ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ

«СКАДИ»

(ООО «СКАДИ»)

СОГЛАСОВАНО **УТВЕРЖДАЮ**

СИСТЕМА КОМПЛЕКСНОГО ОБУЧЕНИЯ ЯДЕРНЫМ ТЕХНОЛОГИЯМ

Описание программы

34185873.425510.005.ОП.80.М

(На - листах)

2025

РАЗРАБОТАЛ

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ 5
1.1 Обозначение и Наименование программ 5
1.2 Программное обеспечение, необходимое для функционирования программы 5
1.3 Языки программирования, на которых написана программа 6
2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ 7
2.1 Классы решаемых задач 7
2.2 Назначение программы и основные функции 7
2.3 Сведения о функциональных ограничениях на применение 7
3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ 8
3.1 Основный метод и механизмы взаимодействия 8
3.2 Структура программы 10
3.2.1 Состав серверных приложений 10
3.2.2 Пути доступа к системам 12
3.2.2 Пользовательские разделы систем 12
3.2.3 Сервисы программного обеспечения 13
3.3 Система регистрации и управления доступом 13
3.3.1 Регистрация приложения в облачном сервисе 13
3.3.2 Алгоритм авторизации СКОЯТ 13
3.3.3 Использование SCADI ID 15
3.4 Система верхнего уровня диспетчерского управления 15
3.4.1 Основные функции системы 15
3.4.2 Подключение в качестве контейнера 16
3.4.3 Формат представления видеокадров 18
3.5 Система управления моделями технологического процесса 20
3.5.1 Общие характеристики 20
3.5.2 Структура программных модулей 20
3.5.3 Идентификация объектов в реляционных таблицах 24
3.5.4 Менеджер моделей 27
3.5.5 Сохранение и восстановление состояния 29
3.5.6 Оболочка расчетных моделей 33
3.5.7 Обновление табличных данных узлов по очередям 35
3.5.8 Групповое отображение данных 37
3.5.9 Регистрация и управление сигнализациями 38
3.5.10 Сервер обмена 38
3.6 Средства связи с внешним тренажером с моделями технологического процесса 39
3.6.1 Структурная схема соединения СКОЯТ - тренажер 39
3.6.2 Основные функции ПО связи 40
3.7 Система контроля действий обучаемого 41
3.7.1 Структура системы контроля действий обучаемого 41
3.7.2 Подсистема архивирования действий обучаемого и данных технологического процесса 42
3.7.3 Подсистема оценивания действий обучаемого на основании данных технологического процесса 43
3.7.4 Подсистема управления и мониторинга предоставления ресурсов 44
3.8 Система учета использованных ресурсов 46

3.8.1 Фиксация индивидуальных и групповых ресурсов 46
3.8.2 Доступ к данным об использовании ресурсов 48
3.9 Система представления обучающих программ 49
3.9.1 Структура фреймового представления графической системы 49
3.9.2 Объектная структура системы представления 50
3.9.3 Скрипты клиентской части 51
4 ИСПОЛЬЗУЕМЫЕ СИСТЕМНЫЕ И ТЕХНИЧЕСКИЕ СРЕДСТВА 53
4.1 Минимальные технические средства развертывания 53
4.2 Системные средства управления ресурсами 53
4.3 Система управления базой данных 53
4.4 Развертывание в облачных сервисах 53
5 ВЫЗОВ И ЗАГРУЗКА 54
5.1 Подготовка серверной части 54
5.2 Загрузка сервисов 54
5.3 Образы и контейнеры 54
6 ВХОДНЫЕ ДАННЫЕ 56
6.1 Описание гипертекста средствами Markup 56
6.2 Входные файлы базы данных реального времени 56
7 ВЫХОДНЫЕ ДАННЫЕ 58
7.1 Протоколы действий обучаемого 58
ПРИЛОЖЕНИЯ 59
Приложение 1 – Состав программных сервисов СКОЯТ 59
Приложение 2 – Общие структуры данных СКОЯТ 60
Приложение 3 – Перечень сообщений СВУ формата JSON 61
Приложение 4 - Данные системных реляционных таблиц 64
Приложение 5 – Таблица значений статусов приложений 85
Приложение 6 – Опции трассировки ПО СКАДИ 86
Приложение 7 – Запросы и ответы сервиса WoMark 87
Приложение 8 – Запросы и ответы сервиса WoData 91
Приложение 9 – Пример Markup гипертекста для УМО 93
Приложение 10 - Данные пользовательских реляционных таблиц 95
Приложение 11 – Формат протоколов действий обучаемого 97
ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ 99
ЛИСТ ИЗМЕНЕНИЙ 100

1 ОБЩИЕ СВЕДЕНИЯ

1.1 ОБОЗНАЧЕНИЕ И НАИМЕНОВАНИЕ ПРОГРАММ

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.ОП.80.М, обозначение по ГОСТ 34.201-89.

«СКОЯТ 1.0» интегрировано с ПО «СКАДИВЕБ» - «Средства веб-доступа к данным для мониторинга и моделирования АЭС и промышленных предприятий». Описание ПО «СКАДИВЕБ», зарегистрированное в реестре программ для ЭВМ №2023685251 от 24.11.2023, представлено в документе 34185873.425510.004.ОП.80.М от 01.12.2023.

«СКОЯТ 1.0» также интегрировано с ПО «СКАДИ» - «Программная платформа АСУ ТП и диагностики для СВБУ АЭС и промышленных применений». Описание ПО «СКАДИ», зарегистрированное в реестре программ для ЭВМ №2021681559 от 23.12.2021, представлено в документе 34185873.425510.001.ОП.80.М от 04.07.2022.

«СКОЯТ 1.0» также использует для интеграции с тренажерами ПО «СКЗА» - «Средства коммуникации зонной архитектуры». Описание ПО «СКЗА», зарегистрированное в реестре программ для ЭВМ №2022669913 от 26.10.2022, представлено в документе 34185873.425510.002.ОП.80.М от 21.10.2022.

1.2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, НЕОБХОДИМОЕ ДЛЯ ФУНКЦИОНИРОВАНИЯ ПРОГРАММЫ

Программное обеспечение СКОЯТ состоит из клиентского ПО и серверного ПО. Пользователь системы СКОЯТ использует только клиентское ПО – браузер, установленное на устройство пользователя (компьютер, планшет). Для корректной работы клиентского ПО СКОЯТ требуется использование браузеров Yandex, Chrome, Edge, Safari, Mozilla.

Для выполнения ПО СКОЯТ необходимо иметь установленное серверное программное обеспечение СКАДИВЕБ.

Для функционирования «СКОЯТ 1.0» требуется наличие операционной системы семейств Linux. Операционное системное программное обеспечение устанавливается на все серверные 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.

Система авторизации СКОЯТ требует наличия доступа в интернет и использует регистрацию согласно Яндекс OAuth.

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

1.3 ЯЗЫКИ ПРОГРАММИРОВАНИЯ, НА КОТОРЫХ НАПИСАНА ПРОГРАММА

«СКОЯТ 1.0» написана на языке JavaScript в клиентских модулях, исполняемых на стороне браузера.

Серверная часть СКОЯТ написана на языке Го. Средства моделирования и все модели технологических систем написаны на языке Оберон. Скрипты контроля действий обучаемого написаны на языке Python.

2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1 КЛАССЫ РЕШАЕМЫХ ЗАДАЧ

СКОЯТ является платформой или интеграционным средством построения интерактивных обучающих систем для работы с технологическими данными. С помощью СКОЯТ осуществляется получение обучаемыми практических навыков в оперативной деятельности на занятиях с моделями технологических подсистем.

Использование СКОЯТ позволяет решать следующие классы задач:

- осуществление перехода от группового обучения на аналитическом тренажере к индивидуальному обучению с возможностью неограниченного персонального доступа к моделям технологических процессов;
- обучение на селективных подробных моделях подсистем, а не только работе АЭС в целом;
- предоставление студентам профильных специальностей доступ к технологическим моделям в режиме 24x7.

2.2 НАЗНАЧЕНИЕ ПРОГРАММЫ И ОСНОВНЫЕ ФУНКЦИИ

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

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.

Интеграция ПО СКОЯТ с ПО СКАДИВЕБ означает обеспечение возможности доступа СКОЯТ к данным моделей технологического процесса в части мониторинга у управления.

2.3 СВЕДЕНИЯ О ФУНКЦИОНАЛЬНЫХ ОГРАНИЧЕНИЯХ НА ПРИМЕНЕНИЕ

СКОЯТ может быть использована как информационная система только совместно с моделями технологического процесса. Использования СКОЯТ в реальных АСУ ТП не предусматривается.

Для разработки учебно-методических материалов используются внешние средства редактирования видеокадров ПО СКАДИ и внешние редакторы содержимого учебных заданий на языке Markup.

3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1 ОСНОВНЫЙ МЕТОД И МЕХАНИЗМЫ ВЗАИМОДЕЙСТВИЯ

Получение практических навыков взаимодействия достигается опытом решения обучаемым задач в условиях, максимально приближенных к реальным. Такие условия создаются управлением технологическим процессов с помощью реальной системы верхнего уровня (СВУ). Естественно, в обучающих системах должны использоваться модели СВУ и технологических подсистем вместо реальных.

Доступ к данным технологического процесса и управление может осуществляться на аналитических тренажерах АЭС. Однако, аналитические тренажеры позволяют осуществлять только групповое обучение в специально оборудованных учебно-тренировочных центрах. Для индивидуального обучения необходимо индивидуальное предоставление ресурсов обучаемым.

Основной метод обучения в СКОЯТ заключается в необходимости индивидуально решить практическую задачу в течении заданного интервала времени. Обучаемый проходит в своем понимании этапы оценки обстановки, выработки решения, управления посредством системы верхнего уровня и контроля за результатами произведенных действий. В дальнейшем, в режиме без временных ограничений, обучаемый может анализировать свои результаты и оценки выполнения, а также изучать материально-техническую часть по документам, заданиям и т.д.

Система СКОЯТ содержит в себе механизмы предоставления информации и ресурсов пользователям. Информация представляет собой веб-контент с документами, заданиями, вспомогательными материалами. Ресурсы предоставляются на временной основе с учетом категории и вида деятельности обучаемого (рисунок 1).

Ресурсы могут предоставляться как индивидуально, так и для группы пользователей. Группы и индивидуальные ресурсы не пересекаются. Пользователь инструктор может получать ресурс в одной группе с пользователем обучаемый.

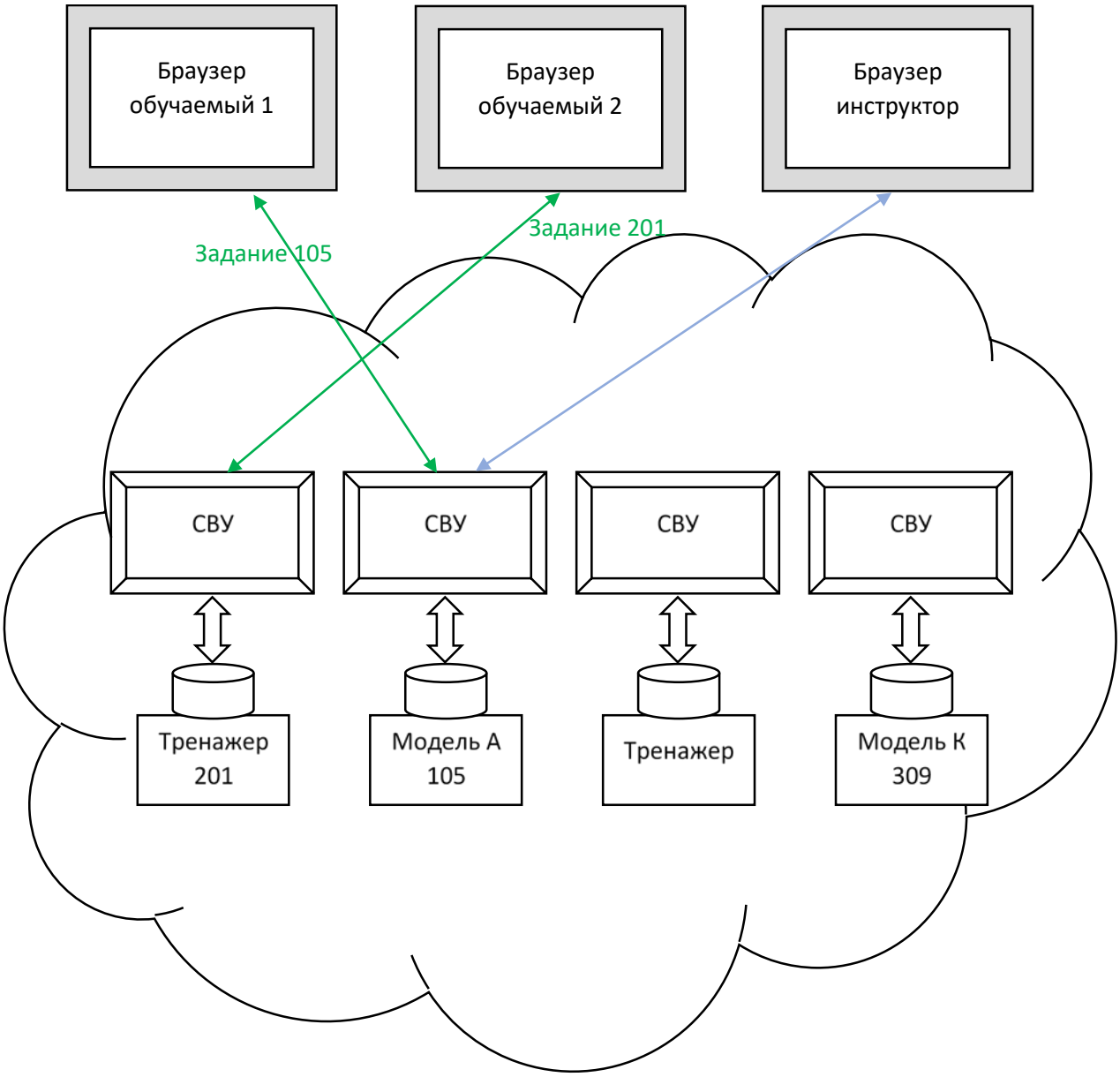

Рисунок 1 – Механизмы предоставления ресурсов СКОЯТ

Каждому обучаемому могут предоставляться селективные модели подсистем либо модель всего тренажера с определенными начальными условиями по запросу. Если пользователю предоставляется определенная селективная модель, например, «Модель К» для задания 309, то данная модель загружается и присоединяется к СВУ, доступ к управлению которой предоставляется. Если пользователю дается модель всего тренажера, то для тренажера выбирается начальное условие (например, 201 для браузера 2), с которого начинается выполнение технологического процесса.

Доступ к информации разделяется на общие и частные разделы. Общие разделы содержат контент с документами, описаниями, материалами. Частные разделы содержат пользовательскую информацию, историю прохождения тестов, статистику использования ресурсов.

3.2 СТРУКТУРА ПРОГРАММЫ

3.2.1 Состав серверных приложений

СКОЯТ является платформой для построения обучающих систем, использующих данные технологических процессов. СКОЯТ построена на основе облачных технологий с доступом из глобальных сетей посредством браузера (рисунок 2).

Программное обеспечение СКОЯТ состоит из следующих систем:

— система регистрации и управления доступом к СКОЯТ;

— система учета использованных ресурсов;

— система верхнего уровня (СВУ) диспетчерского управления;

— система управления моделями технологического процесса;

— средства связи с внешним тренажером с моделями технологического процесса;

— система контроля действий обучаемого;

— подсистема архивирования действий обучаемого и данных технологического процесса;

— система представления обучающих программ.

Средства связи с внешним тренажером с моделями технологического процесса предполагают обмен данными с удаленной моделирующей системой – интерфейсом абстрактного тренажера. Интерфейс абстрактного тренажера предполагает наличие внешней моделирующей системы и ввода/вывода, соединенных посредством средств связи. В реализации внешние модели могут иметь как минимальную функциональность, так и функциональность вплоть до аналитического или полномасштабного тренажера. Здесь и далее будет употребляться связь с внешним тренажером.

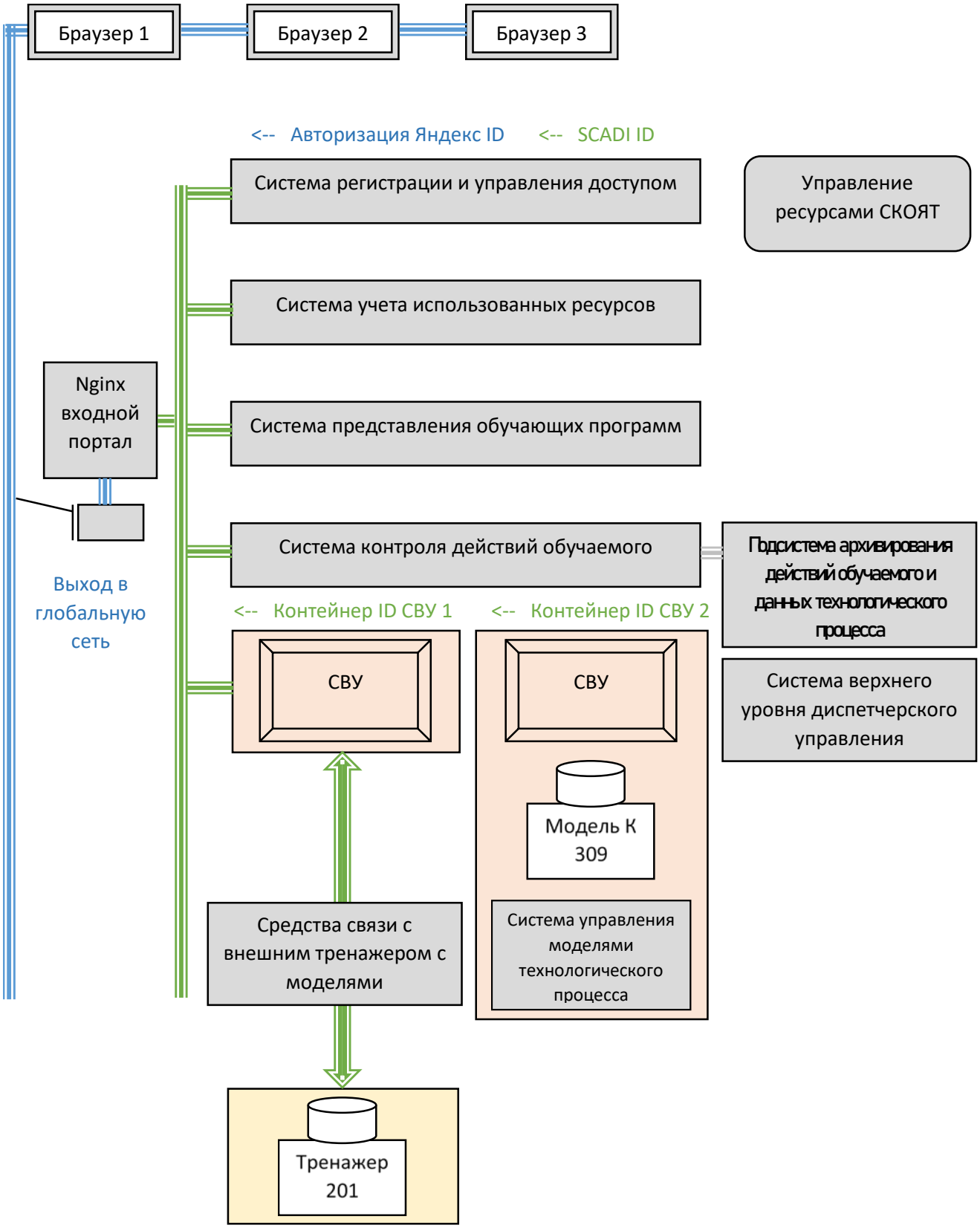

Рисунок 2 – Структура программы СКОЯТ

3.2.2 Пути доступа к системам

Обучающие системы на платформе СКОЯТ размещаются в глобальных сетях. Конкретные системы наполняют программное обеспечение платформы содержанием. Точкой входа в программную систему СКОЯТ из глобальной сети является входной портал на основе Nginx. Доступ осуществляетс через открытые порты по протоколу https. Все запросы к СКОЯТ проксируются Nginx и передаются конкретным серверам СКОЯТ в локальной сети. Составные части изделия доступны через привязку к путям в прокси-сервере (Таблица 1).

Путь	Название системы	Раздел	Описание
/	система регистрации и управления доступом	/account	Информация о пользователе
		/login	Логин
		/logout	Выход из системы
/data/	система учета использованных ресурсов		
/mark/	система оценивания действий обучаемого		
/cntrl/	система верхнего уровня (СВУ) диспетчерского управления	/cntrl/1	Контейнер ID 1 СВУ
		/cntrl/2	Контейнер ID 2 СВУ …
/at/	ПО внешнего тренажера	/at/101	Контейнер ID 101 внешнего тренажера

Таблица 1 – Пути доступа к системам СКОЯТ

Система представления обучающих программ осуществляет доступ к веб-содержимому обучающей системы. Такой доступ структурируется на основании характеристик пользователя, в частности, категории пользователя и вида деятельности.

Программное обеспечение системы верхнего уровня интегрируется вместе с моделями технологического процесса в контейнеры, к которым осуществляется доступ через пути cntrl/<number>, где number представляет собой номер контейнера. Если используется внешний тренажер, то контейнер СВУ соединяется по локальной сети с контейнером тренажера. Вместо тренажера также может использоваться внутренняя система управления моделями технологического процесса. Подсистема управления ресурсами СКОЯТ обеспечивает запуск и завершение работы контейнеров и предоставление ресурсов конкрентным пользователям системы.

Система архивирования действий обучаемого и данных технологического процесса осуществляет запись прохождения задания для последующего использования в качестве исходных данных для оценивания.

Система оценивания действий обучаемого использует данные архивирования для анализа и последующей выдачи оценок результатов проверки.

3.2.2 Пользовательские разделы систем

Доступ к ресурсам СКОЯТ с точки зрения пользователя определяется возможностями выборки кнопок основного управляющего фрейма клиентского окна системы (рисунок 3).

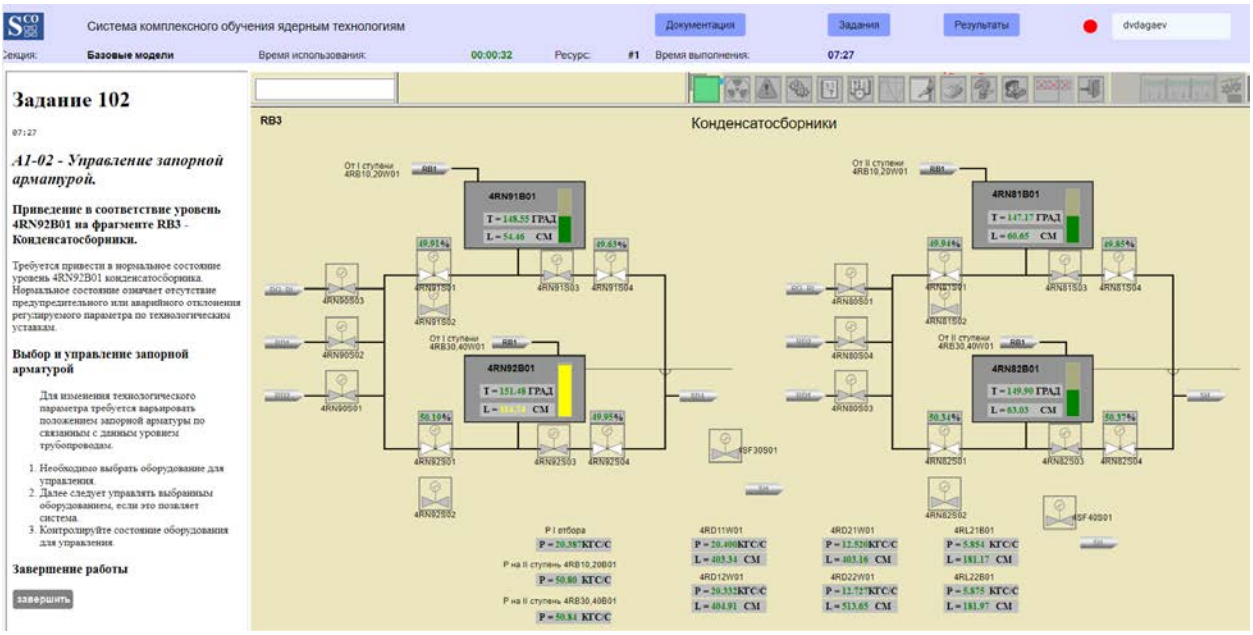

Рисунок 3 – Основное клиентское окно СКОЯТ

При выбранном разделе «Управление» осуществляется переход фрейма на /cntrl и основная экранная область предоставляется содержимому системы верхнего уровня диспетчерского управления.

При выбранном разделе «Документы» осуществляется переход фрейма на /data и основная экранная область предоставляется системе представления обучающих программ. При этом слева появляется еще один фрейм для навигации по содержимому раздела.

Аналогично выбирается раздел «Результаты».

Число разделов и их содержимое может меняться в зависимости от категории пользователя. Пользователям, обладающим административными привилегиями, добавляются дополнительные разделы для управления и контроля происходящих в системе процессов.

Вся фреймоваяч структура графических средств, представленных в браузере, составляет систему представления обучающих программ.

3.2.3 Сервисы программного обеспечения

Неграфическое ПО СКОЯТ состоит из сервисов – программ, выполняющихся на серверной части, и клиентских программных модулей, загружаемых и выполняющихся в браузере.

Состав программных сервисов СКОЯТ приведен в приложении 1.

3.3 СИСТЕМА РЕГИСТРАЦИИ И УПРАВЛЕНИЯ ДОСТУПОМ

3.3.1 Регистрация приложения в облачном сервисе

Приложение СКОЯТ зарегистрировано в облачном сервисе Яндекса. При регистрации приложение получило поля Client ID, Client secret, Redirect URI. Доступ к СКОЯТ предоставляется только пользователям из числа зарегистрированных на Яндексе, следовательно, имеющих учетную запись Яндекса. Учетная запись позволяет осуществить овторизацию по протоколу OAuth2.0.

3.3.2 Алгоритм авторизации СКОЯТ

Алгоритм авторизации показан на схеме рисунка 4.

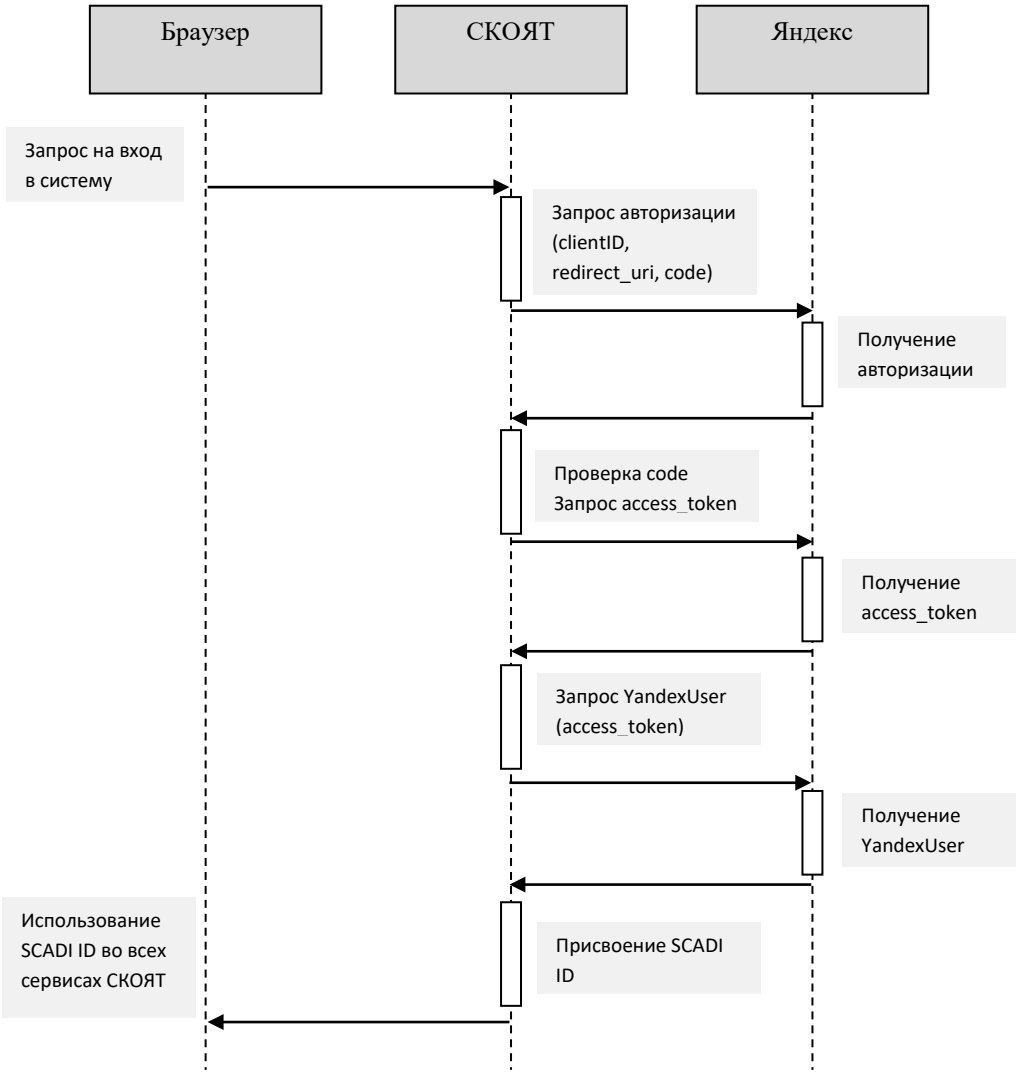

Рисунок 4 – Алгоритм авторизации СКОЯТ

Алгоритм авторизации СКОЯТ имеет в составе стандартный для протокола OAuth2.0 фрагмент. Первым этапом осуществляется запрос Яндекса на авторизацию с передачей результатов по адресу сервиса Redirect URI. При успешном прохождении и совпадении значения параметра code с переданным вторым этапом запрашивается access_token. При успешном запросе для данного значения access_token третьим этапом запрашивается информация о пользователе YandexUser. Если пользователь с учетной записью Яндекса зарегистрирован в СКОЯТ, то на четвертом этапе ему присваивается SCADI ID. Данная структура представляет собой объект пользовательских данных, который используется во всех сервисах СКОЯТ.

Сервис WoAccess реализует алгоритм авторизации СКОЯТ, механизмы регистрации и управления доступом.

Структура данных YandexUser, передаваемая между сервисами Яндекс и WoAccess, представлена в таблице 2-1 приложения 2. Структура данных ScadiID, передаваемая между сервисами СКАДИ, представлена также в таблице 2-2 приложения 2.

3.3.3 Использование SCADI ID

Пользовательская структура SCADI ID включает в себя следующие поля данных.

```
type ScadiID struct {
      ID          string `json:"id"`
      Name        string `json:"name"`
      ResourceID  int    `json:"resource_id"`
      Category    int `json:"category"`
      Activity    string `json:"activity"`
}
```

Идентификатор и имя пользователя используются для отображение персональной информации. Имя ресурса ResourceID проверяется при подключении к контейнерам СВУ. Имя ресурса должно совпадать с именем, назначенным для контейнера. Таким образом гарантируется подключение определенного пользователя к определенному ресурсу.

Также структура содержит категорию и вид деятельности, установленных для данного пользователя.

3.4 СИСТЕМА ВЕРХНЕГО УРОВНЯ ДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ

3.4.1 Основные функции системы

Система верхнего уровня диспетчерского управления реализована средствами ПО СКАДИВЕБ. На платформе СКАДИВЕБ реализованы информационно-управляющая системы для тренажеров АЭС. Программное обеспечение СКАДИВЕБ, предназначенное для построения систем мониторинга и аналитики в области АСУТП и диагностики, реализует следующие функции:

- сбора и выдачи данных, в том числе в режиме мягкого реального времени;
- онлайн-обработка последовательностей сигналов, в том числе с применением сложных диагностических алгоритмов;
- командного управления моделью управляющей системы с выдачей сигнала на модель технологического процесса;
- регистрации и представления сигнализациями;
- обеспечения доступа к архивным данным;
- запроса и управления протоколами;
- диагностики программных и аппаратных средств системы;
- конфигурации параметров системы;
- визуализации и отображения видеокадров;
- импортной и экспортной конвертации видеокадров.

Программное обеспечение верхнего уровня позволяет выводить статические и динамические данные технологических видеокадров в браузер устройств.

На рисунке 5 представлен пример такой визуализации со статическими и динамическими данными.

Окно управления, с которого может выдаваться управляющая команда в модель, выделено на рисунке голубым цветом.

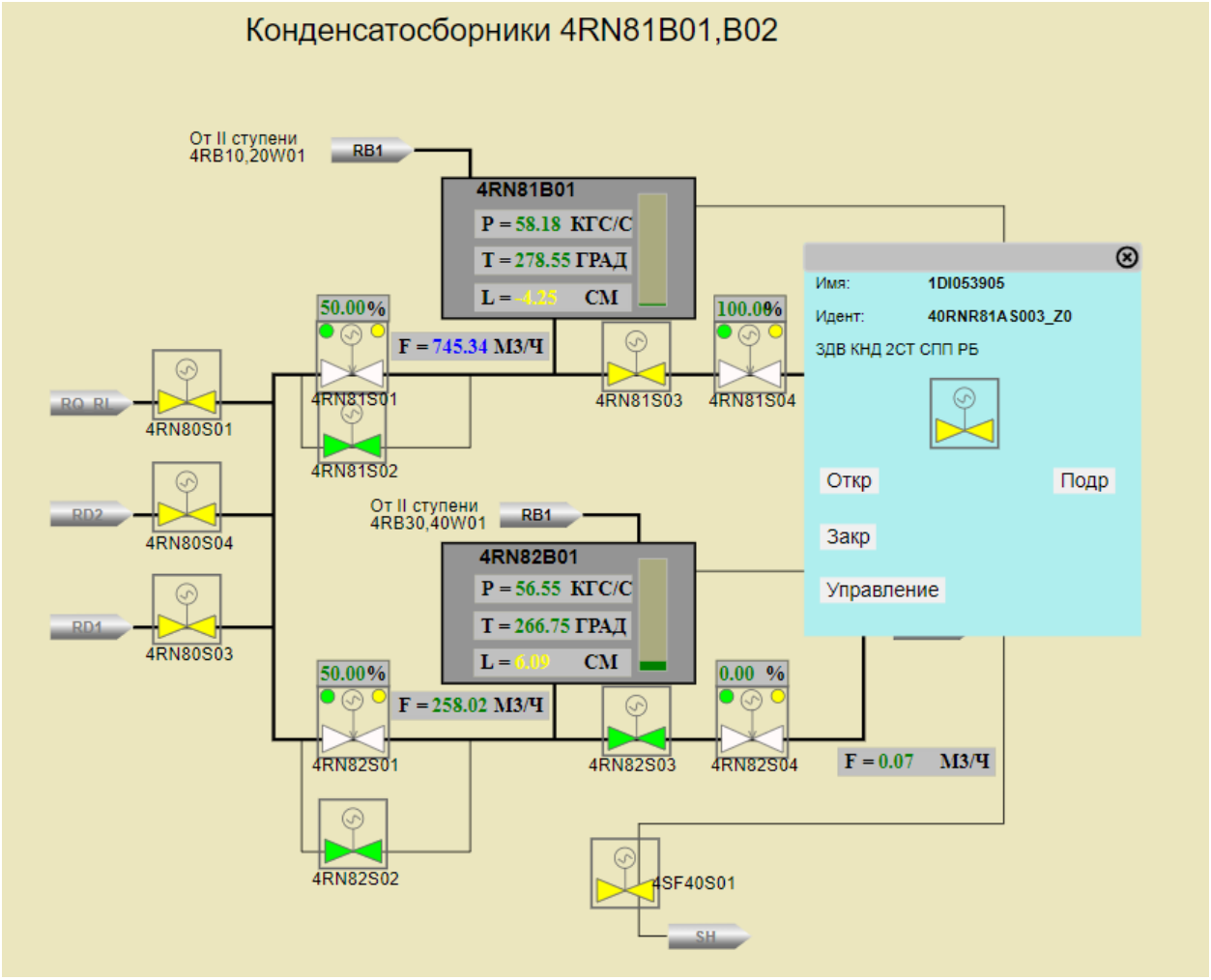

Рисунок 5 – Визуализация видеокадра Конденсатосборники 4RN81B01,B02

В качестве статических данных используется представление данного видеокадра в формате SVG. Программное обеспечение СКАДИВЕБ содержит подсистему скриптов генерации данных. С помощью указанных скриптов видеокадры ПО СКАДИ конвертируются в формат расширенного SVG, используемого в СКАДИВЕБ. При выводе формата расширенного SVG в браузер обеспечивается отображение статических данных.

В расширенном формате SVG содержатся привязки динамических данных, по которым осуществляется подписка.

3.4.2 Подключение в качестве контейнера

Подключение данных СВУ осуществляется через веб-сокет серверы WsShmCntrl, помещенные в контейнеры для каждого используемого ресурса, рисунок 6.

Каждому номеру ресурса поставлен в соответствие контейнер, доступный через nginx по cntrl/<номер ресурса>. Например, ресурс 1 доступен как cntrl/1, а ресурс 2 cntrl/2. Внутри контейнера располагается рантайм СКАДИ-СКАДИВЕБ с веб-сокет сервером.

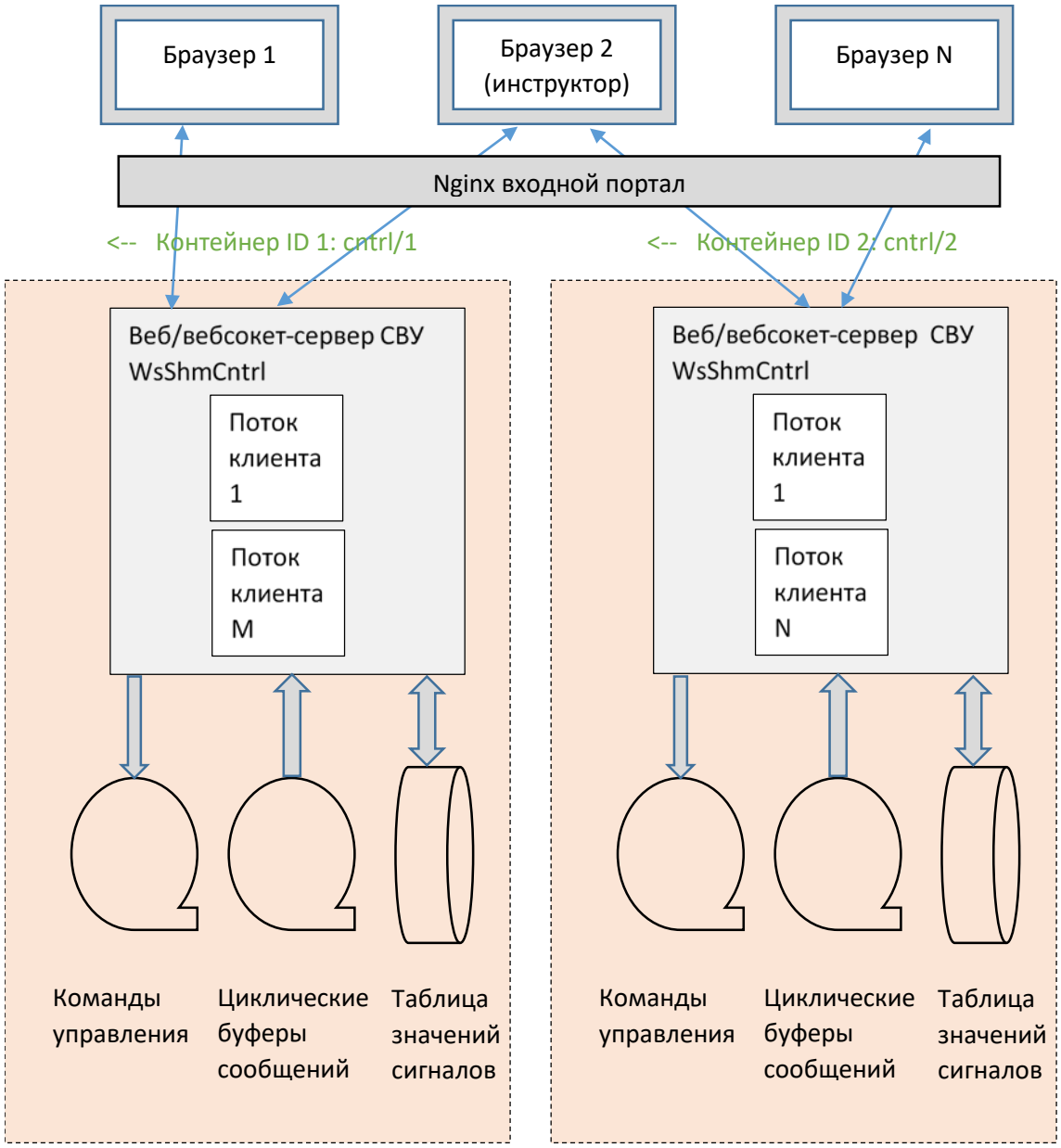

Рисунок 6 – Подключение контейнеров системы СВУ

Экземпляру вебсокет-сервера ставится в соответствие циклический буфер, в который записываются команды управления. Распространение и архивирование буфера с командами управления осуществляется средствами серверного программного обеспечения СКАДИ.

Механизм взаимодействия СКАДИВЕБ с выдачей команд управления имплементируется серверной задачей WsShmCntrl.

Динамические данные СВУ запрашиваются, устанавливаются по подписке и принимаются браузером по протоколу веб-сокета wss. Все динамические данные представляют собой сообщения формата JSON. Перечень сообщений представлен в приложении 4.

Запрос на установку подписки изменения текущих данных видеокадров может осуществляться следующим примером. Запрос представляет собой структуру clientMessage с заполненными полями.

```
sub.publish({"sframe": "frm1", "input": "AI:1AI001343.av AI:1AI001343.units
AI:1AI001343.descr SVALX,AI:~40YDR10CF001_XQ01.av AI:~40YDR10CF001_XQ01.units
AI:~40YDR10CF001_XQ01.descr SVALX,AI:~40YDR10CP001_XQ01.av
NVAL,AI:~40YDR10CP001_XQ01.av AI:~40YDR10CP001_XQ01.units
AI:~40YDR10CP001_XQ01.descr SVALX,DI:~40YDR10CL003_XH03.dv
DI:~40YDR10CL003_XH03.descr"})
```

Для фрейма frm1 устанавливается набор переменных подписки input. По установленному набору передаются сообщения об изменениях значений параметров от предыдущих величин. Для данного набора сообщение dataMessage выглядит следующим образом.

```
Message:
{"sframe":"frm1","nvars":4,"vars":[{"vname":"AI:1AI001343.av","val":"0.498;48
;1704975166","atype":"R4QTS"},{"vname":"SVALX,AI:~40YDR10CF001_XQ01.av","val"
:"0.4904","atype":"R4"},{"vname":"SVALX,AI:~40YDR10CP001_XQ01.av","val":"62.0
9","atype":"R4"},{"vname":"NVAL,AI:~40YDR10CP001_XQ01.av","val":"0.24834582",
"atype":"R4"}]}
```

Команда остановки обновлений freeze реализуется как клиентское сообщение RpcMessage.

```
centrifuge.rpc("RunStatus", {"sframe": "frm1", "rstatus": "freeze"})
```

В ответ сервером посылается сообщение runStatusMessage об остановке.

```
Run Status is:
{"data":{"sframe":"frm1","rmethod":"RunStatus","result":"freeze"}}
```

Команда старта обновлений run реализуется как клиентское сообщение RpcMessage

```
centrifuge.rpc("RunStatus", {"sframe": "frm1", "rstatus": "run"})
```

В ответ сервером посылается сообщение rpcStatusMessage о старте.

```
Run Status is:
{"data":{"sframe":"frm1","rmethod":"RunStatus","result":"run"}}
```

Текущие значения динамических данных СВУ берутся либо из системы управления моделями технологического процесса, либо из подсоединенных внешних моделей/тренажеров.

3.4.3 Формат представления видеокадров

Отображение статических и динамических данных осуществляется браузером в формате расширенного SVG. Базовые элементы в SVG определяются как прямоугольник, круг, овал, линия, ломаная, кривая, текст и т.д. Расширение формата предполагает специальные типы для подмоделей, переменных процесса, определений динамики.

Объект подмодель описывается типом Svd. Объект подмодели представляет собой код SVG, вставляемый в исходный. Параметр onclick задает имя команды по умолчанию, активируемой при нажатии левой кнопки манипулятора.

```
<svg                                                    oname="40RNR82AS002"
onclick="parent.window.goControl('40RNR82AS002,VLV')">
<scadiweb-var  objtype="Var"  name="z0"  vname="z0"  vartype="3"  gflags="2"
feval="DI:~$code_Z0.dv" id="z0" />
<scadiweb-dmisc objtype="Dmisc" dyn="dbackground" refoid="#z0" groid="#name_8"
dt="t" ntt="3" imax="2" fmax="2.0">
<rect  fill="#ffffff"  fill-opacity="0.0"  x="-35.0"  y="-35.0"  width="70.0"
height="70.0" transform="translate(368.0,785.5)" />
<scadiweb-thr ind="1" tr="0" idat="65280" />
<scadiweb-thr ind="2" tr="16384" idat="65280" />
<scadiweb-thr ind="3" tr="32768" idat="16776960" />
</scadiweb-dmisc>
…
```

Каждая вставляемая подмодель представляет собой экземпляр графического объекта со своими параметрами и привязками к переменным состояния.

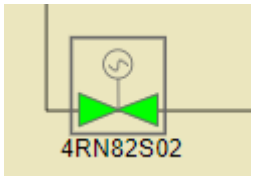

Рисунок 7 – Подмодель задвижки с динамикой

На рисунке 7 показана подмодель задвижки с параметризацией привязки к KKS 40RNR80AS002 и RTM 4RN82S02. Подмодель привязана к соответствующим их технологическим кодам состояниям процессов. Отображение задвижки иллюстрирует закрытое состояние.

Изменение динамических данных процесса приводит к изменению значения переменной DI:~ 40RNR80AS002_Z0.dv, что, в свою очередь приводит к изменению цвета фона dbackground. Последний привязан к значениям переменной, согласно табличным определениям динамики scadiweb-thr, входящем в расширение SVG.

Окна управления также представляют собой видеокады SVG, загружаемые в отдельные фреймы, рисунок 8.

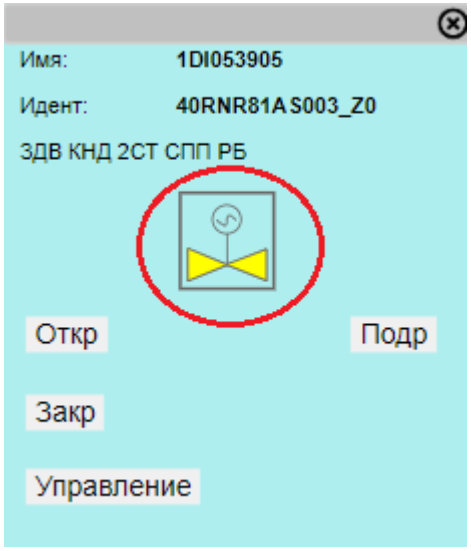

Рисунок 8 – Окно управления задвижной

Окна управления разрабатываются по одному для каждого типа оборудования. При вызове окно управления параметризуется идентификаторами вызванного объекта. Для приведенного рисунка параметризация осуществлена по KKS=40RNR81AS003.

Расширенный SVG код окна управления учитывает возможность параметризации.

```
  <svg oname="$code" onclick="parent.window.goControl('$code,VLV')">
    <rect fill="#ffffff" fill-opacity="0.0" x="-35.0" y="-35.0" width="70.0"
height="70.0" transform="translate(143.0,152.0)" />
    <rect name="name_3" x="-30.125" y="-31.125" width="59.875" height="61.375"
rx="0.0"  ry="0.0"  stroke="#696969"  fill-opacity="0.0"  stroke-width="2.0"
stroke-linecap="bevel" stroke-linejoin="miter" stroke-dashoffset="0.0" stroke-
opacity="1.0" transform="translate(143.0,152.0)" />
    <polygon name="name_4" points="-25.0,5.0 25.0,25.0 25.0,5.0 -25.0,25.0"
stroke="#696969"  fill="#c0c0c0"  stroke-width="1.0"  stroke-linecap="bevel"
stroke-linejoin="round"  stroke-dashoffset="0.0"  stroke-opacity="1.0"  fill-
rule="N" fill-opacity="1.0" transform="translate(143.0,152.0)" />
    <circle name="name_6" cx="-0.375" cy="-14.125" r="9.125" stroke="#696969"
fill="#ffffff"  fill-rule="N"  fill-opacity="0.0"  stroke-width="1.0"  stroke-
linecap="bevel"   stroke-linejoin="miter"   stroke-dashoffset="0.0"   stroke-
opacity="1.0" transform="translate(143.0,152.0)" />
```

```
    <path    name="name_7"    d="M-6.375,-13.5    C5.958330154418945,-3.25    -
6.458330154418945,-23.625  6.125,-13.25 "  stroke="#696969"  fill-opacity="0.0"
stroke-width="1.0"   stroke-linecap="bevel"   stroke-linejoin="miter"   stroke-
dashoffset="0.0" stroke-opacity="1.0" transform="translate(143.0,152.0)" />
    <scadiweb-var  objtype="Var"  name="z0"  vname="z0"  vartype="3"  gflags="2"
feval="DI:~$code_Z0.dv" id="z0" />
    <scadiweb-dmisc     objtype="Dmisc"     dyn="dbackground"     refoid="#z0"
groid="#name_8" dt="t" ntt="3" imax="2" fmax="2.0">
  </svg>
```

Расширение в коде определяют имя как `oname="$code"` и функцию обработки нажатия как `onclick="parent.window.goControl('$code,VLV')"`. Параметризация означает, что управление будет ассоциирована с необходимыми, указанными на видеокадре переменными.

Подмодель задвижки, отмеченная на рисунке красным овалом, вставляется в окно управления аналогичным видеокадру образом. Привязки переменных подмодели осуществляются с учетом параметризации окна управления.

Динамические данные в виде таблиц значений сигналов и циклических буферов сообщений формируются в моделях. Внутренние модели описаны в главе о системе управления моделями технологического процесса. Сигналы с внешних моделей принимаются через средства связи с внешним тренажером.

3.5 СИСТЕМА УПРАВЛЕНИЯ МОДЕЛЯМИ ТЕХНОЛОГИЧЕСКОГО ПРОЦЕССА

3.5.1 Общие характеристики

Система управления моделями технологического процесса реализована средствами расширенного моделирования ПО СКОЯТ-МОДЕЛЬ.

Система используется для работы с автономными моделями, не связанными с реальным технологическим процессом.

База данных реального времени технологического процесса содержит все необходимые расчетные величины в виде глобальных переменных.

В системе обеспечено сохранение и восстановление состояния всех переменных технологического процесса в именованных областях. Сохранение и восстановления значений переменных производится через xxx-файлы маппирования базы данных реального времени. Управление сохранением/восстановлением состояний позволяет стартовать модели из предварительно сохраненных состояний. Предварительно сохраненные состояния содержат:

- значения переменных технологического процесса (давления, расходы, температуры и т.д.);
- состояния технологического оборудования – задвижек, насосов, рекуляторов;
- установленные значения отказов, реализуемых в модели.

В системе реализована возможность запуска до 50 экземпляров автономных моделей для одновременной работы группы пользователей. Каждая запущенная модель локализована в автономном контейнере. Если используются одинаковые модели, т.е. созданные из одного образа, то каждый автономный контейнер создается из указанного образа и не зависит от других контейнеров данного образа.

3.5.2 Структура программных модулей

Логическая структура СКОЯТ-МОДЕЛЬ представлена на рисунке 9. В контейнере модели представлены основные программные модули, основные структуры данных и

файлы базы данных, архива и начальных условий. Связь с внешним миром осуществляется через модуль WsShmCntrl, относящийся к системе верхнего уровня.

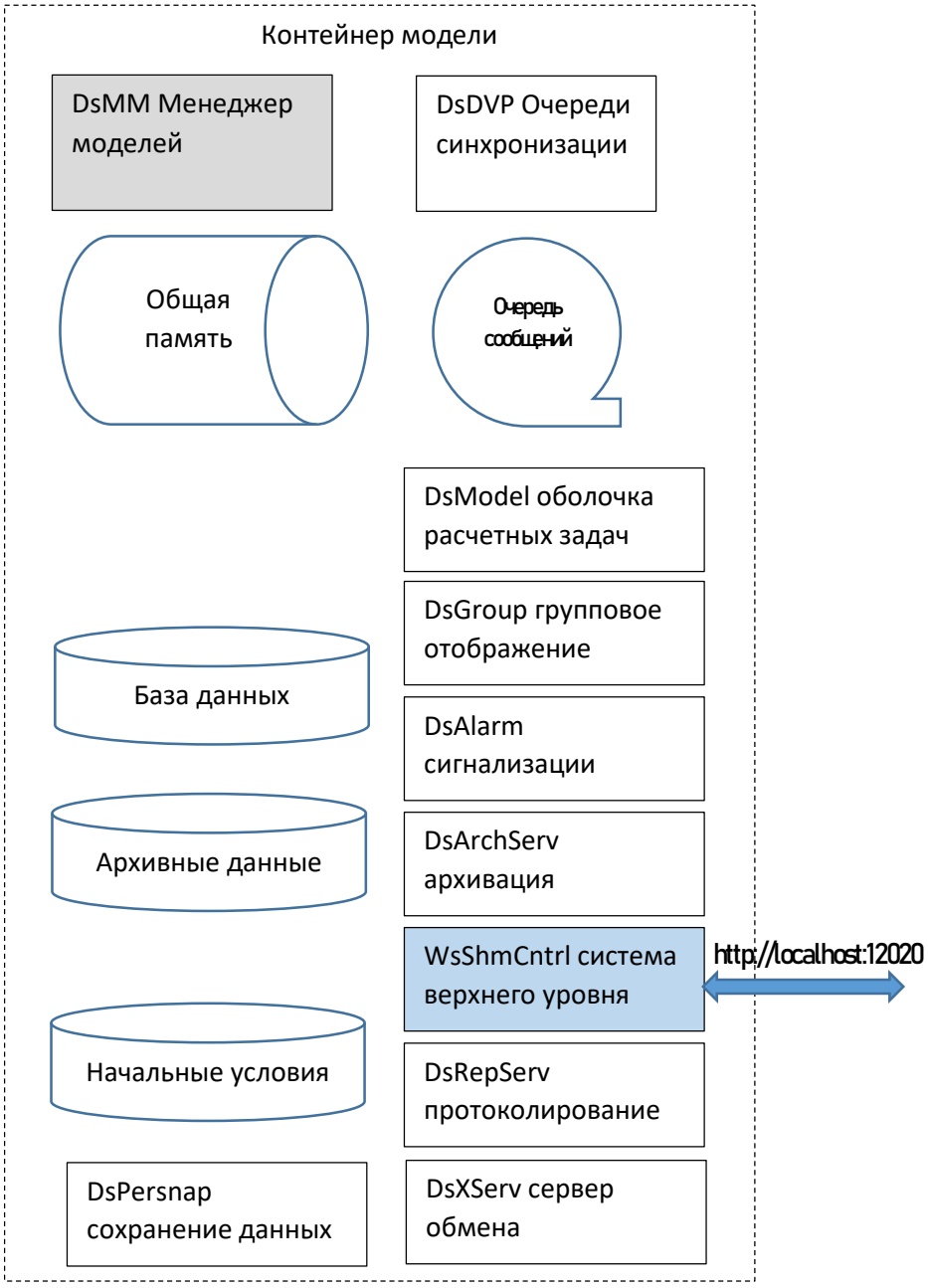

Рисунок 9 – Структура программных модулей

Программные процессы внутри узлов СКОЯТ-МОДЕЛЬ взаимодействуют между собой через распределенную общую память. Распределенная общая память состоит из так называемых глобалов – выделенных при старте системы блоков фиксированного размера. Каждый глобал состоит из заголовка и массива данных определенной структуры.

Заголовок глобала представляет собой 24-байтную структуру в составе:

```
SHDR* = RECORD
      n_total_objs-, n_used_objs-, sz_used_mem-, usr_data-: Tb.Int32;
      length: LONGINT;
```

```
END;
```

Заголовок каждого глобала содержит поля:

- максимальное число объектов n_total_objs;
- текущее число объектов n_used_objs;
- размер используемой памяти sz_used_mem;
- поле переменного значения usr_data, обычно используемое как индекс в очереди;
- длину length, используемую в зависимости от типа глобала.

Массив данных определенной структуры заполняется в зависимости от типа глобала. Список типов глобалов общей памяти приведен в таблице 2.

Тип	Описание	Мета/Данные	Структура
G	Список глобалов	Мета	КонстТаблица
E	Список имен констант Enumerations	Мета	КонстТаблица
T	Типы объектов	Мета	КонстТаблица
A	Типы атрибутов объектов	Мета	КонстТаблица
S	Лог системных сообщений	Сообщения	ЦиклБуфер
s	Системные реляционные таблицы	Данные	Таблица
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 глобалов общей памяти

Все объекты глобалов s и n наследуются от базовых объектов OGeneric. Поля всех наследуемых объектов содержат информацию о полях базового типа OGeneric.

```
OGeneric* = EXTENSIBLE RECORD
      oid-: Tb.Int32;
      ncadr-: Tb.Int32;
      name-: Sb.STag;
      proglags-, alloflags-: Tb.Int8;
      host_local-: Tb.Int16;
END;
```

Все системные типы (типы из глобала s) вначале включают поля OGeneric, а затем специфические для определенного типа поля. Перечень полей OGeneric приведен в таблице 3.

Поле	Наименование	Значения	Размер, байт
oid	Идентификатор объекта	<Число>	4
ncadr	Номер кадра	<Число>	4
name	Имя объекта	<Имя>	32
proflags	Флаги обработки	<Число>	1
alloflags	Флаги аллокирования	<Число>	1
host_local	Номер хоста для локальных данных	<Число>	0 означает отсутствие атрибута

Таблица 3. Поля типа OGeneric

Первым атрибутом описывается поле oid, вторым ncadr, третьим name, и т.д. Как системные, так и пользовательские реляционные таблицы содержат информацию о значениях базового типа. Перечень системных реляционных таблиц приведен в приложении 4.

Расширяемый пользовательский тип vart = 10 наследуется от OConfigurable. Расширение OConfigurable является структурой, наследуемой от OGeneric и добавляющей дополнительные поля для индексации и версионности.

```
OConfigurable* = EXTENSIBLE RECORD (SO.OGeneric)
      revnum*: Tb.Int32;
      revdate*: Tb.Time;
      code*, pin*: Sb.STag;
END;
```

Все пользовательские типы глобалов n, не являющиеся переменными процесса, имеют vart = 10 и наследуется от OConfigurable. Перечень полей OConfigurable приведен в таблице 4.

Поле	Наименование	Значения	Размер, байт
revnum	Номер версии	<Число>	4
revdate	Дата модификации	<Число>	8
code	Технологический код в форматах KKS, PTM, …	<Имя>	32
pin	Контакт или адрес устройства связи с объектом в формате, определяемом этим устройством	BOOL	32

Таблица 4. Поля типа OConfigurable

Объекты с vart = 10 могут описывать какие-то функциональные блоки (регуляторы, блоки ФГУ), не являющиеся переменными процесса.

Управляемые пользовательские типы с vart = 23 наследуются от OManagable. Расширение OManagable является структурой, наследуемой от OConfigurable и добавляющей дополнительные поля, характеризующие запись как переменную процесса.

```
OManagable* = EXTENSIBLE RECORD(OConfigurable)
      descr*: Sb.SDescr;
      dst_mask*: Tb.Int32;
      xch_num*, fun_type*: Tb.Int16;
      block*, ablock*: SO.VariableRef;
      evcu*: SO.E16B;
      almpri*: Tb.Int16;
      qman*, qprev*: Sb.Quality;
      bsim*, chatter*: BOOLEAN;
      nchanges*: Tb.Int32;
END;
```

Перечень полей OManagable приведен в таблице 5.

Поле	Наименование	Значения	Размер, байт
descr	Описание параметра	<Строка>	80
dst_mask	Адресат сигнализации	<Число>	4
xch_num	Номер агента обмена	<Число>	2
fun_type	Тип функции обработчика	<Число>	2
block	Ссылка на объект	<Ссылка>	16
ablock	Альтернативная ссылка	<Ссылка>	16
evcu	Значение сигнализации	<Сигнализация>	16
almpri	Флаги приоритета сигнализации	<Множество>	2
qman	Ручной ввод качества	<Качество>	2
qprev	Предыдущее качество	<Качество>	2
bsim	Включение имитации	BOOL	1
chatter	Дребезг	BOOL	1

nchanges	Число изменений	<Число>	4

Таблица 5. Поля типа OConfigurable

Далее вернемся к процессам на рисунке 9.

Общий мониторинг и управление процессами осуществляет DsMM –менеджер моделей. Он обеспечивает запуск/рестарт всех процессов СКОЯТ-МОДЕЛЬ в данном контейнере.

Программы DsDVP осуществляют обновление табличных данных узлов по очередям сообщений.

Обработка сигналов переменных процесса, представляющую совокупность из аналоговых входных AI, дискретных входных DI и расчетных данных произвольной структуры осуществляется программой DsModel. Конкретные расчетные задачи вычисляются в динамически загружаемых модулях плагинов. Для работы модулей плагинов необходим сервер оболочки плагина DsModel, который может загружать в себя плагины различных типов, по одному экземпляру на каждый тип.

Модуль DsGroup осуществляет подготовку данных для группового отображения для системы верхнего уровня. Он охватывает группы данных различных типов.

DsAlarm является подсистемой регистрации и управления сигнализациями.

DsArchServ представляет собой ПО для архивации текущих данных.

DsRepServ представляет собой ПО для доступа к архивным данным и текущим данным, в зависимости от параметров задания.

DsPersnap является программой, которая осуществляет персистентность данных. Персистентность заключается в периодическом и окончательном сбросе текущих данных в постоянное хранение в виде дисковых файлов. Кроме этого персистентность позволяет работать динамически задаваемыми начальными условиями.

DsXServ представляет собой сервер обмена. Он обменивается данными с агентами обмена, шлюзами передачи и внешними серверами обмена данных по сетевым протоколам.

WsShmCntrl представляет собой веб-сокет сервер системы верхнего уровня.

3.5.3 Идентификация объектов в реляционных таблицах

Информация о типах и атрибутах в глобалах T и A позволяет осуществлять доступ к объектам в реляционных таблицах глобалов s и n.

Идентификация объектов осуществляется по следующему правилу:

```
Объект = Тип:Идент
```

Идентификация полей объектов объектов осуществляется по следующему правилу:

```
Поле = Тип:Идент.Атрибут
```

Значение типа представляет собой символьное наименование из таблицы типов глобала T (поле Type.name). Значение атрибута представляет собой символьное наименование из таблицы атрибутов глобала A (поле Attrs.name).

Значение «Идент» представляет собой идентификатор объекта в глобалах s и n.

Идентификация объекта по номеру oid осуществляется по правилу:

```
Идент = #oid
```

Доступ по номеру является индексным, т.е. самым быстрым за 1 операцию.

Идентификация объекта по имени name осуществляется по правилу:

```
Идент = name
```

Доступ по имени без индексации осуществляется линейным поиском в массиве и является медленным. Для ускорения доступа необходимо использовать индексацию. Индексацию по полю name можно включить для глобалов s и n заданием флага GFL_SORTED{0} в Global.flags. Это осуществляется установкой GLOBAL[glob].flags=S, где glob означает номер глобала s и n. Для индексации по полю name внутри глобалов s и n необходимо иметь отсортированные по возрастанию уникальные имена name.

Индексация по полю name возможна с использованием индексных файлов, но только для объектов глобалов n, наследуемых от OConfigurable. Для этого необходимо создание индексных файлов i, со ссылкой на глобал n. Способ индексации должен быть задан флагом GFL_BYNAME{1} в Global.flags. Это осуществляется установкой GLOBAL[glob].flags=N, где glob означает номер глобала i.

Идентификация объекта по коду code осуществляется по правилу:

```
Идент = ~code
```

Доступ по коду (KKS, PTM) без индексации осуществляется линейным поиском в массиве и является медленным. Для ускорения доступа необходимо использовать индексацию. Индексация по полю code возможна с использованием индексных файлов, но только для объектов глобалов n, наследуемых от OConfigurable. Для этого необходимо создание индексных файлов i, со ссылкой на глобал n. Способ индексации должен быть задан флагом GFL_BYCODE{2} в Global.flags. Это осуществляется установкой GLOBAL[glob].flags=C, где glob означает номер глобала i.

Идентификация объекта по контакту pin осуществляется по правилу:

```
Идент = +pin
```

Доступ по контакту устройства связи с объектом без индексации осуществляется линейным поиском в массиве и является медленным. Для ускорения доступа необходимо использовать индексацию. Индексация по полю code возможна с использованием индексных файлов, но только для объектов глобалов n, наследуемых от OConfigurable. Для этого необходимо создание индексных файлов i, со ссылкой на глобал n. Способ индексации должен быть задан флагом GFL_BYPIN{3} в Global.flags. Это осуществляется установкой GLOBAL[glob].flags=P, где glob означает номер глобала i.

Идентификация объекта по макросу осуществляется по правилу:

```
Идент = @macro
```

Возможные значения макроса зависят от типа таблиц:

- @H означает подстановку индекса хоста в таблице Hosts;
- @T означает подстановку индекса текущей задачи в таблице Tasks;
- @[a-f]<num> означает подстановку номера строки в групповом отображении в таблице Cells. При этом символ в имени типа & обозначает ссылку на объект, а не поле. Наличие ссылки на объект позволяет подключить любой адресуемый атрибут, относящийся к данному объекту;
- @[a-f] означает подстановку номера группы в групповом отображении в таблице CSets.

Поле объекта может быть как значением, так и ссылкой. Собственно, ссылки также представляют собой значения с типом 11 (VREF). Значением поля ссылки будет структура VREF, которая выводится на печать в виде, подобном #27,3,256,-1.10i-1. Поля структуры VREF, описанные в таблице 4, при печати соответствуют полям строки:

otype,oid,attrid,dim.atype<index>

Для большей читаемости ссылки используется символьное представление объекта, на который указывает ссылка. Символьное представление для полей выглядит как Тип:Идент.Атрибут, а для объектов как Тип:Идент.

Для получения значения в ссылке используется префикс «^». При идентификации с использованием префикса ^ в значение подставляется значение поля ссылки, а не сама ссылка.

Получение значений определенных полей через ссылку на объект иногда приводит к длинной, нечитаемой записи вида ^&:@a1.val. В видеокадрах системы визуализации для первой группы ‘a’ используется сокращенная аббревиатура:

@n вместо ^&:@an – только в первой группе системы визуализации.

Возможные способы построения идентификаторов объектов и полей объектов в качестве примеров сведены в таблицу 6.

Идентификация	Значение	Примечание
Host:#0.name	any_computer	Имя компьютера в таблице Hosts[0].name
Host:@H.ip_address	127.0.0.1	IP-адрес компьютера в таблице Hosts[host_id].ip_address для данного хоста
AI:1AI0004.av	145.35	Значение текущей переменной AIs[oid].av, где AIs[oid].name = 1AI0004
AI:~YA12TCP.ainp	0.0	Значение входной переменной AIs[oid].av, где AIs[oid].code = YA12TCP
DI:+D101016.dv	1	Значение текущей переменной DIs[oid].dv, где DIs[oid].pin = D101016
CSet:@a.cp	1	Число страниц первой группы для текущего визуализатора. Макро @a подставляет разные значения для разных экземпляров программ визуализации
Cell:@a1.rfs(1)	AI:1AI0004.av	Ссылка на поле av аналогового объекта. Используется символьное представление ссылки для отображения
^Cell:@a1.rfs(1)	145.35	Значение поля объекта, на который установлена ссылка. Аналогично строке 2, значение текущей переменной AIs[oid].av, где AIs[oid].name = 1AI0004
&:@a2	AI:1AI0001	Ссылка поля 2 группового отображения на адресуемый объект AI:1AI0001 целиком
^&:@a1.av	145.35	Ссылка поля 1 на адресуемый объект AI:1AI0004 и применение операции взятия значения атрибута av. Аналогично строке 2, значение текущей переменной AIs[oid].av, где AIs[oid].name = 1AI0004
@1.av	145.35	Сокращенное для группы ‘a’ системы визуализации наименование предыдущего варианта

Таблица 6. Способы идентификации объектов

Система идентификации охватывает все способы доступа к объектам и полям объектов. Все программное обеспечение использует механизмы доступа к разделенным переменным.

3.5.4 Менеджер моделей

ПО мониторинга и управления процессами реализует DsMM – менеджер моделей.

ПО СКОЯТ-МОДЕЛЬ конфигурируется для определенного проекта <project> и загружается для одноименного каталога. Загрузка СКОЯТ-МОДЕЛЬ осуществляется командой, переходящей в каталог (при наличии опции):

```
DsMM –cd <path/to/project>
```

Команда DsMM стартует менеджер моделей. Менеджер моделей при необходимости инициализирует общую память, вызывая загрузчик для заданного конфигурационного файла (опция –cfg_file <filename.cfg>) или DsmConfig.cfg по умолчанию, либо подключается к имеющейся для <project> загрузке.

После этого DsMM начинает выполнять свою основную функцию мониторинга и управления процессами. Менеджер моделей выполняет периодическое обновление, проверяя значения переменных в общей памяти.

Текущее состояние определяется в полях структур данных реляционных таблиц. Например, значение Task:@T.task_status определяет значение поля task_status для номера текущей задачи в таблицах Task приложения 4-4. В свою очередь, значение Host:@H.host_status относится к текущему номеру хоста и полю host_status для таблицы 4-6.

Значение переменной общей памяти Task:@T.task_status определяет статус текущей задачи, т.е. DsMM в виде перечисления Enum SRV_STAT (таблица 5-1 приложения 5). Задача DsMM всегда имеет номер 0, поэтому для нее всегда можно использовать Task:#0.task_status. По статусу DsMM выставляется состояние заморозки приложений всего узла Host:@H.host_status.

Если статус Task:@T.task_status устанавливается в END, то инициируется завершение DsMM и всех остальных его дочерних процессов. Это можно сделать вызовом утилиты `DsAccess -end all` из командной строки или командой `end` из отладчика DsISD.

Для текущего хоста обновляется параметр Host:@H.current_time и времена модификации, относящиеся к глобалам узла.

Менеджер моделей осуществляет мониторинг и управление задачами СКОЯТ-МОДЕЛЬ. Настройки старта устанавливают, что управляемая задача @T должна стартовать согласно конфигурации проекта. Если задача должна стартовать, то из начального состояния INIT она переходит в состояние начала загрузки LOAD, иначе задача переводится в статус END и не отслеживается менеджером моделей.

При загрузке LOAD менеджер моделей создает дочерний процесс вида ‘Task:@T.exe_path Task:@T.command_line’. Имя программы и параметры command_line также попадают в общую память при загрузке конфигурации. Имя задачи Task:@T.name означает уникальное имя задачи. Одна программа Task:@T.exe_path с одинаковыми или разными параметрами может быть запущена несколько раз как несколько отдельных задач. Например,

```
Task:#8.name = ‘DsModelGC’
Task:#8.exe_path = ‘DsModel’
Task:#8.command_line = -hl DpgcValves,DpgcCNValves,DpgcPumps’
```

для моделей задвижек, насосов, регуляторов;

```
Task:#9.name = ‘DsModelMain’
Task:#9.exe_path = ‘DsModel’
Task:#9.command_line = ‘-hl DpmodelMain,DpgcNodes,DpgcPipes’
```

для моделей теплогидравлических процессов.

Запуск дочернего процесса сопровождается немедленным кодом возврата Task:@T.command_result. Отрицательное значение означает ошибку и установку task_status=ERROR. Неотрицательное значение приводит к статусу WAITING, что означает ожидание завершения загрузки. При этом счетчик Task:@T.watch_count устанавливается в значение Task:@T.watch_limit.

Завершение загрузки и начало работу осуществляет сам дочерний процесс, изменяя свой статус согласно рисунку 10.

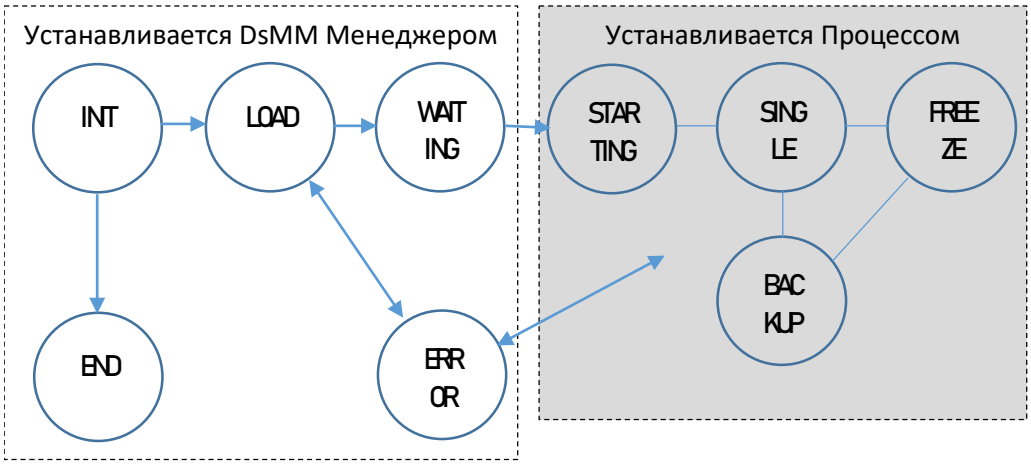

Рисунок 10 – Статус управляемого дочернего процесса

При каждом цикле мониторинга менеджер моделей уменьшает значение счетчика Task:@T.watch_count и по значению этого счетчика может установить статус мониторинга. Если процесс стартовал, то он работает, и в каждом цикле обновления возвращает счетчик Task:@T.watch_count в значение Task:@T.watch_limit. Если цикла обновления дочернего процесса нет, то при уменьшении Task:@T.watch_count <= Task:@T.watch_limit / 2 статус дочернего процесса устанавливается в ERROR, а при уменьшении счетчика Task:@T.watch_count дочерний процесс принудительно рестартует,

если такая возможность не отключена в конфигурации. Описываемый алгоритм известен как сторожевой таймер.

Ограничение числа принудительных рестартов задается параметром Task:@T.restart_limit. Отрицательное значение означает неограниченное количество рестартов. Неотрицательное значение устанавливает предел счетчика рестартов Task:@T.restart_count. Рестарт возможен, если Task:@T.restart_count < Task:@T.restart_limit или в случае, если Task:@T.restart_limit < 0. Рестарт производится при нулевом значении Task:@T.watch_count переходом в статус LOAD. Предыдущий запущенный дочерний процесс при необходимости принудительно завершается, и производится очередной запуск дочернего процесса последующие действия по ожиданию завершения, описанные выше.

Таким образом, работающий менеджер моделей обеспечивает мониторинг и управление всеми процессами, относящимися к данному узлу.

Обеспечение постоянной работы самого менеджера моделей должно осуществляться средствами операционной системы.

3.5.5 Сохранение и восстановление состояния

Все глобалы общей памяти создаются как привязанные к файлам отображения. Для этого используется механизм мэппинга (memory mapping) операционной системы посредством функций mmap. При такой организации глобалы с именем, например, DIs имеет тот же размер и содержание, что и файл db/<project>_DIs.xxx. Вместо <project> должно подставляться имя конкретного проекта. Сохранение пользовательских данных глобала осуществляется при вызове функций синхронизации памяти с имеющимся файлом *.xxx.

Для того, чтобы при последующей загрузке данный глобал получал свои начальные данные из маппированного к нему файла *.xxx, нужны определенные условия:

- Файл *.xxx для данного глобала должен, как минимум, присутствовать;
- Размер файла *.xxx для маппирования должен быть идентичен размеру создаваемого глобала;
- Различные наборы начальных состояний в диапазоне 1-99 создаются из файлов *.x01 - *.x99.

DsPersnap является программой, которая осуществляет сохранение и восстановление состояний данных глобалов в разделяюмую память. Программа DsPersnap обеспечивает персистентность данных: время жизни данных состояний глобалов больше времени жизни программных модулей, их использующих.

Программа DsPersnap работает только с теми глобалами, для которых задан параметр persist_interval=-1 в конфигурации, например, GLOBAL[].persist_interval=-1. Остальные глобалы не сохраняют свое значение.

DsPersnap, как и все другие задачи, периодически отслеживает статус задачи. При Task:@T.task_status = END осуществляется окончательный сброс без проверок изменения данных в файлы *.xxx. Поэтому программа DsPersnap обычно финиширует после окончания работы других программ, сбрасывая данные в файлы.

Алгоритм сохранения состояния иллюстрирует рисунок 11.

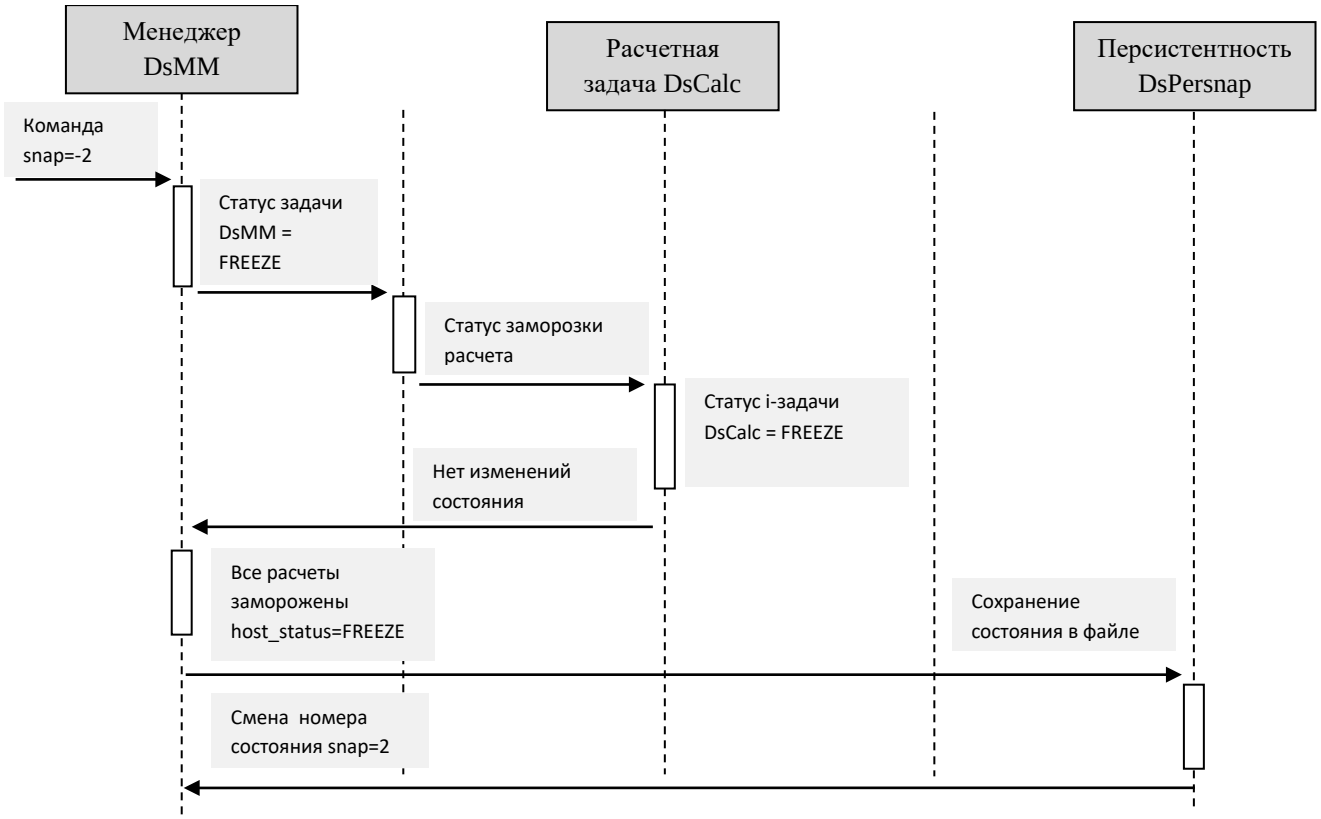

Рисунок 11 – Сохранение состояний командой snap

Запуск алгоритма командой snap N (N – номер начального условия 1-99) устанавливает переменную Host:@H.snap_no отрицательным значением –N. По отрицательному значению менеджер моделей устанавливает свой статус Task:#0.task_status значением FREEZE. Далее каждая расчетная задача устанавливает свой статус во FREEZE, равный статусу менеджера моделей. После этого статус всего узла становится равным Host:@H.host_status = FREEZE. Как только программа персистентности фиксирует статус хоста = FREEZE, она считает, что данные заморожены, и можно сбрасывать значения из разделяемой памяти в файл. При N=2 и бинарных файлах мэппинга сброс осуществляется в файлы db/<project>_<global>s.x02. В конфигурации может быть задан сброс в текстовый файл. В таком случае имя файла сброса будет db/<project>_<global>s.t02.

При сбросе командой snap фиксируются лишь изменения значений переменных системы моделирования. Текущие значения статических параметров, равно как и команды оператора, сбросу не подлежат.

В случае успешного завершения сброса в переменную Host:@H.snap_no устанавливается номер начального состояния N. При неуспешном завершении сброса

(например, вследствие отсутствия места на диске) в переменную Host:@H.snap_no записывается 0, а код ошибки – в лог системных событий.

Алгоритм восстановления иллюстрирует рисунок 12.

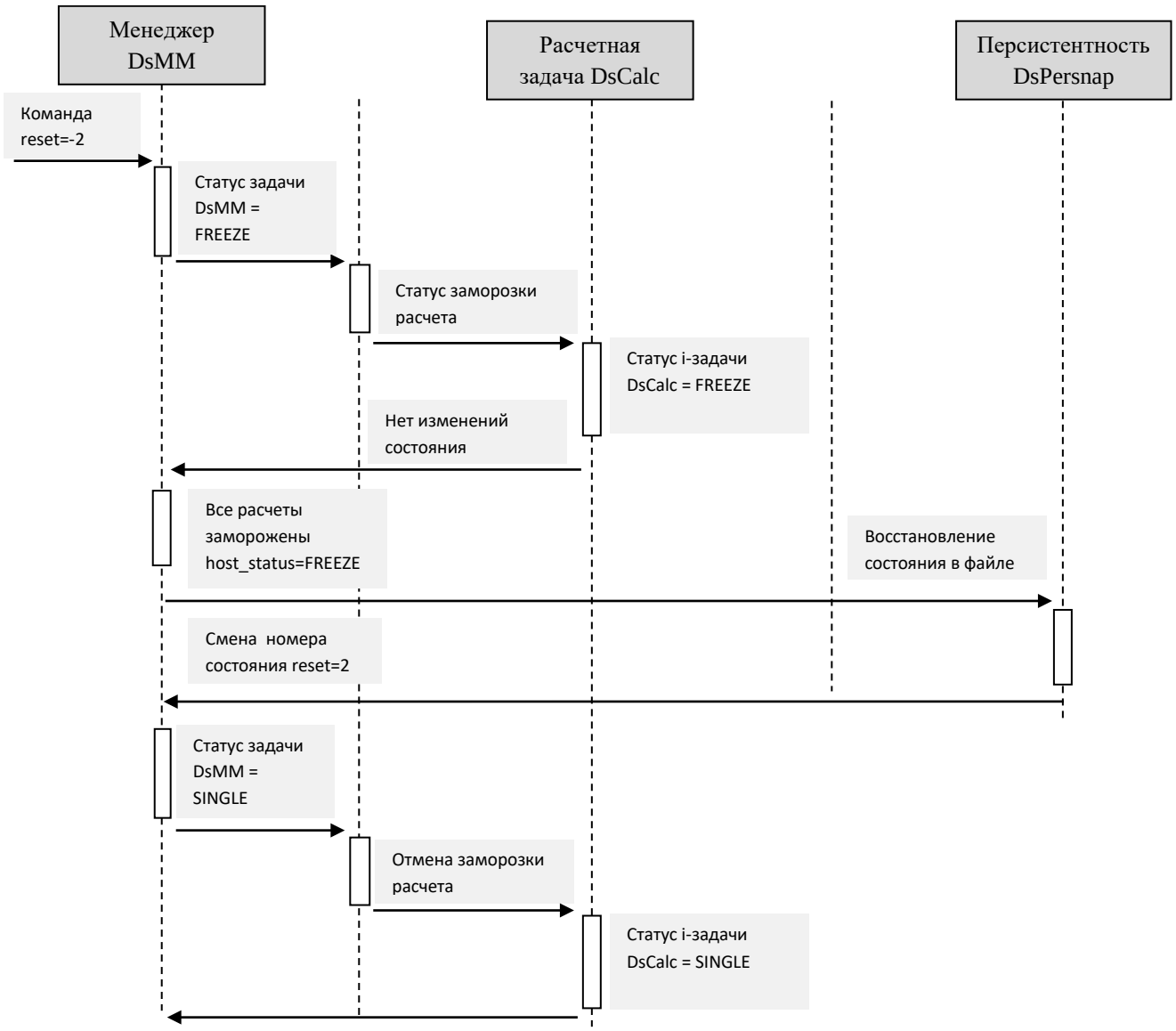

Рисунок 12 – Восстановление состояний командой snap

Запуск алгоритма командой reset N (N – номер начального условия 1-99) устанавливает переменную Host:@H.reset_no отрицательным значением –N. По отрицательному значению менеджер моделей устанавливает свой статус Task:#0.task_status значением FREEZE. Далее каждая расчетная задача устанавливает свой статус во FREEZE, равный статусу менеджера моделей. После этого статус всего узла становится равным Host:@H.host_status = FREEZE. Как только программа персистентности фиксирует статус хоста = FREEZE, она считает, что данные заморожены, и можно восстанавливать значения из файл в разделяемую память. При N=2 и бинарных файлах мэппинга восстановление осуществляется из файлов db/<project>_<global>s.x02. В конфигурации может быть задан сброс в текстовый файл. В таком случае имя файла восстановления будет db/<project>_<global>s.t02.

В случае успешного завершения восстановления в переменную Host:@H.reset_no устанавливается номер начального состояния N. При неуспешном завершении восстановления (например, вследствие отсутствия файлов требуемого состояния на диске) в переменную Host:@H.reset_no записывается 0, а код ошибки – в лог системных событий.

После завершения восстановления менеджер моделей устанавливает статус задачи DsMM в Task:#0.task_status=SINGLE. Это означает отмену заморозки расчетов. Статусы расчетных задач DsCalc становятся в Task:DsCalc.task_status=SINGLE. По окончании этого Host:@H.host_status устанавливается в SINGLE.

При наличии в номере начального состояния флага RESET_FLG_FRZ=1000 переход в статус SINGLE после отмены заморозки не производится. Данный флаг также используется при смене статуса вычисление-заморозка (SINGLE-FREEZE) безотносительно восстановления состояния. При команде Host:@H.reset_no := -RESET_FLG_FRZ производится заморозка расчетов. При команде Host:@H.reset_no := 0 происходит восстановление расчетов. Переход узла в статус FREEZE не означает останов работы. Это означает останов вычислительной функции расчетных задач.

Для отслеживания работы DsPersnap нужно устанавливать уровень трассировки (например, Sb.TRAL_ERRORS, Sb.TRAL_RUNTIME, Sb.TRAL_AUX, Sb.TRAL_DATA согласно приложению 6). Это можно сделать, установив значение Task:DsPersnap.trace_level в отладчике DsISD. Тогда в логе системных событий Syslogs появятся информационные сообщения вида:

```
      10/02/25 16:28:48.085 WARNING 442286# DsPersnap 1@0
DsrmPersnap.OnReadObj Snap size 204 <> 208, resnap advised

      10/02/25 16:28:48.085 INFO 442287# DsPersnap 1@0 DsrmPersnap.ReloadT
HEAs global changes:0 file:db/model_HEAs.t02 res=0()

      10/02/25 16:28:48.085 INFO 442288# DsPersnap 1@0 DsrmPersnap.SaveAll
ResetT cmd:-1002 reset_no:1002
```

Это означает, что модуль DsPersnap периодически сбрасывает данные в файлы сохранения для глобалов AI, DI.

Параметр Const:pers_sync_mode управляют процессом сохранения и восстановления состояний. Биты параметра устанавливают механизмы сохранения. В зависимости от значений этих битов происходит сохранение:

- MODE_SNAP_RESET_T:4 – состояние записывается и востанавливается в текстовых файлах db/<project>_<global>s.t<number>. Все изменения значений переменных относительно нулевых величин сохраняются в тексте. При восстановлении все значения и команды обнуляются. Сохраненные значения восстанавливаются из файлов;
- MODE_SNAP_VAL_T:5 – состояние записывается и востанавливается в виде значений переменных. Все изменения значений переменных относительно нулевых величин сохраняются в тексте. При восстановлении все значения и команды обнуляются. Сохраненные значения восстанавливаются из файлов;

- MODE_SNAP_VAL_T:5 – состояние записывается и востанавливается в виде значений параметров.

3.5.6 Оболочка расчетных моделей

Программы DsModel представляют собой оболочки плагинов расчетных модулей. Каждая задача DsModel работает с несколькими плагинами, работающими с задачами объектов определенных типов, расположенных в базе данных реального времени. Например заданный в конфигурационном файле ниже плагины DpgcValves, DpgcCNValves, DpgcPump загружаются и выполняются задачей DsModelGC.

```
TASK[2].name=DsModelGC
TASK[2].exe_path=DsModel
TASK[2].command_line=-hl DpgcValves,DpgcCNValves,DpgcPumps
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
```

Флаг –hl ИмяМодуля подключает плагины обработки расчетных при инициализации. Модуль должен соответствовать имени динамически загружаемой программы. В приведенном примере DpgcValves подключает плагин модели запорной арматуры, DpgcCNValves плагин модели регулирующего клапана, DpgcPump подключает плагин обработки насоса.

В рамках фазы инициализации подключаются глобалы для типов арматуры, клапанов и насосов и своя очередь расчетных данных обработки. Очередь расчетных данных обработки ассоциируется с данной задачей DsModel в конфигурационном файле.

```
GLOBAL[4].globtype=q
GLOBAL[4].globname=GCq
GLOBAL[4].max_count=2000
GLOBAL[4].reftask=DsModelGC
GLOBAL[4].run_at=STDSERVER
```

Очередь расчетных данных обработки GCq представляет собой глобал q, привязанный к задаче вида DsModel. В приведенном примере привязка к задаче с именем DsModelGC осуществляется установкой параметра reftask= DsModelGC.

Работа DsModel после инициализации происходит в циклическом режиме. Входными данными являются переменные табличных глобалов n, причем только тех, которые состоят из объектов типа, равного типу плагина. Параметр usr_data заголовка пользовательского глобала n является индексом в таблице типов Types, т.е. является типом объектов глобала.

Циклический механизм RefreshWRAP вызывается периодически для глобалов n с указанным типом объекта. Циклическое обновление расчетных данных плагином обработки осуществляется при наличии в плагине флагов OVHW_CYCLIC, OVHW_CYCLIC_ALL поддержки циклического режима. В рамках циклического механизма вызывается обработчик глобала. Обработчик вызывает функции обработки из плагина для каждого объекта глобала. Это происходит периодически, с циклом обновления в миллисекундах задачи Task:@T.update_ms, заданной в конфигурации.

Флаг OVHW_CYCLIC_ALL означает необходимость вызова метода обновления всей модели перед вызовом обновлений для каждого объекта. В реализации метода разработчик не ограничен в выборе способов обращения к переменным процесса.

При установке статуса Host:@H.host_status := FREEZE (через значение состояния флага RESET_FLG_FRZ=1000, описанное в предыдущем разделе) происходит заморозка вычислений. При такой заморозке циклический механизм RefreshWRAP функционирует, но выполнение расчетных задач не производится. При восстановлении расчетов выполнение задач возобновляется. Переход узла в статус FREEZE не означает останов работы DsModel. Модуль периодически обновляет параметры рантайма: Task:@T.task_status, Task:@T.current_time. FREEZE означает останов вычислительной функции расчетных задач.

Кроме этого, в расчетных модулях DsModel реализован механизм пошагового выполнения step N, где N означает число шагов циклического обновления (Рисунок 13). Команда step выполняется из состояния status := FREEZE. Установка команды означает запись в переменную Host:@H.n_steps значения –N. При послуплении этой команды статус задачи DsMM устанавливается в SINGLE, а переменная Host:@H.n_steps получает значение N. По положительному значению расчетные задачи получаю сигнал начала работы и обновляют ровно n_steps шагов, отмечая это во внутренней переменной stepCounter. По истечении n_steps шагов задача переводится в состояние Task:@T.task_status=BACKUP и становится в заморозку. Когда все задачи станут заморожены, то статус DsMM и статус хоста станут status := FREEZE.

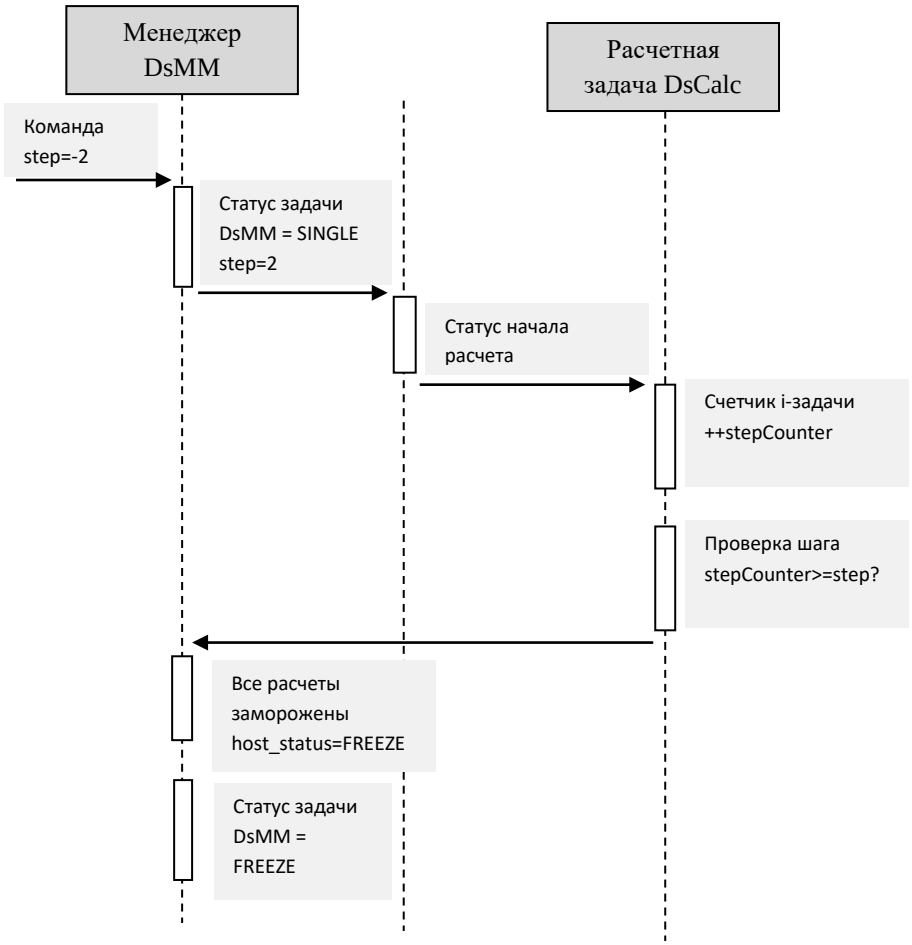

Рисунок 13 – Пошаговое обновление командой step

Очереди расчетных данных обработки (GCq в примере выше) содержат последовательность изменений расчетных значений ассоциированной задачи.

3.5.7 Обновление табличных данных узлов по очередям

Программы DsDVP осуществляют обновление табличных данных узлов по очередям циклических буферов сообщений. Программа DsDVP не взаимодействует с удаленными абонентами по сети, в качестве входных данных используются очереди сообщений, сформированные другими программами, как, например, DsModel.

Общая схема обновления данных по очередям представлена на рисунке 14. Входные очереди сообщений содержат последовательности изменений входных данных. Данные последовательности могут содержать как переменные с временными метками, так и просто значения. В приведенном примере на рисунке входными являются команды управления оборудованием.

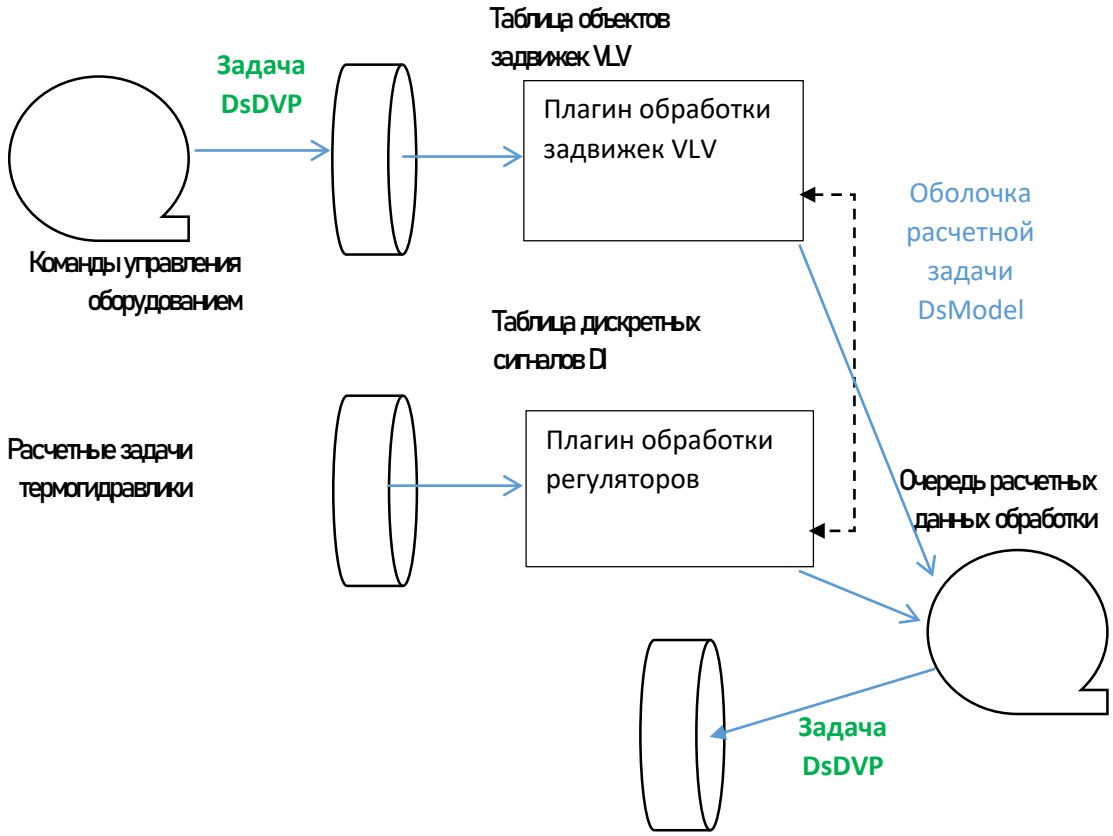

Рисунок 14 – Табличные данные и очереди

Данные входных очередей отслеживаются задачей DsDVP в постоянном режиме и поступают в соответствующие таблицы глобалов n. Изменения поступают в таблицы объектов моделей, обновляя значения последних. Циклически в задачах DsModel вызываются функции обработчики, подключенные в виде плагинов. В результате обработки выходные расчетные данные обработки записываются в выходные очереди сообщений.

Выходная очередь сообщений также отслеживается DsDVP, обновляя значения в глобалах n. Расчетные значения поступают в глобалы аналоговых AI и дискретных DI сигналов, имитирую устройство связи с объектом системы верхнего уровня. По значениям переменных глобалов AI и DI определяются величины соответствующих технологических параметров и состояний оборудования. Например, в примере рисунка выше выходными расчетными параметрами будут состояние и концевые выключатели задвижки, положение регулирующего клапана и регулируемая величина.

Работа DsDVP после инициализации происходит периодически. Событийный механизм осуществляет проверку наличия входных сигналов UpdateDVP с минимальным циклом обновления. Циклический механизм RefreshDVP выполняется с циклом обновления в миллисекундах задачи Task:@T.update_ms, который установлен в конфигурации для данной задачи.

Событийных механизм UpdateDVP периодически считывает новые данные, появившиеся в глобалах q данной группы. При наличии таких данных производится попытка внести эти данные как изменения в табличные глобалы.

Циклический механизм RefreshDVP вызывается периодически для глобалов n. Циклическое обновление расчетных данных плагином обработки осуществляется задачей DsModel.

3.5.8 Групповое отображение данных

Программа DsGroup реализует подготовку данных для группового отображения. Программа DsGroup не взаимодействует с удаленными абонентами по сети и не использует механизмы отслеживания входных очередей сообщений.

В рамках фазы инициализации подключается своя очередь групповых данных отображения. Очередь групповых данных отображения ассоциируется с данной задачей DsGroup в конфигурационном файле.

```
GLOBAL[7].globtype=q
GLOBAL[7].globname=Groupq
GLOBAL[7].max_count=2000
GLOBAL[7].reftask=DsGroup
GLOBAL[7].in_groups=SYNC SYNC2 DVP
```

Для понимания структуры и состава групповых данных отображения необходимо представление тех экранных областей, в которые данные группы отображаются, рисунок 15.

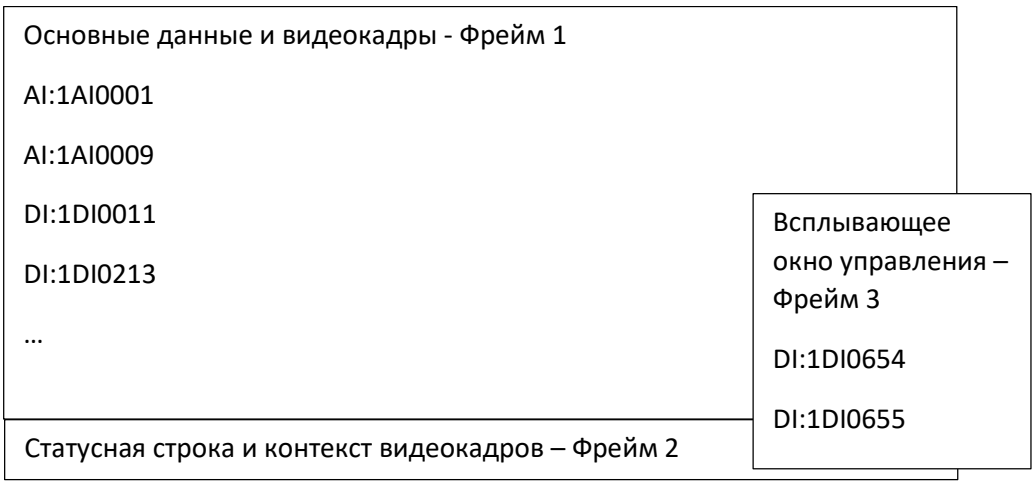

Рисунок 15 – Представление групповых данных в областях

Программа СВУ запускается с отображением видеокадров во фреймах. Часть видеокадров основана на группах. Групповые данные нумеруются слева направо – сверху вниз и подготавливаются программой DsGroup.

Фреймы представляют собой прямоугольные области, на которые делится экран. Фреймы могут быть всплывающими, такие фреймы используют для запуска окон управления оборудованием, окон подробностей.

Фрейм 1 на рисунке представляет собой основную экранную область, в которую выводятся видеокадры. Данные основного фрейма выводятся в табличном виде, т.е.

представляют собой части групп, видимые в экранной области. Размер видимой части группы должен быть достаточным, чтобы разместить видимые в основном видеокадре данные. Реально из соображений эргономики на одном видеокадре может разместиться не более 100 элементов групп.

Фреймы статусной строки и обобщенной сигнализации (фрейм 2) также используют групповое отображение. Окна управления использует меньшее количество данных, чем основная экранная область. Конкретная конфигурация является проектно-зависимой.

3.5.9 Регистрация и управление сигнализациями

Программа DsAlarm реализует функции регистрации и управления сигнализациями. Программа DsAlarm в качестве входных данных используются очереди сообщений, сформированные другими программами, как, например, DsXServ или DsModel.

События, связанные с отклонениями в параметрах процесса или оборудования обрабатываются программой DsAlarm в оперативном режиме. Сформированные в результате обработки данные записываются в массивы сигнализаций, состоящие из объектов Alarm.

Данные из групп сигнализаций поступают в ячейки группового отображения, программа DsGroup считывает их постоянно с циклом обновления при задании - every_cycle T.

3.5.10 Сервер обмена

Программы DsXServ представляет собой сервер обмена. Сервер обмена осуществляет прием/передачу данных, внешних по отношению к системе управления моделями технологического процесса.

Данные от внешней подсистемы передаются агентом обмена. Таким образом, сервер обмена входит в состав ПО управления моделями технологического процесса, а агент обмена является внешним по отношению к системе управления моделями модулем.

Агент обмена A-1 передает данные только на сервер обмена 1. Аналогично агент обмена 1 получает данные только от сервера обмена 1. Такая конфигурация представлена на рисунке 16.

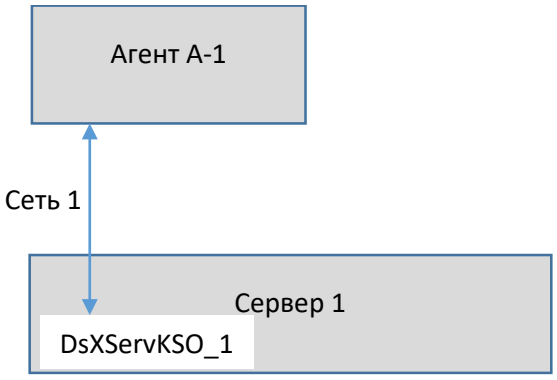

Рисунок 16 – Сервер обмена и агент обмена

Данная конфигурация обмена не пересекается с конфигурацией других сервера и агента обмена. Состояние в время жизни сервера обмена контролируется задачей DsMM менеджера моделей. Состояние агента обмена контролируется средствами опрерационной системы удаленного хоста.

3.6 СРЕДСТВА СВЯЗИ С ВНЕШНИМ ТРЕНАЖЕРОМ С МОДЕЛЯМИ ТЕХНОЛОГИЧЕСКОГО ПРОЦЕССА

3.6.1 Структурная схема соединения СКОЯТ - тренажер

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 (рисунок 17).

Обмен данными будет осуществляться по сети, для этого необходимо установить внешние сетевые соединения для контейнеров и определить конфигурацию точек обмена. Со стороны контейнера модели будет задействован сервер обмена. Со стороны контейнера тренажера будет задействовано ПО средств коммуникации зонной архитектуры (СКЗА), реализованное в виде библиотеки.

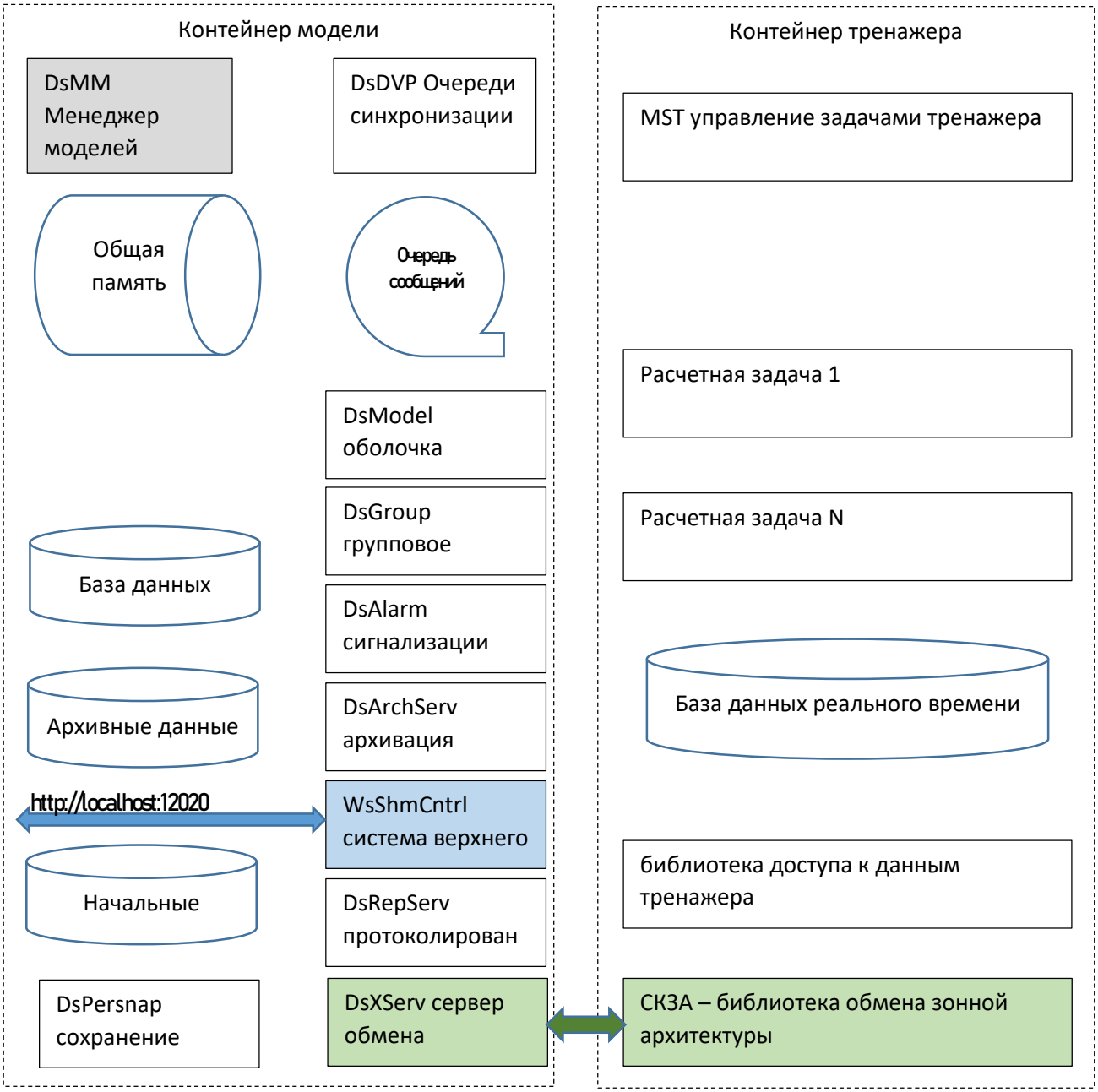

Рисунок 17 – Структурная схема соединения с тренажером

3.6.2 Основные функции ПО связи

Для реализации обмена данными между контейнером модели и контейнером тренажера используется ПО СКЗА – средства коммуникации зонной архитектуры. СКЗА применяется для интеграции систем, работающих в режиме онлайн и реализующих функции получения, обработки, управления, архивирования и представления данных оператору.

Основные функции СКЗА:

- установление данных в виде независимых коммуникационных полей;
- приема данных и команд от удаленных абонентов;
- передачу данных и команд удаленному абоненту;
- конфигурирование в двухточечной и трехточечной схемах связи;
- независимое конфигурирование характеристик связи;
- сопоставление и контроль идентификаторов с зонными номерами;
- трассировка передаваемых данных.

Конфигурирование ПО СКЗА осуществляется на основе протокола TCP и соединения 1 сервер – 1 агент. Библиотека СКЗА осуществляет выдачу значений и прием команд от сервера обмена DsXServ с плагином СКЗА.

Программное обеспечение связи начинает функционировать после загрузки контейнера тренажера и обменивается данными с сервером контейнера модели.

3.7 СИСТЕМА КОНТРОЛЯ ДЕЙСТВИЙ ОБУЧАЕМОГО

3.7.1 Структура системы контроля действий обучаемого

Сервис WoMark совмещает функции следующих подсистем:

- подсистемы архивирования действий обучаемого и данных технологического процесса;
- подсистемы оценивания действий обучаемого;
- подсистемы управления и мониторинга предоставления ресурсов.

Структура системы представлена на рисунке 18.

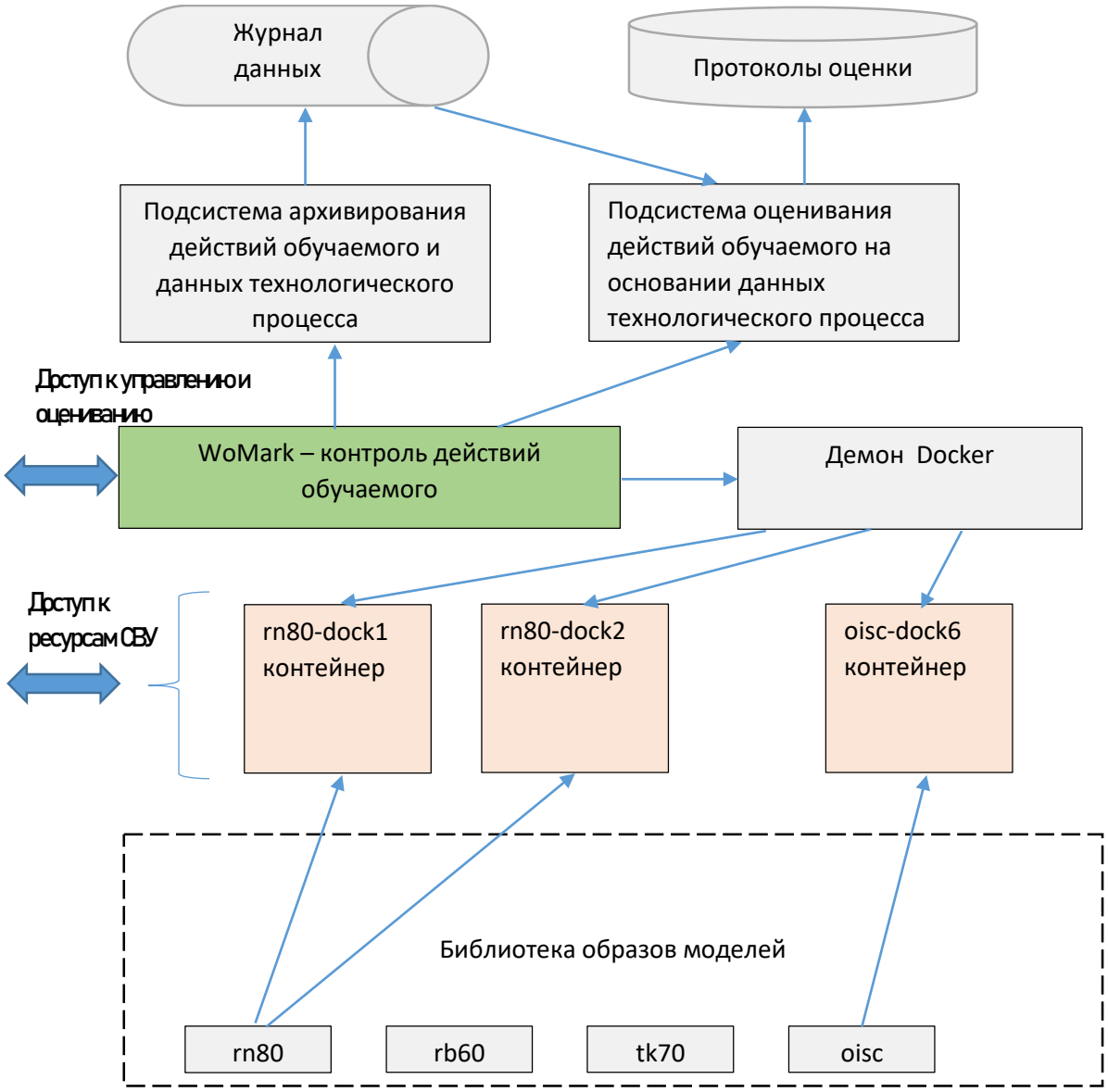

Рисунок 18 – Структурная схема системы контроля действий обучаемого

Сервис WoMark интегрирует и управляет функциональностью подсистем. Как программная единица он взаимодействует с демоном Docker и создаваемыми им контейнерами из библиотеки образов моделей.

Подсистемы программно интегрированы в сервис и описываются ниже.

3.7.2 Подсистема архивирования действий обучаемого и данных технологического процесса

Подсистема архивирования реализует онлайн-фиксацию данных при прохождении заданий и лабораторных работ.

Архивные данные включают в себя действительные, целочисленные, булевские значения и записи команд оператора. Кроме этого, в архивные данные входят выборки идентификаторов технологического оборудования и датчиков. Данные выборки используются как ответы на вопросы и соотносятся с конкретными группами (как, к примеру, аварийные и предупредительные отклонения на видеокадре).

Из текущих значений технологических сигналов фильтруются подлежащие оцениванию согласно списку идентификаторов. Такие значения заносятся в группу VARIABLE_VALUES. Изменения значений, поступаемые из модели, перенаправляются в сервис WoMark.

```
VARIABLES_VALUES=dict()
VARIABLES_VALUES['19_02_2025__16_00_48_127']=['',
[('DI:~40RNR92AS003_Z0.dv','1;48;1739970014','I2QTS')]]
VARIABLES_VALUES['19_02_2025__16_00_48_127']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.58','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_48_127']=['',
[('QUALITY,AI:~40RNR92CL901_XQ57.av','53','I2')]]
VARIABLES_VALUES['19_02_2025__16_00_48_632']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.52','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_49_650']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.41','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_50_668']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.61','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_51_696']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.43','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_52_719']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.79','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_53_733']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.82','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_54_649']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.74','R4')]]
VARIABLES_VALUES['19_02_2025__16_00_54_726']=['',
[('DI:~40RNR92AS003_OU.dcmd','2','I2')]]
```

Типичный пример представленного набора значений содержит величины аналоговых и дискретных сигналов AI:…av, DI:…dv, команды управления оборудованием DI:…dcmd.

Данные выборки в конкретные группы записываются в структуры данных OUTPUT_SELECTION.

```
OUTPUT_SELECTION=dict()
```

Типичный пример представленного набора выборок содержит списки идентификаторов аварийных и предупредительных превышений и снижений. Какого вида должны быть выборки, будет определено конкретным заданием учебно-методического материала.

Архивные данными обрабатываются средствами запроса и анализа. Запрошенные архивные данные используются системой оценивания действий обучаемого при формировании результата и выставлении оценки.

Сохранение архивных данных производится в файлах сессий и доступно в течении сессии работы. Периодическое удаление старых архивных данных определяется настройками конфигурации WoMark.

3.7.3 Подсистема оценивания действий обучаемого на основании данных технологического процесса

Подсистема оценивания действий обучаемого основана на обработке заархивированных протоколов, полученных при ответах или лабораторных занятиях обучаемого.

Подсистема оценивания действий обучаемого реализует контекстный анализ протоколов, выработку результата выполнения задания, выставление оценки. Результаты процеденного контекстного анализа фиксируются в системе для возможной последующей выдачи.

Набор правил оценки привязывается к технологическим кодам оборудования и временным отношениям между происходящими событиями.

В простом случае, например, при невыполнении задания, выводится информация для представлению пользователю.

```
/usr/bin/python3 WoEval results/dvdagaev-department-a1-19_02_2025__16_35_10_811.pdt ./labs/Sc102
{"comments": "Уровень - 114.57 см. Нормализация уровня не достигнута", "result": "2"}
```

Более сложные случае представляют отчет с детализацией действий и более детальными оценками проведенных работ.

Прикладное программное обеспечение содержит скрипты обработки архивных данных технологического процесса и получения результатов.

3.7.4 Подсистема управления и мониторинга предоставления ресурсов

Предоставление ресурсов обучаемому или группе обучаемых осуществляется в виде доступа к контейнерам, содержащих технологические модели и СВУ.

Функционально это осуществляется посредством управления подпиской и выполнения команд сервиса WoMark. Доступ получает веб-сокет клиент на стороне браузера с поддержкой JavaScript. Доступ предоставляет веб-сокет сервер WoMark и передает информацию клиенту по коммуникационному протоколу. Логическая составляющая запросов коммуникационного протокола описана в приложении 7.

При установлении соединения клиент получает возможность обмена сообщений с сервисом WoMark согласно таблице 7-1. При начале работы в первую очередь устанавливается категория пользователя системы сообщением SetCategory. В зависимости от разрешенного списка для данного пользователя устанавливается категория текущей сессии, рисунок 19.

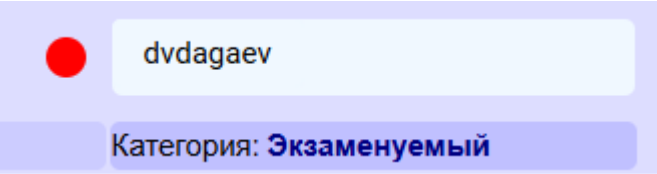

Рисунок 19 – Установленная категория экзаменуемого

В примере на рисунке пользователю установлена категория Examinee – Экзаменуемый (команда SetCategory).

Далее идет проверка предоставленного по данной сессии ресурса командой Getload. Данная команда проверяет состояние ресурса и выдает информацию о нем пользователю.

Состояния ресурсов определяются конечным автоматом, соответствующим рисунку 20.

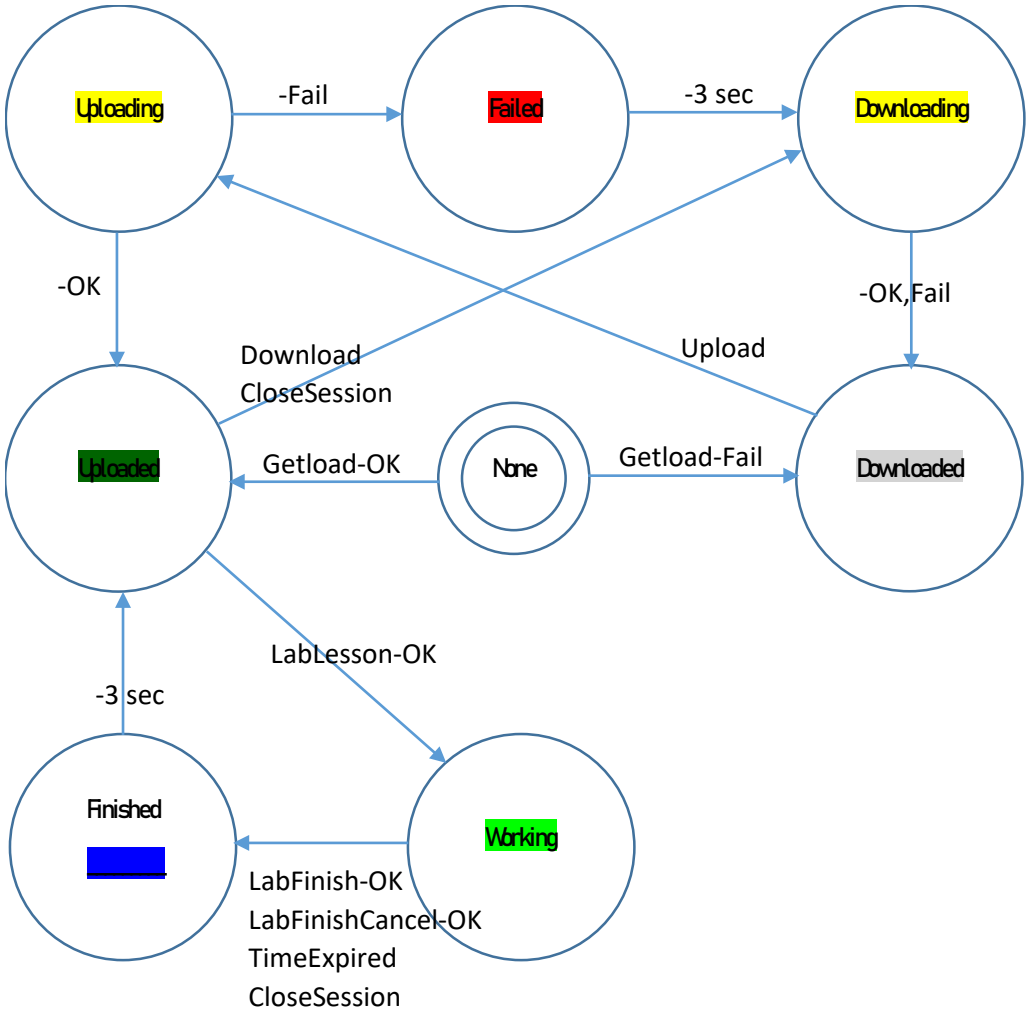

Рисунок 20 – Граф переходов состояний ресурсов

Состояние ресурса передается от сервиса клиенту в поле LStatus структуры SubOutputMessage в таблице 7-4. Первоначально состояние равно None. Если контейнер загружен, состояние становится Uploaded, если контейнер выгружен, состояние становится Downloaded.

При выполнении команды Download состояние из загруженного Uploaded (темно-зеленое) переходит в выгружаемое состояние Downloading (желтое) и далее в выгруженное Downloaded (серое).

При выполнении команды Upload выгруженное состояние Downloaded (серое) становится Uploading (желтое). Далее при успешной загрузке состояние переходит в Uploaded (темно-зеленое). При неуспешной загрузке состояние переходит в Failed (красное), а через 3 сек – в Downloading (желтое) и далее в Downloaded (серое).

Выполнение лабораторной работы возможно только из загруженного состояния. Подпиской LabLesson инициируется выполнение лабораторной работы. При этом состояние переходит из загруженного Uploaded (темно-зеленое) в рабочее Working (светло-зеленое). По окончании работы состояние переходит в завершенное Finished (синее) и далее, через 3 секунды, в загруженное Uploaded (темно-зеленое).

Изменение цвета состояния отображается пользователю в информационном фрейме и фрейме выбора задания. При этом, цвета отображения состояний соответствуют цветам графа переходов.

Рисунок 21 иллюстрирует это цветовое соответствие.

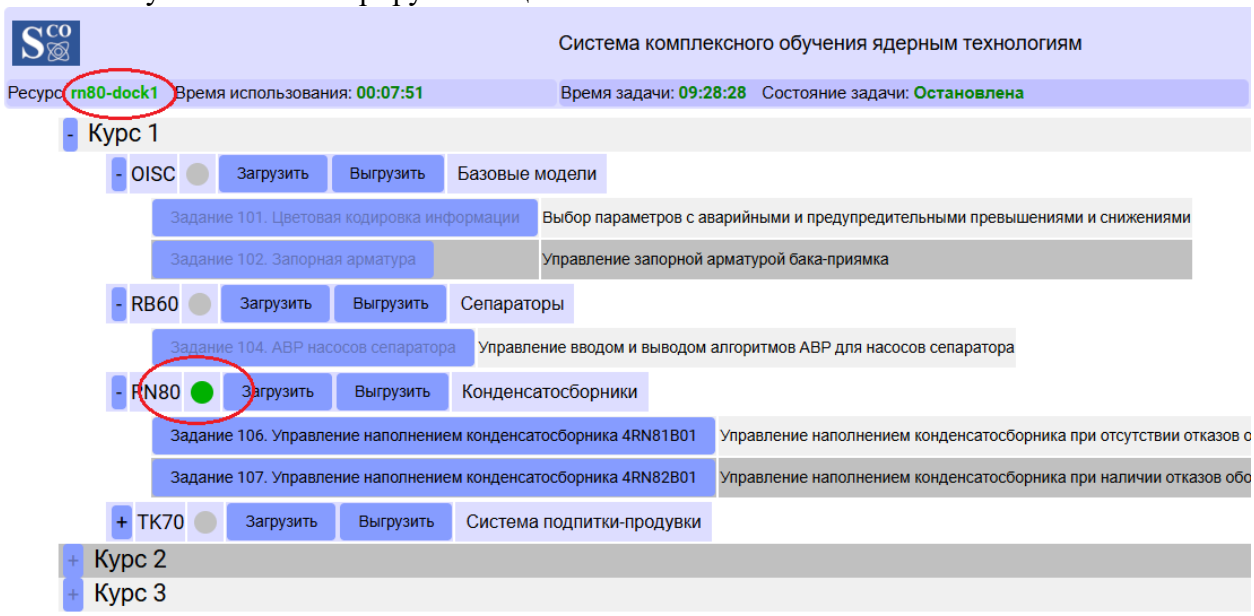

Рисунок 21 – Цвета состояний пользователя

Отмеченные на рисунке цвета загруженного состояния выведены в информационном фрейме вверху и фрейме выбора задания в центре в левой части.

При выполнении задания вместо фрейма выбора появляется содержательная часть задания. При этом цветовое отображение имени контейнера в верхней части будет менять цвет согласно графу переходов состояний.

При завершении работы системы сервисом WoMark отслеживаются события о закрытии сессии пользователем CloseSession. Данное событие приводит к автоматическому переходу в состояние Downloading с последующей выгрузкой используемого ресурса.

При отсутствии действий пользователя в системе в течении 1 часа (время устанавливается настройками) также осуществляется автоматический переход в состояние Downloading.

3.8 СИСТЕМА УЧЕТА ИСПОЛЬЗОВАННЫХ РЕСУРСОВ

3.8.1 Фиксация индивидуальных и групповых ресурсов

Сервис WoMark осуществляет фиксацию времени загрузки и выгрузки ресурсов загрузчиком. При установленной трассировке загрузка контейнера будет сопровождаться сообщениями следующего вида.

```
2025/02/25 08:56:17 docker resource dr1_970 calling Upload
2025/02/25 08:56:17 docker resource dr1_970 lstatus changed to: uploading, section:rn80
2025/02/25 08:56:17 Return method=Upload inp={0 rn80   } out={0 Upload  dock1 1 ok}
2025/02/25 08:56:17 docker uploading {dr1_970 model-rn80:v1.5d model-rn80-dock1 1 rn80} from:
2025/02/25 08:56:17 docker container create:model-rn80-dock1 image:model-rn80:v1.5d
```

```
2025/02/25 08:56:17 container created: model-rn80-dock1
{c79f3646908538edbb386f87f72e86da143ecb3d643b29cb70caf6d9819169fa []}
resource_id: 1
2025/02/25 08:56:17 container bind: 0.0.0.0:12021->12020
2025/02/25 08:56:17 docker container start: model-rn80-dock1
2025/02/25 08:56:17 ? container ping do: model-rn80-dock1 Get
"http://localhost:12021": read tcp 127.0.0.1:40964->127.0.0.1:12021: read:
connection reset by peer
2025/02/25 08:56:20 ? container ping do: model-rn80-dock1 Get
"http://localhost:12021": read tcp 127.0.0.1:40974->127.0.0.1:12021: read:
connection reset by peer
2025/02/25 08:56:23 ? container ping do: model-rn80-dock1 Get
"http://localhost:12021": read tcp 127.0.0.1:55290->127.0.0.1:12021: read:
connection reset by peer
2025/02/25 08:56:26 container ping ok: model-rn80-dock1 &{200 OK 200 HTTP/1.1
1 1 map[Content-Length:[321] Content-Type:[text/html; charset=utf-8] Date:[Tue,
25 Feb 2025 05:56:26 GMT] Last-Modified:[Tue, 25 Feb 2025 05:56:17 GMT]]
0xc0006d33c0 321 [] false false map[] 0xc00098a7e0 <nil>}
2025/02/25 08:56:26 return uploaded rn80 {dr1_970 model-rn80:v1.5d model-rn80-
dock1 1 rn80}
2025/02/25 08:56:27 docker resource dr1_970 lstatus changed to: uploaded,
section:rn80
```

Загрузка начинается с приема команды Upload. Устанавливается статус Uploading, передается команда создания контейнера из образа docker. По созданию в лог пишется сообщение вида docker container create: model-rn80-dock1. Это означает, что создан контейнер dock1 для секции rn80. После создания контейнера производится его старт, сопровождающийся сообщением docker container start: model-rn80-dock1. Старт контейнера означает, что начали инициализацию приложения, расположенные внутри контейнера. В течении этого времени сервис WoMark посылает посылки ping, проверяющие окончание инициализации приложений. По окончании инициализации устанавливается статус Uploaded загружено с выводом сообщения вида return uploaded rn80.

Аналогично, выгрузка контейнера будет спровождаться сообщениями следующего вида.

```
2025/02/25 08:58:53 docker resource dr1_970 calling Download
2025/02/25 08:58:53 docker resource dr1_970 lstatus changed to: downloading,
section:rn80
2025/02/25 08:58:53 Return method=Download inp={0 rn80 } out={0 Download
dock1 1 ok}
2025/02/25 08:58:53 docker downloading {dr1_970 model-rn80:v1.5d model-rn80-
dock1 1 rn80} from:
2025/02/25 08:58:53 docker container stop:
{c79f3646908538edbb386f87f72e86da143ecb3d643b29cb70caf6d9819169fa [/model-
rn80-dock1] model-rn80:v1.5d
sha256:2756e760f6d271d314d7a6c21a4aa7b3a36c70ab6c8edb32358d5057700285d8
/bin/sh -c 'DsMM -cd /home/osco' 1740462977 [{0.0.0.0 12020 12021 tcp}] 0 0
map[] running Up 2 minutes {default map[]} 0xc00011a358 [{bind
/home/user/osco/Dsmodel /home/osco/Dsmodel true rprivate} {bind
/home/user/osco/mweb /home/osco/web true rprivate} {bind /usr/local/scadi
/usr/local/scadi true rprivate} {bind /usr/local/scadiweb
/usr/local/scadiweb true rprivate} {bind /home/user/osco/Dpgc
/home/osco/Dpgc true rprivate} {bind /home/user/osco/Dpmodel
/home/osco/Dpmodel true rprivate}]}
2025/02/25 08:59:03 docker container remove:
{c79f3646908538edbb386f87f72e86da143ecb3d643b29cb70caf6d9819169fa [/model-
rn80-dock1] model-rn80:v1.5d
sha256:2756e760f6d271d314d7a6c21a4aa7b3a36c70ab6c8edb32358d5057700285d8
/bin/sh -c 'DsMM -cd /home/osco' 1740462977 [{0.0.0.0 12020 12021 tcp}] 0 0
map[] running Up 2 minutes {default map[]} 0xc00011a358 [{bind
```

```
/home/user/osco/Dsmodel   /home/osco/Dsmodel         true   rprivate}   {bind
/home/user/osco/mweb /home/osco/web    true rprivate} {bind  /usr/local/scadi
/usr/local/scadi          true   rprivate}   {bind       /usr/local/scadiweb
/usr/local/scadiweb         true   rprivate}   {bind      /home/user/osco/Dpgc
/home/osco/Dpgc         true   rprivate}   {bind      /home/user/osco/Dpmodel
/home/osco/Dpmodel   true rprivate}]}
2025/02/25 08:59:03 return downloaded rn80 {dr1_970 model-rn80:v1.5d model-
rn80-dock1  1 rn80}
2025/02/25 08:59:03 docker resource dr1_970 lstatus changed to: downloaded,
section:rn80
```

Выгрузка начинается с приема команды Download. Устанавливается статус Downloading, передается команда останова контейнера docker container stop: model-rn80-dock1. По останову контейнера начинается выполнение команды удаления контейнера docker container remove: model-rn80-dock1. По окончанию удаления устанавливается статус Downloaded загружено с выводом сообщения вида return downloaded rn80.

Информация о времени начала, времени конца и инициаторе загрузки передается в базу данных для возможности дальнейшего доступа.

Кроме этого, сервис динамически отслеживает время использования ресурса каждой сессии (равное разнице этих времен), рисунок 22.

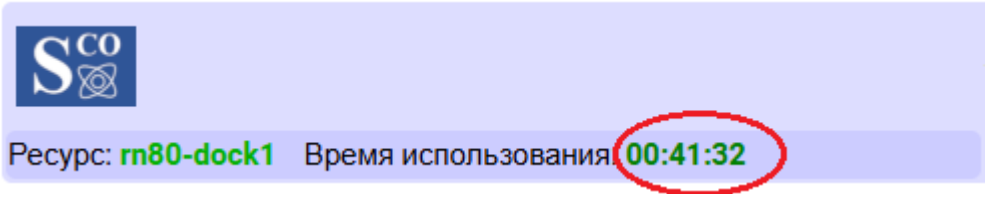

Рисунок 22 – Время использования ресурса для сессии

Все автоматические завершения сессий и переходы в состояние Downloading приводят к фиксации завершения использования указанного ресурса.

3.8.2 Доступ к данным об использовании ресурсов

Сервис WoData реализует алгоритмы доступа к данным об использовании ресурсов пользователями. Данные об использовании содержат информацию о пользователе, сессиях работы, использованных ресурсах в сессиях, характеристиках доступа к ресурсам.

Число пользователей, временной интервал хранения и степень детализации определяются настройками конкретного проекта.

При соединении с сервисом в него передается структура данных ScadiID (таблица 2-2 приложения 2). Примерное содержание структуры следующее.

```
2025/01/20          10:54:08          ScadiID          from          session:
{"id":"1130000064906924","name":"ovchinnikov@scadi.tech","resource_id":2,"cat
egory":15,"activity":"basic"}
```

Для пользователя со ScadiID создается сессия. Запись в лог-файле будет следующего содержания.

```
2025/01/20         10:54:24         client         ovchinnikov@scadi.tech_374
1130000064906924/ovchinnikov@scadi.tech connected via websocket
```

По окончании работы и выходе со страницы СКОЯТ осуществляется разрыв соединения.

```
2025/01/20 10:54:30 client ovchinnikov@scadi.tech_374 disconnected
```

Таким образом, фиксируются события, связанные со входами в сервис в системе. Аналогично фиксируются события, связанные с началом и окончанием использования ресурсов системы.

Данные о сессиях и пользователях доступны посредством Web-сокет запросов к серверу. Перечень и структура запросов и ответов представлены в приложении 8.

Приложение 8 также содержит историю пройденных курсов и полученных оценок, доступную через сервис WoData.

3.9 СИСТЕМА ПРЕДСТАВЛЕНИЯ ОБУЧАЮЩИХ ПРОГРАММ

3.9.1 Структура фреймового представления графической системы

Система представления обучающих программ обеспечивает реализацию интерактивных страниц с веб-контентом. Фреймовая структура интерактивных страниц представлена на рисунке 23.

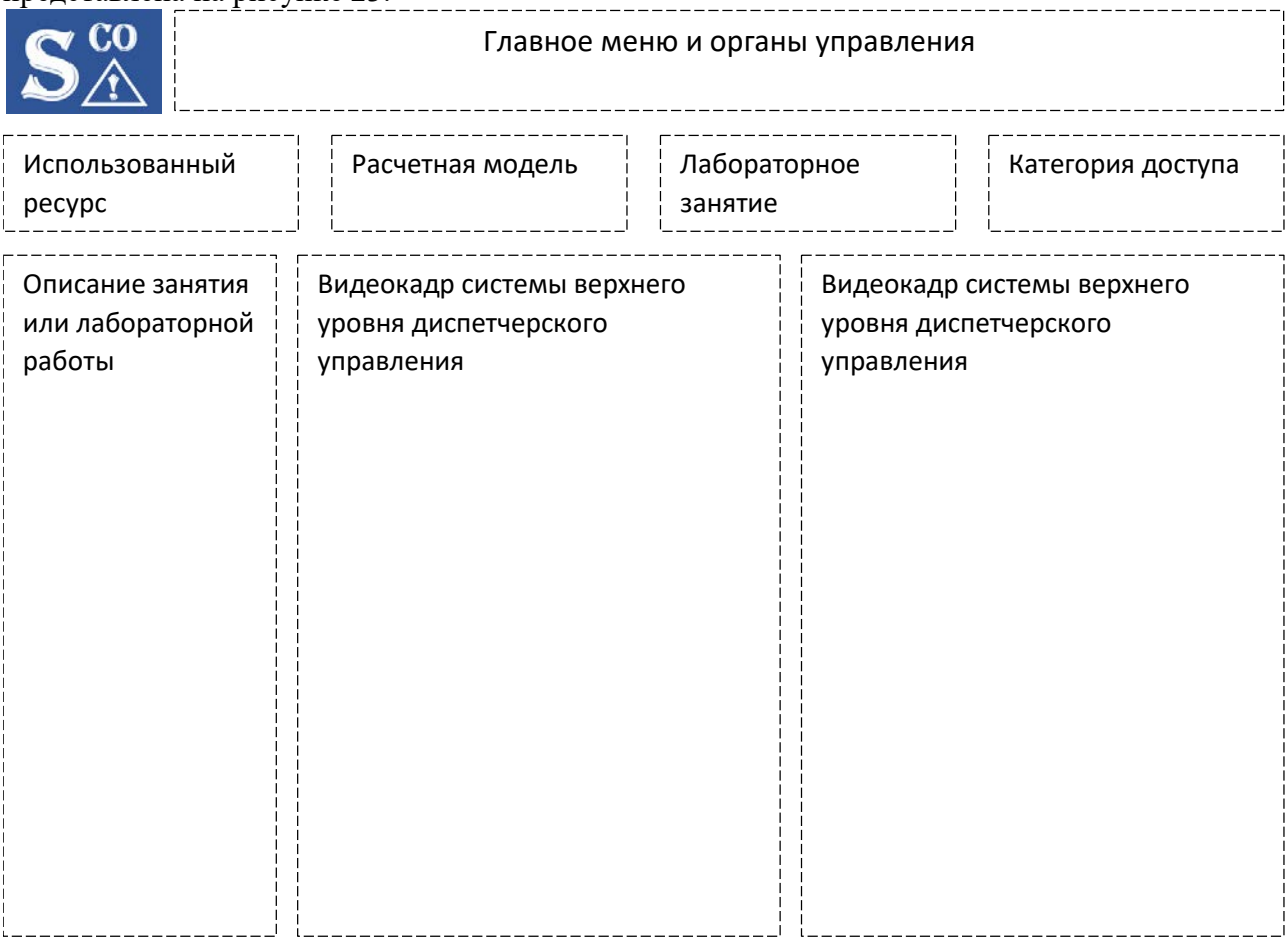

Рисунок 23 – Фреймовая структура интерактивных страниц

В верхней части расположены основные органы управления и название программной реализации. Второй сверху ряд содержит фреймы и структуры данных, а именно:

- использованный ресурс;
- расчетная модель;
- лабораторное занятие;
- категория доступа.

Обе верхних строки остаются постоянными при работе системы. В части структур данных они содержат необходимые веб-сокет объекты, связанные с серверными ресурсами различных систем и подсистем.

Фреймы нижней части динамически меняют как свою структуру, так и содержание. Типовая фреймовая структура, представленная в левой части рисунка, содержит описание

занятия или лабораторной работы в виде гипертекстовых уроков. В средней и правой частях рисунка расположены форматы представления данных технологического процесса. Форматы данных технологического процесса при наличии загруженного ресурса и активной расчетной модели выводятся с динамическими значениями, соответствующих данному процессу.

При запуске интерактивного задания расчетная модель или внешний тренажер запускаются с исходным состоянием, определенным в конфигурации описания занятия. После этого данные технологического процесса начинают пересчитываться, исходя из начальных условий, определенных исходным состоянием.

3.9.2 Объектная структура системы представления

Объектная структура системы представления обучающих программ имеет корневые объекты уровня приложения, располагающиеся в верхнем фрейме графического экрана, рисунок 24.

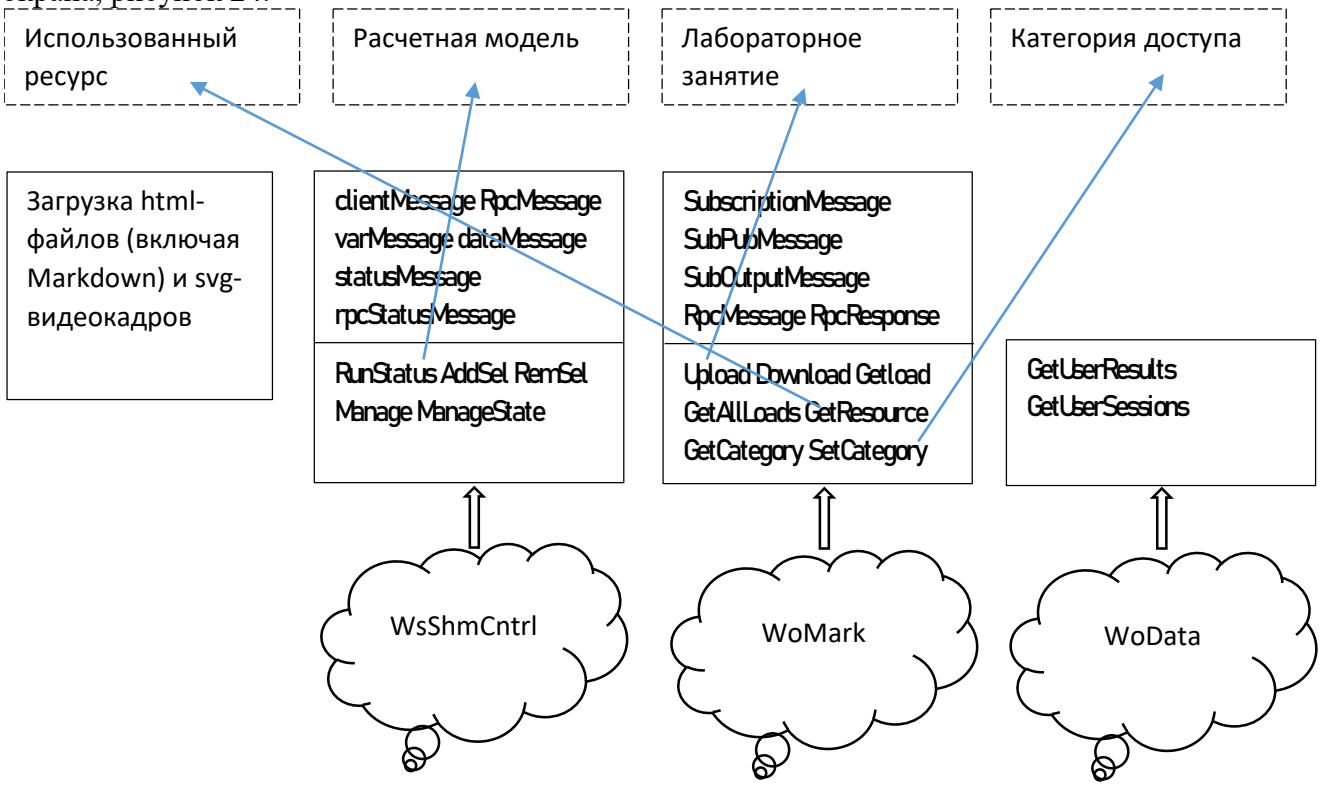

Рисунок 24 – Объекты верхнего фрейма системы представления

Лабораторное занятие управляется объектом связи с WoMark. При взаимодействии с сервисами методами SetCategory и GetCategory устанавливается категория доступа и отображается в верхнем фрейме. Метод GetResource получает в пользование ресурс, который управляет и контролирует жизненный цикл контейнера. Если ресурс запрашивается в единоличное пользование "GetResource", {"group": "", "ownership": "exclusive"}, то он будет использоваться монопольно.

В противном случае "GetResource", {"group": "", "ownership": "shared"} ресурс находится в разделенном доступе для разных пользователей, и запросивший пользователь будет его владельцем. Пользователи, арендующие ресурс "GetResource", {"group": "", "ownership": “rent”}, получают права без возможности управления жизненным циклом ресурса (загрузка/выгрузка).

Контроль за состоянием ресурса осуществляется командой Getload. Загрузка ресурса осуществляется командой Upload, выгрузка – командой Download.

Расчетная модель управляется объектом связи с WsShmCntrl в системе верхнего уровня загружаемого контейнера. При загрузке контейнера объект связи с ним инициализируется, при выгрузке – обнуляется, рисунок 25.

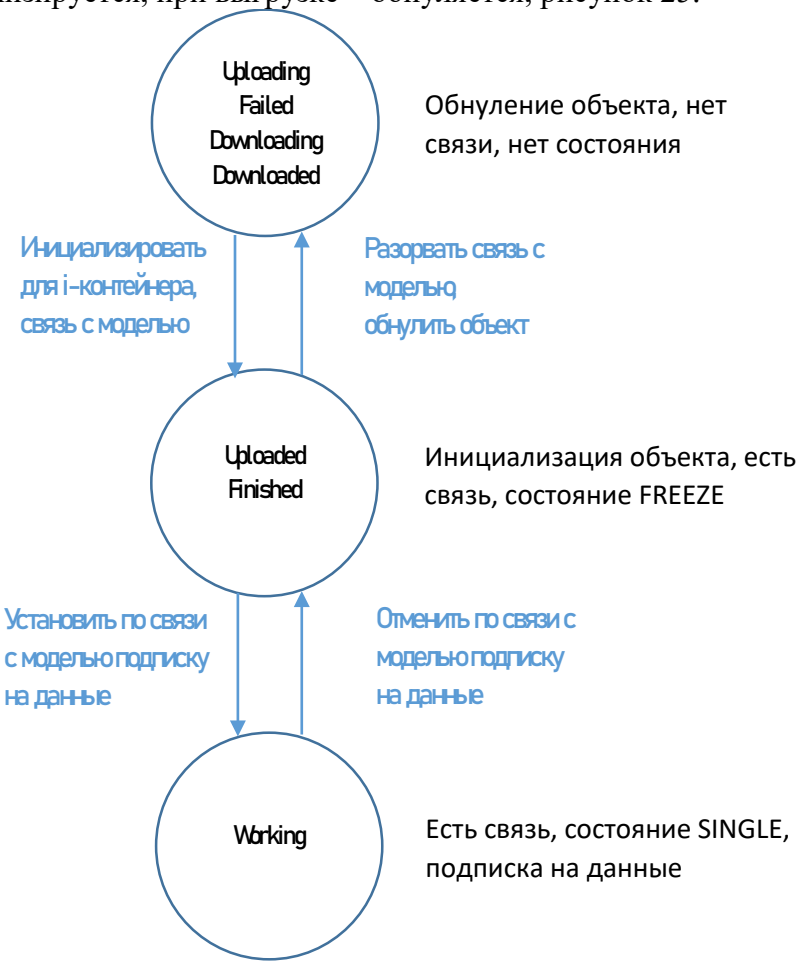

Рисунок 25 – Управление связью с расчетной моделью

Для разных соединений с контейнерами устанавливаются новые экземпляры объектов связи с WsShmCntrl. При выполнении лабораторных работ и прочих расчетных операций ресурс контейнера переходит в состояние Working. В этом состоянии связь не просто присутствует, а устанавливается подписка, по которой обновляются данные и передаются на клюентский браузер. При окончании работы осуществляется переход в состояние Finished, а затем Uploaded. При этом связь с моделью не разрывается, а состояние расчетных задач устанавливается во FREEZE (задача остановлена).

Обмен данных по подписке производится с помощью сообщений clientMessage, dataMessage приложения 3.

Установка состояний SINGLE, FREEZE осуществляется вызовом метода RunStatus.

3.9.3 Скрипты клиентской части

Скрипты, используемые клиентской частью СКОЯТ написаны на языке JavaScript. Часть из них реализована в составе платформы СКАДИВЕБ. Полный список скриптов приведен в Таблице 7.

Имя файла скрипта	Принадлежность к платформе	Назначение
centrifuge.js	СКАДИВЕБ	Обмен данными с веб-сокет сервером
svd.js	СКАДИВЕБ	Библиотека классов, реализующих элементы веб-интерфейса и динамические объекты
dialog.js	СКАДИВЕБ	Реализация окон управления
cosco.js	СКОЯТ	Определение глобальных констант. Установление соединений с сервисами. Обновление верхних фреймов головной страницы
task.js	СКОЯТ	Инициализация и завершение заданий
cfuge.js	СКОЯТ	Реализация функций связи элементов графического интерфейса с веб-сокет сервером

Таблица 7. Состав скриптов клиентской части.

Назначение основных функций, реализованные в скриптах клиентской части СКОЯТ, приведено в Таблице 8.

Имя функции	Имя файла скрипта	Назначение
connect()	cosco.js	Установление соединения с сервисами WoMark и WoData
connectMM()	cosco.js	Установление соединения с сервисом WsShmCntrl
openMenuItem()	cosco.js	Обработка пунктов меню информационного фрейма головной страницы и фрейма выбора заданий
updateHeader()	cosco.js	Обновление информационного фрейма головной страницы
mmFrz()	cosco.js	Остановка вычислений – перевод модели в состояние FREEZE
prepareLabLesson()	task.js	Подготовка данных для задания (путем извлечения их из описания задания)
initLabLesson()	task.js	Инициализация задания
finishTask()	task.js	Завершение задания
initFrame()	cfuge.js	Инициализация фрейма при загрузке задания

Таблица 8. Назначение основных функций

4 ИСПОЛЬЗУЕМЫЕ СИСТЕМНЫЕ И ТЕХНИЧЕСКИЕ СРЕДСТВА

4.1 МИНИМАЛЬНЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА РАЗВЕРТЫВАНИЯ

Развертывание СКОЯТ в минимальной конфигурации требует одну виртуальную машину, на которой развернуто ПО СКОЯТ. Виртуальная машина должна иметь выход в интернет с открытыми портами в nginx.

4.2 СИСТЕМНЫЕ СРЕДСТВА УПРАВЛЕНИЯ РЕСУРСАМИ

Сервис управления ресурсами СКОЯТ на локальной машине реализован на средствах Docker Engine. Основные компоненты представляют собой:

- сервер, на котором запускается фоновым процессом демон docker;
- REST API протокол взаимодействия с демоном docker;
- реализованная в WoMark клиентская часть взаимодействия через Go client for the Docker Engine API.

4.3 СИСТЕМА УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ

Сервис базы данных использует СУБД в зависимости от необходимого объема системы. Для небольших задач на локальной машине используется пакет sqlite3.

Большие объемы и больший уровень масштабирования системы требует автономных серверных СУБД.

4.4 РАЗВЕРТЫВАНИЕ В ОБЛАЧНЫХ СЕРВИСАХ

Разветывание системы в облачном сервисе производится методом генерации виртуальной машина с помощью ПО vagrant и размещение ее в облаке.

5 ВЫЗОВ И ЗАГРУЗКА

5.1 ПОДГОТОВКА СЕРВЕРНОЙ ЧАСТИ

Развертывание СКОЯТ в минимальной конфигурации требует одну виртуальную машину. Перед развертыванием требуется заполнение базы данных проектными исходными данными. Заполнение осуществляется вызовом скрипта mkdb_users.sh.

Также предполагается, что в развернутом проекте заполнены прикладные данные в части курсов, задач, гипертекстовых условий выполнения и проч.

Должен быть развернут сервис управления ресурсами СКОЯТ на локальной машине реализован на средствах Docker Engine.

Поскольку приложения на основе СКОЯТ должны работать в глобальных сетях, поэтому при подготовке устанавливается окружение для регистрации в Яндекс OAuth. В частности, через окружение задаются параметры Client secret и Redirect URI для сервиса WoAccess.

5.2 ЗАГРУЗКА СЕРВИСОВ

Каждый из сервисов WoAccess, WoMark, WoData конфигурируется отдельно и запускается в операционной системе.

Запуск осуществляется системным сервисом systemd

Для кажого сервиса описан свой Unit файл.

WoAccess	woaccess.service
WoMark	womark.service
WoData	wodata.service

5.3 ОБРАЗЫ И КОНТЕЙНЕРЫ

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 26 – 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.

При создании контейнера к имени образа добавляется постфикс, например, dock1. На рисунке 27 представлен перечень доступных контейнеров.

```
user@osco:~/osco$
user@osco:~/osco$ docker ps --all
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORT
S                      NAMES
96f558466516   model-oisc:v6d     "/bin/sh -c 'DsMM -c…"   23 seconds ago   Up 22 seconds   0.0.
0.0:12021->12020/tcp   model-oisc-dock1
1d4b7caf3322   model-rb60:v1.1d   "/bin/sh -c 'DsMM -c…"   28 minutes ago   Up 28 minutes   0.0.
0.0:12022->12020/tcp   model-rb60-dock2
user@osco:~/osco$
```

Рисунок 27 – Список доступных контейнеров

Имя model-oisc-dock1 образовано из имени образа model-oisc и постфикса dock1. При работе контейнер загружается, его статус становится активным Up. Выведенные наружу объекты контейнера доступны через связанные с контейнером порты. В приведенном примере порт 12021 связан с контейнером model-oisc-dock1, а порт 12022 связан с контейнером model-oisc-dock2.

По окончании использования контейнер сначала выгружается, а потом удаляется из системы. При этом образ в текущей версии хранится на протяжении длительного времени, пока не возникнет необходимости в замене образа на другую версию.

6 ВХОДНЫЕ ДАННЫЕ

6.1 ОПИСАНИЕ ГИПЕРТЕКСТА СРЕДСТВАМИ MARKUP

Входными данными для системы являются учебно-методические материалы, подготавливаемые обучающим персоналом. Учебно-методические материалы формируются в виде гипертекста. Гипертекстовая информация редактируется средствами Markup редакторов. Использование внешнего редактора для подготовки учебно-методических материалов представлено на рисунке 28.

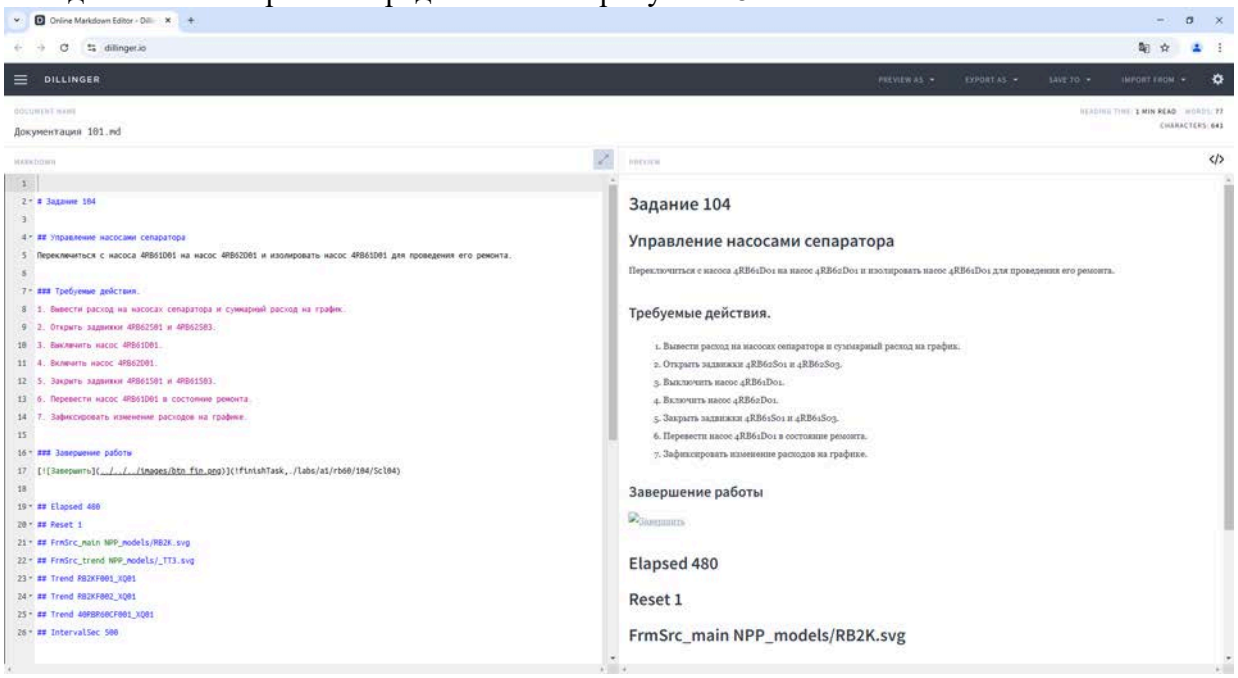

Рисунок 28 – Редактирование гипертекста средствами Dillinger.io

В левой части редактора представлен текст, как он вводится в редактор. В правой части – отображение гипертекста, как оно будет выглядеть в системе СКОЯТ. В приложении 9 приведен полный гипертекст для указанного материала.

Материал содержит тестовые данные с возможностью форматирования разделов и абзацев.

Также материал содержит команды, выполняемые, например, по завершению выполнения задания ([Завершить] в приложении).

Кроме этого, материал содержит параметры для выполнения задания (время, интервал на графике, номер начального условия и т.д.). Описание материалов приведено в приложении 9.

6.2 ВХОДНЫЕ ФАЙЛЫ БАЗЫ ДАННЫХ РЕАЛЬНОГО ВРЕМЕНИ

Система СКОЯТ использует параметры базы данных реального времени СКАДИ, состоящие из аналоговых AI, дискретных DI и целочисленных NI параметров (приложение 10). Аналоговые данные загружаются в пользовательские реляционные таблицы AIs. Пример структур данных приведен ниже.

```
Obj 1000 -name 1AI001001 -code 40TLR02CM002_XQ01 -descr {М АЭРФ 4TL02D002 0-
100} -units % -lf 0 -hf 100 -la ~ -ha ~ -lw 60 -hw 80 -lr ~ -hr ~ -almpri 0 -
```

```
dst_mask 0 -xarea_id 101 -proflags 31 -alloflags 31 -is_work TRUE -fun_type 0
-zone 0.0 -ndigits 0
Obj 1001 -name 1AI001002 -code 40TLR02CP001_XQ01 -descr {P-Р АЭРФ 4TL02D001
0-100} -units КГС/М -lf 0 -hf 100 -la ~ -ha ~ -lw ~ -hw 95 -lr ~ -hr ~ -
almpri 1033 -dst_mask 0 -xarea_id 101 -proflags 31 -alloflags 31 -is_work
TRUE -fun_type 0 -zone 0.0 -ndigits 0
Obj 1002 -name 1AI001003 -code 40TLR02CP002_XQ01 -descr {P-Р АЭРФ 4TL02D002
0-100} -units КГС/М -lf 0 -hf 100 -la ~ -ha ~ -lw ~ -hw 95 -lr ~ -hr ~ -
almpri 1033 -dst_mask 0 -xarea_id 101 -proflags 31 -alloflags 31 -is_work
TRUE -fun_type 0 -zone 0.0 -ndigits 0
```

Дискретные данные загружаются в пользовательские реляционные таблицы DIs. Пример структур данных приведен ниже.

```
Obj 1000 -name 1DI001001 -code 40TKR23CT005C_XH10 -descr {Т ОБМОТ СТАТ 3Ф >30
} -almpri 0 -dst_mask 1 -xarea_id 101 -proflags 7 -alloflags 7 -is_work TRUE
-fun_type 6 -inver FALSE
Obj 1001 -name 1DI001002 -code 40TKR23CT005C_XH58 -descr {Т ОБМОТ СТАТ 3Ф <25
} -almpri 0 -dst_mask 1 -xarea_id 101 -proflags 7 -alloflags 7 -is_work TRUE
-fun_type 6 -inver FALSE
Obj 1002 -name 1DI001003 -code 40TKR23CT005C_XM06 -descr {Т ОБМОТ СТАТ 3Ф
ИМИТ} -almpri 0 -dst_mask 1 -xarea_id 101 -proflags 7 -alloflags 7 -is_work
TRUE -fun_type 6 -inver FALSE
Obj 1003 -name 1DI001004 -code 40TKR23CT005C_XM31 -descr {Т ОБМОТ СТАТ 3Ф
РАСС} -almpri 0 -dst_mask 1 -xarea_id 101 -proflags 7 -alloflags 7 -is_work
TRUE -fun_type 6 -inver FALSE
```

Целочисленные данные загружаются в пользовательские реляционные таблицы NIs.

Видеокадры системы верхнего уровня разрабатываются таким образом, чтобы осуществить привязку к базам данных реального времени конкретного проекта.

7 ВЫХОДНЫЕ ДАННЫЕ

7.1 ПРОТОКОЛЫ ДЕЙСТВИЙ ОБУЧАЕМОГО

Выходными данными для системы являются протоколы действий обучаемого. Формат протоколов действий обучаемого представлен в приложении 11.

Протокол создается подсистемой архивирования действий обучаемого как онлайн-фиксацию данных при прохождении заданий и лабораторных работ.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1 – СОСТАВ ПРОГРАММНЫХ СЕРВИСОВ СКОЯТ

Состав программных сервисов СКОЯТ приведен в таблице 1-1 приложения.

Имя	Наименование	Система
WoAccess	Сервер управления доступом	Регистрации и управления доступом
WsShmCntrl	СКАДИВЕБ	Верхнего уровня диспетчерского управления
DsMM	СКАДИ-МОДЕЛЬ	Управления моделями технологического процесса
WoData	Сервер доступа к данным об использовании ресурсов	Система учета использованных ресурсов
WoMark	Сервер контроля действий обучаемого	Оценивание действий обучаемого

Приложение 1-1 Состав программных сервисов

ПРИЛОЖЕНИЕ 2 – ОБЩИЕ СТРУКТУРЫ ДАННЫХ СКОЯТ

Структура данных YandexUser, передаваемая между сервисами Яндекс и WoAccess.

```
type YandexUser struct {
      Login         string `json:"login"`
      ID            string `json:"id"`
      ClientId      string `json:"client_id"`
      PsUID         string`json:"psuid"`
      Email         string`json:"default_email"`
}
```

Поля структуры YandexUser приведены в таблице 2-1.

Поле	Наименование	Значения	Имя JSON
Login	Имя логина	<строка>	json:"login"
ID	Идентификатор Яндекса	<строка>	json:"id"
ClientID	Идентификатор клиента	<строка>	json:"client_id"
PsUID	Имя логина пользователя	<строка>	json:"psuid"
Email	Электронная почта	<строка>	json:"default_email"

Таблица 2-1. Поля типа YandexUser

Структура данных ScadiID, передаваемая между сервисами СКАДИ.

```
type ScadiID struct {
      ID          string `json:"id"`
      Name        string `json:"name"`
      ResourceID  int   `json:"resource_id"`
      Category    int `json:"category"`
      Activity    string `json:"activity"`
}
```

Поля структуры ScadiID приведены в таблице 2-2.

Поле	Наименование	Значения	Имя JSON
ID	Идентификатор Яндекса для СКАДИ	<строка>	json:"id"
Name	Имя логина	<строка>	json:"name"
ResourceID	Номер ресурса СКОЯТ	<число>	json:"resource_id"
Category	Поддержка категорий доступа	<число>	json:"category"
Activity	Вид деятельности	<строка>	json:"activity"

Таблица 2-2. Поля типа ScadiID

Поле ResourceID принимает значения 1-n_resources в зависимости от ассоциированного с пользователем ресурса. Значение 0 означает отсутствие ресурса.

Поле Category определяет поддерживаемые категории доступа, суммируемые как битовые маски таблицы 2-3.

Имя	Значение	Роль	Деятельность
VISITORc	1	Наблюдатель	Изучение и мониторинг моделя
STUDENTc	2	Обучаемый	Мониторинг и управление
EXAMINEEc	4	Экзаменуемый	Сдача лабораторной по мониторингу и управлению
INSTRUCTORc	8	Инструктор	Контроль и управление экзаменуемыми

Таблица 2-3. Поля категорий доступа

ПРИЛОЖЕНИЕ 3 – ПЕРЕЧЕНЬ СООБЩЕНИЙ СВУ ФОРМАТА JSON

Сообщение от клиента clientMessage устанавливает подписку для данного фрейма клиента.

```
type clientMessage struct {
	Timestamp int64  `json:"timestamp"`
	SFrame    string `json:"sframe"`
	Input     string `json:"input"`
	FragName    string `json:"fragname"`
	Inips    string `json:"inips"`
	VarNames    string `json:"varnames"`
	VFile    string `json:"vfile"`
	AttrNames    string `json:"attrnames"`
}
```

Перечень параметров сообщения приведен в таблице 3-1.

Имя	Описание	Значение	Примечания
timestamp	Метка времени	время	64бит
sframe	Имя фрейма	строка	Уникальное имя
input	Набор переменных подписки	строка	Пробелы как разделители
fragname	Имя фрагмента	строка	
inips	Строка инициализации группового отображения	строка	По спецификации СКАДИ
varnames	Имена переменных	строка	Пробелы как разделители
vfile	Имя файла	строка	Файл из разделяемой памяти
attrnames	Имена атрибутов	строка	Пробелы как разделители

Таблица 3-1. Параметры сообщения clientMessage

Сообщение от клиента RpcMessage запрашивает вызов клиентом удаленной процедуры на стороне сервера.

```
type RpcMessage struct {
	Timestamp int64  `json:"timestamp"`
	SFrame    string `json:"sframe"`
	RStatus     string `json:"rstatus"`
	PageSet     int `json:"pageset"`
}
```

Перечень параметров сообщения приведен в таблице 3-2.

Имя	Описание	Значение	Примечания
timestamp	Метка времени	время	64бит
sframe	Имя фрейма	строка	Уникальное имя
rstatus	Статус выполнения	строка	run|freeze
pageset	Уставка номера страницы	число	

Таблица 3-2. Параметры сообщения RpcMessage

Сообщение от сервера dataMessage передает клиенту очередную порцию данных по подписке. Сообщение содержит динамический массив структур varMessage по одной структуре на каждую передаваемую переменную.

```
type varMessage struct {
	Vname string `json:"vname"`
	Val         string `json:"val"`
	Atype string `json:"atype"`
}
type dataMessage struct {
	SFrame      string `json:"sframe"`
	NVars int `json:"nvars"`
```

```
	Vars []varMessage `json:"vars"`
}
```

Перечень параметров сообщений приведен в таблице 3-3.

Имя	Описание	Значение	Примечания
vname	Имя переменной	строка	
val	Значение переменной	строка	
atype	Тип переменной	строка	Поле «Имя»
sframe	Имя фрейма	строка	Уникальное имя
nvars	Число переменных	число	
vars	Массив переменных	[]varMessage	Массив varMessage

Таблица 3-3. Параметры сообщения dataMessage

Сообщение от сервера statusMessage передает клиентам телеграммы жизни, включающие состояние и текущее время удаленного сервера.

```
type statusMessage struct {
	TInput		string `json:"input"`
	SrvStatus	string `json:"srvstatus"`
	SrvClock	int64 `json:"srvclock"`
}
```

Перечень параметров сообщения приведен в таблице 3-4.

Имя	Описание	Значение	Примечания
input	Имя телеграммы	строка	Heartbeat
srvstatus	Статус сервера	число	
srvclock	Время сервера	время	64бит Unix

Таблица 3-4. Параметры сообщения statusMessage

Сообщение от сервера rpcStatusMessage является ответом на запрос клиента RpcMessage с методом «RunStatus».

```
type rpcStatusMessage struct {
	SFrame	string `json:"sframe"`
	RMethod string `json:"rmethod"`
	Result string	`json:"result"`
}
```

Перечень параметров сообщения приведен в таблице 3-5.

Имя	Описание	Значение	Примечания
sframe	Имя фрейма	строка	Уникальное имя
rmethod	Имя метода	строка	RunStatus
result	Результат выполнения	строка	run|freeze

Таблица 3-5. Параметры сообщения rpcStatusMessage

Сообщение от сервера rpcPagesMessage является ответом на запрос клиента RpcMessage с методом «NextPage». Сообщение содержит вложенную структуру rpcPages с номерами страниц.

```
type rpcPages struct {
	Cp	int32 `json:"cp"`
	Fp	int32 `json:"fp"`
	Np	int32 `json:"np"`
}
type rpcPagesMessage struct {
	SFrame	string `json:"sframe"`
	RMethod string `json:"rmethod"`
	Result string	`json:"result"`
	Pages rpcPages	`json:"pages"`
```

```
}
```

Перечень параметров сообщения приведен в таблице 3-6.

Имя	Описание	Значение	Примечания
cp	Текущая страница	число32	
fp	Первая страница	число32	
np	Число страниц	число32	
sframe	Имя фрейма	строка	Уникальное имя
rmethod	Имя метода	строка	NextPage
result	Результат выполнения	строка	
pages	Структура страниц	rpcPages	

Таблица 3-6. Параметры сообщения rpcPagesMessage

Сообщение от сервера rpcVFilesMessage является ответом на запрос клиента RpcMessage с методом «GetVFiles». Сообщение содержит динамический массив структур rpcVFile по одной структуре на каждый передаваемый файл.

```
type rpcVFile struct {
      Fname string      `json:"fname"`
      Descr string      `json:"descr"`
      Nvars int         `json:"nvars"`
}
type rpcVFilesMessage struct {
      SFrame      string `json:"sframe"`
      RMethod string `json:"rmethod"`
      Result string     `json:"result"`
      Vfiles []rpcVFile `json:"vfiles"`
}
```

Перечень параметров сообщения приведен в таблице 3-7.

Имя	Описание	Значение	Примечания
fname	Имя файла	число32	
descr	Описание файла	число32	
nvars	Число переменных в нем	число	
sframe	Имя фрейма	строка	Уникальное имя
rmethod	Имя метода	строка	GetVFiles
result	Результат выполнения	строка	
vfiles	Массив файлов	[]rpcVFile	Массив rpcVFile

Таблица 3-7. Параметры сообщения rpcVFilesMessage

ПРИЛОЖЕНИЕ 4 - ДАННЫЕ СИСТЕМНЫХ РЕЛЯЦИОННЫХ ТАБЛИЦ

Типы данных СКОЯТ-МОДЕЛЬ пересекуются с типами данных СКАДИ.

Тип Const предназначен для хранения целочисленных констант. Структура Const – это запись следующего вида, наследуемая от OGeneric.

```
Const* = RECORD (SO.OGeneric)
      val-, vdef-, vmin-, vmax-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-1.

Поле	Наименование	Значения	Примечание
val	Текущее значение	<Число>	Все значение целочисленные
vdef	Значение по умолчанию	<Число>	-1 если не задано
vmin	Минимально возможное значение	<Число>	-1 если не задано
vmax	Минимально возможное значение	<Число>	-1 если не задано

Таблица 4-1. Список полей структуры Const

Значения Const представляют собой именованные целочисленные константы. Константы имеют также диапазон и значение по умолчанию. Доступ к значениям структуры Const осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Characteristic предназначен для хранения характеристик системы взаимодействия. Структура Characteristic - это запись следующего вида, наследуемая от OGeneric.

```
Characteristic* = RECORD (OGeneric)
      parent-: Tb.Int32;
      cname-: Sb.STag;
      descr-: Sb.SDescr;
END;
```

Список полей структуры приведен в таблице 4-2.

Поле	Наименование	Значения	Примечание
parent	Идентификатор родительской характеристики	<Число>	
cname	Имя родительской характеристики	<Имя>	Значения из предопределенных
descr	Описание характеристики	<Строка>	

Таблица 4-2. Список полей структуры Characteristic

Значения Characteristic представляют собой именованные характеристики системы взаимодействия. Характеристики определяют физически возможное в данной архитектуре взаимодействие (управление, обновление данных, синхронизацию. Доступ к тем или иным возможностям системы характеристиками не определяется. Доступ к значениям структуры Characteristic осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Usergroup предназначен для хранения име пользовательских групп. Структура Usergroup - это запись следующего вида, наследуемая от OGeneric.

```
Usergroup* = RECORD (OGeneric)
      chids-: Tb.Int32;
      dst_id-: Tb.Int32;
      lname-: Sb.STag;
      descr-: Sb.SDescr;
```

```
END;
```

Список полей структуры приведен в таблице 4-3.

Поле	Наименование	Значения	Примечание
chids	Набор идентификаторов характеристик		Множесто oid из таблицы Characteristics
dst_id	Адресат сигнализации	<Число>	
lname	Имя группы пользователей	<Строка>	
descr	Описание группы пользователей	<Строка>	

Таблица 4-3. Список полей структуры Usergroup

Значения Usergroup представляют собой группы пользователей, имеющих набор ролей пользовательского доступа. Доступ к значениям структуры Usergroup осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Task предназначен для хранения задач СКАДИ. Структура Task - это запись следующего вида, наследуемая от OGeneric.

```
Task* = RECORD @UNTAG (OGeneric)
      process_id-, program_id-, n_program-: Tb.Int32;
      exe_path-: Sb.String64;
      command_line-: Sb.String128;
      dependencies-: Sb.STag;
      watch_limit-, watch_count-, restart_limit-,
            restart_count-: Tb.Int32;
      is_disabled-, is_debug-, is_master-, must_start-: BOOLEAN;
      run_at-: Tb.Int32;
      command_result-, task_status-: Tb.Int32;
      update_ms-: Tb.Int32;
      start_time-, writeq_time-, current_time-, cycle_period-,
            delta_time-: Tb.Time;
      error_count-, error_code_last-, init_trace_level-, trace_level-,
            init_priority-, priority-: Tb.Int32;
      error_time_first-, error_time_last-, process_start-,
            process_end-: Tb.Time;
      current_load-, maximum_load-, writeq_dsec-,
            statistics_interval-: Tb.Int16;
      write_q-, counter_n-: Tb.Int32;
      for_recv_q-, recv_q-, for_send_q-, send_q-, resend_q-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-4.

Поле	Наименование	Значения	Примечание
process_id	Идентификатор процесса	<Число>	getpid()
program_id	Идентификатор задачи	<Роли>	
n_program	Порядковый номер программы данного типа	<Число>	0, 1, 2
exe_path	Имя исполняемой задачи	<Строка>	
command_line	Аргументы командной строки	<Строка>	Запуск: <exe_path> <command_line>
dependencies	Зависимости – загруженные задачи	<Строка>	
watch_limit	Предел счетчика исправности	<Число>	
watch_count	Значение счетчика исправности	<Число>	Перезапуск при 0

restart_limit	Предел счетчика рестартов	<Число>	
restart_count	Значение счетчика рестартов	<Число>	Прекращение при 0
is_disabled	Задача отключена	BOOL	
is_debug	Задача в отладке	BOOL	
is_master	Задача является основной	BOOL	
must_start	Задача стартует автоматически	BOOL	
run_at	Уровень запуска	<Число>	
command_result	Результат выполнения команды	<Число>	Код возврата процесса
task_status	Статус задачи	0-OFF 1-INIT 2-LOAD 3-ERROR 4-END 5-WAITING 6-STARTING 7-SINGLE 8-PRIMARY 9-BACKUP 10-FAILOVER	Enum SRV_STAT
update_ms	Время обновления	<Интервал>	Миллисекунд
start_time	Время старта	<Время>	
writeq_time	Время записи данных в циклический буфер	<Время>	
current_time	Текущее время	<Время>	
cycle_period	Период цикла	<ИнтервалВрем>	
delta_time	Время обработки	<ИнтервалВрем>	
error_count	Счетчик ошибок	<Число>	
error_code_last	Код последней ошибки	<Число>	0 - нет
init_trace_level	Начальное значение уровня трассировки	<Число>	0 - нет
trace_level	Текущее значение уровня трассировки	<Число>	0 - нет
init_priority	Начальное значение приоритета	<Число>	
priority	Текущее значение приоритета	<Число>	
error_time_first	Время первой ошибки	<Время>	
error_time_last	Время последней ошибки	<Время>	
process_start	Время старта процесса	<Время>	
process_end	Время окончания процесса	<Время>	
current_load	Текущая загрузка процесса	<Число>	
maximum_load	Максимальная загрузка процесса	<Число>	
writeq_dsec	Интервальный размер q-глобала	<Интервал>	История в секундах

statistics_interval	Интервал сбора статистики	<Число>	
write_q	Очередь сообщений для записи	<Номер>	Номер глобала
counter_n	Счетчик изменений глобала n	<Число>	
for_recv_q	Счетчик изменений для очереди приема	<Число>	
recv_q	Счетчик изменений очереди приема	<Число>	
for_send_q	Счетчик изменений для очереди передачи	<Число>	
send_q	Счетчик изменений очереди передачи	<Число>	
resend_q	Счетчик изменений очереди повторной передачи	<Число>	

Таблица 4-4. Список полей структуры Task

Значения Task представляют данные, относящиеся к функционированию текущей задачи. Идентификация задачи осуществляется по имени, заданном в поле name структуры OGeneric.

Тип HGroup предназначен для хранения групп хостов. Каждой группе хостов соответствует область обмена внутри ПО СКАДИ. Или, по другому, каждой области обмена соответствует группа задействованной в ней хостов. Структура HGroup (группы хостов) - это запись следующего вида, наследуемая от OGeneric.

```
HGroup* = RECORD (OGeneric)
      task_id-: Tb.Int32;
      multicast_address-: Sb.String32;
      multicast_address_2-: Sb.String32;
      pl_polling-: Sb.String64;
      pl_protocol-: Sb.String64;
      pl_tezas-: Sb.String64;
      cu_polling-: Sb.String32;
      cu_protocol-: Sb.String32;
      cu_tezas-: Sb.String32;
      cu_tezas_params-: Sb.String128;
      protocol-, rtdb_port-, rtdb_port_2-: Tb.Int16;
      netnum-: Tb.Int16;
      hgtype-: Tb.Int16;
      tp_discovery_timeout-: Tb.Int32;
      fix_port-: Tb.Int16;
      max_resend_counter-: Tb.Int16;
      tp_resend_delay-: Tb.Int32;
      max_send_polls-: Tb.Int16;
      so_keepalive-: Tb.Int16;
      so_reuseaddr-: Tb.Int16;
      so_sndbuf-: Tb.Int32;
      so_rcvbuf-: Tb.Int32;
      tp_connect_timeout-: Tb.Int32;
      tp_disconnect_timeout-: Tb.Int32;
      tcp_nodelay-: Tb.Int16;
      listen_backlog-: Tb.Int16;
      max_send_length-: Tb.Int32
```

```
        tp_wait_tmo-: Tb.Int32
    END;
```

Список полей структуры приведен в таблице 4-5.

Поле	Наименование	Значения	Примечание
task_id	Номер в таблице Tasks	<Число>	
multicast_address	Адрес групповой передачи	<IP>	
multicast_address2	Адрес групповой передачи 2	<IP>	
pl_polling	Плагин поллинга	<Имя>	Имя динамически загружаемого модуля
pl_protocol	Плагин протокола	<Имя>	Аналогично
pl_tezas	Плагин технологического закрытия	<Имя>	Аналогично
cu_polling	Имя поллинга	<Имя>	
cu_protocol	Имя протокола	<Имя>	
cu_tezas	Имя закрытия	<Имя>	
cu_tezas_params	Параметры закрытия	<Строка>	
protocol	Протокол обмена	1-ICMP 2-UDP 3-FTUDP 4-MUDP 5-FTMUDP 6-TCP 7-FTTCP	Enum PROTOCOL FT Означает дублированное Fault Tolerant расширение
rtdb_port	Порт синхронизации БДРВ	<Число>	
rtdb_port_2	Порт 2 синхронизации БДРВ	<Число>	
netnum	Номер подсети	<Число>	
hgtype	Тип группы хостов	1-PING 2-SYNC 3-DVP	Определяет способ синхронизации
tp_discovery_timeout	Таймаут системы обнаружения	<Время>	
fix_port	Фиксированные порт соединения	<Число>	
max_resend_counter	Максимальное значение счетчика повтора посылок	<Число>	
tp_resend_delay	Задержка повтора посылок	<Время>	
max_send_polls	Максимальное значение посылок поллинга	<Число>	
so_keepalive	Параметр связи keepalive	<Число>	POSIX параметры сокета
so_reuseaddr	Параметр связи reuseaddr	<Число>	Аналогично
so_sndbuf	Параметр связи sndbuf	<Число>	Аналогично
so_recvbuf	Параметр связи sndbuf	<Число>	Аналогично
tp_connect_timeout	Таймаут соединения	<Время>	
tp_disconnect_timeout	Таймаут разрыва соединения	<Время>	
tcp_nodelay	Параметр tcp_nodelay	<Число>	Аналогично
listen_backlog	Параметр listen_backlog	<Число>	Аналогично
max_send_length	Максимальная длина передачи	<Число>	

tp_wait_tmo	Время ожидание ответа агента	<Интервал>	

Таблица 4-5. Список полей структуры HGroup

Значения HGroup представляют собой группы хостов для внутренней синхронизации. Доступ к значениям структуры HGroup осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Host предназначен для хранения хостов, обменивающихся в рамках групп HGroup. Структура Host - это запись следующего вида, наследуемая от OGeneric.

```
Host* = RECORD @UNTAG (OGeneric)
      host_id-, ugroup_id-, host_status-: Tb.Int32;
      chids-: Tb.Int32;
      in_groups-: Tb.Int32;
      ip_address-: Sb.String32;
      ip_address_2-: Sb.String32;
      nsync_set-, hmi_set-, restart_limit-,
            restart_count-: Tb.Int32;
      version-: Tb.Int32;
      start_time-, current_time-, monitor_time-, version_date-: Tb.Time;
      std_server-: Tb.Int8;
      is_server-, is_master-: BOOLEAN;
      sync_wdata-: Tb.Int8;
      snap_no-, reset_no-: Tb.Int32;
      cpu_load-, net_mask-: Tb.Int16;
      ping_state-, ping_state_2-: Tb.Int32;
      ping_last-, ping_last_2-: Tb.Time;
      master_time-: Tb.Time;
      master_id-: Tb.Int32;
      comm_status-, comm_status_2-: Tb.Int32;
      ta_flags-, ta_flags_2-: Tb.Int32;
      ta_trans_flags-, ta_trans_flags_2-: Tb.Int32;
      ta_poll_events-, ta_poll_events_2-: Tb.Int32;
      ta_link_stat-, ta_link_stat_2-: Tb.Int32;
      n_recv-: Tb.Int32;
      t_recv-, last_recv-: Tb.Time;
      n_rejected-, n_steps-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-6.

Поле	Наименование	Значения	Примечание
host_id	Идентификатор хоста	<Число>	
ugroup_id	Идентификатор группы пользователей	<Число>	
host_status	Статус хоста	0-OFF 1-INIT 2-LOAD 3-ERROR 4-END 5-WAITING 6-STARTING 7-SINGLE 8-PRIMARY 9-BACKUP 10-FAILOVER 11-FREEZE	Enum SRV_STAT

chids	Набор идентификаторов характеристик	Приложение 6	Множесто oid из таблицы Characteristics
in_groups	Множество групп хостов	<Множество>	Число представляет собой множество идентификаторов oid
ip_address	IP-адрес	<IP>	
ip_address2	IP-адрес 2	<IP>	
nsync_set	Список номеров без синхронизации	<Число=Список>	
hmi_set	Список ЧМИ без обновления	<Число=Список>	
restart_limit	Предел счетчика рестартов	<Число>	
restart_count	Значение счетчика рестартов	<Число>	Прекращение при 0
version	Номер версии	<Число>	
start_time	Время старта	<Время>	
current_time	Текущее время	<Время>	
monitor_time	Время мониторинга	<Время>	
version_date	Дата обновления	<Время>	
std_server	Хост является стандартным сервером	<Число>	0 – не сервер
is_server	Хост является сервером	BOOL	
is_master	Хост является основной	BOOL	
sync_wdata	Синхронизация данными окна	<Число>	
snap_no	Номер Snap	<Число>	< 0, команда установить
reset_no	Номер Reset	<Число>	< 0, команда установить
cpu_load	Текущая загрузка ЦПУ	<Число>	
net_mask	Текущая маска сети	<Число>	
ping_state	Текущее состояние пинга	<Число>	
ping_state_2	Текущее состояние пинга 2	<Число>	
ping_last	Время последнего пинга	<Время>	
ping_last_2	Время последнего пинга 2	<Время>	
master_time	Время основного сервера	<Время>	
master_id	Идентификатор основного сервера	<Число>	oid таблицы Hosts
comm_status	Статус коммуникации	См host_status	Enum SRV_STAT
comm_status_2	Статус коммуникации 2	См host_status	Enum SRV_STAT
ta_flags	Флаги уровня линка канала	<Число>	Число представляет собой множество флагов
ta_flags_2	Флаги уровня линка канала 2	<Число>	Аналогично
ta_trans_flags	Флаги транспортного уровня канала	<Число>	Аналогично

ta_trans_flags_2	Флаги транспортного уровня канала 2	<Число>	Аналогично
ta_poll_events	События поллинга канала	<Число>	
ta_poll_events_2	События поллинга канала 2	<Число>	
ta_link_stat	Статус линка канала	<Число>	
ta_link_stat_2	Статус линка канала 2	<Число>	
n_recv	Счетчик принятых сообщений	<Число>	
t_recv	Время приема на сокете	<Время>	
last_recv	Время приема от станции	<Время>	
n_rejected	Счетчик отклоненных сообщений	<Число>	
n_steps	Счетчик шагов выполнения модели	<Число>	< 0, команда установить

Таблица 4-6. Список полей структуры Host

Значения Host представляют собой описания хостов по отношению к группе обмена и синхронизации. Доступ к значениям структуры Host осуществляется по именам, заданным в поле name структуры OGeneric.

Тип XGroup предназначен для хранения хранения групп хостов, включенных в области связи с агентами обмена. Структура XGroup – это запись следующего вида, наследуемая от OGeneric.

```
XGroup* = RECORD (OGeneric)
      task_id-: Tb.Int32;
      chids-: Tb.Int32;
      multicast_address-: Sb.String32;
      multicast_address_2-: Sb.String32;
      pl_prot_format-: Sb.String64;
      pl_polling-: Sb.String64;
      pl_protocol-: Sb.String64;
      pl_tezas-: Sb.String64;
      cu_prot_format-: Sb.String32;
      cu_polling-: Sb.String32;
      cu_protocol-: Sb.String32;
      cu_tezas-: Sb.String32;
      cu_tezas_params-: Sb.String128;
      protocol-: Tb.Int16;
      netnum-: Tb.Int16;
      max_ndatas-, nack_history-: Tb.Int32;
      tp_ping_timeout-: Tb.Int32;
      enable_ping-: BOOLEAN;
      update_parm-: BOOLEAN;
      with_crc_ref-: BOOLEAN;
      packed_size-: Tb.Int8;
      tp_discovery_timeout-: Tb.Int32;
      fix_port-: Tb.Int16;
      max_resend_counter-: Tb.Int16;
      tp_resend_delay-: Tb.Int32;
      max_send_polls-: Tb.Int16;
      so_keepalive-: Tb.Int16;
      so_reuseaddr-: Tb.Int16;
      so_sndbuf-: Tb.Int32;
      so_rcvbuf-: Tb.Int32;
      tp_connect_timeout-: Tb.Int32;
      tp_disconnect_timeout-: Tb.Int32;
```

```
        tcp_nodelay-: Tb.Int16;
        listen_backlog-: Tb.Int16;
        max_send_length-: Tb.Int32;
        pl_usercom-: Sb.String64;
        cu_usercom-: Sb.String32;
        params_usercom-: Sb.String256;
        tp_wait_tmo-: Tb.Int32;
        tp_nconn_tmo-: Tb.Int32;
        tp_lknown_tmo-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-7.

Поле	Наименование	Значения	Примечание
task_id	Номер в таблице Tasks	<Число>	
chids	Набор идентификаторов характеристик	Приложение 6	Множесто oid из таблицы Characteristics
multicast_address	Адрес групповой передачи	<IP>	
multicast_address2	Адрес групповой передачи 2	<IP>	
pl_prot_format	Плагин логического формата протокола	<Имя>	Имя динамически загружаемого модуля
pl_polling	Плагин поллинга	<Имя>	Аналогично
pl_protocol	Плагин протокола	<Имя>	Аналогично
pl_tezas	Плагин технологического закрытия	<Имя>	Аналогично
cu_prot_format	Имя логического формата протокола	<Имя>	
cu_polling	Имя поллинга	<Имя>	
cu_protocol	Имя протокола	<Имя>	
cu_tezas	Имя закрытия	<Имя>	
cu_tezas_params	Параметры закрытия	<Строка>	
protocol	Протокол обмена	1-ICMP 2-UDP 3-FTUDP 4-MUDP 5-FTMUDP 6-TCP 7-FTTCP	Enum PROTOCOL FT Означает дублированное Fault Tolerant расширение
netnum	Номер подсети	<Число>	
max_ndatas	Максимальное число структур Data в сообщении	<Число>	
nack_history	История несквитированных сигналов	<Число>	
tp_ping_timeout	Таймаут посылок пингования	<Время>	В рамках протокола PROTOCOL
enable_ping	Включение пингования	BOOL	
update_parm	Обновление параметров	BOOL	
with_crc_ref	С контрольными суммами	BOOL	
packed_size	Размер упакованного сообщения	<Число>	

tp_discovery_timeout	Таймаут системы обнаружения	<Время>	
fix_port	Фиксированные порт соединения	<Число>	
max_resend_counter	Максимальное значение счетчика повтора посылок	<Число>	
tp_resend_delay	Задержка повтора посылок	<Время>	
max_send_polls	Максимальное значение посылок поллинга	<Число>	
so_keepalive	Параметр связи keepalive	<Число>	POSIX параметры сокета
so_reuseaddr	Параметр связи reuseaddr	<Число>	Аналогично
so_sndbuf	Параметр связи sndbuf	<Число>	Аналогично
so_recvbuf	Параметр связи sndbuf	<Число>	Аналогично
tp_connect_timeout	Таймаут соединения	<Время>	
tp_disconnect_timeout	Таймаут разрыва соединения	<Время>	
tcp_nodelay	Параметр tcp_nodelay	<Число>	Аналогично
listen_backlog	Параметр listen_backlog	<Число>	Аналогично
max_send_length	Максимальная длина передачи	<Число>	
pl_usercom	Плагин пользовательского протокола	<Имя>	Имя динамически загружаемого модуля
cu_usercom	Имя пользовательского протокола	<Имя>	
params_usercom	Параметры подключения пользовательского протокола	<Строка>	
tp_wait_tmo	Максимальный таймаут ожидания	<Интервал>	
tp_nconn_tmo	Максимальный таймаут обрыва соединения	<Интервал>	
tp_lknown_tmo	Максимальный таймаут необновлени	<Интервал>	

Таблица 4-7. Список полей структуры XGroup

Значения XGroup представляют собой группы хостов для связи с агентами обмена. Константы имеют также диапазон и значение по умолчанию. Доступ к значениям структуры XGroup осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Xch предназначен для хранения хостов агентов обмена, обменивающихся в рамках групп XGroup. Структура Xch - это запись следующего вида, наследуемая от OGeneric.

```
Xch* = RECORD (OGeneric)
       xgroup-: Tb.Int32;
       serv_host_id-: Tb.Int32;
       station_id-: Tb.Int32;
       ip_address-: Sb.String32;
       ip_address_2-: Sb.String32;
```

```
        rtdb_port-, rtdb_port_2-: Tb.Int16;
        xch_reserve-: Tb.Int32;
        comm_status-, comm_status_2-: Tb.Int32;
        ta_flags-, ta_flags_2-: Tb.Int32;
        ta_trans_flags-, ta_trans_flags_2-: Tb.Int32;
        ta_poll_events-, ta_poll_events_2-: Tb.Int32;
        ta_link_stat-, ta_link_stat_2-: Tb.Int32;
        dummy: Tb.Int32;
        t_recv-, last_recv-: Tb.Time;
        n_recv-, n_rejected-: Tb.Int32;
        n_datas-, n_used-: Tb.Int32;
        srvstatus-: Tb.Int16;
        data_err-, last_data_err-: Tb.Int16;
        detail-: Tb.Int32;
        flags-: Tb.Int32;
        clock-, srvclock-, lastclock-: Tb.Time;
        params_usercom-: Sb.String256;
END;
```

Список полей структуры приведен в таблице 4-8.

Поле	Наименование	Значения	Примечание
xgroup	Идентификатор группы хостов	<Число>	Из таблицы XGroup
serv_host_id	Идентификатор сервера	<Число>	Из таблицы Hosts
station_id	Номер станции	<Число>	Station для данной Area
ip_address	IP-адрес	<IP>	
ip_address2	IP-адрес 2	<IP>	
rtdb_port	Порт синхронизации БДРВ	<Число>	
rtdb_port_2	Порт 2 синхронизации БДРВ	<Число>	
xch_reserve	Номер резервного агента	<Число>	
comm_status	Статус коммуникации	См Host.host_status	Enum SRV_STAT
comm_status_2	Статус коммуникации 2	См Host.host_status	Enum SRV_STAT
ta_flags	Флаги уровня линка канала	<Число>	Число представляет собой множество флагов
ta_flags_2	Флаги уровня линка канала 2	<Число>	Аналогично
ta_trans_flags	Флаги транспортного уровня канала	<Число>	Аналогично
ta_trans_flags_2	Флаги транспортного уровня канала 2	<Число>	Аналогично
ta_poll_events	События поллинга канала	<Число>	
ta_poll_events_2	События поллинга канала 2	<Число>	
ta_link_stat	Статус линка канала	<Число>	
ta_link_stat_2	Статус линка канала 2	<Число>	
t_recv	Время приема на сокете	<Время>	
last_recv	Время приема от станции	<Время>	
n_recv	Счетчик принятых сообщений	<Число>	

n_rejected	Счетчик отклоненных сообщений	<Число>	
n_datas	Полное количество данных	<Число>	
n_used	Количество использованных данных	<Число>	
srvstatus	Статус сервера	<Число>	
data_err	Ошибка получения данных	<Число>	
last_data_err	Последняя ошибка получения данных	<Число>	
detail	Детализация ошибки данных	<Число>	
flags	Флаги очереди	<Число>	Число представляет собой множество флагов, относящиеся к очереди сообщений
clock	Время сообщения Xch	<Время>	
srvclock	Время хоста Xch	<Время>	
lastclock	Время последнего сообщения Xch	<Время>	
params_usercom	Параметры подключения пользовательского протокола	<Строка>	

Таблица 4-8. Список полей структуры Xch

Значения Xch представляют собой описания хостов агентов обмена по отношению к группе XGroup. Доступ к значениям структуры Xch осуществляется по именам, заданным в поле name структуры OGeneric.

Тип CSet предназначен для хранения групп, отображаемых в экранные области видеокадров. Каждому объекту соответствует группа для данной области, номера программы, имени хоста. Структура CSet - это запись следующего вида, наследуемая от OGeneric.

```
CSet* = RECORD @UNTAG (OGeneric)
      ndpr-, maxdpr-, ngrp-, maxgrp-: Tb.Int32;
      icell-, max_cell-: Tb.Int32;
      report_id-: Tb.Int32;
      small-: BOOLEAN;
      dispcmd-: Sb.String512;
      disp_req-: Sb.String512;
      inips-: Sb.String512;
      subscr_id-, subscr_crt-, rep_done_crt-: Tb.Int32;
      colnames-: Sb.String256;
      cds-: Sb.String256;
      cds_id-, cds_crt-: Tb.Int32;
      cp-, fp-, np-: Tb.Int32;
      n_found-, pagesize-, n_cols-: Tb.Int32;
      nxt-, nxt_crt-: Tb.Int32;
      more_pages-: BOOLEAN;
      res-, rep_res-: Tb.Int32;
      fst-, fen-: Tb.Time;
END;
```

Список полей структуры приведен в таблице 4-9.

Поле	Наименование	Значения	Примечание

ndpr	Номер программы-вьюера	<Число>	
maxdpr	Число программ-вьюеров	<Число>	ndpr<maxdpr
ngrp	Номер группы визуализации	<Число>	
maxgrp	Число групп визуализации	<Число>	ngrp<maxgrp
icell	Идентификатор первой ячейки	<Число>	
max_cell	Число ячеек в наборе	<Число>	
report_id	Индентификатор отчета	<Число>	
small	Признак уменьшенного набора	BOOL	
dispcmd	Команда текущего отображения	<Строка>	
disp_req	Заданная команда текущего отображения	<Строка>	Установленное задание на отображение видеокадра
inips	Параметры инициализации группы	<Строка>	
subscr_id	Идентификатор требуемой подписки	<Число>	
subscr_crt	Идентификатор текущей подписки	<Число>	
rep_done_crt	Идентификатор созданного текущего отчета	<Число>	
colnames	Набор имен столбцов	<Строка>	
cds	Команда установки данных	<Строка>	Фиксированный перечень команд, отрабатываемый групповым отображением
cds_id	Идентификатор требуемой команды установки	<Число>	
cds_crt	Идентификатор текущей команды установки	<Число>	
cp	Текущая страница	<Число>	
fp	Первая страница	<Число>	
np	Число страниц	<Число>	
n_found	Число найденных записей для группы	<Число>	
pagesize	Заданный размер страницы	<Число>	
n_cols	Число столбцов	<Число>	
nxt	Следующая запрашиваемая страница	<Число>	
nxt_crt	Следующая текущая страница	<Число>	
more_pages	Есть еще страницы	BOOL	
res	Результат обработки задания подписки	<Число>	0 или код ошибки

rep_res	Результат обработки задания отчета	<Число>	0 или код ошибки
fst	Время начала	<Время>	
fen	Время окончания	<Время>	

Таблица 4-9. Список полей структуры CSet

Значения CSet представляют собой параметры отображаемых групп. Доступ к значениям структуры CSet осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Cell предназначен для хранения записей, входящих в группы CSet. Структура Cell - это запись следующего вида, наследуемая от OGeneric.

```
Cell* = RECORD (OGeneric)
      crec-: VariableRef;
      rfs-: ARRAY @UNTAG N_CELL_MAX_COLS OF VariableRef;
      icset-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-10.

Поле	Наименование	Значения	Примечание
crec	Ссылка на текущую запись	<Ссылка>	
rfs	Массив ссылок на значения	<Массив Ссылок>	Ссылки на значения переменных в столбцах CSet
icset	Идентификатор группы CSet	<Число>	Номер в таблице CSet

Таблица 4-10. Список полей структуры Cell

Значения Cell представляют собой ячейки, в которых хранятся относящиеся к группам записи. Доступ к значениям структуры Cell осуществляется по именам, заданным в поле name структуры OGeneric.

Тип VFile предназначен для файлов разделенного доступа, состоящих из переменных/параметров. Структура VFile - это запись следующего вида, наследуемая от OGeneric.

```
VFile* = RECORD (OGeneric)
      fname-: Sb.STag;
      descr-: Sb.SDescr;
      ftype-: Tb.Int32;
      fvbegin-: Tb.Int32;
      n_vars-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-11.

Поле	Наименование	Значения	Примечание
fname	Имя файла	<Имя>	
descr	Описание файла	<Строка>	
ftype	Тип файла	<Число>	1-VFILE 2-EKP 3-ALARM 4-DATABOOK 5-TS 6-DOV
fvbegin	Индекс первой переменной в таблице FVars	<Число>	
n_vars	Количество относящихся к файлу переменных	<Число>	

Таблица 4-11. Список полей структуры VFile

Значения VFile представляют собой именованные наборы переменных/параметров, условно названные файлами. Доступ к значениям структуры VFile осуществляется по именам, заданным в поле name структуры OGeneric.

Тип FVar предназначен для хранения переменных/параметров, относящихся к файлам VFile. Структура FVar - это запись следующего вида, наследуемая от OGeneric.

```
FVar* = RECORD (OGeneric)
      vref-: VariableRef;
      params-: Sb.String256;
      vfoid-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-12.

Поле	Наименование	Значения	Примечание
vref	Ссылка на элемент файла	<Ссылка>	Ссылка на переменную процесса или объект
params	Параметры, устанавливаемые для переменной	<Строка>	Например, требуемые значения
vfoid	Индекс файла в таблице	<Число>	

Таблица 4-12. Список полей структуры FVar

Значения FVar представляют собой переменные/параметры, относящиеся к файлам. Доступ к значениям структуры FVar осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Arch предназначен для хранения типов архивов. Структура Arch - это запись следующего вида, наследуемая от OGeneric.

```
Arch* = RECORD (OGeneric)
      nglobal-: Tb.Int32;
      cglobal-: Tb.Int32;
      afmt-: Tb.Int16;
      aclass-: Tb.Int16;
      artype-: Tb.Int16;
      n_files-: Tb.Int16;
      t_hh-: Tb.Int32;
      t_mm-: Tb.Int32;
      t_ss-: Tb.Int32;
      delta_t-: Tb.Int32;
      aperture-: Tb.Real32;
      period-: Tb.Int32;
      t_activated-: Tb.Time;
      update_time-: Tb.Time;
      record_time-: Tb.Int32;
      n_updated-: Tb.Int32;
      n_recorded-: Tb.Int32;
      n_updt_ctr-: Tb.Int32;
      n_rec_ctr-: Tb.Int32;
      update_res-: Tb.Int32;
      record_res-: Tb.Int32;
      start_clock-: Tb.Time;
      end_clock-: Tb.Time;
      mp_fileno-: Tb.Int32;
      mp_mpos-: Tb.Int32;
      max_mpos-: Tb.Int32;
      is_activated-: BOOLEAN;
END;
```

Список полей структуры приведен в таблице 4-13.

Поле	Наименование	Значения	Примечание
nglobal	Номер архивного глобала	<Число>	Глобал a
cglobal	Номер заголовочного глобала архива	<Число>	Глобал c
afmt	Формат файла архива	0-нет 1-fta 2-fba 3-foa	fta – текстовый формат архива; fba – двоичный формат архива foa – двоичный формат оперативного архива
aclass	Класс реализации алгоритма архивации	0-нет 1-QUEUE 2-OLD 3-CALC 4-LOG 5-PERIOD	QUEUE –архив по очередям сообщений; OLD – архив изменений значения/качества; CALC – архив расчетных параметров; LOG – архив сигнализаций; PERIOD – непрерывный архив
artype	Тип архива	0-нет 1-NGLOB 2-SYSLOG 3-OPER 4-STA 5-LTA 6-TRA	NGLOB – текущие данные глобала n; SYSLOG – архив системных событий; OPER – оперативный архив; STA – среднесрочный архив; LTA – долговременный архив; TRA – аварийный архив
t_hh	Время в часах	<Число>	
t_mm	Время в минутах	<Число>	
t_ss	Время в секундах	<Число>	
delta_t	Интервал времени	<Число>	
aperture	Апертура по умолчанию	<Значение>	
period	Период непрерывной архивации	<Число>	
t_activated	Время активации	<Время>	
update_time	Время формирования архивной записи	<Время>	
record_time	Время сброса архивной записи на носитель	<Время>	
n_updated	Количество сформированных данных	<Число>	
n_recorded	Количество записанных данных	<Число>	
n_updt_ctr	Счетчик обновленных данных	<Число>	
n_rec_ctr	Счетчик записанных данных	<Число>	
update_res	Результат формирования архива	<Число>	
record_res	Результат записи архива на носитель	<Число>	

start_clock	Время старта	<Время>	
end_clock	Время останова	<Время>	
mp_fileno	Номер файла позиции	<Число>	Адрес записи: (fileno,mpos)
mp_mpos	Номер записи позиции	<Число>	Адрес записи: (fileno,mpos)
max_mpos	Предел записи позиции	<Число>	
is_activated	Флаг активации	<0|1>	

Таблица 4-13. Список полей структуры Arch

Значения Arch представляют собой параметры, относящиеся к архивным файлам в целом. Доступ к значениям структуры Arch осуществляется по именам, заданным в поле name структуры OGeneric.

Тип Gstat предназначен для хранения ролей с правами пользователей. Структура Gstat - это запись следующего вида, наследуемая от OGeneric.

```
Gstat* = RECORD (SO.OGeneric)
      globtype-: Tb.Char8;
      done_saved-: BOOLEAN;
      refresh_counter-: Tb.Int32;
      t_modified-, t_parmodified-, t_saved-, t_parsaved-: Tb.Time;
      task_modified-, task_saved-: Tb.Int32;
      persist_interval-, persist_count-: Tb.Int16;
      in_groups-, in_xgroups-: Tb.Int32;
      snd_done-, snd_missed-, snd_redone-: Tb.Int32;
      rcv_done-, rcv_missed-, rcv_redone-, rcv_lost-,
            rcv_err-: Tb.Int32;
      ncadr_sum-, netcadr_sum-: Tb.Int32;
      ncadr_last-, cnt_val_sync-: Tb.Int32;
      chids-: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-14.

Поле	Наименование	Значения	Примечание
globtype	Тип глобала	<Символ>	Перечень типов приведен в таблице 1
done_saved	Сохранение выполнено	BOOL	
refresh_counter	Счетчик обновлений	<Число>	
t_modified	Время модификации	<Время>	
t_parmodified	Время модификации параметра	<Время>	
t_saved	Время сохранения	<Время>	
t_parsaved	Время сохранения параметра	<Время>	
task_modified	task_id модифицирующей задачи	<Число>	Номер в таблице Tasks
task_saved	task_id сохраняющей задачи	<Число>	Аналогично
persist_interval	Интервал сохранения в файл	<Число>	
persist_count	Счетчик сохранения в файл	<Число>	
done_saved	Сохранение выполнено	BOOL	
parm_modified	Параметр модифицирован	BOOL	

in_groups	Входит во множество серверных групп	<Число>	Число представляет собой множество идентификаторов групп
in_xgroups	Входит во множество групп агентов обмена	<Число>	Аналогично
snd_done	Счетчик отправленных	<Число>	
snd_missed	Счетчик пропущенных при отправке	<Число>	
snd_redone	Счетчик повторенных при отправке	<Число>	
rcv_done	Счетчик принятых	<Число>	
rcv_missed	Счетчик пропущенных при приеме	<Число>	
rcv_redone	Счетчик повторенных при приеме	<Число>	
rcv_lost	Счетчик потерянных при приеме	<Число>	
rcv_err	Ошибка приема	<Число>	Согласно таблице кодов ошибок
ncadr_sum	Суммарный номер кадра	<Число>	
netcadr_sum	Сетевой номер кадра	<Число>	
ncadr_last	Последний номер кадра	<Число>	
cnt_val_sync	Значение счетчика синхронизации	<Число>	
chids	Набор идентификаторов характеристик	Приложение 6	Множесто oid из таблицы Characteristics

Таблица 4-14. Список полей структуры Gstat

Значения Gstat представляют собой именованные роли пользовательского доступа. Доступ к значениям структуры Gstat осуществляется по именам, заданным в поле name структуры OGeneric.

Gstat предоставляет программам дополнительную информацию о глобалах, доступных из самих глобалов. Следует отметить, программа, получающая доступ к глобалам, уже имеет информацию общего характера о подключенных глобалах.

Тип Alarm предназначен для хранения значений сигнализаций. Структура Alarm - это запись следующего вида, наследуемая от OGeneric.

```
Alarm* = RECORD @UNTAG (OGeneric)
      e16-: E16B;
      vref-: VariableRef;
      timestamp-: Tb.Time;
      t_modified-: Tb.Time;
      nchanges-: Tb.Int32;
      dummy: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 4-15.

Поле	Наименование	Значения	Примечание
e16	Структура сигнализации	<Сигнализация>	

vref	Ссылка на элемент, вызвавший сигнализацию	<Ссылка>	Ссылка на переменную объекта со значением
timestamp	Метка времени сигнализации	<Время>	Время возникновения
t_modified	Время изменения	<Время>	Меняется при любой модификации
nchanges	Число изменений	<Число>	

Таблица 4-15. Список полей структуры Alarm

Значения Alarm представляют собой массив сигнализаций, разбитый на области, в зависимости от условий выборок. Доступ к значениям структуры Alarm осуществляется по индексам, соответствующим областям выборок.

Тип Repdata предназначен для хранения данных отчетов в текстовом и бинарном виде. Структура Repdata - это запись следующего вида, наследуемая от OGeneric.

```
Repdata* = RECORD @UNTAG (OGeneric)
      atype-: Sb.VarType;
      srvstatus-: Tb.Int16;
      source_id-, subscr_id-, flags-: Tb.Int32;
      srec-: StringSRec;      (* 272 bytes *)
END;
```

Список полей структуры приведен в таблице 4-16.

Поле	Наименование	Значения	Примечание
atype	Тип атрибута	<Тип>	
srvstatus	Статус сервера	<Число>	Enum SRV_STAT
source_id	Идентификатор источника данных	<Число>	
subscr_id	Идентификатор подписки	<Число>	
flags	Флаги передачи	<Число>	Множество
srec	Данные текстовые или двоичные		

Таблица 4-16. Список полей структуры Repdata

Значения Repdata представляют собой текстовые или бинарные данные, в зависимости от значения atype, что иллюстрируется таблицей 4-17.

Имя	Значение	Описание
SG.SEQ_DATA	-1	Каждая структура Repdata содержит 1-6 записей SG.ARecSeq (Приложение 11), начиная от адреса Repdata.atype
SG.SEQ_NACK	-2	
SG.SEQ_TEXT	-3	Поле Repdata.srec представляет собой отформатированный текст протокола
SG.SEQ_PROT_PAGE	-4	Разрыв страницы протокола
SG.SEQ_PROT_END	-5	Окончание данных протокола
--	0	Нет данных
	>0	Данные бинарной последовательности

Таблица 4-17. Список значений Repdata.atype

Доступ к значениям структуры Repdata осуществляется по индексам, соответствующим строкам протокола.

Тип Report предназначен для хранения управляющих данных отчетов, по записи на каждый отчет. Структура Report - это запись следующего вида, наследуемая от OGeneric.

```
      Report* = RECORD @UNTAG (OGeneric)
            cset_id-, task_id-: Tb.Int32;
            irepdata-, max_repdata-: Tb.Int32;
            inips-: Sb.String256;
            rep_id-, rep_crt-, rep_done_crt-: Tb.Int32;
            irow-: Tb.Int32;
            n_found-: Tb.Int32;
            page_rec_max-, page_rec_count-, pages_max-, pages_count-:
Tb.Int32;
            nset_rep-, nset_id-, nset_crt-: Tb.Int32;
            pct_done-: Tb.Int32;
            res-: Tb.Int32;
            dummy: Tb.Int32;
            st-, en-: Tb.Time;
      END;
```

Список полей структуры приведен в таблице 4-18.

Поле	Наименование	Значения	Примечание
cset_id	Номер записи CSet	<Число>	
task_id	Имя задачи	<Число>	Динамически присваивается по нагрузке
irepdata	Номер записи Repdata	<Число>	
max_repdata	Максимальное количество записей Repdata	<Число>	
rep_id	Идентификатор подписки	<Число>	subscr_id в структуре CSet
rep_crt	Идентификатор текущей подписки	<Число>	subscr_crt
rep_done_crt	Идентификатор выполненой подписки	<Число>	
irow	Память строки протокола	<Число>	При многостраничном останове
n_found	Число найденных строк	<Число>	
page_rec_max	Максимальное число строк на странице	<Число>	
page_rec_count	Число строк на странице	<Число>	
pages_max	Максимальное число страниц	<Число>	
pages_count	Число страниц	<Число>	
nset_rep	Идентификатор набора страниц протокола	<Число>	
nset_id	Идентификатор набора страниц	<Число>	
nset_crt	Идентификатор текущего набора страниц	<Число>	
pct_done	Процент готовности	<0-100>	
res	Результат выполнения	<Число>	0, <0 при ошибке
st	Время начала	<Время>	
en	Время окончания	<Время>	

Таблица 4-18. Список полей структуры Report

Значения Repdata представляют собой массив отчетов, разбитый на области, определяющие состояние отчета. Каждый отчет связан с элементом CSet и частью массива Repdata.

ПРИЛОЖЕНИЕ 5 – ТАБЛИЦА ЗНАЧЕНИЙ СТАТУСОВ ПРИЛОЖЕНИЙ

Перечень значений и описаний статусов приведен в таблице 5-1.

Значение	Статус	Описание	Цвет
0	OFF	Не используется	малиновый
1	INIT	Инициализация	малиновый
2	LOAD	Загрузка	желтый
3	ERROR	Ошибка мониторинга	красный
4	END	Задача остановлена	малиновый
5	WAITING	Ожидание работы	красный
6	STARTING	Стартует	желтый
7	SINGLE	Одиночный	зеленый
8	PRIMARY	Основной	зеленый
9	BACKUP	Резервный	темно-зеленый
10	FAILOVER	Аварийное переключение	синий
11	FREEZE	Заморозка расчета	серый

Таблица 5-1. Значения и описания статусов

ПРИЛОЖЕНИЕ 6 – ОПЦИИ ТРАССИРОВКИ ПО СКАДИ

Для всех процессов СКОЯТ-МОДЕЛЬ можно динамически устанавливать и менять уровень трассировки. Изменение переменной `Task:@T.trace_level` отслеживается динамически задачами СКОЯТ-МОДЕЛЬ. Трассировка для задач задается следующими способами:

- из командной строки указанием `–trace_level 34`
- в конфигурационном файле `trace_level=1022`
- динамически из отладчика. Добавить переменную `Task:#0.trace_level`, задать значение `1=36`

Значение величины трассировки представляет собой сумму значений бит, согласно таблице 6-1.

Бит	Значение	Коммуникация	Наименование	Обозначение
0	1	нет	Цикл обновления	Tb.TRAL_RUNTIME
1	2	да	Транспортный уровень	Tb.TRAL_TRANSPORT
2	4	да	Состояние поллинга	Tb.TRAL_POLLING
3	8	да	Логический уровень сообщений	Tb.TRAL_MESSAGE
4	16	да	Детализация уровня сообщений	Tb.TRAL_MESSAGE_DATA
5	32	да	Ошибки коммуникации	Tb.TRAL_ERRORS
6	64	нет	Ошибки задачи	Sb.TRAL_ERRORS
7	128	нет	Уровень обработки в задаче	Sb.TRAL_RUNTIME
8	256	нет	Детализация уровня обработки	Sb.TRAL_AUX
9	512	нет	Данные уровня обработки	Sb.TRAL_DATA
10	1024	нет	Времена обработки	Sb.TRAL_TIMES

Таблица 6-1. Список значений трассировки

Для примера, значение 1022 означает установленные биты {1..9} из таблицы. Все проверки на вхождение бита завершатся включением консольной печати или вывода в лог сообщений для Tb.TRAL_TRANSPORT, Tb.TRAL_POLLING, Tb.TRAL_MESSAGE, Tb.TRAL_MESSAGE_DATA, Tb.TRAL_ERRORS, Sb.TRAL_ERRORS, Sb.TRAL_RUNTIME, Sb.TRAL_AUX, Sb.TRAL_DATA.

Консольная печать, требуемая для отладки, включается после установки соответствующих опций трассировки. Следует заметить, что при необходимости трассировки инициализации задачи, нужно установить опцию trace_level в самом начале конфигурационного файла. Например trace_level=1 в начале конфигурационного файла позволяет трассировать разбор параметров, заданных в конфигурационном файле непосредственно за опцией трассировки.

ПРИЛОЖЕНИЕ 7 – ЗАПРОСЫ И ОТВЕТЫ СЕРВИСА WOMARK

Интерфейс сервиса WoMark реализует функции управления подпиской и выполнения команд.

Перечень выполняемых команд представлен в таблице 7-1.

Имя	Наименование	Примечание
PutVariableValues	Передать массив переменных	Специальное сообщение об изменении данных
OutputSelection	Передать массив выборок пользователя	Сообщение о выборе объектов пользователем
GetResource	Получить имя ресурса контейнера	
GetCategory	Имя курса	"Visitor", "Student", "Examinee", "Instructor", ""
SetCategory	Имя урока или лабораторной работы	"Visitor", "Student", "Examinee", "Instructor", ""
Upload	Загрузить ресурс	
Download	Выгрузить ресурс	
Getload	Проверка загрузки ресурса	
GetAllLoads	Запрос всех активных видимых ресурсов	

Таблица 7-1. Перечень команд WoMark

Структура данных управления подпиской SubscriptionMessage, передаваемая между клиентом и сервисом WoMark, имеет следующее описание.

```
type SubscriptionMessage struct {
	Timestamp   int64  `json:"timestamp"`
	Topic       string `json:"topic"`
	Section           string `json:"section"`
	Course            string `json:"course"`
	Lessname    string `json:"lessname"`
	Elapsed           int `json:"elapsed"`
	Command           string `json:"command"`
}
```

Поля структуры SubscriptionMessage приведены в таблице 7-2. Функций управления подпиской всего 2, это команды Topic=”LabLesson” и Topic=”LabFinish”.

Поле	Наименование	Значения	Имя JSON
Timestamp	Метка времени	<время>	json:"timestamp"
Topic	Наименование темы подписки	<строка>	json:"topic"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Course	Имя курса	<строка>	json:"course"
Lessname	Имя урока или лабораторной работы	<строка>	json:"lessname"
Elapsed	Число секунд на выполнение задания	<число>	json:"elapsed"
Command	Имя команды оценивания	<строка>	json:"command"

Таблица 7-2. Поля типа SubscriptionMessage

В ответ на SubscriptionMessage от клиента сервис WoMark передает сообщение SubPubMessage, адресуемое клиенту. На каждое SubscriptionMessage от клиента

передается только одно ответное сообщение SubPubMessage, которое имеет следующее описание.

```
type SubPubMessage struct {
	Timestamp   int64  `json:"timestamp"`
	Topic       string `json:"topic"`
	Section           string `json:"section"`
	Course            string `json:"course"`
	Lessname    string `json:"lessname"`
	Tused       string `json:"tused"`
	Result            string `json:"result"`
	Comments    string `json:"comments"`
	Report            string `json:"report"`
}
```

Поля структуры SubPubMessage приведены в таблице 7-3.

Поле	Наименование	Значения	Имя JSON
Timestamp	Метка времени	<время>	json:"timestamp"
Topic	Наименование темы подписки	<строка>	json:"topic"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Course	Имя курса	<строка>	json:"course"
Lessname	Имя урока или лабораторной работы	<строка>	json:"lessname"
Tused	Время, потраченное на выполнение задания	<строка>	json:"tused"
Result	Результат выполнения задания	<строка>	json:"result"
Comments	Текстовый комментарий оценивания действий	<строка>	json:"comments"
Report	Имя скрипта выполнения оценки	<строка>	json:"report"

Таблица 7-3. Поля типа SubPubMessage

После установлении подписки сервис начинает передавать данные клиентскому ПО. Структура данных SubOutputMessage, постоянно передаваемая от сервиса WoMark к клиенту по подписке, имеет следующее описание.

```
type SubOutputMessage struct {
	Timestamp   int64 `json:"timestamp"`
	Section           string      `json:"section"`
	Lstatus           string      `json:"lstatus"`
	Isup        bool  `json:"isup"`
	CountUsage  string      `json:"countusage"`
	CountDown   string      `json:"countdown"`
}
```

Поля структуры SubOutputMessage приведены в таблице 7-4.

Поле	Наименование	Значения	Имя JSON
Timestamp	Метка времени	<время>	json:"timestamp"
Section	Имя секции – имя контейнера	<строка>	json:"section"
LStatus	Статус выполнения	"none", "uploading", "uploaded", "failed", "downloading", "downloaded", "working", "finished"	json:"lstatus"

Isup	Флаг состояния загрузки	<булевское>	json:"isup"
CountUsage	Счет использования ресурса	<строка>	json:"countusage"
CountDown	Обратный отсчет оставшегося времени выполнения задания	<строка>	json:"countdown"

Таблица 7-4. Поля типа SubOutputMessage

Функции выполнения команд инициируются сообщением RpcMessage, передаваемым от клиента сервису WoMark. RpcMessage имеет следующее описание.

```
type RpcMessage struct {
	Timestamp   int64  `json:"timestamp"`
	Section           string `json:"section"`
	Group       string `json:"group"`
	VarNames    string `json:"varnames"`
	Command           string `json:"command"`
}
```

Поля структуры RpcMessage приведены в таблице 7-5.

Поле	Наименование	Значения	Имя JSON
Timestamp	Метка времени	<время>	json:"timestamp"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Group	Имя ресурса группы	<строка>	json:"group"
VarNames	Передаваемые имена переменных	<строка>	json:"varnames"
Command	Команда выполнения на сервисе	<строка>	json:"command"

Таблица 7-5. Поля типа RpcMessage

Отдельными сообщениями осуществляется выполнением команды PutVariableValues. Команда передает изменившиеся данные от клиента на сервис WoMark. PutVariableValues использует сообщения VarsValsMessage и ValMessage со следующими описаниями.

```
type ValMessage struct {
	Vname string `json:"vname"`
	Val         string `json:"val"`
	Atype string `json:"atype"`
}

type VarsValsMessage struct {
	Topic       string `json:"topic"`
	Group       string `json:"group"`
	Vars []ValMessage `json:"vars"`
}
```

Поля структуры ValMessage приведены в таблице 7-6.

Поле	Наименование	Значения	Имя JSON
vname	Имя переменной	<строка>	
val	Значение переменной	<строка>	
atype	Тип переменной	<строка>	Поле «Имя»

Таблица 7-6. Поля типа ValMessage

Поля структуры VarsValsMessage приведены в таблице 7-7. Формирование данного сообщения на стороне клиента осуществляется на основе принимаемого от СВУ сообщения dataMessage. Данные от СВУ проходят фильтрацию, требуемые в контексте обучающей задачи и формируют массив VarsValsMessage.

Поле	Наименование	Значения	Имя JSON
Topic	Наименование темы подписки	<строка>	json:"topic"

Group	Имя ресурса группы	<строка>	json:"group"
vars	Массив переменных	[]valMessage	Массив valMessage

Таблица 7-7. Поля типа VarsValsMessage

Ответом на посылку RpcMessage и массив данных в VarsValsMessage является сообщение RpcResponse, передаваемое от сервиса WoMark клиенту. RpcResponse имеет следующее описание.

```
type RpcResponse struct {
	Timestamp   int64  `json:"timestamp"`
	Topic       string `json:"topic"`
	Section           string `json:"section"`
	Group       string `json:"group"`
	ResourceID  int `json:"resource_id"`
	Result string     `json:"result"`
}
```

Поля структуры RpcResponse приведены в таблице 7-8.

Поле	Наименование	Значения	Имя JSON
Timestamp	Метка времени	<время>	json:"timestamp"
Topic	Наименование темы подписки	<строка>	json:"topic"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Group	Имя ресурса группы	<строка>	json:"group"
ResourceID	Передаваемые имена переменных	<число>	json:"resource_id"
Result	Результат выполнения задания	<строка>	json:"result"

Таблица 7-8. Поля типа RpcResponse

ПРИЛОЖЕНИЕ 8 – ЗАПРОСЫ И ОТВЕТЫ СЕРВИСА WODATA

Сервис WoData на запрос GetUserInfo возвращает client.Info().
Интерфейс сервиса WoData реализует функции выполнения команд.
Перечень выполняемых команд представлен в таблице 8-1.

Имя	Наименование	Примечание
GetUserInfo	Получить информацию о пользователе	Структура ScadiID, таблица 2-2 приложения 2
QueryMarks	Получить информацию об оценках работ	Структура MarksMessage, таблица 8-4

Таблица 8-1. Перечень команд WoMark

Структура данных запроса оценок MarksQueryMessage, передаваемая между клиентом и сервисом WoData, имеет следующее описание.

```
type MarksQueryMessage struct {
	Username    string      `json:"username"`
	Course          string      `json:"course"`
	Section         string      `json:"section"`
	Lessname    string      `json:"lessname"`
	WithCancel  bool  `json:"withcancel"`
}
```

Поля структуры MarksQueryMessage приведены в таблице 8-2.

Поле	Наименование	Значения	Имя JSON
Username	Имя пользователя	<строка>	json:"username"
Course	Имя курса	<строка>	json:"course"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Lessname	Имя урока или лабораторной работы	<строка>	json:"lessname"
WithCancel	Включая незавершенные задания	<логический>	json:"with_cancel"

Таблица 8-2. Поля типа MarksQueryMessage

В ответ на MarksQueryMessage от клиента сервис WoData передает сообщение MarksMessage, адресуемое клиенту. На каждое MarksQueryMessage от клиента передается только одно ответное сообщение MarksMessage, которое состоит из сообщений MarkMessage для каждой оценки.

```
type MarkMessage struct {
	Username    string      `json:"username"`
	Course          string      `json:"course"`
	Section         string      `json:"section"`
	Lessname    string      `json:"lessname"`
	Tdone           int64 `json:"tdone"`
	Tupdate         int64 `json:"tupdate"`
	Result          string      `json:"result"`
	Comments    string      `json:"comments"`
	Report          string      `json:"report"`
	FinStatus   string      `json:"finstatus"`
}
type MarksMessage struct {
	Topic       string      `json:"topic"`
	Results         []MarkMessage     `json:"marks"`
}
```

Поля структуры MarkMessage приведены в таблице 8-3.

Поле	Наименование	Значения	Имя JSON
Username	Имя пользователя	<строка>	json:"username"
Course	Имя курса	<строка>	json:"course"
Section	Имя секции – имя контейнера	<строка>	json:"section"
Lessname	Имя урока или лабораторной работы	<строка>	json:"lessname"
Tdone	Время выполнения задания	<время>	json:"tdone"
Tdone	Время обновления результата	<время>	json:"tupdate"
Result	Результат выполнения задания	<строка>	json:"result"
Comments	Текстовый комментарий оценивания действий	<строка>	json:"comments"
FinStatus	Статус окончания	wrk closed cancelled finished expired	json:"finstatus"

Таблица 8-3. Поля типа MarkMessage

Поля структуры MarksMessage приведены в таблице 8-4.

Поле	Наименование	Значения	Имя JSON
Topic	Наименование темы подписки	<строка>	json:"topic"
Marks	Оценки	[]MarkMessage	json:"marks"

Таблица 8-4. Поля типа MarksMessage

ПРИЛОЖЕНИЕ 9 – ПРИМЕР MARKUP ГИПЕРТЕКСТА ДЛЯ УМО

Данный гипертекст приведен для УМО управления насосами сепаратора.

В гипертексте могут использоваться следующие команды управления, таблица 9-1.

Команда	Наименование	Параметры
[Завершить]	Завершение выполнения задания и вызов скрипта оценивания	[(кнопка)] (!finishTask,скрипт оценки)
[Выбрать]	Добавление выбранной переменной в данную серверную группу	[(кнопка)](!addCode,имя группы)
[Удалить]	Удаление выбранной переменной из данной серверной группы	[(кнопка)](!removeCode,имя группы)

Таблица 9-1. Команды управления и кнопки

В гипертексте могут использоваться следующие параметры запуска задания, таблица 9-2:

Команда	Наименование	Параметры
Reset	Начальное состояние задания	<число> – номер начального состояния
RstObj	Имя объекта, в котором задается начальное состояние задания	<строка> - имя объекта. Если такой объект не определен в серверной БД, то данный параметр не задается
Elapsed	Время, отведенное на выполнение задания	<секунды>
Interceptor	Имя переменной, значение которой передается на сервер при ее изменении	<строка> - имя (код) переменной
FrmSrc_main	Имя видеокадра, загружаемого в головной фрейм системы верхнего уровня	<строка>
FrmSrc_trend	Имя видеокадра, загружаемого во фрейм отображения графика системы верхнего уровня	<строка>
Trend	Имя переменной, выводимой на график	<строка> - имя (код) переменной
IntervalSec	Интервал отрисовки графика	<секунды>. Рекомендуемое значение – время, отведенное на выполнение задания + 20 секунд

Таблица 9-2. Параметры запуска задания

```
# Задание 104


## Управление насосами сепаратора
```

Переключиться с насоса 4RB61D01 на насос 4RB62D01 и изолировать насос 4RB61D01 для проведения его ремонта.

Требуемые действия.

1. Вывести расход на насосах сепаратора и суммарный расход на график.
2. Открыть задвижки 4RB62S01 и 4RB62S03.
3. Выключить насос 4RB61D01.
4. Включить насос 4RB62D01.
5. Закрыть задвижки 4RB61S01 и 4RB61S03.
6. Перевести насос 4RB61D01 в состояние ремонта.
7. Зафиксировать изменение расходов на графике.

Завершение работы

[![Завершить](../../../images/btn_fin.png)](!finishTask,./labs/a1/rb60/104/Scl04)

Elapsed 480

Reset 1

FrmSrc_main NPP_models/RB2K.svg

FrmSrc_trend NPP_models/_TT3.svg

Trend RB2KF001_XQ01

Trend RB2KF002_XQ01

Trend 40RBR60CF001_XQ01

IntervalSec 500

ПРИЛОЖЕНИЕ 10 - ДАННЫЕ ПОЛЬЗОВАТЕЛЬСКИХ РЕЛЯЦИОННЫХ ТАБЛИЦ

Тип AI предназначен для обработки аналоговых сигналов. Объекты AI представляют собой переменные процесса и наследуются от OGeneric, OConfigurable, OManagable. Объекты включают в себя поля OGeneric, OConfigurable, OManagable и поля, относящиеся к непосредственно AI. Структура AI – это запись следующего вида, наследуемая от OManagable.

```
AI* = @EXTENS RECORD (OManagable)
      units*: Sb.String16;
      lf*, hf*, la*, ha*, lw*, hw*, lr*, hr*: Tb.Real32;
      ainp*, av*: SO.R4QTS;
      av_sim*, aper*, zone*: Tb.Real32;
      ndigits*: Tb.Int32;
END;
```

Список полей структуры приведен в таблице 10-1.

Поле	Наименование	Значения	Примечание
units	Единицы измерения	<Строка>	
lf	Нижний предел	<Предел>	Значение или ~ если не задано
hf	Верхний предел	<Предел>	Аналогично
la	Нижняя аварийная граница	<Предел>	Аналогично
ha	Верхняя аварийная граница	<Предел>	Аналогично
lw	Нижняя предупредительная граница	<Предел>	Аналогично
hw	Верхняя предупредительная граница	<Предел>	Аналогично
lr	Нижняя регламентная граница	<Предел>	Аналогично
hr	Верхняя регламентная граница	<Предел>	Аналогично
ainp	Входное аналоговое значение	<Значение, Качество, МеткаВремени>	
av	Текущее аналоговое значение	<Значение, Качество, МеткаВремени>	
av_sim	Имитируемое аналоговое значение	<Значение>	
aper	Апертура архивации	<Значение>	
zone	Зона возврата гистерезиса	<Значение>	
ndigits	Число значащих цифр	<Число>	

Таблица 10-1. Список полей структуры AI

Значения AI представляют собой поля объектов, определяющих аналоговые переменные. Переменные имеют значение, входное значение, а также диапазон и границы сигнализации. Доступ к значениям структуры AI осуществляется по именам, заданным в поле name структуры OGeneric.

Тип DI предназначен для обработки дискретных сигналов. Объекты DI представляют собой переменные процесса и наследуются от OGeneric, OConfigurable,

OManagable. Объекты включают в себя поля OGeneric, OConfigurable, OManagable и поля, относящиеся к непосредственно DI. Структура DI – это запись следующего вида, наследуемая от OManagable.

```
DI* = @EXTENS RECORD (OManagable)
      dinp*, dv*: SO.I2QTS;
      dv_sim*: Tb.Int8;
      dv_cmd*: SO.C8B;
      dtype*: Tb.Char8;
      inver*: BOOLEAN;
END;
```

Список полей структуры приведен в таблице 10-2.

Поле	Наименование	Значения	Примечание
dinp	Входное дискретное значение	<Число1, Качество, МеткаВремени>	Число1 размером 1 байт, булевские и целые значения
dv	Текущее дискретное значение	<Число1, Качество, МеткаВремени>	Аналогично
dv_sim	Имитируемое дискретное значение	<Число1>	Аналогично
dv_cmd	Команда управления	Структура C8	
dtype	Тип дискрета	<Символ>	
inver	Прямой или инверсный	BOOL	

Таблица 10-2. Список полей структуры DI

Значения DI представляют собой поля объектов, определяющих дискретные переменные. Переменные имеют значение, входное значение, а также параметры типизации. Доступ к значениям структуры DI осуществляется по именам, заданным в поле name структуры OGeneric.

Тип NI предназначен для обработки целочисленных сигналов. Объекты NI представляют собой переменные процесса и наследуются от OGeneric, OConfigurable, OManagable. Объекты включают в себя поля OGeneric, OConfigurable, OManagable и поля, относящиеся к непосредственно NI. Структура NI – это запись следующего вида, наследуемая от OManagable.

```
NI* = @EXTENS RECORD (OManagable)
      ninp*, nv*: SO.I4QTS;
END;
```

Список полей структуры приведен в таблице 10-3.

Поле	Наименование	Значения	Примечание
ninp	Входное целочисленное значение	<Число, Качество, МеткаВремени>	Число размером 4 байт содержит целые значения
nv	Текущее целочисленное значение	<Число, Качество, МеткаВремени>	Аналогично

Таблица 10-3. Список полей структуры NI

Значения NI представляют собой поля объектов, определяющих аналоговые переменные. Переменные имеют текущее и входное значения. Доступ к значениям структуры NI осуществляется по именам, заданным в поле name структуры OGeneric.

ПРИЛОЖЕНИЕ 11 – ФОРМАТ ПРОТОКОЛОВ ДЕЙСТВИЙ ОБУЧАЕМОГО

Формат протокола действий обучаемого представляет собой скрипт на языке Python с заранее определенным набором глобальных параметров.

Список параметров протокола приведен в таблице 11-1.

Имя	Наименование	Значения	Примечание
USERINFO	Информация о пользователе	‘name’ ‘category’ ‘activity’ ‘department’	Зачения из БД и текущей сессии
LESSON	Описание лабораторной работы	‘section’ ‘cource’ ‘lessname’ ‘elapsed’ ‘time’	Значения из проведенного занятия (лабораторной работы)
OUTPUT_SELECTION	Выборки в группы по заданию		
VARIABLES_VALUES	Список значений переменных	'05_03_2025__15_19_31_764' - время	[('SVALX, AI:~40RNR92CL901_XQ57.av', '114.64','R4')]
STATUS	Состояние сессии	‘finished’	

Таблица 11-1. Список параметров протокола

Ниже представлен типовой протокол действий обучаемого.

```
USERINFO=dict()
USERINFO['name']='dvdagaev'
USERINFO['category']='Examinee'
USERINFO['activity']='basic'
USERINFO['department']='department'
LESSON=dict()
LESSON['section']='oisc'
LESSON['course']='a1'
LESSON['lessname']='102'
LESSON['elapsed']='480'
LESSON['time']='06_03_2025__14_10_55_156'
OUTPUT_SELECTION=dict()
VARIABLES_VALUES=dict()
VARIABLES_VALUES['06_03_2025__14_02_56_608']=['',
[('DI:~40RNR92AS003_Z0.dv','1;48;1741258975','I2QTS')]]
VARIABLES_VALUES['06_03_2025__14_02_56_608']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.99','R4')]]
VARIABLES_VALUES['06_03_2025__14_02_56_609']=['',
[('QUALITY,AI:~40RNR92CL901_XQ57.av','53','I2')]]
VARIABLES_VALUES['06_03_2025__14_02_57_639']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.76','R4')]]
VARIABLES_VALUES['06_03_2025__14_02_58_660']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.60','R4')]]
VARIABLES_VALUES['06_03_2025__14_02_59_678']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.96','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_00_695']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.71','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_01_560']=['',
[('DI:~40RNR92AS003_OU.dcmd','2','I2')]]
VARIABLES_VALUES['06_03_2025__14_03_01_718']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.75','R4')]]
```

```
VARIABLES_VALUES['06_03_2025__14_03_02_738']=['',
[('DI:~40RNR92AS003_Z0.dv','2;48;1741258982','I2QTS')]]
VARIABLES_VALUES['06_03_2025__14_03_02_738']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.54','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_03_753']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','114.85','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_04_676']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','112.35','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_05_764']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','109.12','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_06_685']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','105.54','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_07_718']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','102.94','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_08_738']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','99.63','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_08_738']=['',
[('QUALITY,AI:~40RNR92CL901_XQ57.av','48','I2')]]
VARIABLES_VALUES['06_03_2025__14_03_09_760']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','97.26','R4')]]
VARIABLES_VALUES['06_03_2025__14_03_10_779']=['',
[('SVALX,AI:~40RNR92CL901_XQ57.av','93.95','R4')]]
STATUS='finished'
```

ПЕРЕЧЕНЬ ТЕРМИНОВ И СОКРАЩЕНИЙ

АЭС	- Атомная Электростанция
ОС	- Операционная система
ПО	- Программное обеспечение
СВУ	- Система верхнего уровня
СКАДИ	- Средства комплексной автоматизации и диагностики
УМО	- Учебно-методическое обеспечение
HMI	- Human Machine Interface
JSON	- JavaScript Object Notation
SVG	- Scalable Vector Graphics

ЛИСТ ИЗМЕНЕНИЙ

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