cycle
  -task_name string
        Project path to be connected via shm (default "WsShm")
  -trace_level int
        Level of printing trace info
```

Рисунок 5-1 – WsShm с опциями командной строки

СКАДИВЕБ использует следующие опции командной строки, таблица 5-1.

| Опция | Описание | Возможные значения |
|---|---|---|
| -ast_invalid | Символ недостоверного значения | *** ??? |
| -cset | Возможность групповых запросов | true(по умолчанию) false |
| -hbinterval | Интервал сердцебиения | 1s(по умолчанию), 100ms, 300ms, … |
| -interval | Интервал опроса | 1s(по умолчанию), 100ms, 300ms, … |
| -ndigits | Число значащих цифр | 4(по умолчанию) 1-10 |
| -noempty | Посылка только непустых сообщений | true(по умолчанию) false |
| -port | Номер порта | 8010(по умолчанию) число |
| -projpath | Путь к проекту | Для WsShmRd |
| -sendall | Посылка всех данных, а не только измененных | false(по умолчанию) true |
| -task_name | Имя задачи | Задается DsRM при запуске |
| -trace_level | Уровень трассировки | Число |

Таблица 5-1 – Перечень опций командной строки

# ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

**АЭС** - Атомная Электростанция

**КСА** - Комплекс средств автоматизации

**ОС** - Операционная система

**ПО** - Программное обеспечение

**СВБУ** - Система верхнего блочного уровня

**СКАДИ** - Средства комплексной автоматизации и диагностики

**ЧМИ** - Человеко-машинный интерфейс

**ICMP** - Internet Control Message Protocol

**TCP** - Transmission Control Protocol

**UDP** - User Datagram Protocol

## ЛИСТ ИЗМЕНЕНИЙ

| Изм. | Номера листов | | | | Всего листов в изме-ненном документе | Номер документа, утверждаю-щего изменение | Подп. | Дата |
|---|---|---|---|---|---|---|---|---|
| | измененных | замененных | новых | аннулированных | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |