Акционерное общество «Научно-технический центр «Диапром»

СИСТЕМА ПРИДЕКТИВНОЙ АНАЛИТИКИ

ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ «DIAPROM PREDICT»

Руководство по эксплуатации

ДКНБ.

Количество листов – 88

Москва 2022

Содержание

1 Общие сведения

В состав ППО «DIAPROM PREDICT» входят следующие программные модули:

– DatabaseWriter;

– Server;

– MathManagerDaemon;

– xdevmon;

– SystemWatcher;

– ModelsLauncher;

– OldRmsod;

– SpaGui.

Также в состав ППО «DIAPROM PREDICT» входят вспомогательные файлы, скрипты и библиотеки:

– набор динамических библиотек и вспомогательных файлов;

– скрипты развертывания и партиционирования таблиц БД.

В настоящем документе приведены сведения, необходимые для установки и наладки прикладного ПО.

ППО «DIAPROM PREDICT» имеет модульную структуру. Структура ППО представлена на рисунке, иллюстрирующем его модульный состав, а также движение информационных потоков между модулями (см. рисунок 1).

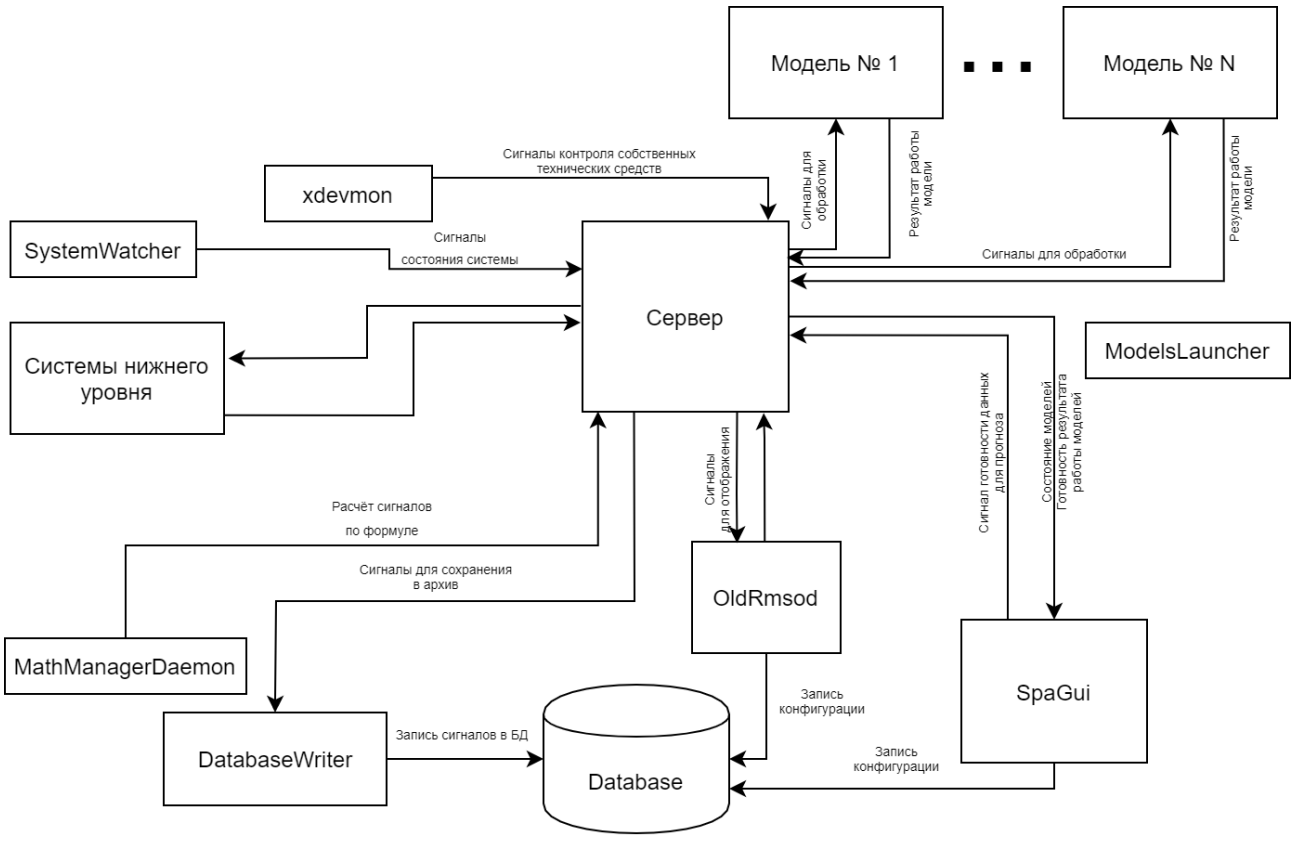
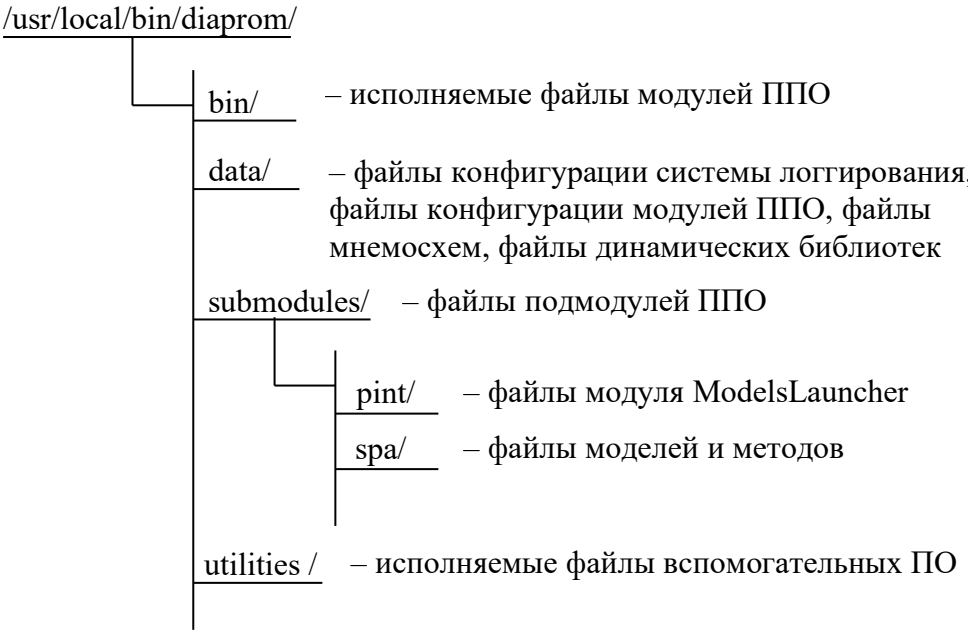


Рисунок 1 – Структурная схема ППО «DIAPROM PREDICT»

Структура директорий ППО «DIAPROM PREDICT» представлена на рисунке 2.



/etc/init.d/ – скрипты запуска модуля в качестве системного сервиса

/etc/ld.so.conf.d/skpt.conf – файл конфигурации линковщика динамических библиотек

/var/log/ – лог-файлы модулей ППО

Рисунок 2 – Структура директорий ППО «DIAPROM PREDICT»

2 Модули программного обеспечения

2.1 DatabaseWriter

2.1.1 Общие сведения о программе

DatabaseWriter является системным демоном и предназначен для сохранения данных, поступающих от источников, зарегистрированных в системе.

Запуск DatabaseWriter производится автоматически во время старта операционной системы. В случае остановки сервиса пользователем или аварийного прекращения работы сервис будет перезапущен автоматически с помощью cron.

При нормальной эксплуатации не требует никаких действий со стороны пользователя.

Сохранение данных производится без задержки по мере их поступления в модуль. Модуль не производит квантования по времени, по уровню или любых других манипуляций с данными для уменьшения их объема в базе данных, а сохраняет данные в том виде, в котором они в него поступили.

Во время запуска DatabaseWriter считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL, на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

Прочитав параметры, DatabaseWriter подключается к серверу базы данных и считывает описание всех сигналов из таблиц: sakt, system_statistics_values, calculated_values, diagnostic_knowledge_base, cla_sakt_output, predict_sakt, import_sakt.

Модуль определяет, какие сигналы подлежат сохранению. Для этого анализируется значение stor в прочитанных ранее таблицах. Если значение stor равно 1, то данные, поступающие по этому сигналу, будут сохраняться в базу данных, если значение равно нулю, то данные сохраняться не будут.

Определив, какие сигналы подлежат сохранению, модуль открывает соединение с серверами оперативных данных, регистрируется в них в качестве наблюдателя и передает список идентификаторов сигналов, которые должны сохраняться. В ответ серверы высылают срез значений, признаков достоверности и меток времени по сигналам, на которые подписался модуль. Эти данные сохраняются в базу данных. После этого DatabaseWriter переходит в “слушающий” режим. Это означает, что модуль не запрашивает данные у серверов напрямую,

а ждет, пока серверы вышлют уведомление об изменении значения или признака достоверности по сигналам, на которые он подписан. Получив уведомление, модуль сохраняет значения в базу данных.

Если у сигнала изменилась только метка времени, но значение и признак достоверности остались прежними, то серверы не вышлют уведомление об изменении, и сохранение в базе данных не произойдет.

Список серверов оперативных данных, к которым подключается DatabaseWriter, читается из таблицы communication_servers базы данных.

При потере связи с серверами DatabaseWriter автоматически пытается восстановить соединение. В случае успешного восстановления связи модуль отправляет повторный запрос на подписку на изменения сохраняемых сигналов так же, как при своем старте. Дальнейшее функционирование модуля осуществляется так же, как при запуске модуля.

Модуль выполняет логирование информации о начале и завершении своей работы, а также информации об ошибках, возникающих в процессе выполнения им своих функций.

2.1.2 Структура программы

Модуль состоит из исполняемого файла DatabaseWriter, а также нескольких дополнительных файлов конфигурации, скриптов запуска и лог-файла. Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл DatabaseWriter;

– /usr/local/bin/diaprom/data/database_writer.properties - файл конфигурации системы логирования;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /etc/init.d/DatabaseWriter - скрипт запуска модуля в качестве системного сервиса;

– /var/log/DatabaseWriter.log - лог-файл модуля. Создается при первом запуске программного модуля автоматически;

– /root/Check_services - скрипт для запуска сервисов по расписанию cron.

Помимо перечисленных файлов для корректной работы модуля необходимы:

– сервер базы данных PostgreSQL с полностью корректно установленной базой данных;

– серверы оперативных данных;

– служба запуска сервисов по расписанию cron.

Для старта во время загрузки операционной системы, сервис должен быть прописан в chkservices на уровнях 2, 3 и 5.

2.1.3 Настройка программы

Процесс настройки модуля можно разделить на три этапа.

Первый этап заключается в первичной установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Второй этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации системы логирования (database_writer.properties) наладчик может изменить степень подробности логирования. Не допускается изменение любых других параметров в этом файле.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP-адрес сервера базы данных PostgreSQL. Единственно возможный вариант - 127.0.0.1. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс.

В скрипте запуска сервисов (Check_services) указываются условия запуска модуля и команда для его выполнения. Скрипт должен быть прописан в журнале cron (/etc/crontab) с интервалом запуска, равным одной минуте.

Третий этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта, необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.1.4 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена двумя способами.

Способ первый. Выполнить команду в терминале ps -A | grep DatabaseWriter. Если модуль запущен, то ответом на команду будет номер процесса.

Способ второй. Проверить статус сервиса, выполнив команду sudo service DatabaseWriter status.

Проверка работоспособности программного модуля может быть осуществлена в несколько шагов.

Проверка лог-файлов программного модуля. Если в лог-файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их

возникновения (см. раздел Сообщения системному программисту) перед тем, как переходить к следующим проверкам.

Проверка перечня сохраняемых сигналов. Очевидно, что для сохранения данных в базу, перечень сохраняемых сигналов не должен быть пустым. Поэтому, подключившись к базе данных проекта через терминал, нужно выполнить SQL запрос к таблицам с описанием сигналов (sakt, system_statistics_values, calculated_values, diagnostic_knowledge_base, cla_sakt_output, predict_sakt, import_sakt):

SELECT count(*) FROM TABLE_NAME WHERE stor != 0;

Ответом будет строка с числом сигналов в таблице, которые должны сохраняться. Если сумма всех цифр этого числа равна нулю, то модулю сохранять нечего. В результате этого модуль будет запущен, в лог-файлах будут отсутствовать сообщения об ошибках, но сохранение данных в базу производиться не будет.

Проверка сохранения данных в базе. Для этой проверки не обязательно иметь какие-либо источники данных. Достаточно чтобы были запущены серверы оперативных данных и сервер базы данных. При каждом запуске DatabaseWriter получит срез данных по всем сигналам (значение 0, признак достоверности “Не инициализировано”, а в качестве метки времени - время запуска сервера оперативных данных), которые необходимо сохранять, и запишет их в базу. Проверить наличие данных в базе можно двумя способами.

Способ первый (предпочтительный). Воспользоваться программой с графическим пользовательским интерфейсом и, следуя руководству оператора, выбрать интересующие сигналы из архива.

Способ второй. Через терминал подключиться к базе данных и отправить соответствующий SQL запрос. При отсутствии данных еще раз убедиться, что в лог-файлах отсутствуют сообщения об ошибках, и число сохраняемых сигналов не нулевое. Если ошибок нет - связаться с разработчиками. При наличии ошибок - сначала исправить их, а затем повторить проверку сохранения данных в базе.

2.1.5 Дополнительные возможности

DatabaseWriter может быть запущен в двух режимах: демон и не демон. Первый режим запуска является штатным способом запуска программного модуля и должен быть использован в режиме эксплуатации. Второй режим работы подразумевает запуск программного модуля вручную из терминала с указанием дополнительного ключа -nd. Этот режим нужен только для отладки. В этом режиме все сообщения, которые система логирования записывает в лог-файл, отображаются в окне терминала, где запущен модуль. Функции, выполняемые программным модулем одинаковы для обоих режимов.

По умолчанию при конфигурировании базы данных все сигналы, проходящие через серверы оперативных данных, сохраняются (stror равен 1). Однако в процессе эксплуатации может возникнуть необходимость не сохранять часть сигналов. Тогда нужно выполнить команду и перезапустить серверы оперативного архива и DatabaseWriter:

UPDATE TABLE_NAME SET stor = 0 WHERE param_id=ID;

Для обеспечения максимальной производительности при записи данных в базу, DatabaseWriter использует методы записи, доступные только если и сервер базы данных, и сам модуль находятся на одной физической машине. Поэтому в качестве IP-адреса сервера базы данных может быть указан только 127.0.0.1.

2.1.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует DatabaseWriter сохраняются в лог-файл /var/log/DatabaseWriter.log.

2.2 MathManagerDaemon

2.2.1 Общие сведения о программе

MathManagerDaemon является системным демоном и предназначен для расчета вычисляемых значений на основе сигналов, поступающих от источников, зарегистрированных в системе.

Запуск MathManagerDaemon производится автоматически во время старта операционной системы. В случае остановки сервиса пользователем или аварийного прекращения работы сервис будет перезапущен автоматически с помощью cron.

При нормальной эксплуатации не требует никаких действий со стороны пользователя.

Во время запуска MathManagerDaemon считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL. на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

Из секции [BufferServer]:

– MathManagerBufferServerId - числовой идентификатор сервера (значение поля Id в таблице communication_servers базы данных), с которым должен взаимодействовать MathManagerDaemon;

– AdcBufferLimit - количество значений для каждого сигнала, которые будут храниться в модуле. При превышении данного лимита “устаревшие” данные будут удаляться.

Из секции [MathManager]:

– SensorMalfunctionDiagnosticIsOn - включить режим диагностики неисправностей датчиков.

Прочитав параметры, MathManagerDaemon подключается к серверу базы данных и считывает описание всех сигналов из таблиц sakt, system_statistics_values, cla_sakt_output, predict_sakt, import_sakt, диагностические признаки из таблиц diagnostic_features, диагнозы из diagnostic_knowledge_base, пороги из таблицы calculated_treshold и вычисляемые значения из таблицы calculated_values.

Затем модуль формирует общий список значений для расчета, распределяя их исходя из уровней зависимости (т.е. вычисляемые значения могут зависеть от других вычисляемых значений), и прочитанные из базы данных формулы конвертируются в обратную польскую запись (форма записи математических и логических выражений, в которой операнды расположены перед знаками операций).

Считав необходимые для работы данные, модуль открывает соединение с установленным параметром MathManagerBufferServerId сервером оперативных данных, регистрируется в нём в качестве наблюдателя и подписывается на сигналы, которые участвуют в расчете. После этого MathManagerDaemon переходит в “слушающий” режим. Это означает, что модуль не запрашивает данные у серверов напрямую, а ждет, пока серверы вышлют уведомление о поступлении новых данных по сигналам, на которые он подписан. Получив уведомление, модуль сохраняет у себя новые данные, производит расчет вычисляемых значений по поступившим данным и отправляет результаты серверу оперативных данных.

При потере связи с серверами MathManagerDaemon автоматически пытается восстановить соединение. В случае успешного восстановления связи модуль отправляет повторный запрос на подписку на сигналы так же, как при своем старте. Дальнейшее функционирование модуля осуществляется так же, как при запуске модуля.

Модуль выполняет логирование информации о начале и завершении своей работы, а также информации об ошибках, возникающих в процессе выполнения им своих функций.

Схема информационных потоков модуля представлена на рисунке 2.2.1.

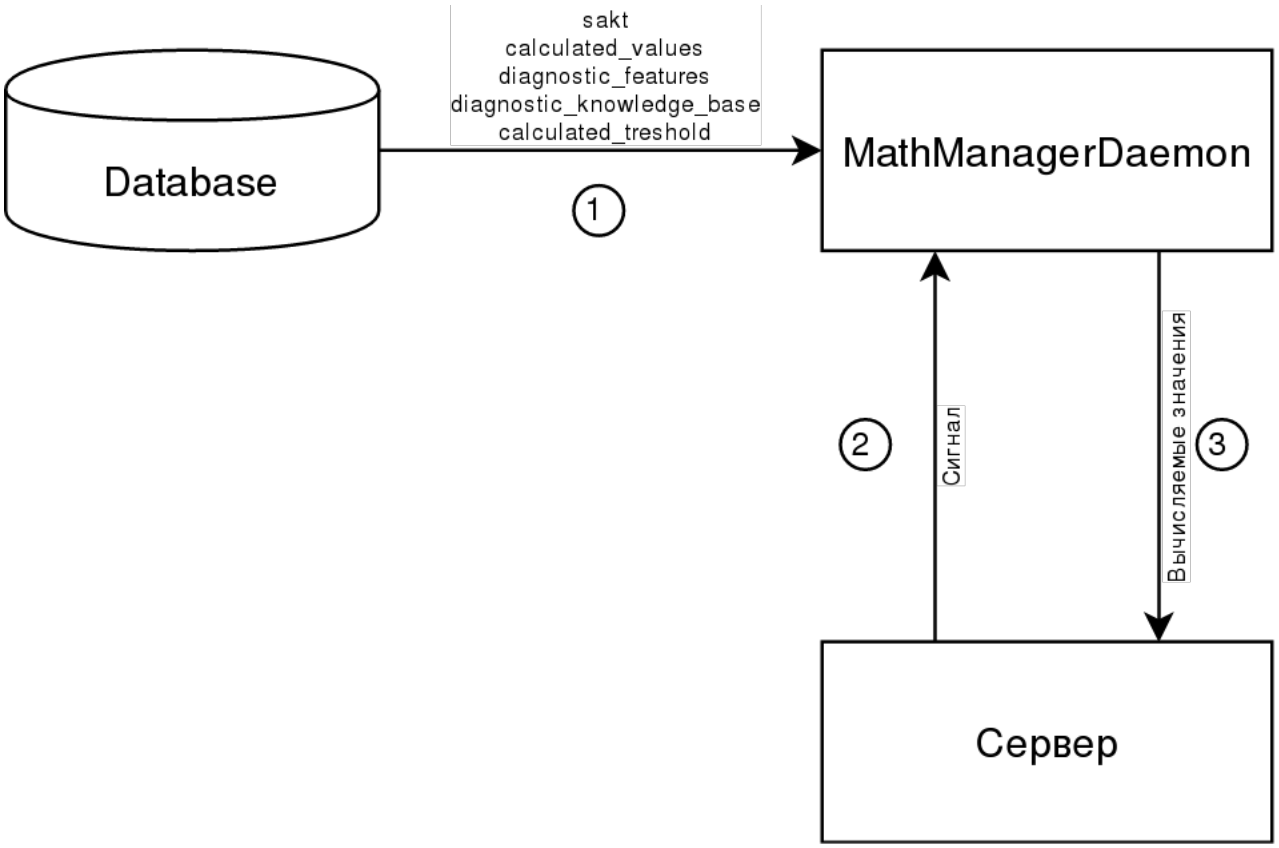


Рисунок 2.2.1

2.2.2 Структура программы

Модуль состоит из исполняемого файла MathManagerDaemon, а также нескольких дополнительных файлов конфигурации, скриптов запуска и лог-файла. Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл MathManagerDaemon;

– /usr/local/bin/diaprom/data/MathManagerDaemon.properties - файл конфигурации системы логирования;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /etc/init.d/MathManagerDaemon - скрипт запуска модуля в качестве системного сервиса;

– /var/log/MathManagerDaemon.log - лог-файл модуля. Создается при первом запуске программного модуля автоматически;

– /root/Check_services - скрипт для запуска сервисов по расписанию cron.

Помимо перечисленных файлов для корректной работы модуля необходимы:

– сервер базы данных PostgreSQL с полностью корректно установленной базой данных;

– серверы оперативных данных;

– служба запуска сервисов по расписанию cron.

Для старта во время загрузки операционной системы сервис должен быть прописан в chkservices на уровнях 2, 3 и 5.

Модуль в своей работе использует следующие таблицы:

– sakt - хранит информацию о сигналах;

– system_statistics_values - хранит информацию о сигналах состояния системы;

– cla_sakt_output – хранит информацию о выходных сигналах системы комплексного анализа течей (если установлена)

– predict_sakt – хранит информацию о создаваемых для моделей сигналах;

– import_sakt – хранит информацию о сигналах для импорта данных;

– diagnostic_features - содержит информацию о значениях, которые являются диагностическими признаками;

– calculated_treshold - содержит информацию о вычисляемых порогах;

– diagnostic_knowledge_base - содержит диагнозы;

– calculated_values - содержит информацию о вычисляемых значениях.

2.2.3 Настройка программы

Настройку модуля можно разделить на четыре этапа.

Первый этап - остановка модуля. Выполняется с помощью команды sudo service MathManagerDaemon stop с последующим вводом пароля пользователя если это потребуется. Если после выполнения вышеуказанной команды будет получено сообщение “MathManagerDaemon: нераспознанная служба”, значит установка модуля производится впервые.

Второй этап заключается в установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Третий этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP-адрес сервера базы данных PostgreSQL. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс. Также в этом файле необходимо указать идентификатор сервера оперативных данных, с которым должен взаимодействовать модуль, в параметре MathManagerBufferServerId секции BufferServer, количество значений для каждого сигнала, которые будут храниться в модуле, в параметре AdcBufferLimit и состояние режима диагностики неисправностей (включен или

выключен) в параметре SensorMalfunctionDiagnosticIsOn секции MathManager (см. Общие сведения о программе).

В таблицы diagnostic_features, calculated_treshold, diagnostic_knowledge_base, calculated_values значения могут быть добавлены посредством использования программы FormulaCreator программного комплекса (см. раздел FormulaCreator).

При настройке или устранении неполадок стоит обратить особое внимание на поле formula таблиц calculated_treshold, diagnostic_knowledge_base, calculated_values, в котором представлена формула для вычисления, соответственно, порога, диагноза и вычисляемого значения. Формулы также можно редактировать через FormulaCreator.

Для каждой строки в таблице diagnostic_features:

– Если значение feature_type равно 0, то в таблице sakt должна существовать строка со значением поля param_id, равным значению поля external_id таблицы diagnostic_features;

– Если значение feature_type равно 1, то в таблице calculated_values должна существовать строка со значением поля param_id, равным значению поля external_id таблицы diagnostic_features.

В скрипте запуска сервисов (Check_services) указываются условия запуска модуля и команда для его выполнения. Скрипт должен быть прописан в журнале cron (/etc/crontab) с интервалом запуска, равным одной минуте.

Четвертый этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новые. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.2.4 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена двумя способами.

Способ первый. Выполнить команду в терминале ps -A | grep MathManagerDaemon. Если модуль запущен, то ответом на команду будет номер процесса.

Способ второй. Проверить статус сервиса, выполнив команду sudo service MathManagerDaemon status.

Проверка работоспособности программного модуля. Проверка лог-файлов программного модуля. Если в лог-файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

Также следует обратить внимание на строку в логе Calculated values count <число> - если число равно нулю, то у модуля нет таких значений, которые надо вычислять.

Для проверки на наличие в БД вычисляемых значений следует выполнить следующие SQL запросы:

SELECT COUNT(*) FROM diagnostic_features WHERE feature_type!=0;

SELECT COUNT(*) FROM calculated_values;

SELECT COUNT(*) FROM diagnostic_knowledge_base;

SELECT COUNT(*) FROM calculated_treshold;

Для проверки на наличие в БД информации о сигналах следует выполнить следующие SQL запросы:

SELECT COUNT(*) FROM sakt;

SELECT COUNT(*) FROM predict_sakt;

SELECT COUNT(*) FROM import_sakt;

Ответом будет строка с числом. Если сумма всех цифр этого числа равна нулю, то модулю нечего рассчитывать. В результате этого модуль будет запущен, в лог-файлах будут отсутствовать сообщения об ошибках, но расчет данных производится не будет.

2.2.5 Дополнительные возможности

MathManagerDaemon может быть запущен в двух режимах: демон и не демон. Первый режим запуска является штатным способом запуска программного модуля и должен быть использован в режиме эксплуатации. Второй режим работы подразумевает запуск программного модуля вручную из терминала с указанием дополнительного ключа -nd. Этот режим нужен только для отладки. В этом режиме все сообщения, которые система логирования записывает в лог-файл, отображаются в окне терминала, где запущен модуль. Функции, выполняемые программным модулем, одинаковы для обоих режимов.

2.2.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует MathManagerDaemon сохраняются в лог-файл /var/log/MathManagerDaemon.log.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.2.6.1 Информационные сообщения, выводимые при штатной работе модуля

– Exit with return code: <число> - работа модуля была завершена. Если <число> не равно нулю - это может сигнализировать о проблемах в работе модуля;

– Calculated values count <число> - общее количество вычисляемых значений. Далее за этим сообщением следует список имен вычисляемых значений;

– detected deps in formula <числа через пробел> - id сигналов, от которых зависят вычисляемые значения. Именно на эти сигналы модуль подпишется при подключении к серверу;

– Buffers observiceng <числа через пробел> - адреса сигналов, от которых зависят вычисляемые значения;

– values to write <вещественные числа через пробел> - отправленные серверу посчитанные значения;

– Formula <строка> Addr <число> - формула для расчета вычисляемого значения сигнала и соответствующий этому сигналу адрес;

– %%%%%%%%%%%%%%%%%%%%%% <строка> - формула вычисляемого значения, преобразованная в обратную польскую запись;

– ARGS FOR CALCULATION <строка> - операнды для операции.

2.2.6.2 Информационные сообщения, сигнализирующие о проблемах

– Database INSTANCE ERROR: <подробное описание ошибки> - происходит ошибка запроса к базе данных. Требуется подключиться вручную по конфигурации, которую можно найти в skpt.ini к базе данных, после чего выполнить запрос, который будет описан в сообщении. После этого можно будет получить более подробное сообщение об ошибке уже от самой базы или убедиться в некорректности работы модуля;

– Wrong client IP - отсутствуют права доступа в системе. Проверьте таблицу access_permissions на наличие записи с необходимым клиентом;

– Connections limit exceeded - достигнут лимит подключений. Проверьте, не запущен ли дубликат демона (ps -A | grep MathManagerDaemon);

– Received ban on the server - получен бан на сервере (несоответствие версий клиента и сервера). Возможно, неправильно была проведена процедура установки/обновления;

– CANT FIND CALCULATED VALUE. ID <число> - не было найдено вычисляемого значения для сигнала с Id <число>;

– EMPTY STACK - стек, в котором хранятся операнды и операции, на момент окончания расчетов оказался пуст, хотя в нем должен был остаться результат вычислений.

2.3 Server

2.3.1 Общие сведения о программе

Server является системным демоном и служит сервером оперативных данных. Обмен между клиентами (модулями программного комплекса и внешними источниками данных) происходит через него. Далее по тексту указывается как “Сервер”.

Запуск Server производится автоматически во время старта операционной системы. В случае остановки сервиса пользователем или аварийного прекращения работы сервис будет перезапущен автоматически с помощью cron.

При нормальной эксплуатации не требует никаких действий со стороны пользователя.

Во время запуска Server считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - зашифрованный пароль пользователя базы данных.

Из секции [BufferServer]:

– Port - порт, на котором Сервер будет слушать входящие соединения по протоколу CSP;

– ModbusPort - порт, на котором Сервер будет слушать входящие соединения по протоколу ModbusTCP.

Из секции [BufferServer]:

– ModbusValuesCacheSize - максимальное количество значений сигналов, хранящихся в кэше;

– CheckCacheInterval – интервал, с которым значения сигналов, хранящихся в кэше, будут отправляться подписанным на него клиентам;

– MaxSecsWithoutTimestamp - максимальный интервал (в секундах), по истечении которого клиент, работающий по протоколу ModbusTCP, будет отсоединен, если отсутствует активность в открытом соединении (операции чтения/записи);

– TimestampTimerTimeout - интервал проверки времени последнего запроса от клиента, работающего по протоколу ModbusTCP.

Сервер работает по двум протоколам: CSP (Communication Service Protocol) и ModbusTCP.

CSP (Communication Service Protocol) используется при работе с внутренними клиентами системы:

– Демоны:

– DatabaseWriter;

– MathManagerDaemon;

– SystemWatcher;

– xdevmon,

– Приложения:

– OldRMSOD;

– SpaGui.

ModbusTCP используется для взаимодействия с внешними клиентами нижнего уровня (подсистемы САКТ, СКВ, СКТВ, СОСП).

Прочитав параметры, Server подключается к серверу базы данных и считывает описание всех сигналов и подсистем, прав доступа из таблиц sakt, system_statistics_values, calculated_values, diagnostic_knowledge_base, cla_sakt_output, predict_sakt, import_sakt, subsys, access_permissions.

Часть прав доступа, описывающая соответствие идентификаторов клиентов (поле client_id), их типов (поле client_type), IP-адресов (поле ip), лимита количества соединений (поле max_connections) и соответствующих серверов оперативных данных (поле server_id), прописана в таблице access_permissions базы данных. Часть прав доступа, описывающая разрешенные адреса для записи и чтения генерируется динамически во время чтения таблицы access_permissions на основе информации о сигналах из таблиц sakt, system_statistics_values, calculated_values, diagnostic_knowledge_base, cla_sakt_output, predict_sakt, import_sakt следующим образом:

– разрешенные для записи адреса формируются списком из входящих и исходящих адресов сигналов, принадлежащих подсистеме, к которой принадлежит клиент (поле client_id), и двух дополнительных адресов для хранения метки времени (дата/время и миллисекунды);

– разрешенные для чтения адреса формируются списком из всех адресов, разрешенных для записи.

Считав необходимые данные из базы, модуль начинает слушать указанные в параметрах порты для клиентов, работающих по протоколу CSP и ModbusTCP.

Затем происходит настройка межсетевого экрана в соответствии с данными, представленными в таблице access_permissions. Сначала удаляются уже установленные правила, затем для каждого указанного в таблице клиента:

– разрешается установление входящих подключений через порт, соответствующий его типу (значение Port или ModbusPort конфигурационного файла skpt.ini) с ограничением в 3 подключения в секунду;

– запрещается установление входящих подключений, соответствующих пункту 1, если в текущий момент времени достигнут лимит количества соединений для данного клиента;

– разрешаются уже существующие установленные подключения для текущего клиента.

Для всех остальных адресов соединения на порты Port и ModbusPort (параметры конфигурационного файла skpt.ini секции BufferServer) запрещаются.

При подключении клиента Server открывает с этим клиентом отдельный канал связи и пытается зарегистрировать клиента. Процедура регистрации зависит от типа клиента.

Для клиента, работающего по протоколу CSP:

– сервер ожидает передачи идентификатора клиента - зарезервированного имени клиента, которое соответствует значению поля client_id таблицы access_permissions;

– при получении идентификатора проверяется, не установлен ли запрет на клиента с этим идентификатором. Запрет для клиента устанавливается в случае попытки доступа к запрещенным адресам или несоответствии версии клиента и Сервера (время запрета - 1 час). В случае наличия запрета клиенту высылается сообщение об этом, и соединение разрывается. В этой ситуации следует проверить конфигурацию и работоспособность клиента, на которого был установлен запрет (см. Сообщения системному программисту), и перезагрузить Сервер, чтобы снять запрет (sudo service Server restart);

– проверяется, соответствует ли реальный IP-адрес клиента указанному в таблице access_permissions базы данных. В случае отсутствия в базе данных строки с IP-адресом клиента и переданным идентификатором, клиенту высылается сообщение об этом и соединение разрывается.

– проверяется, не превышен ли лимит количества соединений от данного клиента (поле max_connections таблицы access_permissions). В случае превышения клиенту высылается сообщение об этом и соединение разрывается.

– в случае успешного прохождения вышеперечисленных проверок клиент регистрируется на Сервере.

Для клиента, работающего по протоколу ModbusTCP:

– проверяется наличие в таблице access_permissions базы данных IP-адреса, соответствующего IP-адресу подключившегося клиента. В случае отсутствия в базе данных строки с IP-адресом клиента клиенту высылается сообщение об этом, и соединение разрывается.

– проверяется, не превышен ли лимит количества соединений от данного клиента (поле max_connections таблицы access_permissions). В случае превышения - клиенту высылается сообщение об этом, и соединение разрывается.

– в случае успешного прохождения вышеперечисленных проверок клиент регистрируется на Сервере.

Внутренние клиенты работают по системе подписок. Они не запрашивают данные у Серверов напрямую, а ждут, пока Сервер вышлет уведомление о поступлении новых данных, на которые подписан модуль. Внешние клиенты запрашивают данные напрямую.

Схема информационных потоков модуля представлена на рисунке 2.3.1.

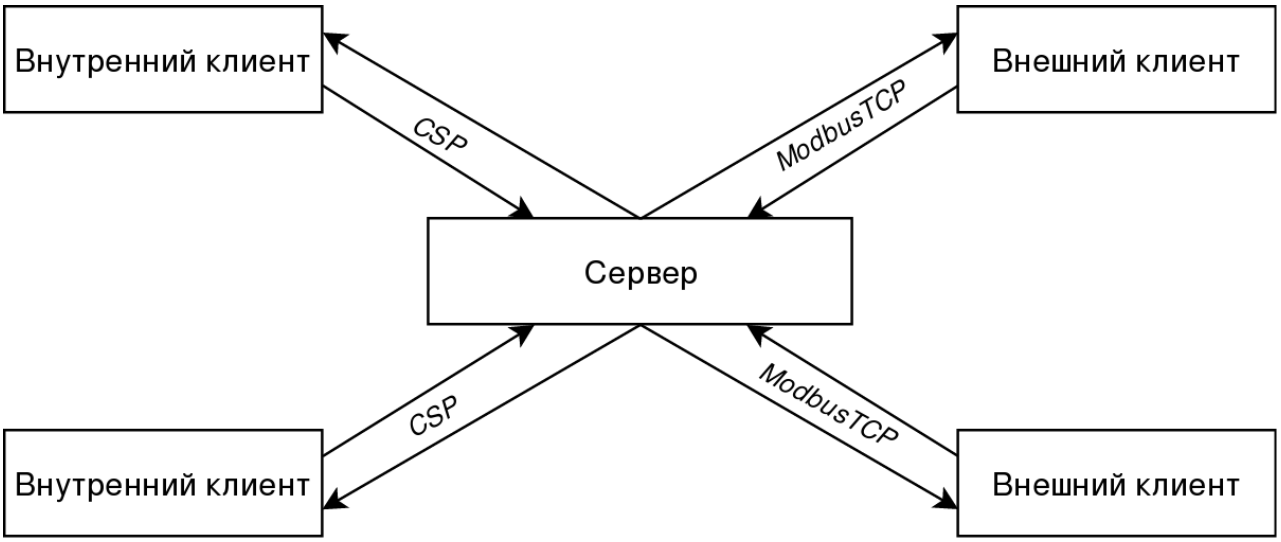


Рисунок 2.3.1

Если клиент подписывается на один из типов подписки на определённые сигналы, то модуль после успешной регистрации высылает срез значений и сопутствующих данных (признаков достоверности и меток времени для сигналов) по сигналам, на которые подписался клиент. Впоследствии модуль будет высылать уведомления каждый раз, когда будут изменяться значения, на которые подписан клиент.

Если у значения сигнала изменилась только метка времени, но значение и признак достоверности остались прежними, то модуль не вышлет уведомление об изменении. Это справедливо только для значений сигналов от источников, зарегистрированных в системе.

Для оптимизации передачи данных внутри системы Сервер использует оперативную память (кеш) для накопления значений сигналов и их последующей отправки большими пакетами. Количество накапливаемых значений сигналов в кеше (количество значений сигналов) устанавливается параметром ModbusValuesCacheSize секции BufferServer конфигурационного файла skpt.ini. Накопленные в кеше значения отправляются подписанным клиентам в том случае, если количество значений доходит до установленного лимита или по истечении интервала времени, установленного параметром CheckCacheInterval (в миллисекундах).

Для проверки наличия соединения с внутренним клиентом (который работает по протоколу CSP) используется процедура ping-запроса. Клиент каждые 5 секунд (если он не посылает какие-то другие данные) посылает Серверу ping-запрос. При отсутствии активности Сервер ожидает 10 секунд и, если в течении этого времени не было какой-либо активности от клиента (запросы на чтение/запись или ping-запросы), отсоединяет данного клиента.

Наличие соединения с внешним клиентом проверяется следующим образом: каждый интервал времени, установленный параметром TimestampTimerTimeout секции BufferServer конфигурационного файла skpt.ini, проверяется, как давно был последний запрос на

чтение/запись от данного клиента. Если разница между временем последнего запроса и текущим временем больше, чем установлено параметром MaxSecsWithoutTimestamp, то соединение закрывается.

Описанные выше процедуры проверки наличия соединения необходимы для того, чтобы не держать открытое соединение, когда клиент по каким-либо причинам потерял связь с сервером и при восстановлении связи с сервером мог корректно повторно зарегистрироваться.

В случае если клиент потерял связь с Сервером, после восстановления соединения клиент отправляет повторный запрос на подписку, и Сервер подписывает клиента на сигналы, на которые он был подписан ранее. Данная процедура повторяет ту, что и при начальном подключении клиента.

Сервер выполняет логирование информации о начале и завершении своей работы, а также информации об ошибках, возникающих в процессе выполнения им своих функций.

2.3.2 Структура программы

Модуль состоит из исполняемого файла Server, а также нескольких дополнительных файлов конфигурации, скриптов запуска и лог-файла. Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл Server;

– /usr/local/bin/diaprom/data/Server.properties - файл конфигурации системы логирования;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /etc/init.d/Server - скрипт запуска модуля в качестве системного сервиса;

– /var/log/Server.log - лог-файл модуля. Создается при первом запуске программного модуля автоматически;

– /root/Check_services - скрипт для запуска сервисов по расписанию cron.

Помимо перечисленных файлов для корректной работы модуля необходимы:

– сервер базы данных PostgreSQL с полностью корректно установленной базой данных;

– служба запуска сервисов по расписанию cron.

Для старта во время загрузки операционной системы сервис должен быть прописан в chkservices на уровнях 2, 3 и 5.

2.3.3 Настройка программы

Настройку модуля можно разделить на четыре этапа.

Первый этап - остановка модуля. Выполняется с помощью команды sudo service Server stop с последующим вводом пароля пользователя если это потребуется. Если после

выполнения вышеуказанной команды, будет получено сообщение “Server: нераспознанная служба”, значит установка модуля производится впервые.

Второй этап заключается в первичной установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Третий этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP-адрес сервера базы данных PostgreSQL. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс. Также в этом файле указываются порты, на которых будут ожидаться соединения по протоколам CSP и ModbusTCP.

В базе данных в таблице access_permissions должны быть введены корректные сведения о клиентах:

– идентификатор клиента (поле client_id, строго зарезервирован, менять категорически запрещено);

– тип клиента (поле client_type, значение “CSC” для внутренних и “ModbusTcp” для внешних клиентов);

– IP-адрес клиента (поле ip);

– идентификатор сервера, к которому может подключаться данный клиент (поле server_id, значение которого должно быть равно соответствующему Серверу в таблице communication_servers, поле id);

– лимит количества одновременных соединений (поле max_connections).

В скрипте запуска сервисов (Check_services) указываются условия запуска модуля и команда для его выполнения. Скрипт должен быть прописан в журнале cron (/etc/crontab) с интервалом запуска, равным одной минуте.

Четвертый этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.3.4 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена двумя способами.

Способ первый. Выполнить команду в терминале ps -A | grep Server. Если модуль запущен, то ответом на команду будет номер процесса.

Способ второй. Проверить статус сервиса, выполнив команду sudo service Server status.

Проверка работоспособности программного модуля может быть осуществлена путем проверки лог-файлов программного модуля. Если в лог-файлах отсутствуют какие-либо сообщения об ошибках, то это указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту). Если ошибок нет - связаться с разработчиками. При наличии ошибок - сначала исправить их, а затем повторить проверку.

2.3.5 Дополнительные возможности

Server может быть запущен в двух режимах: демон и не демон. Первый режим запуска является штатным способом запуска программного модуля и должен быть использован в режиме эксплуатации. Второй режим работы подразумевает запуск программного модуля вручную из терминала с указанием дополнительного ключа -nd. Этот режим нужен только для отладки. В этом режиме все сообщения, которые система логирования записывает в лог-файл, отображаются в окне терминала, где запущен модуль. Функции, выполняемые программным модулем, одинаковы для обоих режимов.

2.3.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует Server, сохраняются в лог-файл /var/log/Server.log.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.3.6.1 Информационные сообщения, выводимые при штатной работе модуля

– Server started successfully! - сообщение, сигнализирующее о том, что часть Сервера, работающая с клиентами по протоколу CSP, стартовала успешно;

– Modbus server started successfully! - сообщение, сигнализирующее о том, что часть Сервера, работающая с клиентами по протоколу ModbusTCP, стартовала успешно;

– New client connected - подключение нового внутреннего клиента;

– Old modbus client connected - подключение нового внешнего клиента;

– Too many old modbus incoming connections - ведется регистрация другого клиента;

– Client disconnected - отключение внутреннего клиента;

– Modbus client disconnected (IP <IP-адрес>) - отключение внешнего клиента;

– Ping timeout. The client will be disconnected. Client ID: <строка>; IP address: <IP-адрес> - превышение времени ожидания данных от внутреннего клиента. Клиент будет отсоединен;

– The client with IP <IP-адрес> has been disconnected. No timestamps within last <число> seconds. - превышение времени ожидания данных от внешнего клиента. Клиент будет отсоединен;

– Ping request received from ClientId=<строка> and HOST=<IP-адрес> - получен ping-запрос от клиента;

– Register client ID request received. Client ID: <строка>; IP address: <IP-адрес> - регистрация клиента прошла успешно;

– Client IP: <IP-адрес>. Modbus values quality updated for addresses <список чисел> - обновлены признаки достоверности у значений;

– Client IP: <IP-адрес>. Write modbus values request received (Old). Count: <число>. Addresses: <список чисел> - получен запрос на запись значений;

– Client IP: <IP-адрес>. Write timestamp request received (Old) - получена новая временная метка;

– Client IP: <IP-адрес>. Read modbus values request received (Old): <число> values requested (from addresses: <список чисел>) - получен запрос на чтение значений;

– Sending modbus values notification. Count: <число> - клиентам отправлено уведомление о получении новых значений сигналов;

– About to send changed adc values. Buffer id: <число>; Value count: <число>

– About to send changed rms values. Buffer id: <число>; Value count: <число>

– About to send notification about spectral calculation finish for View id: <число>

– About to send notification about sensor malfunction of type=<число>, hash=<строка>; Client ID: <число>

Sending notification about sensor malfunction of type=<число>, id=<строка>; Client ID: <число> - клиентам отправлено уведомление о неисправности датчика;

– Write modbus values request received. Values to write: - получен запрос на запись значений сигналов;

– Read modbus values request received. Address count: <число> - получен запрос на чтение значений сигналов;

– Observe modbus values request received. Value count: <число> - получен запрос на подписку на значения сигналов;

– Sensor malfunction notification received: type=<число>, hash=<строка> - получено уведомление о неисправности датчика;

– Observe sensor malfunction request received - получен запрос на подписку на неисправности датчиков;

2.3.6.2 Информационные сообщения, сообщающие об ошибках

– Unable to listen to port <номер> - модуль не может начать прослушивание необходимого для его работы порта во время старта. После этого сообщения модуль завершает свою работу. Убедитесь, что не запущен другой экземпляр модуля (sudo service Server status), а в конфигурационном файле (skpt.ini) прописаны корректные порты в параметрах Port и ModbusPort секции BufferServer (1-65535);

– IPTABLES PROCESS ERROR: <строка> - ошибка в процессе настройки межсетевого экрана. <строка> уточняет ошибку;

– IPTABLES PROCESS CRASHED - процесс командной утилиты iptables для настройки межсетевого экрана аварийно завершился;

– IPTABLES PROCESS ERROR: ExitCode: <число>. Error: <строка> - процесс командной утилиты iptables для настройки межсетевого экрана завершился с ошибкой;

– An error occured: <сообщение> - ошибка при получении данных от клиента. Проверьте сетевое соединение;

– Unknown client command: <номер> - неизвестная команда от клиента. Проверьте работоспособность внутренних клиентов;

– The client has no permissions to write to the passed registers: <список чисел> - у клиента отсутствуют права на запись в запрошенные регистры. Следом за этим сообщением следует запрет подключения для данного клиента на Сервере (1 час);

– The client has no permissions to read from the passed registers: <список чисел> - у клиента отсутствуют права на чтение запрошенных регистров. Следом за этим сообщением следует запрет подключения для данного клиента на Сервере (1 час);

– The client has no permissions to observe values from the passed registers: <список чисел> - у клиента отсутствуют права на слежение (подписку) за запрошенными регистрами. Следом за этим сообщением следует запрет подключения для данного клиента на Сервере (1 час);

– Wrong protocol version in ping request received from ClientId=<строка> and HOST=<IP-адрес> - получен ping-запрос с отличающимся идентификатором версии. Следом за этим сообщением следует запрет подключения для данного клиента на Сервере (1 час). Возможно, при установке/обновлении были неправильно заменены файлы;

– No client ID - в процессе регистрации клиент передал пустой идентификатор. Проверьте работоспособность клиентов;

– Client with ID <строка> and IP address <IP-адрес> disconnected. Reason: Banned until <дата/время> - в процессе регистрации обнаружен запрет подключения для данного клиента. Возможно была некорректно произведена процедура обновления. Для сброса запрета перезапустите Server (sudo service Server restart);

– The client has been disconnected. Access forbidden. Wrong client IP address. Client tries to connect from <IP-адрес>. - в таблице access_permissions базы данных отсутствует строка с IP-адресом клиента. Проверьте таблицу access_permissions базы данных на наличие данных о внешних клиентах;

– Error write data into socket. Connection will be closed - ошибка при передаче данных клиенту. Клиент будет отсоединен. Проверьте сетевое соединение;

– The client has been disconnected. Access forbidden. Too match connection from this host (<IP-адрес>). Active connections count: <число>. Maximum expected connections: <число> - превышен лимит подключений с данного IP-адреса;

– The client has been disconnected. Undefined subsys ID for this client (IP: <IP-адрес>, Name: <строка>) - не найдена подсистема для данного IP-адреса. Проверьте поле client_id таблицы access_permissions, значение в этом поле должно соответствовать одной из подсистем в таблице subsys (поле name);

– !!! ERROR when <строка> (client IP <IP-адрес>) !!! Timestamp for subsys <число> NOT UPDATED because requested address does not contained in a configuration table. Requested addresses: <список чисел>. If timestamp will be not updated <число> seconds - that connection will be closed automatically. - произошла операция чтения/записи, но время последнего запроса не обновилось, так как адрес некорректен;

– !!! ERROR !!!. Client IP: <IP-адрес>. Write modbus values request received (Old). Incorrec modbus adress: <число>. Access denied - попытка записи в некорректный адрес. Проверьте работоспособность внешних клиентов;

– !!! ERROR !!!. Client IP: <IP-адрес>. Read modbus values request received (Old). Incorrec modbus adress: <число>. Access denied - попытка чтения из некорректного адреса. Проверьте работоспособность внешних клиентов.

2.4 xdevmon

2.4.1 Общие сведения о программе

xdevmon является системным демоном и предназначен для опроса состояния устройств, работающих по протоколу SNMP (свитчи, роутеры, блоки питания, кулеры, двери и т.п.).

Схема информационных потоков приведена ниже на рисунке 2.4.1.

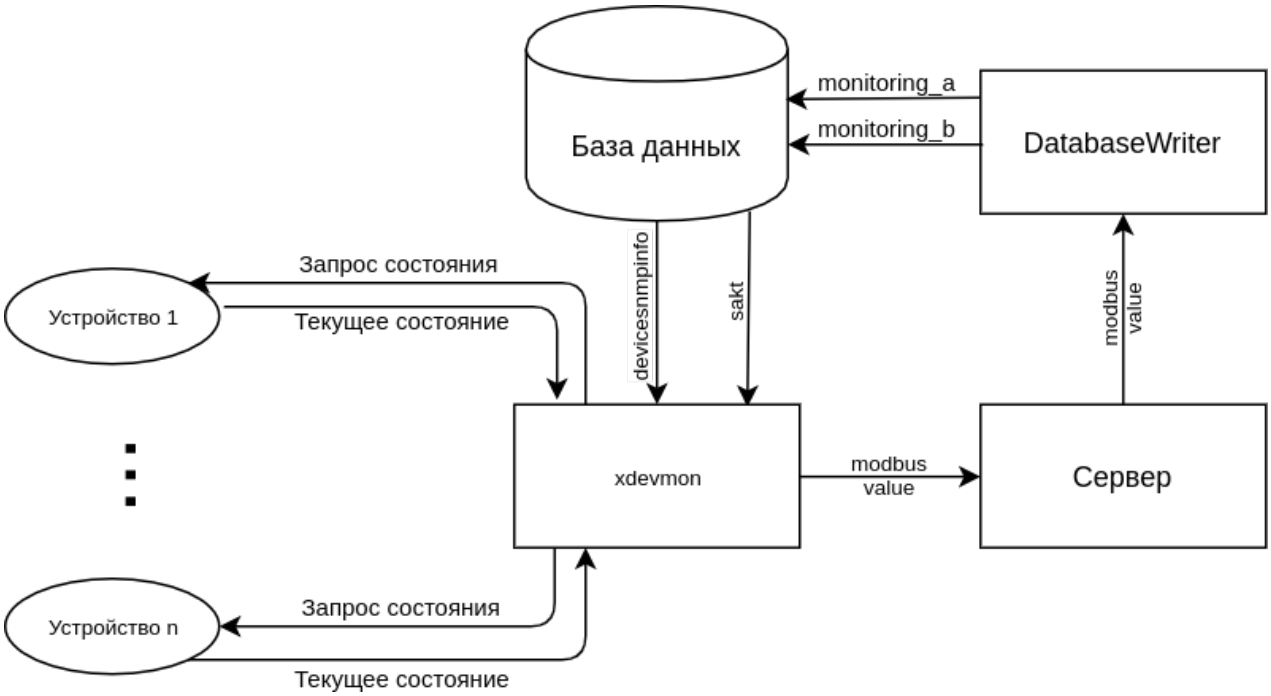


Рисунок 2.4.1

Во время запуска xdevmon считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL, на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

Также будут прочитаны все переменные секции xdevmon:

– TimerInterval - интервал времени между опросами устройств.

После чтения параметров конфигурационного файла skpt.ini xdevmon считывает описание серверов из базы данных (таблица communication_servers) и подключается к ним. Также будет прочитан список сигналов из таблицы sakt, которые относятся к подсистеме xdevmon и таблица devicesnmpinfo.

После инициализации модуль начинает опрашивать SNMP устройства по списку из таблицы devicesnmpinfo. Опрос будет идти с интервалом TimerInterval из skpt.ini. Ответы устройств отправляются на сервер. Если запущен DatabaseWriter, ответы устройств будут записаны в базу данных в соответствующие таблицы (monitoring_a для аналоговых сигналов, monitoring_b для бинарных).

При потере связи с серверами xdevmon автоматически пытается восстановить соединение. Никаких дополнительных действий после совершения переподключения совершено не будет. Ответы устройств, полученные во время отсутствия подключения к серверу, отправлены не будут.

Приложение ведет логи, которые должны храниться в файле /var/log/xdevmon.log.

2.4.2 Структура программы

Модуль имеет следующую структуру:

– /usr/local/bin/diaprom/bin - исполняемый файл xdevmon;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /usr/local/bin/diaprom/data/xdevmon.properties - файл конфигурации системы логирования;

– /etc/init.d/xdevmon - скрипт запуска модуля в качестве системного сервиса;

– /var/log/xdevmon.log - лог-файла модуля. Создается автоматически при первом запуске;

– /root/Check_services - скрипт для запуска сервисов по расписанию cron.

Помимо перечисленных файлов для корректной работы модуля необходимы:

– сервер базы данных PostgreSQL с полностью корректно установленной базой данных;

– серверы оперативных данных.

2.4.3 Настройка программы

Для работы модуля требуются корректно инициализированные таблицы sakt, subsys, communication_servers и devicessnmpinfo.

Таблица devicesnmpinfo хранит в себе описание необходимых запросов. Для каждого устройства может существовать несколько записей с запросами. Каждый запрос должен иметь уникальный modbusaddress и соответствующую ему запись в таблице sakt. В таблице sakt все записи, относящиеся к snmp запросам, должны соответствовать подсистеме Monitoring (поле source_subsys таблицы sakt <-> subsys_id таблицы subsys). В случае если взаимно однозначного соответствия между записями таблиц не будет, xdevmon завершится с ошибкой.

Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.4.4 Проверка программы

Проверка лог-файлов программного модуля. Если в лог-файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование

модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

Модуль должен быть запущен вместе с Server. После запуска xdevmon начнет опрос устройств и будет отправлять отчеты/информацию об их состоянии на сервер. Рекомендуется открыть skd_gui и создать в нем экран с любым сигналом подсистемы Monitoring для отображения изменений в реальном времени. Вместо skd_gui для проверки можно использовать DatabaseWriter. Тогда значения сигналов будут записаны в таблицы monitoring_a и monitoring_b (в зависимости от типа).

2.4.5 Дополнительные возможности

xdevmon может быть запущен в двух режимах: демон и не демон. Первый режим запуска является штатным способом запуска программного модуля и должен быть использован в режиме эксплуатации. Второй режим работы подразумевает запуск программного модуля вручную из терминала с указанием дополнительного ключа -nd. Этот режим нужен только для отладки. В этом режиме все сообщения будут выводиться на экран, а не в лог-файл.

2.4.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует xdevmon, сохраняются в лог-файл /var/log/xdevmon.log

Список возможных сообщений приведен ниже:

– Could not find id for subsys name имя_подсистемы - у xdevmon не получилось найти свой id подсистемы. Без этого не получается провести соответствие между SNMP сигналами и modbus сигналами. Требуется проверить таблицу subsys;

– getAllSignals(): error (count of signals = 0). - в базе данных не найдено описания сигналов. Нужно подключиться к базе по конфигурации, указанной в skpt.ini, если в таблице SAKT нет записей, то нужно будет полностью развернуть базу с нуля;

– getAllSignals(): devmonSignals.count(%%) != snmpSignals.count(%%) - таблица описания сигналов snmp конфликтует с общей таблицей сигналов modbus;

– run(): %ERROR_TYPE% (peer = %%, oid = %%) - snmp устройство не ответило на запрос. %ERROR_TYPE% может быть представлен одним из вариантов:

– CANNOT_OPEN - устройство не может быть открыто;

– BAD_RESPONSE - устройство прислало некорректный ответ;

– TIMEOUT - устройство отвечало слишком долго или недоступно;

– API_ERROR - ответ противоречит стандарту;

– UNKNOWN_ERROR - неизвестная ошибка;

– run(): exiting with ERROR (nSNMPGetErrors >= MAX_SNMP_GET_ERRORS) - демон завершает свою работу, потому что превышено предельно допустимое число ошибок;

– Client::onSocketStateChanged() UnconnectedState - модуль не может подключиться к серверу. Требуется убедиться, что сервер запущен, а в конфигурационном файле указан правильный IP-адрес сервера;

– Database INSTANCE ERROR: <подробное описание ошибки> - происходит ошибка запроса к базе данных. Требуется подключиться вручную по конфигурации, которую можно найти в skpt.ini к базе данных, после чего выполнить запрос, который будет описан в сообщении. После этого можно будет получить более подробное сообщение об ошибке уже от самой базы или убедиться в некорректности работы модуля;

– server connection error: <подробное описание ошибки> - ошибка подключения к серверу. Обычно вызвана несколькими запущенными копиями программы, неправильным описанием IP-адреса и порта сервера, некорректной конфигурацией сервера, нарушением сетевой связности (например, передавленный сетевой кабель) или сбоем на сервере. Более подробно установить причину можно, посмотрев вторую часть сообщения и логи сервера.

ping response received - сообщение выводится при проверке состояния соединения, если был получен ответ на сервере. Актуальность сообщения - несколько секунд. Выводится только на уровне логирования DEBUG.

2.5 partition

2.5.1 Общие сведения о программе

partition является консольным приложением для создания и поддержки партиционирования таблиц с данными.

Запуск partition производится автоматически с помощью cron. Запускаться partition должен от пользователя root, так как требуется доступ к каталогу /var/lock на запись.

partition является запакованным приложением, написанным на языке Python, поэтому при запуске он распаковывается во временном каталоге (по умолчанию /tmp), а при окончании работы приложения распакованные файлы удаляются.

partition принимает следующие параметры командной строки:

– опциональные параметры:

– -h - показ справки;

– -v - показ версии приложения;

Во время запуска partition считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение.

Из секции [Partition]:

– PartitionParamNum - количество сигналов в одной дочерней таблице при разбиении по ID сигнала. По умолчанию: 1;

– PartitionSize - временной интервал, занимаемый дочерней таблицей при разбиении по времени (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 1 день;

– FutureInterval - временной интервал в будущем, который должен быть покрыт дочерними таблицами после работы приложения (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 365 суток;

– PastInterval - временной интервал в прошлом, таблицы, покрывающие этот интервал, удаляются только при превышении дисковой квоты (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 365 суток;

– MaxDiskUsage - максимальная дисковая квота в процентах, при превышении которой старые таблицы удаляются, даже если попадают в интервал PastInterval. По умолчанию 80%;

– PartitionBackwards - если true, то приложение при необходимости создает в прошлом пустые таблицы, так чтобы интервал, заданный параметром PastInterval, был ими полностью покрыт;

– DumpDir - директория, в которую будут записываться дампы таблиц. По умолчанию: текущая;

– LogFile - полное имя файла, в который будет записываться лог. По умолчанию: /var/log/partitionmanager.log;

– Debug - если true, то приложение работает в режиме отладки (логгер переходит в режим DEBUG, если false - INFO);

– Loop - если true, то операция повторяется в цикле, параметр задает интервал времени, через который операция перезапускается.

Прочитав параметры, partition блокирует файл /var/lock/subsys/tm_partition для предотвращения запуска двух одновременно работающих экземпляров приложения.

Тип партиционирования, используемый приложением, - двухуровневое, по ID сигнала и по дате/времени.

Алгоритм работы приложения следующий:

– чтение списка таблиц с данными на основе таблиц subsys (подсистемы), signal_types (типы сигналов), sakt, system_statistics_values, calculated_values, cla_sakt_output, predict_sakt, import_sakt (сигналы);

– создание таблицы partition_data (сведения о партициях) если её не существует;

– для каждой таблицы с данными:

– если разбиение на партиции уже происходило ранее - обновление партиций (создание новых, сохранение в дамп и удаление старых), иначе - начальное разбиение на партиции (при начальном разбиении, после разбиения данные переносятся из родительской таблицы в дочерние партиции);

– анализ дискового пространства, занимаемого данными в базе данных;

– в случае если квота использования дискового пространства (параметр MaxDiskUsage) превышена, то партиции с самыми старыми данными сохраняются в виде дампа и удаляются из базы данных.

Для корректной работы приложения требуются:

– наличие свободного пространства во временном каталоге (по умолчанию /tmp) не менее 12 мегабайт;

– таблицы с данными подсистем и таблица partition_data с корректной структурой, или полностью отсутствующие.

2.5.2 Структура программы

Модуль состоит из исполняемого файла partition.

/usr/local/bin/diaprom/data/dbscripts/partition - исполняемый файл partition

Помимо исполняемого файла для корректной работы модуля необходим сервер базы данных PostgreSQL с полностью корректно установленной базой данных.

2.5.3 Настройка программы

Обновление модуля заключается в замене исполняемого файла partition на новый.

Дополнительной настройки модуль не требует.

2.5.4 Проверка программы

Проверка работоспособности программного модуля осуществляется посредством проверки лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

2.5.5 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.5.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует partition, сохраняются в лог файл, указанный параметром LogFile секции Partition конфигурационного файла skpt.ini, при каждом запуске.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.5.6.1 Информационные сообщения, выводимые при штатной работе модуля

– Запущено обновление партиционирования БД <дата/время> - информационное сообщение, сигнализирующее о запуске приложения;

– Завершение работы <дата/время> - информационное сообщение, сигнализирующее о завершении работы приложения;

– INFO ### BEGIN EXECUTION, DTIME=”<дата/время> ### - информационное сообщение, сигнализирующее о запуске приложения;

– Table partition_data does not exist, creating. - таблица partition_data не существует, происходит её создание;

– *** Updating table=<название таблицы>, dtime=<дата/время> *** - обновляется разбиение для указанной таблицы;

– *** Reading partition tree from DB *** - чтение дерева партиций;

– *** Cleaning unsaved partitions *** - удаление несохраненных партиций;

– *** No partition data found for table <название таблицы>, building new partition tree *** - первоначальное разбиение таблицы;

– *** Updating partition tree for table <название таблицы> - обновляется разбиение для таблицы;

– *** Writing changes to partition tree to database *** - запись изменений дерева партиций в базу данных;

– *** Copiing data from old table to new partition tree *** - копирование данных из родительской таблицы в партиции;

– *** Creating indices *** - создание индексов для партиций;

– *** Saving partition tree and finalizing changes *** - завершение записи изменений в базу данных;

– *** Initial tree for <название таблицы> created, relaunching update *** - изначальное разбиение проведено, запущено обновление разбиения для указанной таблицы;

– Partition <название таблицы> is empty, not dumping - партиция не содержит данных;

– Add new partition <название таблицы> as child of <название таблицы> - добавлена новая партиция;

– Delete partition <название таблицы> because of age - партиция со старыми данными помечена для удаления;

– Delete partition <название таблицы> to free up disk space - партиция помечена для удаления для освобождения дискового пространства;

– *** Building new partition tree for table=<название таблицы>, dtime=<дата/время> *** - изначальное разбиение для таблицы;

– Partition <название таблицы> read from partition_data - прочитана информация о партиции;

– Table <название таблицы> not recorded in partition_data; drop - запись о партиции отсутствует в таблице partition_data, партиция удаляется;

– !!! Failed to copy data, read process returned error code !!! - процесс чтения данных из корневой таблицы завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Failed to copy data, write process returned error code !!! - процесс записи данных в партиции завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Failed to copy data, num_lines check failed, <число> != <число> - копирование данных произошло с ошибкой, число прочитанных строк не равно числу записанных;

– *** Disk usage by DB is <число> *** - процент использования дискового пространства базой данных;

– *** Disk usage is above treshold <число>, trying to free up disk space, cycle <число> *** - квота дискового пространства превышена, попытка очистки свободного пространства;

– !!! Failed to free up disk space !!! - количество циклов освобождения дискового пространства превысило пять, процесс освобождения дискового пространства остановлен;

2.5.6.2 Информационные сообщения, сигнализирующие о проблемах

– IOError: [Errno 13] Permission denied: '/var/lock/subsys/tm_partition' - невозможно открыть для записи lock-файл. Возможно, partition был запущен не от пользователя root;

– IOError: [Errno 13] Permission denied: '<путь к файлу лога>' - невозможно открыть для записи файл лога. Проверьте путь к файлу лога и права доступа на родительскую директорию;

– IOError: [Errno 11] Resource temporarily unavailable - невозможно открыть для записи lock-файл, т.к. он уже заблокирован. Проверьте, не запущен ли ещё один экземпляр partition;

– ERROR Cannot connect to DB '<строка>' psycopg2.OperationalError: ВАЖНО: база данных "<строка>" не существует - база данных не существует, проверьте название базы данных, переданное приложению;

– psycopg2.ProgrammingError: ОШИБКА: отношение "<название таблицы>" не существует - таблица не найдена, проверьте корректность базы данных;

– !!! PSQL subprocess returned error code !!! - внешний процесс psql, завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Cannot open or create directory <путь к директории>: <текст ошибки> !!! - невозможно открыть или создать папку для сохранения дампов. Обратите внимание на текст ошибки, следует проверить возможность доступа к указанной папке и права доступа к ней;

– !!! Failed to chmod directory <путь к директории>: <текст ошибки> !!! - невозможно сменить права доступа для папки с дампами. Обратите внимание на текст ошибки, следует проверить возможность доступа к указанной папке и возможность смены прав доступа к ней;

– !!! Cannot open dump file <путь к файлу>: <текст ошибки> !!! - невозможно открыть файл для сохранения дампа в режиме записи. Проверьте возможность открытия файла и права доступа к нему;

– !!! pg_dump returned non-zero exit status !!! - процесс pg_dump завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Partition tree for table <название таблицы> is broken !!! - дерево партиций для таблицы некорректно, проверьте логи предыдущих запусков partition;

– !!! Show data_directory failed, cannot estimate disk usage !!! - запрос директории, в которой находятся данные базы данных, завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Cannot calculate disk usage, df returned error code <число> !!! !!! <сообщение об ошибке> !!! - запуск команды df для определения свободного места на разделе с данными базы данных завершился с ошибкой. Проверьте доступность и работоспособность команды df;

– !!! Couldn't parse df output, unexpected format <вывод df> !!! - получен нестандартный вывод команды df. Проверьте работоспособность команды df;

– !!! Failed to tar gzip directory <путь к директории> !!! - процесс tar для создания архива .tar.gz завершился с ошибкой. Проверьте доступность и работоспособность команды tar и наличие и доступность директории с дампами;

– !!! Failed to chmod file <путь к файлу>: <текст ошибки>!!! - невозможно сменить права доступа для архива с дампами. Обратите внимание на текст ошибки, следует проверить возможность доступа к указанному файлу и возможность смены прав доступа к нему.

2.6 extremum

2.6.1 Общие сведения о программе

extremum является консольным приложением для расчета экстремумов на интервалах 5 минут, 1 час, 12 часов.

Запуск extremum производится автоматически с помощью cron. Запускаться extremum должен от пользователя root, так как требуется доступ к каталогу /var/lock на запись.

extremum является запакованным приложением, написанным на языке Python, поэтому при запуске он распаковывается во временном каталоге (по умолчанию /tmp), а при окончании работы приложения распакованные файлы удаляются.

extremum принимает следующие параметры командной строки:

– Опциональные параметры:

– -h - показ справки;

– –v - показ версии приложения;

Во время запуска partition считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение.

Из секции [Extremum]:

– LogFile - полное имя файла, в который будет записываться лог. По умолчанию: /var/log/extremumcalculator.log;

– Debug - если true, то приложение работает в режиме отладки (логгер переходит в режим DEBUG, если false - INFO);

– StartTime - начальное время расчета экстремумов. Если не задано, то используется время окончания прошлого интервала обработки, сохраненное в таблице extremum_data. Если в таблице extremum_data нет информации по какой-либо таблице, то используется минимальная временная метка в этой таблице;

– EndTime - конечное время расчета экстремумов. Если не задано, то используется текущее время или максимальная временная метка в таблице.

Прочитав параметры, extremum блокирует файл /var/lock/subsys/tm_extremum для предотвращения запуска двух одновременно работающих экземпляров приложения.

Алгоритм работы приложения следующий:

– чтение списка таблиц с данными на основе таблиц subsys (подсистемы), signal_types (типы сигналов), sakt, system_statistics_values, calculated_values, cla_sakt_output, predict_sakt, import_sakt (сигналы)

– создание таблицы extremum_data (сведения о вычисленных экстремумах), если её не существует;

– для каждой таблицы с данными (только аналоговые):

– если таблицы для экстремумов не существует, то она создается;

– запускается процесс чтения данных со стандартного входа (STDIN);

– вычисляются экстремумы, исключаются дубляжы данных, данные записываются в таблицу экстремумов;

– по окончании вычисления экстремумов для таблицы с данными, в таблице extremum_data обновляется дата последних вычисленных экстремумов.

Для корректной работы приложения требуются:

– наличие свободного пространства во временном каталоге (по умолчанию /tmp) не менее 12 мегабайт

– таблицы с данными подсистем и таблица extremum_data с корректной структурой, или полностью отсутствующие.

2.6.2 Структура программы

Модуль состоит из исполняемого файла extremum.

/usr/local/bin/diaprom/data/dbscripts/extremum - исполняемый файл extremum.

Помимо исполняемого файла для корректной работы модуля необходим сервер базы данных PostgreSQL с полностью корректно установленной базой данных.

2.6.3 Настройка программы

Обновление модуля заключается в замене исполняемого файла extremum на новый.

Дополнительной настройки модуль не требует.

2.6.4 Проверка программы

Проверка работоспособности программного модуля осуществляется посредством проверки лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

2.6.5 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.6.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует extremum, сохраняются в лог файл, указанный параметром LogFile секции Extremum конфигурационного файла skpt.ini, при каждом запуске.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.6.6.1 Информационные сообщения, выводимые при штатной работе модуля

Запущено вычисление экстремумов <дата/время> - информационное сообщение, сигнализирующее о запуске приложения.

Завершение работы <дата/время> - информационное сообщение, сигнализирующее о завершении работы приложения.

– INFO ### BEGIN EXECUTION, DTIME=”<дата/время> ### - информационное сообщение, сигнализирующее о запуске приложения;

– *** Updating table <название таблицы> *** - обновление экстремумов для заданной таблицы;

– Using working interval [<дата/время>,<дата/время>] - вывод обрабатываемого интервала времени;

– Calculating extremums of type <тип экстремумов> on interval [<дата/время>,<дата/время>] - информационное сообщение, сигнализирующее о начале вычисления экстремумов для заданного типа экстремумов и интервала времени;

– No data in table <название таблицы>, nothing to calculate - в заданной таблице нет данных, пропускается.

2.6.6.2 Информационные сообщения, сигнализирующие о проблемах

– IOError: [Errno 13] Permission denied: '/var/lock/subsys/tm_extremum' - невозможно открыть для записи lock-файл. Возможно, extremum был запущен не от пользователя root;

– IOError: [Errno 13] Permission denied: '<путь к файлу лога>' - невозможно открыть для записи файл лога. Проверьте путь к файлу лога и права доступа на родительскую директорию;

– IOError: [Errno 11] Resource temporarily unavailable - невозможно открыть для записи lock-файл, так как он уже заблокирован. Проверьте, не запущен ли ещё один экземпляр extremum;

– ERROR Cannot connect to DB '<строка>' psycopg2.OperationalError: ВАЖНО: база данных "<строка>" не существует - база данных не существует, проверьте название базы данных, переданное приложению;

– psycopg2.ProgrammingError: ОШИБКА: отношение "<название таблицы>" не существует - таблица не найдена, проверьте корректность базы данных;

– !!! psql COPY returned error !!! - процесс psql, записывающий данные экстремумов завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! PSQL subprocess returned error code !!! - внешний процесс psql завершился с ошибкой. Проверьте работоспособность базы данных.

2.7 extremum_partition

2.7.1 Общие сведения о программе

Extremum_partition является консольным приложением для создания и поддержки партиционирования таблиц с экстремумами.

Запуск extremum_partition производится автоматически с помощью cron. Запускаться extremum_partition должен от пользователя root, так как требуется доступ к каталогу /var/lock на запись.

extremum_partition является запакованным приложением, написанным на языке Python, поэтому при запуске он распаковывается во временном каталоге (по умолчанию /tmp), а при окончании работы приложения распакованные файлы удаляются.

extremum_partition принимает следующие параметры командной строки:

– опциональные параметры:

– -h - показ справки;

– -v - показ версии приложения;

Во время запуска extremum_partition считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение.

Из секции [ExtremumPartition]:

– PartitionParamNum - количество сигналов в одной дочерней таблице при разбиении по ID сигнала. По умолчанию: 1;

– PartitionSize - временной интервал, занимаемый дочерней таблицей при разбиении по времени (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 1 день;

– FutureInterval - временной интервал в будущем, который должен быть покрыт дочерними таблицами после работы приложения (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 365 суток;

– PastInterval - временной интервал в прошлом, который должен быть покрыт, если параметр PartitionBackwards=true (в секундах или в сутках если указано d в конце, например, -P 10d задает интервал в 10 суток, -P 3600 задает интервал в 1 час). По умолчанию: 365 суток;

– PartitionBackwards - если true, то приложение при необходимости создает в прошлом пустые таблицы, так чтобы интервал, заданный параметром PastInterval, был ими полностью покрыт;

– LogFile - полное имя файла, в который будет записываться лог. По умолчанию: /var/log/partitionmanager.log;

– Debug - если true, то приложение работает в режиме отладки (логгер переходит в режим DEBUG, если false - INFO);

Прочитав параметры, partition блокирует файл /var/lock/subsys/tm_partition для предотвращения запуска двух одновременно работающих экземпляров приложения.

Тип партиционирования, используемый приложением, - двухуровневое, по ID сигнала и по дате/времени.

Алгоритм работы приложения следующий:

– чтение списка таблиц с данными на основе таблиц subsys (подсистемы), signal_types (типы сигналов), sakt, system_statistics_values, calculated_values, cla_sakt_output, predict_sakt, import_sakt (сигналы);

– создание таблицы extremum_partition_data (сведения о партициях) если её не существует;

– для каждой таблицы с экстремумами:

– если разбиение на партиции уже происходило ранее - обновление партиций (создание новых, сохранение в дамп и удаление старых), иначе - начальное разбиение на партиции (при начальном разбиении, после разбиения данные переносятся из родительской таблицы в дочерние партиции).

Для корректной работы приложения требуются:

– наличие свободного пространства во временном каталоге (по умолчанию /tmp) не менее 12 мегабайт;

– таблицы с данными подсистем и таблица extremum_partition_data с корректной структурой, или полностью отсутствующие.

2.7.2 Структура программы

Модуль состоит из исполняемого файла extremum_partition.

/usr/local/bin/diaprom/data/dbscripts/extremum_partition - исполняемый файл extremum_partition

Помимо исполняемого файла для корректной работы модуля необходим сервер базы данных PostgreSQL с полностью корректно установленной базой данных.

2.7.3 Настройка программы

Обновление модуля заключается в замене исполняемого файла extremum_partition на новый.

Дополнительной настройки модуль не требует.

2.7.4 Проверка программы

Проверка работоспособности программного модуля осуществляется посредством проверки лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

2.7.5 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.7.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует partition, сохраняются в лог файл, указанный параметром LogFile секции ExtremumPartition конфигурационного файла skpt.ini, при каждом запуске.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.7.6.1 Информационные сообщения, выводимые при штатной работе модуля

– Запущено обновление партиционирования экстремумов <дата/время> - информационное сообщение, сигнализирующее о запуске приложения;

– Завершение работы <дата/время> - информационное сообщение, сигнализирующее о завершении работы приложения;

– INFO ### BEGIN EXECUTION, DTIME=”<дата/время> ### - информационное сообщение, сигнализирующее о запуске приложения;

– Table extremum_partition_data does not exist, creating. - таблица partition_data не существует, происходит её создание;

– *** Updating table=<название таблицы>, dtime=<дата/время> *** - обновляется разбиение для указанной таблицы;

– *** Reading partition tree from DB *** - чтение дерева партиций;

– *** Cleaning unsaved partitions *** - удаление несохраненных партиций;

– *** No partition data found for table <название таблицы>, building new partition tree *** - первоначальное разбиение таблицы;

– *** Updating partition tree for table <название таблицы> - обновляется разбиение для таблицы;

– *** Writing changes to partition tree to database *** - запись изменений дерева партиций в базу данных;

– *** Copiing data from old table to new partition tree *** - копирование данных из родительской таблицы в партиции;

– *** Creating indices *** - создание индексов для партиций;

– *** Saving partition tree and finalizing changes *** - завершение записи изменений в базу данных;

– *** Initial tree for <название таблицы> created, relaunching update *** - изначальное разбиение проведено, запущено обновление разбиения для указанной таблицы;

– Add new partition <название таблицы> as child of <название таблицы> - добавлена новая партиция;

– *** Building new partition tree for table=<название таблицы>, dtime=<дата/время> *** - изначальное разбиение для таблицы;

– Partition <название таблицы> read from extremum_partition_data - прочитана информация о партиции;

– Table <название таблицы> not recorded in extremum_partition_data; drop - запись о партиции отсутствует в таблице extremum_partition_data, партиция удаляется;

– !!! Failed to copy data, read process returned error code !!! - процесс чтения данных из корневой таблицы завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Failed to copy data, write process returned error code !!! - процесс записи данных в партиции завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Failed to copy data, num_lines check failed, <число> != <число> - копирование данных произошло с ошибкой, число прочитанных строк не равно числу записанных.

2.7.6.2 Информационные сообщения, сигнализирующие о проблемах

– IOError: [Errno 13] Permission denied: '/var/lock/subsys/extremum_partition' - невозможно открыть для записи lock-файл. Возможно, extremum_partition был запущен не от пользователя root;

– IOError: [Errno 13] Permission denied: '<путь к файлу лога>' - невозможно открыть для записи файл лога. Проверьте путь к файлу лога и права доступа на родительскую директорию;

– IOError: [Errno 11] Resource temporarily unavailable - невозможно открыть для записи lock-файл, т.к. он уже заблокирован. Проверьте, не запущен ли ещё один экземпляр partition;

– ERROR Cannot connect to DB '<строка>' psycopg2.OperationalError: ВАЖНО: база данных "<строка>" не существует - база данных не существует, проверьте название базы данных, переданное приложению;

– psycopg2.ProgrammingError: ОШИБКА: отношение "<название таблицы>" не существует - таблица не найдена, проверьте корректность базы данных;

– !!! PSQL subprocess returned error code !!! - внешний процесс psql, завершился с ошибкой. Проверьте работоспособность базы данных;

– !!! Cannot open or create directory <путь к директории>: <текст ошибки> !!! - невозможно открыть или создать папку для сохранения дампов. Обратите внимание на текст ошибки, следует проверить возможность доступа к указанной папке и права доступа к ней;

– !!! Failed to chmod directory <путь к директории>: <текст ошибки> !!! - невозможно сменить права доступа для папки с дампами. Обратите внимание на текст ошибки, следует проверить возможность доступа к указанной папке и возможность смены прав доступа к ней;

– !!! Partition tree for table <название таблицы> is broken !!! - дерево партиций для таблицы некорректно, проверьте логи предыдущих запусков partition.

2.8 ModelsLauncher

2.8.1 Общие сведения о программе

Модуль предназначен для инициализации, запуска и управления работой сконфигурированных моделей. В задачи модуля также входит контроль состояния работающих моделей, передача информации о работе модели.

Для работы модуля необходимо соответствие пакетов anaconda следующим версиям:

- json5==0.9.5;
- jsonschema==3.2.0;
- Keras-Preprocessing==1.1.2;
- matplotlib==3.3.2;
- numpy==1.19.2;
- numpydoc==1.1.0;
- pandas==1.2.1;
- pandocfilters==1.4.3;
- ruptures==1.1.3;
- scikit-image==0.17.2;
- scikit-learn==0.23.2;

- scipy==1.6.0;
- seaborn==0.11.1;
- statsmodels==0.12.1;
- tensorboard==2.4.0;
- tensorboard-plugin-wit==1.7.0;
- tensorflow==2.4.0;
- tensorflow-estimator==2.4.0;
- tensorflow-probability==0.12.0;
- tqdm==4.56.0.

Запуск ModelsLauncher производится через SpaGui или автоматически с помощью cron (для режима онлайн). Для запуска используется скрипт start_all.py, который может принимать следующие параметры командной строки:

- config – путь к глобальному конфигурационному файлу. В общем случае файл формируется автоматически во время работы модуля SpaGui и раположен в /home/diaprom/.diaprom/spa/global_config.json. Если файл отсутствует или некорректен, модуль завершит свое выполненеие;
- model-name – имя запускаемой модели;
- expert-mode – запуск в режиме эксперта (для модуля SpaGui);
- expert-mode-with-train - запуск в режиме эксперта с обучением (для модуля SpaGui);

Для корректной работы ModelsLauncher требуется запущенный Server, запуск которого производится автоматически во время старта операционной системы. В случае остановки сервиса пользователем или аварийного прекращения работы сервис будет перезапущен автоматически с помощью cron.

Во время запуска ModelsLauncher считывает всю информацию из глобального конфигурационного файла, по пути, переданному через аргумент «config». В случае, если конфигурационный файл не прошел валидацию, модуль завершает свою работу с указанием ошибки. Файл должен соответствовать типу JSON. Для модели, указанной аргументом «model-name», или всех сконфигурированных моделей в файле (при отсутствии аргумента) считываются соответствующие конфигурации демона.

Для доступа к БД модуль считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL, на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

Если не указано имя запускаемой модели, то для каждой конфигурации запускается процесс в режиме демона через скрипт model_daemon.py. Демон представляет собой подмодуль, который передает данные модели и отправляет результат предсказаний на сервер.

Этапы работы демона:

– обучение модели (производится один раз для модели);

– выдача предсказаний.

Если не указан опциональный аргумент «not-train-model», то запускается обучение модели, для этого:

– считываются данные для обучения из файлов, которые указаны в конфигурации демона;

– считанные данные передаются модулю препроцессинга, так как перед передачей данных в модель (для обучения/переобучения и предсказаний) они должны быть подготовлены;

– обработанные данные передаются модели для обучения.

Для получения данных осуществляется подписка на сигналы, которые указаны в конфигурации. По таймеру с интервалом, равным шагу сэмплирования, передаются все ранее полученные данные в модуль препроцессинга. Обработанные данные добавляются в буфер данных для модели. При достижении необходимого числа данных (или при запуске демона после обучения) запускается прогнозирование.

Результаты прогнозирования отправляются на сервер.

2.8.2 Структура программы

Файловая структура модуля следующая:

– /usr/local/bin/diaprom/submodules/pint – каталог файлов модуля:

- src – каталог исходных кодов модуля;
- data – файлы данных;
- work_dir/logs – файлы логов;
- work_dir/daemon_cfg – конфигурационные файлы демонов;
- work_dir/output_data –данные моделей;
- work_dir/logs – файлы логов;

Помимо перечисленных файлов для корректной работы модуля необходимы сервер базы данных PostgreSQL с полностью корректно установленной базой данных, сервер оперативных данных, а также исполняемые файлы конфигурируемых моделей.

2.8.3 Настройка программы

Процесс настройки модуля можно разделить на три этапа.

Первый этап заключается в первичной установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Второй этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP-адрес сервера базы данных PostgreSQL. Единственно возможный вариант - 127.0.0.1. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс.

В скрипте запуска сервисов (Check_services) указываются условия запуска модуля и команда для его выполнения. Скрипт должен быть прописан в журнале cron (/etc/crontab) с интервалом запуска, равным одной минуте.

Третий этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта, необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.8.4 Проверка программы

Проверка работоспособности программного модуля осуществляется посредством проверки лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

2.8.5 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.8.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует ModelsLauncher, сохраняются в каталоге /usr/local/bin/diaprom/submodules/pint/work_dir/logs. Для каждой запускаемой модели создается свой собственный файл логов.

Список возможных сообщений приведен ниже:

- INFO:root:[<Наименование функции>] – информационное сообщение, сигнализирующее о выполнении соответствующей функции;
- INFO:root:Preprocessing model <Имя модели> is ready to forecast – информационное сообщение, сигнализирующее о готовности к прогнозированию модели предварительной обработки;
- INFO:root:[<Наименование функции>] Эпоха <Номер эпохи>/<Количество эпох> – информационное сообщение, сигнализирующее о прогрессе обучения.
- INFO:root:[<Наименование функции>], path: <Путь> - информационное сообщение, сигнализирующее о сохранении файлов по пути <Путь>;
- CRITICAL:root:('model state = c_model_states_enum.any_error, error = <Ошибка модели>') – сообщение, сигнализирующее о возникновении ошибки во время работы модели. Для решения необходимо обратится к документации модели, при работе которой произошла ошибка.

2.9 OldRmsod

2.9.1 Общие сведения о программе

OldRmsod является программой с графическим интерфейсом, предназначенной для взаимодействия с ПТК и наблюдения за ним.

Во время запуска OldRmsod считывает информацию из секции [Database] конфигурационного файла skpt.ini:

- DatabaseHost - IP адрес устройства, на котором запущен сервер базы данных PostgreSQL;
- DatabasePort - порт сервера базы данных PostgreSQL, на котором он слушает входящие соединения;
- DatabaseName - название базы данных проекта;
- DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;
- DatabasePass - пароль пользователя базы данных;

Прочитав параметры, OldRmsod подключается к серверу базы данных и читает сведения о пользователях из таблицы users, сведения о группах пользователей из таблицы groups. Затем выводится форма авторизации, в которой требуется выбрать нужного

пользователя из списка и ввести пароль этого пользователя. При успешной авторизации начнется загрузка основного окна программы.

Для корректной работы приложения таблицы users и groups должны иметь корректную структуру и содержать корректные данные о пользователях.

В случае если таблица groups имеет некорректную структуру, следует пересоздать её, выполнив следующую команду в терминале (внимание: данные, содержащиеся в таблице groups, при выполнении следующего действия будут удалены):

psql -U postgres -d DB_NAME

< /usr/local/bin/diaprom/data/dbscripts/tables/common/groups.sql

, где DB_NAME - имя базы данных

В случае если таблица users имеет некорректную структуру, следует пересоздать её, выполнив следующую команду в терминале (внимание: данные, содержащиеся в таблице users, при выполнении следующего действия будут удалены):

psql -U postgres -d DB_NAME

< /usr/local/bin/diaprom/data/dbscripts/tables/common/users.sql

, где DB_NAME - имя базы данных

Примечание: если обе таблицы имеют некорректную структуру, то сначала следует пересоздать таблицу groups и только после этого пересоздавать таблицу users.

В процессе инициализации основного окна OldRmsod читает список всех подсистем из таблицы subsys базы данных и подписывается на изменения списка клиентов для вывода состояния каждой подсистемы.

Когда основное окно программы загружено, OldRmsod читает следующие параметры из конфигурационного файла skpt.ini:

– из секции [FsWatcher]:

– MaxDiskUsagePercent - порог занятого дискового пространства (в процентах от общего объема раздела), свыше которого будет выводиться информационное сообщение о том, что место на диске заканчивается;

– MinDiskDays - количество дней, на которые следует составлять прогноз использования дискового пространства;

– CheckSpaceMinutesInterval - интервал (в минутах), с которым будет запускаться процедура проверки занятости дискового пространства;

– Hosts - список адресов машин, на которых следует проверять занятость дискового пространства;

– MessagesInterval - интервал, с которым будет выводиться информационное сообщение о том, что место на диске заканчивается.

– из секции [UI]

– isDiagnosticFormatVisible - отображать ли вкладку “Основной экран”;
– isMnemoFormatVisible - отображать ли вкладку “Мнемосхемы”;
– isChartsFormatVisible - отображать ли вкладку “Графики”;
– isTablesFormatVisible - отображать ли вкладку “Таблицы”.

Также читаются данные об истории использования дисковых разделов из таблицы fs_status базы данных. Используя полученные данные, OldRmsod проверяет занятость дискового пространства на машинах, указанных в параметре Hosts, посредством удаленного запуска команды “df -h” через ssh и записывает их в таблицу fs_status, а также прогнозирует занятость дискового пространства вперед на количество дней, указанных параметром MinDiskDays.

2.9.2 Сведения о разделах

2.9.2.1 Основной экран

Для корректной работы раздела требуются следующие корректно настроенные таблицы:

– communication_servers - сведения о серверах оперативных данных;
– mnemo_leds - сведения об индикаторах каналов на мнемосхеме Основного окна.

Затем приложение подключается к базе данных, считывает информацию о серверах оперативных данных из таблицы communication_servers, открывает соединение с ними и регистрируется в них в качестве наблюдателя, подписываясь на уведомления.

Это позволяет приложению не запрашивать данные у серверов оперативных данных напрямую, а обрабатывать данные по мере поступления от серверов уведомлений, на которые оно подписано.

При потере связи с серверами OldRmsod автоматически пытается восстановить соединение с периодичностью 1 раз в 5 секунд. В случае успешного восстановления связи приложение отправляет повторный запрос на подписку на перечисленные уведомления, как при своем старте.

2.9.2.2 Мнемосхемы

Для корректной работы раздела требуются следующие корректно настроенные таблицы:

– subsys - сведения о подсистемах;
– sakt - сведения о сигналах;
– signal_types - сведения о типах сигналов;
– quality - сведения о признаках достоверности;
– units - сведения о единицах измерения;
– transfer_direction - сведения о направлениях передачи данных;

– communication_servers - сведения о серверах оперативных данных;
– calculated_values - сведения о вычисляемых значениях;
– diagnostic_knowledge_base - сведения о диагностических признаках;
– таблицы с данными подсистем (<имя_подсистемы>_<тип сигнала>);
– mnemo - сведения о мнемосхемах;
– mnemoleds - сведения об индикаторах на мнемосхемах и их привязке к сигналам.

Также для корректной работы раздела требуется наличие svg-файлов - фонов вкладок-мнемосхем, имена которых прописаны в таблице mnemo, в столбце pic_source.

При инициализации раздела OldRmsod читает из таблицы communication_servers базы данных сведения о серверах оперативных данных, открывает с ними соединение, регистрируется в качестве наблюдателя и подписывается на сигналы, к которым привязаны индикаторы. После этого OldRmsod переходит в “слушающий” режим. Это означает, что модуль не запрашивает данные у серверов напрямую, а ждет, когда серверы вышлют уведомление о поступлении новых данных по сигналам, на которые он подписан. Получив уведомление, модуль обновляет состояние индикаторов в соответствии с полученными данными сигналов.

При потере связи с серверами OldRmsod автоматически пытается восстановить соединение. В случае успешного восстановления связи, модуль отправляет повторный запрос на подписку на сигналы. Дальнейшая процедура функционирования модуля повторяет ту, что и при инициализации раздела.

2.9.2.3 Графики

Для корректной работы раздела требуются следующие корректно настроенные таблицы:

– subsys - сведения о подсистемах;
– sakt - сведения о сигналах;
– communication_servers - сведения о серверах оперативных данных;
– calculated_values - сведения о вычисляемых значениях;
– таблицы с данными подсистем (<имя_подсистемы>_<тип сигнала>);
– таблицы с данными экстремумов (<имя_подсистемы>_<тип сигнала>_<тип_экстремумов>), где тип экстремумов - один из (“5m”, “1h”, “12h”).

Раздел предназначен для визуализации имеющихся данных посредством построения графиков. Возможно создание нового экрана (кнопка “Создать экран”) и его настройка посредством диалога “Настройка графика” и открытие ранее созданного и сохраненного в файле Xml-формата экрана (кнопка “Открыть”), хранящегося в директории /usr/local/bin/diaprom/data/OnlineViewerApp.

Для чтения текущих данных необходим запущенный сервер оперативных данных и сведения о нем или о них (если серверов несколько) в таблице communication_servers.

2.9.2.4 Таблицы

Для корректной работы раздела требуются следующие корректно настроенные таблицы:

- subsys - сведения о подсистемах;
- sakt - сведения о сигналах;
- signal_types - сведения о типах сигналов;
- quality - сведения о признаках достоверности;
- units - сведения о единицах измерения;
- communication_servers - сведения о серверах оперативных данных;
- calculated_values - сведения о вычисляемых значениях;
- diagnostic_knowledge_base - сведения о диагностических признаках;
- таблицы с данными подсистем (<имя_подсистемы>_<тип сигнала>).

Раздел предназначен для просмотра имеющихся данных в виде таблицы и их сохранения. Для чтения текущих данных необходим запущенный сервер оперативных данных и сведения о нем или о них (если серверов несколько) в таблице communication_servers. Для сохранения архивных данных необходимо наличие приложения DumpCreator, находящегося в директории /usr/local/bin/diaprom/data/dumpcreator, и сопутствующих ему файлов (файлы локализации и логотип).

При инициализации раздела OldRmsod читает из таблицы communication_servers базы данных сведения о серверах оперативных данных, открывает с ними соединение, регистрируется в качестве наблюдателя и подписывается на выбранные пользователем сигналы. После этого OldRmsod переходит в “слушающий” режим. Это означает, что модуль не запрашивает данные у серверов напрямую, а ждет, когда серверы вышлют уведомление о поступлении новых данных по сигналам, на которые он подписан. Получив уведомление, модуль обновляет таблицу в соответствии с полученными данными сигналов.

При потере связи с серверами OldRmsod автоматически пытается восстановить соединение. В случае успешного восстановления связи модуль отправляет повторный запрос на подписку на сигналы. Дальнейшая процедура функционирования модуля повторяет ту, что и при выборе пользователем текущих значений.

2.9.3 Структура программы

Модуль состоит из исполняемого файла skd_gui, а также нескольких дополнительных файлов конфигурации, скриптов запуска и лог файла.

Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл skd_gui;

– /usr/local/bin/diaprom/data/skd_gui.properties - файл конфигурации системы логирования;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /home/diaprom/.diaprom/log/skd_gui.log - лог файл модуля. Создается при первом запуске программного модуля автоматически.

Помимо перечисленных файлов для корректной работы модуля необходимы сервер базы данных PostgreSQL с полностью корректно установленной базой данных и серверы оперативных данных.

2.9.4 Настройка программы

Настройку модуля можно разделить на три этапа.

Первый этап заключается в первичной установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Второй этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации системы логирования (log.properties) наладчик может изменить степень подробности логирования. Не допускается изменение любых других параметров в этом файле.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP адрес сервера базы данных PostgreSQL. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс.

Третий этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.9.5 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена следующим способом: выполнить команду в терминале ps -A | grep skd_gui. Если модуль запущен, тогда ответом на команду будет номер процесса.

Проверка работоспособности программного модуля может быть осуществлена в несколько шагов.

Проверка лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

Проверка посредством запуска модуля. Если приложение запускается и возможна авторизация в нем, то это косвенно указывает на нормальное функционирование модуля.

2.9.5.1 Проверка разделов

Раздел “Основной экран”. Если запущен сервер оперативных данных - выбрать несколько сигналов на гистограмме или температурном графике. Если данные отображаются, то это указывает на нормальное функционирование раздела.

Раздел “Мнемосхемы”. Если запущен сервер оперативных данных, то индикаторы будут отображать состояние привязанного сигнала, будут активными (то есть не черными).

Раздел “Графики”. Создать экран с одним или несколькими графиками с данными из архива или текущими значениями (если сервер оперативных данных запущен). Если данные отображаются, то это указывает на нормальное функционирование раздела.

Раздел “Таблицы”. Сделать выборку данных из архива или текущих значений (если сервер оперативных данных запущен). Если данные отображаются, то это указывает на нормальное функционирование раздела.

2.9.6 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.9.7 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует OldRmsod, сохраняются в лог файл /home/diaprom/.diaprom/log/skd_gui.log

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.9.7.1 Информационные сообщения, выводимые при штатной работе модуля

– <дата> <время> INFO - Starting application '<название АЭС>. Блок<номер блока>. <название системы>. Версия - <версия системы>' from diaprom - www.diaprom.ru. All rights reserved. - информационное сообщение, сигнализирующее о запуске;

– Application '<название АЭС>. Блок<номер блока>. <название системы>. Версия - <версия системы>' finished with return code <число>. - информационное сообщение, сигнализирующее о закрытии приложения;

– Хост: <ip-адрес>

Устройство: <путь к файлу устройства>

Точка монтирования: <точка монтирования>

Осталось <число>% пространства из <размер раздела>

Пожалуйста, освободите место на соответствующем разделе! - сообщение, сигнализирующее о заполненности одного из разделов диска;

– LOADING FILES (<список>) - список обнаруженных файлов с сохраненными экранами;

– void SystemInfoController::refreshModel() looks like info gathering task takes too much time - предыдущий процесс сбора информации о системе ещё не завершился.

2.9.7.2 Информационные сообщения, сигнализирующие о проблемах

– Database INSTANCE ERROR: <подробное описание ошибки> - происходит ошибка запроса к базе данных. Требуется подключиться вручную по конфигурации, которую можно найти в skpt.ini к базе данных, после чего выполнить запрос, который будет описан в сообщении. После этого можно будет получить более подробное сообщение об ошибке уже от самой базы или убедиться в некорректности работы модуля;

– something wrong with bash. Is bash installed? - команда запущена не была. Проверьте наличие и работоспособность командной оболочки bash;

– bash with args -c <команда> could not finish in time <число> ms check if your hard drive is still alive - запущенный процесс не был завершен за <число> миллисекунд. Проверьте работоспособность системы.

2.10 SpaGui

2.10.1 Общие сведения о программе

SpaGui является программой с графическим интерфейсом, предназначенной для многомерного анализа временных рядов, их прогнозирования, обнаружения их разладки, а также решения задач классификации и кластеризации.

Программное обеспечение может работать в двух режимах: экспертный – предназначен для первичной обработки данных, их подготовки для дальнейшего исследования и последующего применения как одномерных, так и многомерных моделей, включая различные нейронные сети; онлайн – для запуска в режиме реального времени на объекте исследования.

Первоначально SpaGui запускается в экспертном режиме. Во время запуска проверяется наличие необходимых каталогов по пути /home/diaprom/.diaprom/spa (создаются при отсутствии) и актуальность конфигурационных файлов моделей и методов. Если обнаружена сохраненная сессия происходит восстановление состояния приложения из проектных конфигурационных файлов.

При запуске также считываются следующие параметры из конфигурационного файла skpt.ini:

– из секции [Database]:

– DatabaseHost - IP адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL, на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

– из секции [UI]:

– Style – стиль графического интерфейса.

Прочитав параметры, SpaGui подключается к серверу базы данных и считывает описание всех сигналов из таблиц: sakt, system_statistics_values, calculated_values, diagnostic_knowledge_base, cla_sakt_output, predict_sakt, import_sakt. Также, SpaGui устанавливает стиль графического интерфейса в соответствии с считанным из конфигурационного файла.

Для возможности работы с методами и моделями необходимо их наличие в каталоге /usr/local/bin/diaprom/submodules/spa согласно требованиям раздела Структура программы.

Для запуска моделей необходим корректно установленный модуль ModelsLauncher в директории /usr/local/bin/diaprom/submodules/pint. В режиме эксперта SpaGui запуск и управление модулем ModelsLauncher полностью осуществляется через интерфейс пользователя при нажатии соответствующей кнопки для сконфигурированной модели. В режиме онлайн запуск ModelsLauncher осуществляется автоматически через cron. Для перехода в данный режим необходимо выбрать соответствующий пункт меню в графическом интерфейсе, после чего SpaGui установит необходимые значения в проектный конфигурационный файл и завершит свою работу. Выход из онлайн режима произойдет автоматически после повторного запуска SpaGui.

2.10.2 Структура программы

Модуль состоит из исполняемого файла spa_gui, нескольких файлов конфигурации, скриптов запуска, файлов конфигурации и запуска моделей и методов, лог файлов.

Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл spa_gui;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /usr/local/bin/diaprom/utilities/ - исполняемый файл вспомогательного ПО ImportWizard;

– /usr/local/bin/diaprom/submodules/spa - каталог файлов для работы с методами и моделями:

- Common - служебный каталог, хранящий общие файлы исходных кодов;
- models - каталог, содержащий информацию о моделях;
- preprocessing/calculation - каталог, содержащий информацию о методах предобработки.

Для добавления нового метода или модели необходимо выполнить следующие шаги:

1) создать каталог с именем метода/модели в соответствующем каталоге для хранения информации;
2) добавить файл исходного кода метода/модели формата: <Имя метода/модели>_model.py;
3) добавить конфигурационный файл метода/модели формата: <Имя метода/модели>_conf.json;
4) добавить файлы описания формата: <Имя метода/модели>_help.pdf, <Имя метода/модели>_report.pdf;

– /home/diaprom/.diaprom/spa – каталог для хранения файлов сессии и проектных конфигурационных файлов, создается автоматически при первом запуске:

- global_config.json – глобальный проектный конфигурационный файл, содержит информацию о настроенных методах и моделях;
- gui_config.json – конфигурационный файл графического интерфейса, содержит информацию о всех элементах добавленных во время работы для текущей сессии;
- settings.ini – файл настроек, хранит информации о размещении и размерах некоторых элементов графического интерфейса;
- ui_log.txt – файл логов, хранит информацию о действиях пользователя;
- signals_data – служебный каталог для хранения загруженных данных сигналов;
- graphs – служебный каталог для хранения данных графиков, формируемых моделями;
- image_files – служебный каталог для хранения файлов обученных моделей;
- predict_data – служебный каталог для хранения результатов работы моделей.

Помимо перечисленных каталогов и файлов для корректной работы модуля необходимы сервер базы данных PostgreSQL с полностью корректно установленной базой данных и сервер оперативных данных.

2.10.3 Настройка программы

Настройку модуля можно разделить на три этапа.

Первый этап заключается в первичной установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Второй этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP адрес сервера базы данных PostgreSQL. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс.

Третий этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новую. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.10.4 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена следующим способом: выполнить команду в терминале ps -A | grep spa_gui. Если модуль запущен, тогда ответом на команду будет номер процесса.

Проверка работоспособности программного модуля может быть осуществлена в несколько шагов.

Проверка лог файлов программного модуля. Если в лог файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

Проверка посредством запуска модуля. Если приложение запускается без ошибок и отображается основной экран, то это косвенно указывает на нормальное функционирование модуля.

2.10.5 Дополнительные возможности

Приложение не содержит дополнительных возможностей.

2.10.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует SpaGui, сохраняются в лог файл /home/diaprom/.diaprom/spa/ui_log.txt

Список возможных сообщений и действий системного программиста приведен ниже:

– <дата> <время>:: SpaGui started. - информационное сообщение, сигнализирующее о запуске;

– <дата> <время>:: Button <Имя кнопки> clicked - информационное сообщение, сигнализирующее о нажатии кнопки;

– <дата> <время>:: <Действие >: <Параметры>- информационное сообщение, сигнализирующее о выполнении пользователем действия <Действие> с параметрами <Параметры>;

2.11 SystemWatcher

2.11.1 Общие сведения о программе

SystemWatcher является системным демоном и предназначен для сбора данных о состоянии системы, на которой демон запущен.

Запуск SystemWatcher производится автоматически во время старта операционной системы. В случае остановки сервиса пользователем или аварийного прекращения работы сервис будет перезапущен автоматически с помощью cron.

При нормальной эксплуатации не требует никаких действий со стороны пользователя.

Может быть запущен в двух режимах:

- штатный режим работы
- режим установки – необходимо передать аргумент --setup при запуске. В таком режиме SystemWatcher создает сигналы в БД, в таблице system_statistics_values для дальнейшей работы. После создания сигналов – завершается. Обычно, в таком режиме демон запускается только один раз – сразу после развертывания БД.

Во время запуска SystemWatcher считывает информацию из секции [Database] конфигурационного файла skpt.ini:

– DatabaseHost – IP-адрес устройства, на котором запущен сервер базы данных PostgreSQL;

– DatabasePort - порт сервера базы данных PostgreSQL. на котором он слушает входящие соединения;

– DatabaseName - название базы данных проекта;

– DatabaseUser - имя пользователя, от имени которого будет выполнено соединение;

– DatabasePass - пароль пользователя базы данных.

Из секции [SystemStatistics]:

– FirstModbusAddress – первый Modbus-адрес для создаваемых сигналов. Используется в режиме установки. По умолчанию: 52000;

– LastModbusAddress – Последний возможный Modbus-адрес для создаваемых сигналов. Используется в режиме установки. По умолчанию: 52998;

– RaidController – целевой RAID-контроллер, информацию о котором следует собирать. По умолчанию: c0;

– RaidLogicalDiskObject – целевой логический диск, информацию о котором следует собирать. По умолчанию: u0;

– RaidTwCliPath – путь к утилите tw_cli, для получения информации о состоянии RAID-массива. Если установлен как пустой, то информация о RAID-массиве собираться не будет. По умолчанию: /opt/3ware/CLI/tw_cli;

– RaidSubunits – массив целевых RAID-единиц, информацию о которых следует собирать. Если установлен как пустой, то информация о RAID-массиве собираться не будет.

Прочитав параметры, SystemWatcher подключается к серверу базы данных и считывает список подсистем из таблицы subsys и список прав доступа из таблицы access_permissions.

Считав необходимые для работы данные, модуль начинает работу: каждые 10 секунд происходит цикл сбора данных. Собираются следующие данные:

- Общий объем оперативной памяти;
- Объем свободной оперативной памяти;
- Объем кэшированной оперативной памяти;
- Объем буфферизованной оперативной памяти;
- Объем используемой оперативной памяти;
- Общий объем swap раздела;
- Свободный объем swap раздела;
- Процент загрузки ЦП;
- Температура ЦП;
- Процент занятого пространства на локальных разделах;
- Статус дисков в RAID-массиве;
- Скорости вентиляторов БСБ.

Собираемые данные отправляются на сервер оперативных данных.

Также, SystemWatcher подписывается на изменение списка подключенных к серверу клиентов и при подключении/отключении какого-либо клиента, это событие регистрируется и отправляется на сервер оперативных данных.

Модуль выполняет логирование информации о начале и завершении своей работы, а также информации об ошибках, возникающих в процессе выполнения им своих функций.

Схема информационных потоков модуля представлена на рисунке 2.11.1.

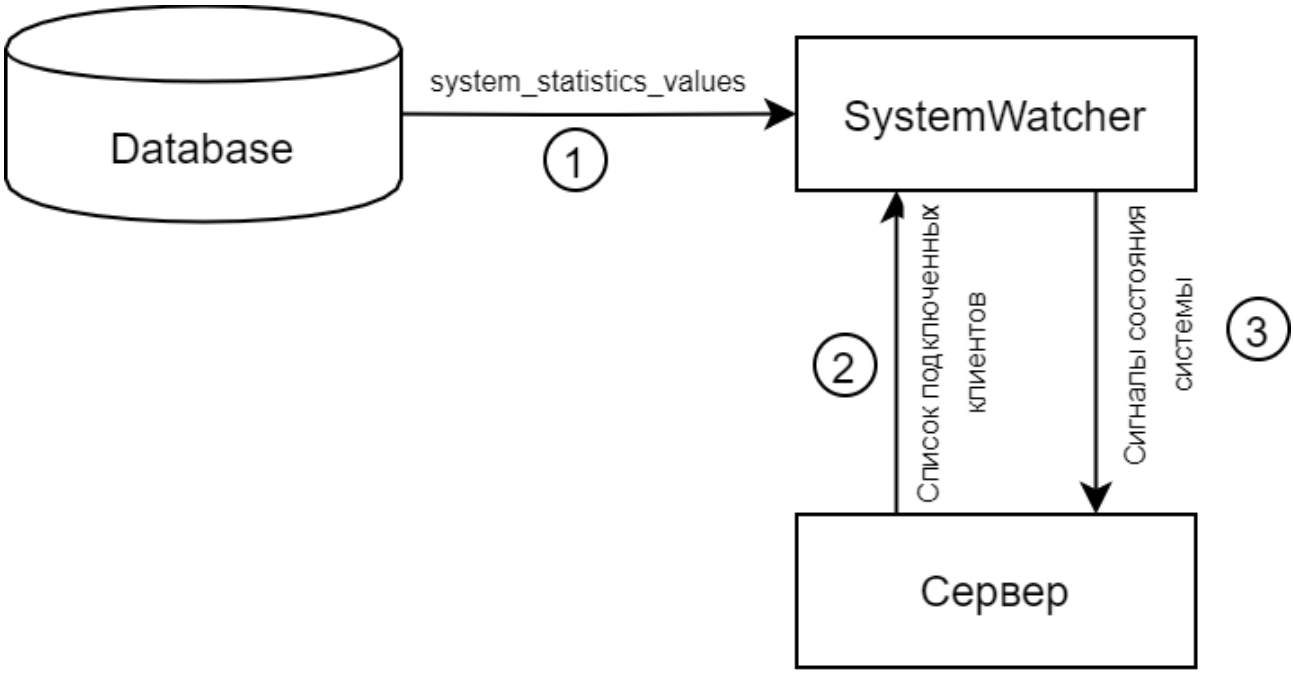


Рисунок 2.11.1

2.11.2 Структура программы

Модуль состоит из исполняемого файла SystemWatcher, а также нескольких дополнительных файлов конфигурации, скриптов запуска и лог-файла. Файловая структура модуля следующая:

– /usr/local/bin/diaprom/bin - исполняемый файл SystemWatcher;

– /usr/local/bin/diaprom/data/SystemWatcher.properties - файл конфигурации системы логирования;

– /usr/local/bin/diaprom/data/skpt.ini - файл конфигурации проектных модулей;

– /etc/init.d/SystemWatcher - скрипт запуска модуля в качестве системного сервиса;

– /var/log/SystemWatcher.log - лог-файл модуля. Создается при первом запуске программного модуля автоматически;

– /root/Check_services - скрипт для запуска сервисов по расписанию cron.

Помимо перечисленных файлов для корректной работы модуля необходимы:

– сервер базы данных PostgreSQL с полностью корректно установленной базой данных;

– серверы оперативных данных;

– служба запуска сервисов по расписанию cron.

Для старта во время загрузки операционной системы сервис должен быть прописан в chkservices на уровнях 2, 3 и 5.

Модуль в своей работе использует следующие таблицы:

– system_statistics_values - хранит информацию о сигналах состояния системы.

2.11.3 Настройка программы

Настройку модуля можно разделить на четыре этапа.

Первый этап - остановка модуля. Выполняется с помощью команды sudo service SystemWatcher stop с последующим вводом пароля пользователя если это потребуется. Если после выполнения вышеуказанной команды будет получено сообщение “SystemWatcher: нераспознанная служба”, значит установка модуля производится впервые.

Второй этап заключается в установке всех компонент программного модуля, описанных в разделе Структура программы, в соответствующие разделы файловой системы. Не допускается устанавливать компоненты программного модуля в пути, отличные от указанных.

Третий этап - конфигурирование. На этом этапе производится уточнение первоначальных параметров в файлах конфигурации и базе данных.

В файле конфигурации проекта (skpt.ini) наладчик указывает IP-адрес сервера базы данных PostgreSQL. Также указывается порт сервера базы данных, имя пользователя базы данных - postgres. Пароль пользователя предустановлен и хранится в файле конфигурации в зашифрованном виде. Его изменение доступно только через пользовательский интерфейс. Также в этом файле необходимо указать параметры для сбора информации о состоянии RAID-массива и общие параметры для создаваемых сигналов состояния системы, если есть необходимость в значениях, отличных от устанавливаемых по умолчанию.

В скрипте запуска сервисов (Check_services) указываются условия запуска модуля и команда для его выполнения. Скрипт должен быть прописан в журнале cron (/etc/crontab) с интервалом запуска, равным одной минуте.

Четвертый этап - обновление. Обновление заключается в замене файлов конфигурации, скриптов запуска и исполняемых файлов проекта на новые. В случае замены исполняемого файла проекта необходима полная замена папки /bin на новые. Частичная замена файлов в этом каталоге приведет к некорректному поведению модулей или к полной их неработоспособности.

2.11.4 Проверка программы

Проверка статуса (запущен или не запущен) модуля может быть осуществлена двумя способами.

Способ первый. Выполнить команду в терминале ps -A | grep SystemWatvcher. Если модуль запущен, то ответом на команду будет номер процесса.

Способ второй. Проверить статус сервиса, выполнив команду sudo service SystemWatcher status.

Проверка работоспособности программного модуля. Проверка лог-файлов программного модуля. Если в лог-файлах отсутствуют какие-либо сообщения об ошибках, то это косвенно указывает на нормальное функционирование модуля. В случае если найдены сообщения об ошибках, нужно устранить причины их возникновения (см. раздел Сообщения системному программисту) перед тем как переходить к следующим проверкам.

Для проверки на наличие в БД информации о сигналах следует выполнить следующий SQL запрос:

SELECT COUNT(*) FROM system_statistics_values;

Ответом будет строка с числом. Если это число равн нулю, то модуль функционировать не будет, т.к. нет сигналов, которые он может использовать для отправки данных на сервер оперативных данных.

2.11.5 Дополнительные возможности

SystemWatcher может быть запущен в двух режимах: демон и не демон. Первый режим запуска является штатным способом запуска программного модуля и должен быть использован в режиме эксплуатации. Второй режим работы подразумевает запуск программного модуля вручную из терминала с указанием дополнительного ключа -nd. Этот режим нужен только для отладки. В этом режиме все сообщения, которые система логирования записывает в лог-файл, отображаются в окне терминала, где запущен модуль. Функции, выполняемые программным модулем, одинаковы для обоих режимов.

2.11.6 Сообщения системному программисту

Все ошибки и информационные сообщения, которые формирует SystemWatcher сохраняются в лог-файл /var/log/SystemWatcher.log.

Список возможных сообщений и действий системного программиста разбит на категории и приведен ниже.

2.11.6.1 Информационные сообщения, выводимые при штатной работе модуля

– Exit with return code: <число> - работа модуля была завершена. Если <число> не равно нулю - это может сигнализировать о проблемах в работе модуля;

– Found main ip: “<адрес>” on interface “<имя сетевого интерфейса>” – найден ip-адрес сетевого интерфейса, через который происходит взаимодействие с сервером оперативных данных;

– Signal “<строка> not available. Please re-execute ./SystemWatcher -nd –setup” – сигнал не найден. Часто такое происходит, если было подключено внешнее запоминающее устройства (например, USB-диск).

2.11.6.2 Информационные сообщения, сигнализирующие о проблемах

– Database INSTANCE ERROR: <подробное описание ошибки> - происходит ошибка запроса к базе данных. Требуется подключиться вручную по конфигурации, которую можно найти в skpt.ini к базе данных, после чего выполнить запрос, который будет описан в сообщении. После этого можно будет получить более подробное сообщение об ошибке уже от самой базы или убедиться в некорректности работы модуля;

– Wrong client IP - отсутствуют права доступа в системе. Проверьте таблицу access_permissions на наличие записи с необходимым клиентом;

– Connections limit exceeded - достигнут лимит подключений. Проверьте, не запущен ли дубликат демона (ps -A | grep SystemWatcher);

– Received ban on the server - получен бан на сервере (несоответствие версий клиента и сервера). Возможно, неправильно была проведена процедура установки/обновления.

Приложение А

(справочное)

Структура конфигурационных таблиц базы данных

Таблица А.1 – Таблица описания прав доступа для клиентов (access_permissions)

Имя поля	Тип	Назначение	Примечание
id	serial	первичный ключ - идентификатор строки таблицы прав доступа	
client_id	varchar (100)	идентификатор клиента	
client_type	varchar (100)	тип клиента	может принимать значения CSC (указывает на работу по внутреннему протоколу CSP) или ModbusTCP (указывает на работу по протоколу Modbus/TCP)
ip	varchar (20)	IP-адрес клиента	
server_id	smallint	вторичный ключ - идентификатор сервера из таблицы communication_servers	
max_connections	smallint	максимально разрешенное число соединений для данного клиента	

Таблица А.2 – Таблица описания типов событий телеграмм (alarm_types)

Имя поля	Тип	Назначение	Примечание

id_event	smallint	первичный ключ - идентификатор типа события	
event_type_name	text	название типа события	
operator	text		
signal	text		
denomination	text		
marker	text		
remark	text	описание типа события	
Имя поля	Тип	Назначение	Примечание
units_is	smallint	идентификатор типа значения телеграммы (таблица unit_bits)	
bit	smallint	установленный бит	

Таблица А.3 – Таблица описания вычисляемых порогов (calculated_treshold)

Имя поля	Тип	Назначение	Примечание
id	serial	идентификатор вычисляемого порога	
diagnostic_feature_id	smallint	идентификатор диагностического признака	
formula	text	формула для вычисления порога	
modb_addr_in	bigint	адрес MODBUS во внутреннем обмене данных	
modb_addr_out	bigint	адрес массива при передаче в СВБУ или СВРК	
dependencies	text	список зависимостей вычисляемых порогов	
time_interval	int	интервал вычислений	

Таблица А.4 – Таблица описания сигналов вычисляемых значений (calculated_values)

Имя поля	Тип	Назначение	Примечание
param_id	serial	первичный ключ – уникальный идентификатор сигнала (параметра)	
param_name	varchar(300)	имя сигнала полное	
param_short_name	varchar(100)	имя сигнала краткое	
formula	text	Формула для вычисления значения сигнала	
param_kks	varchar(300)	код сигнала в системе KKS	зависит от проекта, может быть в другой системе кодирования
param_units	varchar(50)	единицы измерения	
param_units_id	int	идентификатор единиц измерения	
Имя поля	Тип	Назначение	Примечание
dependecies	text	список зависимостей вычисляемого сигнала	
time_interval	int	интервал вычислений	
deadband	numeric	апертура при сохранении во внутренний архив	
min_value	numeric	нижний предел измерения	
max_value	numeric	верхний предел измерения	
modb_addr_in	bigint	адрес MODBUS во внутреннем обмене данных	
modb_addr_out	bigint	адрес массива при передаче в СВБУ или СВРК	

signal_type	int	0 – binary – бинарный сигнал (0/1); 1 – integer – дискретный сигнал (целое число); 2 – telegram – телеграмма (битовая маска, соответствует используемому в СВБУ типу NERI); 3 – float – аналоговый сигнал (вещественное число).	
stor	int	признак сохранения сигнала во внутренний архив: 0 – не сохранять; 1 – сохранять	
table_id	int	идентификатор таблицы внутреннего архива для сохранения значений сигнала	
mgr_action	int	признак действия при передаче данных (направление передачи данных): 0 – из DTS в MODBUS, 1 – из MODBUS в DTS; 2 – внутренний обмен (т.е. никуда)	используется менеджером, при отсутствии внешнего обмена значение этого поля равно 2, поле modb_addr_out игнорируется
source_subsys	int	индекс подсистемы-источника сигнала	соответствует параметру subsys_id таблицы subsys

convert_rule	int	правило преобразования дискретных и бинарных сигналов: 0 – нет преобразования; 1 и далее – номер правила преобразования	Атрибут convert_rule применяется в старых версиях ППО «DIAPROM PREDICT» для преобразования многопозиционных дискретных сигналов в бинарные (для выполнения требования СВБУ по пересылке только бинарных сигналов). В новых реализациях ППО «DIAPROM PREDICT» подобное преобразование стало частью функционала программного модуля MathManager, который конфигурируется отдельно.
param_RTM	varchar(50)	код оборудования по классификации РТМ 039-86 или другая строковая информация, если применение кода RTM не является	Для СВБУ, имеет смысл только для выходных сигналов системы. Применение в

		обязательным (например, псевдоним)	«DIAPROM PREDICT» и «DIAPROM PREDICT» пока не предполагается (возможно для отбора по группе оборудования с данным префиксом PTM, однако такой отбор возможен и по коду KKS)
HA	numeric	верхняя аварийная граница	
HW	numeric	верхняя предупредительная граница	
HT	numeric	верхняя технологическая (диагностическая) граница	
Имя поля	Тип	Назначение	Примечание
LT	numeric	нижняя технологическая (диагностическая) граница	
LW	numeric	нижняя предупредительная граница	
LA	numeric	нижняя аварийная граница	
display	int	число знаков после запятой	
z_verw	int	уровень сигнализации: 1 – самый высокий; 4 – самый низкий.	для СВБУ
hist	int	признак архивирования в архиве СВБУ:	аналог поля stor, для сигналов, не

		0 – не сохранять; 1 – сохранять.	передаваемых в DTS, не имеет смысла и должен быть равен 0
histdeadband	numeric	апертура при архивировании в архиве СВБУ	аналог поля deadband, для сигналов, не передаваемых в DTS, не имеет смысла и должен быть равен 0
alarm	varchar(5)	принцип срабатывания сигнализации в СВБУ по данному параметру: "X-" – переход в 0; "-X" – переход в 1; "XX" – любое изменение; "--" – нет сигнализации.	устанавливается для бинарных сигналов, передаваемых в СВБУ, для остальных сигналов не имеет смысла и должен быть равен "--"
invert	int	признак инвертирования для СВБУ: 0 – не инвертировать; 1 – инвертировать.	устанавливается для бинарных сигналов, передаваемых в СВБУ, для остальных сигналов не имеет смысла и должен быть равен 0

Таблица А.5 – Таблица описания серверов оперативных данных (communication_servers)

Имя поля	Тип	Назначение	Примечание
id	serial	идентификатор сервера	
ip	text	IP-адрес сервера	
port	integer	Порт, на котором сервер слушает соединение по протоколу CSP	По умолчанию 43561

Таблица А.6 – Таблица описания зависимостей для формул (dependecies)

Имя поля	Тип	Назначение	Примечание
id	serial	идентификатор зависимостей	
dependecies	text	список зависимостей	

Таблица А.7 – Таблица описания устройств работающих по протоколу SNMP (devicesnmpinfo)

Имя поля	Тип	Назначение	Примечание
requestId	bigint	первичный ключ - идентификатор запроса	
requestname	varchar (60)	имя запроса	
description	varchar (300)	описание	
peername	inet	IP-адрес устройства	
objectIdName	varchar (300)	идентификатор объекта в протоколе SNMP (OID)	
signalType	smallint	тип сигнала	

period	numeric	временной интервал запроса	
modbusAddress	bigint	адрес MODBUS	
units	varchar (60)	текстовое отображение единиц измерения	
minRange	numeric	минимально возможное значение	
maxRange	numeric	максимально возможное значение	
lowerCriticalThreshold	numeric	нижняя критическая граница	
upperCriticalThreshold	numeric	верхняя критическая граница	
lowerWarningThreshold	numeric	нижняя предупредительная граница	
upperWarningThreshold	numeric	верхняя предупредительная граница	
reduceCoefficient	numeric	понижающий коэффициент	

Таблица А.8 – Таблица описания диагностических признаков (diagnostic_features)

Имя поля	Тип	Назначение	Примечание
id	serial	уникальный идентификатор	Первичный ключ
external_id	smallint	идентификатор оригинального сигнала	Внешний ключ
feature_type	smallint	тип значения	

Таблица А.9 – Таблица описания диагнозов (diagnostic_knowledge_base)

Имя поля	Тип	Назначение	Примечание
id	serial	первичный ключ – уникальный идентификатор сигнала (параметра)	
name	varchar(300)	имя сигнала полное	
short_name	varchar(100)	имя сигнала краткое	
formula	text	Формула для вычисления значения сигнала	
kks	varchar(300)	код сигнала в системе KKS	зависит от проекта, может быть в другой системе кодирования
param_units_id	int	идентификатор единиц измерения	
dependecies	text	список зависимостей вычисляемого сигнала	
time_interval	int	интервал вычислений	
modb_addr_in	bigint	адрес MODBUS во внутреннем обмене данных	
modb_addr_out	bigint	адрес массива при передаче в СВБУ или СВРК	

signal_type	int	0 – binary – бинарный сигнал (0/1); 1 – integer – дискретный сигнал (целое число); 2 – telegram – телеграмма (битовая маска, соответствует используемому в СВБУ типу NERI); 3 – float – аналоговый сигнал (вещественное число).	
table_id	int	идентификатор таблицы внутреннего архива для сохранения значений сигнала	
mgr_action	smallint	признак действия при передаче данных (направление передачи данных): 0 – из DTS в MODBUS, 1 – из MODBUS в DTS; 2 – внутренний обмен (т.е. никуда)	используется менеджером, при отсутствии внешнего обмена значение этого поля равно 2, поле modb_addr_out игнорируется
source_subsys	int	индекс подсистемы-источника сигнала	соответствует параметру subsys_id таблицы subsys

Таблица А.10 – Таблица описания псевдонимов значений (feature_alias)

Имя поля	Тип	Назначение	Примечание
id	serial	первичный ключ - идентификатор псевдонима	
external_id	smallint	идентификатор оригинального сигнала	
feature_type	smallint		

alias	text	псевдоним	

Таблица А.11 – Таблица истории использования дисковых разделов (fs_status)

Имя поля	Тип	Назначение	Примечание
hostname	text	IP-адрес машины	
device_name	text	имя диска	имеет вид /dev/sda1
mount_point	text	точка монтирования	
Имя поля	Тип	Назначение	Примечание
space_used	int	процент использованного дискового пространства	
datetime	timestamp	дата/время проверки	

Таблица А.12 – Таблица описания мнемосхем (mnemo)

Имя поля	Тип	Назначение	Примечание
mnemo_id	bigint	первичный ключ - идентификатор мнемосхемы	
name	varchar (60)	название мнемосхемы	
pic_source	varchar (500)	имя файла-рисунка мнемосхемы	

Таблица А.13 – Таблица описания индикаторов на мнемосхемах (mnemoleds)

Имя поля	Тип	Назначение	Примечание
led_id	bigint	первичный ключ - идентификатор индикатора	
name	varchar (500)	название индикатора	

led_type	bigint	тип индикатора: для дискретных (binary, telegram, integer), для аналоговых (float)	
signal_id	bigint	идентификатор оригинального сигнала	
mnemo_id	bigint	идентификатор мнемосхемы	
cx	numeric	координата X индикатора на мнемосхеме	
cy	numeric	координата Y индикатора на мнемосхеме	
sig_type	bigint	тип привязанного сигнала (binary, telegram, integer, float)	

Таблица А.14 – Таблица описания объектов, сведения о которых передаются в телеграммах (objects)

Имя поля	Тип	Назначение	Примечание
object_id	smallint	первичный ключ - идентификатор объекта	
KKS	text	префикс ККС, идентифицирующий объект	
object_name	text	название объекта	
Имя поля	Тип	Назначение	Примечание
object_type	text	тип объекта	
object_char	text	дополнительная информация об объекте	
drive_type	text	тип оборудования	
object_where	text	место установки	

object_docs	text	документы	
system_id	smallint	идентификатор системы, к которой принадлежит объект	

Таблица А.15 – Таблица соответствия объектов, сведения о которых передаются в телеграммах, сигналам (param_object)

Имя поля	Тип	Назначение	Примечание
param_id	smallint	первичный ключ - идентификатор сигнала	
KKS	text	ККС	
object_id	smallint	идентификатор объекта	

Таблица А.16 – Таблица описания признаков достоверности (quality)

Имя поля	Тип	Назначение	Примечание
id	smallint	первичный ключ - идентификатор признака достоверности	
name	varchar (100)	название признака достоверности	

Таблица А.17 – Таблицы описания сигналов (sakt, cla_sakt_output, import_sakt, predict_sakt, system_statistics_values)

Имя поля	Тип	Назначение	Примечание
param_id	serial	первичный ключ – уникальный идентификатор сигнала (параметра)	
param_name	varchar(300)	имя сигнала полное	
param_short_name	varchar(100)	имя сигнала краткое	

param_kks	varchar(300)	код сигнала в системе KKS	зависит от проекта, может быть в другой системе кодирования
Имя поля	Тип	Назначение	Примечание
param_units	varchar(50)	единицы измерения	
param_units_id	int	идентификатор единиц измерения	
deadband	numeric	апертура при сохранении во внутренний архив	
min_value	numeric	нижний предел измерения	
max_value	numeric	верхний предел измерения	
modb_addr_in	bigint	адрес MODBUS во внутреннем обмене данных	
modb_addr_out	bigint	адрес массива при передаче в СВБУ или СВРК	
signal_type	int	0 – binary – бинарный сигнал (0/1); 1 – integer – дискретный сигнал (целое число); 2 – telegram – телеграмма (битовая маска, соответствует используемому в СВБУ типу NERI); 3 – float – аналоговый сигнал (вещественное число).	
stor	int	признак сохранения сигнала во внутренний архив: 0 – не сохранять;	

		1 – сохранять	
table_id	int	идентификатор таблицы внутреннего архива для сохранения значений сигнала	
mgr_action	int	признак действия при передаче данных (направление передачи данных): 0 – из DTS в MODBUS, 1 – из MODBUS в DTS; 2 – внутренний обмен (т.е. никуда)	используется менеджером, при отсутствии внешнего обмена значение этого поля равно 2, поле modb_addr_out игнорируется
source_subsys	int	индекс подсистемы-источника сигнала	соответствует параметру subsys_id таблицы subsys

convert_rule	int	правило преобразования дискретных и бинарных сигналов: 0 – нет преобразования; 1 и далее – номер правила преобразования	Атрибут convert_rule применяется в старых версиях ППО «DIAPROM PREDICT» для преобразования многопозиционных дискретных сигналов в бинарные (для выполнения требования СВБУ по пересылке только бинарных сигналов). В новых реализациях ППО «DIAPROM PREDICT» подобное преобразование стало частью функционала программного модуля MathManager, который конфигурируется отдельно.

param_RTM	varchar(50)	код оборудования по классификации РТМ 039-86 или другая строковая информация, если применение кода RTM не является обязательным (например, псевдоним)	Для СВБУ, имеет смысл только для выходных сигналов системы. Применение в «DIAPROM PREDICT» и «DIAPROM PREDICT» пока не предполагается (возможно для отбора по группе оборудования с данным префиксом РТМ, однако такой отбор возможен и по коду KKS)
HA	numeric	верхняя аварийная граница	
HW	numeric	верхняя предупредительная граница	
HT	numeric	верхняя технологическая (диагностическая) граница	
LT	numeric	нижняя технологическая (диагностическая) граница	
LW	numeric	нижняя предупредительная граница	
LA	numeric	нижняя аварийная граница	
Имя поля	Тип	Назначение	Примечание
display	int	число знаков после запятой	
z_verw	int	уровень сигнализации: 1 – самый высокий; 4 – самый низкий.	для СВБУ

hist	int	признак архивирования в архиве СВБУ: 0 – не сохранять; 1 – сохранять.	аналог поля stor, для сигналов, не передаваемых в DTS, не имеет смысла и должен быть равен 0
histdeadband	numeric	апертура при архивировании в архиве СВБУ	аналог поля deadband, для сигналов, не передаваемых в DTS, не имеет смысла и должен быть равен 0
alarm	varchar(5)	принцип срабатывания сигнализации в СВБУ по данному параметру: "X-" – переход в 0; "-X" – переход в 1; "XX" – любое изменение; "--" – нет сигнализации.	устанавливается для бинарных сигналов, передаваемых в СВБУ, для остальных сигналов не имеет смысла и должен быть равен "--"
invert	int	признак инвертирования для СВБУ: 0 – не инвертировать; 1 – инвертировать.	устанавливается для бинарных сигналов, передаваемых в СВБУ, для остальных сигналов не имеет смысла и должен быть равен 0

Таблица А.18 – Таблица неисправностей датчиков (sensor_malfunction_log)

Имя поля	Тип	Назначение	Примечание
id	serial	первичный ключ - идентификатор неисправности	
date_time	timestamp	время обнаружения	

sensors_id	int[]	список идентификаторов неисправных датчиков	
formulas_id	int[]	список формул, которые вызвали ошибку	
operation_type	int	тип уведомления о неполадке	

Таблица А.19 – Таблица описания типов сигналов (signal_types)

Имя поля	Тип	Назначение	Примечание
id	smallint	идентификатор типа сигналов	
name	varchar(30)	Наименование типа сигналов	
suffix	varchar(30)	Суффикс имени таблицы	

Таблица А.20 – Таблица описания подсистем (subsys)

Имя поля	Тип	Назначение	Примечание
subsys_id	bigint	идентификатор подсистемы	
alias	varchar (60)	псевдоним подсистемы	
name	varchar (60)	рабочее название подсистемы	

Таблица А.21 – Таблица идентификаторов ККС систем (systems)

Имя поля	Тип	Назначение	Примечание
id_system	smallint	идентификатор системы	
KKS	text	фрагмент ККС идентифицирующий систему	

descr	text	описание системы	

Таблица А.22 – Таблица направлений передачи данных (transfer_direction)

Имя поля	Тип	Назначение	Примечание
id	smallint	идентификатор направления передачи данных	
description	text	описание направления передачи данных	

Таблица А.23 – Таблица значений битов телеграмм (unit_bits)3

Имя поля	Тип	Назначение	Примечание
units_id	smallint	идентификатор	
n_bits	smallint	количество битов	
bits	text	строка номеров битов, разделитель - “;”	
comment	text	текстовый комментарий	

Таблица А.24 – Таблица единиц измерения (units)

Имя поля	Тип	Назначение	Примечание
unit_id	integer	первичный ключ - идентификатор единицы измерения	
units_text	varchar (150)	текстовое обозначение единицы измерения	
signal_type	integer	тип сигнала	
text1	varchar (50)	текст, соответствующий значению 1	
text2	varchar (50)	текст, соответствующий значению 2	
text3	varchar (50)	текст, соответствующий значению 3	
text4	varchar (50)	текст, соответствующий значению 4	
value1	integer	значение 1 (для типа сигнала целочисленный)	

value2	integer	значение 2 (для типа сигнала целочисленный)	
value3	integer	значение 3 (для типа сигнала целочисленный)	
value4	integer	значение 4 (для типа сигнала целочисленный)	
comment	varchar (1000)	комментарий	
SVRK_id	integer	идентификатор единицы измерения в СВРК	

Таблица А.25 – Таблица пользователей (users)

Имя поля	Тип	Назначение	Примечание
user_id	serial	идентификатор пользователя	
user_name	varchar (50)	имя пользователя	
group_id	integer	идентификатор группы пользователя	
Имя поля	Тип	Назначение	Примечание
password	varchar	зашифрованный пароль пользователя	
salt	varchar	соль, используемая при шифровании пароля	

Таблица А.26 – Таблица групп пользователей (groups)

Имя поля	Тип	Назначение	Примечание
group_id	serial	идентификатор группы пользователей	
group_name	varchar (50)	название группы пользователей	
permissions	bigint	права доступа пользователей - битовая маска	
isadmin	boolean	имеются ли у группы административные права (полный доступ)	

Таблица А.27 – Таблица прав доступа виджетов (widget_permissions)

Имя поля	Тип	Назначение	Примечание

id	serial	идентификатор виджета	
name	varchar (50)	название виджета	
permissions	bigint	права виджета - битовая маска	