Функциональные характеристики модуля «Подсистема сбора данных» (версия Postgre SQL)

Введение.

Подсистема сбора данных АС «ТЕКОН-Диспетчеризация» (версия PostgreSQL) позволяет принимать результаты измерений физических величин технологических параметров, произведенных на объектах диспетчеризации (оборудованием, установленном непосредственно на объекте) и записывать эти результаты в структуры БД Системы для долговременного хранения в не модифицированном виде и дальнейшей обработки. Подсистема сбора данных АС «ТЕКОН-Диспетчеризация» (версия PostgreSQL) позволяет осуществлять информационный обмен с объектовым оборудованием в соответствии с требованиями открытого международного стандарта в области промышленной автоматики и диспетчеризации – OPC (OLE for process control) и в соответствии с протоколом PushEvent, применяемым в контроллерах производства группы компаний ТЕКОН.

Архитектура программного комплекса подсистемы сбора данных результатов измерений, полученных от контроллеров MFK1500

Программный комплекс реализован как клиент-серверное приложение. В качестве клиента выступает прикладное программное обеспечение «Сервер MFK1500». Клиент с одной стороны осуществляет информационное взаимодействие с контроллерами MFK1500, установленными на объектах диспетчеризации, а с другой, осуществляет передачу данных в агрегатор серверов MFK1500 под управлением Java EE для распределения её по структурам БД.

Сервер MFK1500 состоит из двух основных блоков:

- блок работы с контроллерами MFK1500;
- технологический блок сервера MFK1500.

Блок работы с контроллерами MFK1500 предназначен для непосредственного информационного взаимодействия с контроллерами MFK1500 и состоит из следующих модулей:

- Модуль мониторинга трафика. Модуль контролирует объемы использованного трафика информационного обмена с контроллерами и ограничивает передачу данных в случае превышения порогового значения.
- Модуль протокола PushEvent. Модуль обеспечивает получение данных часовых архивов от контроллера MFK1500 (контроллеры MFK1500 передают часовые архивы по протоколу PushEvent).
- Модуль протокола isacom. Модуль предназначен для получения мгновенных результатов измерений и ряда служебной информации с контроллера MFK1500 передаваемой по протоколу isacom.
- Модуль прямого доступа к контроллеру по ssh. Сервисный модуль позволяет подключиться к контроллеру MFK1500 по протоколу ssh для поднятия конфигураций, установленных в контроллере, считывания меток времени и синхронизации времени на контроллере с общесистемным.
- Модуль DA параметров. Модуль предназначен для получения и фильтрации результатов мгновенных измерений от контроллера MFK1500.
- Модуль параметров контроллера. С помощью данного модуля возможно получить значения системных параметров контроллера.
- Модуль конфигурации контроллера. Модуль получает от контроллера список мгновенных и архивных переменных.

Технологический блок сервера MFK1500 предназначен для решения сервисных функций сервера. Он состоит из следующих компонентов:

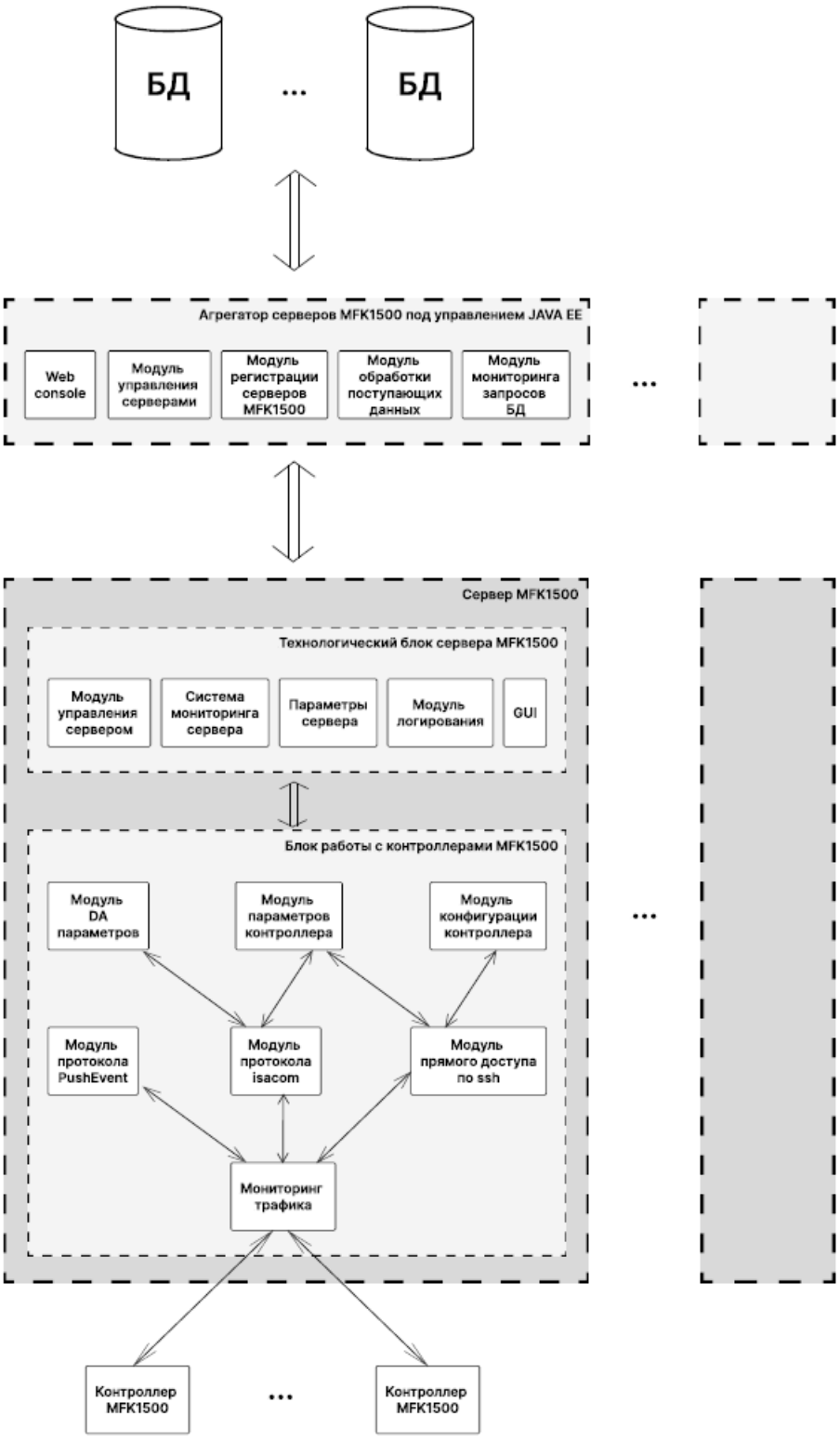
- Модуль управления сервером. Модуль обрабатывает пришедшие команды управления.
- Система мониторинга сервера. Система предназначена для мониторинга состояния сервера и решения возникших проблем.
- Блок хранения всех настроечных параметров сервера.
- Модуль логирования. Модуль протоколирует все происходящие ситуации на сервере для дальнейшего анализа в случае возникновения непредвиденной ситуации.
- GUI. Модуль расширения для возможности визуального наблюдения за работой сервера.

Данные от серверов MFK1500 передаются в агрегатор серверов MFK1500 под управлением Java EE. Он состоит из следующих компонентов:

- Web console. Консоль для просмотра состояния подключенных серверов и контроллеров MFK1500 (пользовательский интерфейс).
- Модуль управления серверами. Модуль передает управляющие команды серверам MFK1500.

- Модуль регистрации серверов MFK1500. Данный модуль хранит всю техническую информацию о подключенных серверах MFK1500.
- Модуль обработки поступающих данных. Данный модуль получает, обрабатывает и передает информацию в структуры БД АС «ТЕКОН-Диспетчеризация» (версия PostgreSQL).
- Модуль мониторинга запросов БД. Данный модуль следит за поступающими запросами от БД АС «ТЕКОН-Диспетчеризация» (версия PostgreSQL) на получения результатов мгновенных измерений и получения конфигураций контроллеров MFK1500.

Данный программный комплекс реализован с возможностью горизонтального расширения. В случае увеличения количества контроллеров добавляется новый сервер MFK1500, а в случае большого количества серверов MFK1500 есть возможность добавлять агрегаторы серверов MFK1500.



Архитектура программного комплекса подсистемы сбора данных результатов измерений, полученных от контроллеров MFK1500.

1. Описание графического интерфейса

Графический интерфейс пользователя (GUI) подсистемы сбора данных результатов измерений, полученных от контроллеров MFK1500 производства группы компаний «ТЕКОН» предназначен для подключения новых объектов, управления процессов информационного взаимодействия, просмотра статистики и для работы с информацией о поступающих в АС «ТЕКОН-Диспетчеризацию» данных приборов автоматики, расположенных непосредственно на объектах диспетчеризации.

Пользовательский интерфейс позволяет:

- Подключить новый объект к подсистеме сбора данных и управлять опросами подключенных объектов;
- наблюдать за процессом передачей данных объектами, непосредственно от приборов автоматики до АС «ТЕКОН-Диспетчеризации»;
- Осуществлять контроль информационного трафика для каждого объекта в рамках выделенного ежесуточного и ежемесячного лимитов;
- Осуществлять синхронизацию локального времени приборов на объектах с общесистемным временем;
- Просматривать архивы передачи групп данных и служебной информации с определённой периодичностью.

Внешний вид пользовательского интерфейса представлен на рисунке ниже.

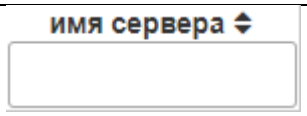
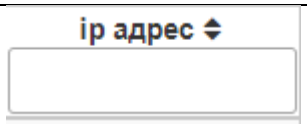
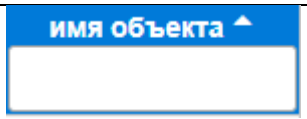
№	имя сервера	ip адрес	имя объекта	сокет	статус	сеанс связи	трафик				параметры
							входящий	исходящий	общий суточный	общий месячный	
1	MFK1500-MK	10.99.4.59		1	не слинковано	07.11.2022 00:05:44	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	14.0 KiB (126 B) из 90.0 MiB	
2	MFK1500-1	10.99.20.173		1	не слинковано	07.11.2022 00:06:18	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	27.0 KiB (519 B) из 90.0 MiB	
3	MFK1500-1	10.99.20.146		1	не слинковано	07.11.2022 00:06:21	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	26.0 KiB (519 B) из 90.0 MiB	
4	MFK1500-1	10.99.20.137		1	не слинковано	07.11.2022 00:06:25	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	14.0 KiB (126 B) из 90.0 MiB	
5	MFK1500-1	10.99.21.104		1	не слинковано	07.11.2022 00:08:37	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	14.0 KiB (126 B) из 90.0 MiB	
6	MFK1500-3	10.99.3.117		4	не слинковано	01.11.2022 15:24:34	4.0 KiB (48 B)	4.0 KiB (24 B)	8.0 KiB (72 B) из 3.0 MiB	8.0 KiB (72 B) из 90.0 MiB	
7	MFK1500-3	10.99.3.16		4	не слинковано	01.11.2022 15:23:45	4.0 KiB (48 B)	4.0 KiB (24 B)	8.0 KiB (72 B) из 3.0 MiB	8.0 KiB (72 B) из 90.0 MiB	
8	MFK1500-2	10.99.23.13		1	не слинковано	06.11.2022 00:05:18	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
9	MFK1500-2	10.99.23.14		1	не слинковано	06.11.2022 00:06:04	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
10	MFK1500-2	10.99.23.15		1	не слинковано	06.11.2022 00:06:18	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
11	MFK1500-2	10.99.23.16		1	не слинковано	06.11.2022 07:32:27	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
12	MFK1500-2	10.99.23.11		1	не слинковано	06.11.2022 00:05:28	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
13	MFK1500-2	10.99.23.12		1	не слинковано	07.11.2022 10:37:53	9.0 KiB (234 B)	4.0 KiB (213 B)	13.0 KiB (447 B) из 3.0 MiB	23.0 KiB (537 B) из 90.0 MiB	
14	MFK1500-2	10.99.22.109		1	не слинковано	06.11.2022 00:07:10	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
15	MFK1500-2	10.99.22.100		1	не слинковано	06.11.2022 00:08:42	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
16	MFK1500-2	10.99.23.22		1	не слинковано	06.11.2022 00:06:50	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	10.0 KiB (90 B) из 90.0 MiB	
17	MFK1500-2	10.99.4.93		1	не слинковано	06.11.2022 00:08:03	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
18	MFK1500-2	10.99.22.197		1	не слинковано	06.11.2022 00:10:03	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
19	MFK1500-2	10.99.22.181		1	не слинковано	06.11.2022 00:09:41	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
20	MFK1500-2	10.99.22.180		1	не слинковано	06.11.2022 00:10:48	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
21	MFK1500-2	10.99.22.182		1	не слинковано	06.11.2022 00:09:55	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	
22	MFK1500-2	10.99.21.137		1	не слинковано	06.11.2022 00:11:11	1.0 KiB (12 B)	1.0 KiB (6 B)	2.0 KiB (18 B) из 3.0 MiB	12.0 KiB (108 B) из 90.0 MiB	

Всего объектов: 1307

Выгрузка в Excel

1.1. Основное окно пользовательского интерфейса

Таблица основного окна содержит следующие информационные колонки:

Колонка	Определение
№	Порядковый номер.
имя сервера 	Имя сервера, зарегистрированного в подсистеме сбора данных. В настоящий момент доступны 4 сервера: MFK1500-1, MFK1500-2, MFK1500-3, MFK1500-MK. Колонка оснащена фильтром, с помощью которого можно осуществлять поиск сервера по имени. Результат отображается по мере ввода. *По данной колонке предусмотрена возможность осуществлять в таблице сортировку* ***по имени сервера***.
ip адрес 	IP-адрес объекта, осуществляющего информационный обмен. Колонка оснащена фильтром, с помощью которого можно осуществлять поиск IP-адреса по номеру. Результат отображается по мере ввода. *По данной колонке предусмотрена возможность осуществлять в таблице сортировку* ***по номеру ip-адреса объекта***.
имя объекта 	Имя объекта. ***Отображается в таблице*** при успешной линковке объекта в системе АС «ТЕКОН-Диспетчеризация» (версия PostgreSQL). ***Не отображается в таблице*** при статусе объекта «Не слинковано». За исключением случая, в котором при первичном присвоении объекту статуса «Свободно», далее изменилась конфигурация прибора автоматики. Колонка оснащена фильтром, с помощью которого можно осуществлять поиск объекта по имени. Результат отображается по мере ввода. *По данной колонке предусмотрена возможность осуществлять в таблице сортировку* ***по имени объекта***.

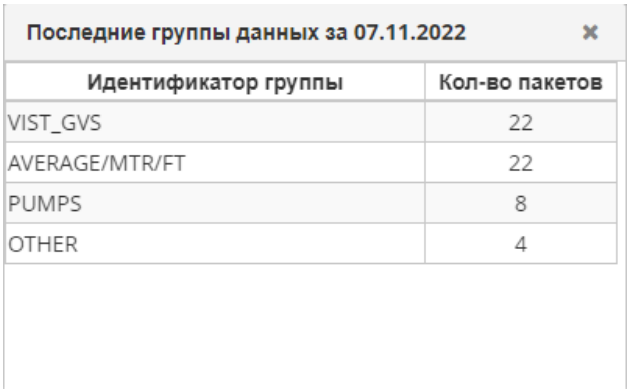
Колонка		Определение
сокет ⇕		Количество открытых соединений с объектом. *По данной колонке предусмотрена возможность осуществлять в таблице сортировку* ***по количеству соединений****.*
статус ⇕		Статус объекта. Доступны 4 статуса: **Свободно** – успешное взаимодействие объекта и Системы: объект слинкован, его данные передаются в Систему и распознаются ей. **Превышение трафика** – превышен общий суточный трафик (общий месячный трафик). Запросы о передаче данных объекта в Систему автоматически остановлены до истечения суток. **Ошибка сервера** – набор данных, посланных объектом, не распознан в Системе. **Не слинковано** – объект не зарегистрирован в Системе. *По данной колонке предусмотрена возможность осуществлять в таблице сортировку* ***по статусу объекта****.*
сеанс связи		Дата и точное время ***последнего*** информационного обмена с объектом.
трафик	входящий	Количество трафика, потраченного объектом для отправки данных.
	исходящий	Количество трафика, потраченное Системой для отправки ответа объекту (напр., подтверждение о получении данных).
	общий суточный	Суммарное количество трафика (входящее, исходящее), указанное в рамках общего лимита трафика на сутки.
	общий месячный	Суммарное количество трафика за прошедшее количество дней месяца, указанное в рамках общего лимита трафика в месяц.
параметры		Кнопка, осуществляющая переход в окно управления системными параметрами объекта.

В нижней части таблицы отображено поле «Всего объектов», которое содержит общее количество объектов, подключенных к Системе.

Кнопка 

осуществляет выгрузку данной таблицы на ПК пользователя в формате электронных таблиц.

При нажатии правой кнопкой мыши на запись об объекте отображается выпадающий список, с помощью функций которого можно осуществлять следующие действия над объектом:

Функция	Определение
Блокировать	Прекратить информационный обмен с объектом.
Разблокировать	Возобновить информационный обмен с объектом.
Снять ограничение трафика	Снять ограничения с общего трафика на один день.
Вернуть ограничение трафика	Возобновить ограничения общего трафика.
Последние данные	Предоставление информации по количеству переданных объектом пакетов последних данных в Систему. 
Переподписать объект	При изменении конфигурации и статуса объекта, с заданной функцией доступна возможность перепроверить и обновить информацию об объекте.
Удалить	Удалить данные об объекте, в том числе архивные. При повторном возобновлении связи объектом, он также будет отображаться в интерфейсе драйвера.

1.2. Окно управления системными параметрами объекта

При нажатии в столбце «Параметры» кнопки появляется окно управления системными параметрами объекта.

Данное окно представлено на рисунке ниже.

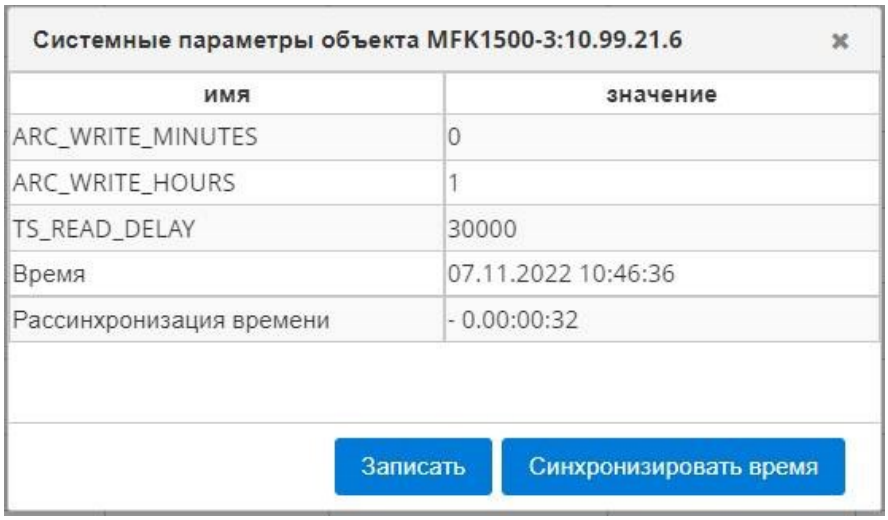


Таблица окна управления системными параметрами объекта содержит следующие колонки и кнопки:

Колонка	Определение
ARC_WRITE_MINUTES	Время, за которое один раз в указанное количество ***минут*** контроллером агрегируются данные с первичных измерителей.
ARC_WRITE_HOURS	Время, за которое один раз в указанное количество ***часов*** контроллером агрегируются данные с первичных измерителей.
TS_READ_DELAY	Время ожидания ответа от подключенного к контроллеру теплосчетчика. Время указано ***в миллисекундах***.
Время	Время прибора на объекте.
Рассинхронизация времени	Несоответствие времени прибора на объекте и времени Системы.

Кнопка	Определение
[Записать]	Сохранить измененные настройки параметров ARC_WRITE_MINUTES, ARC_WRITE_HOURS, TS_READ_DELAY.
[Синхронизировать время]	Уточнить время прибора на объекте до времени Системы.

Приложение №1
Описание протокол Push Event

1. Архитектура и принципы работы Push Event.

1.1. Общее описание.

В своей работе push events активно использует возможности системы архивов TeNIX, такие как подписка на новые события и чтение из большого циклического буфера в RAM. Текущая реализация использует три архива TeNIX:

- Системный архив (sys) - сообщения диагностики TeNIX и т.п., доступны через TUNER. Частично сохраняется при перезагрузке.
- Архив трендов (ta_trends) - сообщения, генерируемые сервисом трендов (входит в состав ISaGRAF) и присвоения метки времени вне цикла ISaGRAF (iowatch). Частично сохраняется при перезагрузке.
- Архив трендов в RAM (ta_ram) - аналогично архиву трендов, но более быстрый, так как не сохраняется при перезагрузке.

Используя возможности “подписки” на архивы TeNIX, сервис push_events постоянно ожидает появления новых данных по всем архивам, и в случае появления устанавливает инициативное соединение с указанным удаленным сервером (если оно еще не установлено) и передает ему данные. После этого сервис push_events ожидает подтверждения от удаленного сервера в течение определенного таймаута. Если подтверждения нет, соединение разрывается и делается повторная попытка соединения. Всего для удаленного сервера может быть задано до 4 IP-адресов, соединение устанавливается с первым ответившим IP.

Для исключения посылки старых данных используется монотонно возрастающий индекс с переполнением при достижении значения 2^32, который предоставляет система архивов TeNIX. Когда удаленный сервер подтверждает прием событий, индекс последнего переданного события запоминается в специальной ГП (last confirmed var), которая сохраняется при перезагрузке. При потере связи или перезагрузке контроллера передача данных архивов начнется с номера, следующего за значением, сохраненным в last confirmed var.

Push events также может диагностировать потери событий, связанные с затиранием данных в циклическом буфере при длительном отсутствии связи с

удаленным сервером. Для этого анализируется разница между индексом самого старого события в архиве и индексом последнего переданного события.

1.2. Логика работы

1. После запуска сервис подписывается на все заданные архивы.
2. Ожидается появление событий на всех архивах.
3. Если текущего соединения еще нет, оно устанавливается со всеми указанными IP, и первый корректно ответивший IP становится главным, а остальные соединения закрываются. Если соединение не удалось, делается повторная попытка, пока соединение не удастся.
4. Собираются события из архивов и формируется посылка.
5. Посылка отсылается удаленной стороне и ожидается подтверждение. Если подтверждение пришло, записывается индекс последнего переданного события для каждого архива, и происходит переход к шагу 2.
6. Если подтверждение не пришло в течение определенного времени - главное соединение закрывается, и происходит переход к шагу 3.

2. Протокол инициативной передачи данных push events.

2.1. Общая информация для всех версий протоколов

2.1.1. **Описание протокола**

Протокол push events реализован как прикладной уровень поверх TCP/IP. Протокол является бинарным, с переменной длиной посылок. Длина посылки передается в первых двух байтах посылки, и указывается без учета этих первых двух байт. Служебные поля в посылках используют порядок байт Motorola (Big Endian), порядок байт внутри посылок зависит от платформы контроллера.

Инициатором соединения выступает контроллер, он подключается к удаленному серверу, используя TCP порт 20100. Сразу после установления соединения контроллер посылает посылку идентификации, информирующую удаленный сервер о версии протокола, порядке байт, платформе контроллера, и ожидает такую же информацию от удаленного сервера, что позволяет сторонам согласовать версию протокола. До того, как идентификация завершена, никаких других сообщений не отсылается.

2.1.2. **Идентификация**

Посылается контроллером инициативно, сразу после установления соединения.

Таблица 1 Формат посылки идентификации

Номер байта	**Описание**
0-1	Длина посылки (Big Endian)
2	Значение 01 - пакет идентификации
3	Максимально поддерживаемая клиентом версия протокола (биты 4-7 - старшая версия, бит 3 равен 0, биты 0-2 - младшая версия)
4	Порядок байт (0 - Little Endian, 1 - Big Endian)
5	Номер контроллера в системе
6 и до конца посылки	Идентификатор модели контроллера в текстовом виде (P06, MFC, MFC3000, TKM410 etc)

В ответ на посылку идентификации сервер присылает либо подтверждение идентификации, либо отказ в идентификации.

2.1.3. **Подтверждение идентификации**

Посылается удаленным сервером в ответ на посылку идентификации.

Таблица 2 Формат подтверждения идентификации

Номер байта	**Описание**
0-1	Длина посылки (Big Endian)
2	Значение 02 - пакет подтверждения идентификации
3	Максимально поддерживаемая сервером версия протокола (биты 4-7 - старшая версия, бит 3 равен 1, биты 0-2 - младшая версия)
4	Порядок байт (0 - Little Endian, 1 - Big Endian)
5	Номер сервера в системе
6 и до конца посылки	Идентификатор модели сервера в текстовом виде (обычно PC)

В случае получения подтверждения идентификации удаленным сервером контроллер переходит в режим, когда он может посылать инициативные данные.

Следует обратить внимание, что в версии протокола, возвращаемой сервером, бит 3 должен быть установлен в 1. Если клиент обнаруживает, что

этот бит в посылке равен 0, то он должен считать, что максимально поддерживаемая сервером версия протокола равна 1.0.

2.1.4. **Отказ в идентификации**

Посылается удаленным сервером в ответ на посылку идентификации при возникновении ошибки при идентификации (например, контроллер с переданным номером не зарегистрирован в системе).

Таблица 3 Формат отказа в идентификации

Номер байта	**Описание**
0-1	Длина посылки (Big Endian)
2	Значение 03 - пакет отказа в идентификации
3	Максимально поддерживаемая сервером версия протокола (биты 4-7 - старшая версия, бит 3 равен 1, биты 0-2 - младшая версия)
4	Порядок байт (0 - Little Endian, 1 - Big Endian)
5	Номер сервера в системе

В случае получения отказа в идентификации контроллер должен закрыть соединение с удаленным сервером, и не открывать его более (до перезагрузки). Также он должен записать факт отказа в системный журнал. Контроллер может интерпретировать отказ идентификации как отказ или ошибку контроллера для системы диагностики.

2.2. Версии протоколов

В настоящий момент существует 2 версии протокола передачи инициативных сообщений. Версию 1.0 обязаны поддерживать любые реализации сервиса push_events и удаленного сервера. Версию 2.0 следует использовать только если обе стороны в пакетах идентификации и подтверждения идентификации в байте версии передали 0x20. В противном случае используется версия 1.0.

2.2.1. **Описание версии протокола 1.0**

Базовая версия протокола была выпущена в сентябре 2008 года. Она предусматривает размер посылки максимум в 65535 байт, и количество сообщений в одной посылке до 255. Реализации, использующие этот протокол, должны передавать в номере версии число 0x10.

2.2.2. **Описание версии протокола 2.0**

Расширенная версия протокола была выпущена в январе 2014 года. В отличие от версии 1.0 в ней расширено количество сообщений в одной посылке (до 65535) при сохранении общей длины посылки не более 65535 байт. Также отличием является возможность передачи удаленным сервером индекса последнего подтвержденного сообщения для каждого архива после идентификации, а также включение индексов сообщений в инициативную посылку. Причины введения новой версии в устранении потери сообщений при перезагрузке удаленного сервера, а также для оптимизации передачи большого количества инициативных данных.

2.3. Посылки протокола версии 1.0

2.3.1. **Посылка событий**

Посылается контроллером инициативно при появлении новых событий. Состоит из фиксированной части - заголовка, и переменной части - списка событий.

Таблица 4 Заголовок посылки событий (версия 1.0)

Номер байта	Описание
0-1	Длина посылки (Big Endian)
2	Значение 03 - инициативные сообщения
3	Количество инициативных событий в посылке
4 и до конца посылки	Список событий, количество задано в байте 3

После заголовка идет столько инициативных сообщений, сколько указано в заголовке. Сообщения идут без каких-либо промежуточных данных, после окончания первого сообщения сразу следует второе и т.д.

Каждое инициативное сообщение, в свою очередь, тоже содержит фиксированный заголовок и переменную часть - дополнительные данные.

Таблица 5 Заголовок сообщения (версия 1.0)

Номер байта	Описание
0-3	Номер байта Описание Количество полных секунд, прошедших с 00:00:00 1 января 1970 года по UTC до момента события
4-7	Количество наносекунд, прошедших с начала текущей секунды до момент возникновения события

8	Уникальный номер буфера
9-12	Код события
13	Количество дополнительных данных (0 если их нет)
14	Дополнительные данные (кол-во задано в байте 13)

К каждому событию могут быть прикреплены дополнительные данные - значения переменных, номера каналов и т.д. Протоколом никак не регламентируется их количество, что означают каждые данные и т.п. Тем не менее, в протоколе указывается тип каждых дополнительных данных, если он известен контроллеру. Если не известен - используется тип UNKNOWN.

Таблица 6 Поддерживаемые типы дополнительных данных (версия 1.0)

Номер	**Тип**	**Размер**	**Примечание**
255	BOOL	1 байт	логическое значение (0 - false, 1 - true)
254	BYTE	1 байт	без знаковое 8-битное целое
253	INT	4 байта	знаковое 32-битное целое
252	UINT	4 байта	без знаковое 32-битное целое
251	QUAD	8 байт	знаковое 64-битное целое
250	FLOAT	4 байта	число с плавающей точкой IEEE 754
249	SYM	1 байт	символ (элемент текстовой строки)
64	UNKNOWN	1 байт	1 байт тип данных не представим в этой версии протокола, передается в виде набора байт

В будущих версиях возможно расширение протокола и добавление новых типов. В случае если шлюз не поддерживает эту версию протокола, тип данных должен быть указан как UNKNOWN.

Таблица 7 Формат дополнительных данных (версия 1.0)

Номер байта	**Описание**
0	Тип данных, как указано в6↑
1	Количество элементов этого типа
2	Собственно данные (длина равна размеру типа, умноженному на количество элементов)

Для типа данных BOOL длина округляется до ближайшего большего или равного числа, кратного 8. Первый элемент типа BOOL соответствует байту 0 биту 0, восьмой - соответствует байту 1 биту 0 и т.д. То есть используется "упаковка" битовых данных.

Получив инициативные сообщения, удаленный сервер должен подтвердить их получение, отослав ответ специального вида. Пока контроллер не получил подтверждение, он не должен посылать новые инициативные

сообщения. Если связь по каким-то причинам разорвалась, то после восстановления связи контроллер должен повторить посылку инициативного сообщения еще раз. Контроллер может ожидать подтверждения получения в течение 60 секунд, после чего считать посылку недоставленной и попытаться доставить ее еще раз. При этом контроллеру разрешается разорвать соединение и пере подключиться.

2.3.2. **Подтверждение получения событий**

Высылается удаленным сервером в ответ на посылку с событиями.

Таблица 8 Посылка подтверждения получения событий (версия 1.0)

Номер байта	Описание
0-1	Длина посылки (Big Endian), должна быть равна “00 02”
2	Байт со значением 04 (подтверждение получения посылки)
3	Количество полученных инициативных сообщений в посылке

Количество полученных инициативных сообщений должно совпадать с количеством отосланных сообщений, иначе посылка считается неполученной и посылается еще раз. Контроллер может разорвать соединение, если несовпадения количества происходят более 3 раз подряд.

При получении корректного подтверждения посылки удаленным сервером контроллер может послать следующие инициативные сообщения, если они присутствуют в очереди.

2.4. Посылки протокола версии 2.0

2.4.1. **Запрос метки последней принятой инициативной посылки**

Посылается контроллером сразу после идентификации, если обе стороны поддерживают протокол версии 2.0 и выше. Метка посылки позволит контроллеру продолжить передачу сообщений с той точки, на которой она была прервана перезагрузкой сервера.

Таблица 9 Посылка запроса метки последней принятой посылки (версия 2.0)

Номер байта	Описание
0-1	Длина посылки (Big Endian), должна быть равна “00 01”

2	Байт со значением 10 (0x0A, запрос метки)

2.4.2. **Передача метки последней принятой инициативной посылки**

Посылается удаленным сервером в ответ на запрос метки последней посылки. В случае если у сервера нет данных о метке последней принятой инициативной посылки, он может указать пустую метку (длина = 0). В этом случае контроллер действует по тому же принципу, как и в протоколе версии 1.0, используя метку, сохраненную на стороне контроллера.

Таблица 10 Передача метки последней принятой посылки (версия 2.0)

Номер байта	**Описание**
0-1	Длина посылки (Big Endian)
2	Байт со значеним 11 (0x0B, передача метки)
3	Длина метки в байтах (0 если сохраненной метки нет)
4 и до конца посылки	Метка, в точности как было передано контроллером

2.4.3. **Посылка событий**

Посылается контроллером инициативно при появлении новых событий. Состоит из фиксированной части - заголовка, и переменной части - метки посылки и списка событий.

Таблица 11 Заголовок посылки событий (версия 2.0)

Номер байта	**Описание**
0-1	Длина посылки (Big Endian)
2	Значение 03 - инициативные сообщения
3-4	Количество инициативных событий в посылке (Big Endian)
5	Длина метки посылки
6 - (5 + длина метки)	Метка посылки (длина этого поля равна значению поля 5 и не превышает 255 байт)
(6 + длина метки) и до окончания посылки	Список событий, количество задано в байтах 3-4

После заголовка идет столько инициативных сообщений, сколько указано в заголовке. Сообщения идут без каких-либо промежуточных данных, после окончания первого сообщения сразу следует второе и т.д.

Каждое инициативное сообщение, в свою очередь, тоже содержит фиксированный заголовок и переменную часть - дополнительные данные.

Таблица 12 Заголовок сообщения (версия 2.0)

Номер байта	Описание
0-3	Количество полных секунд, прошедших с 00:00:00 1 января 1970 года по UTC до момента события
4-7	Описание Количество наносекунд, прошедших с начала текущей секунды до момент возникновения события
8	Статус контроллера на момент возникновения события (0 - неизвестно, 1 - мастер, 2 - не мастер)
9	Уникальный номер буфера
10-13	Код события
14	Количество дополнительных данных (0 если их нет)
15	Дополнительные данные (кол-во задано в байте 14)

Формат дополнительных данных такой же, как и в версии 1.0.

При получении посылки событий удаленный сервер может сохранить переданную метку в общем хранилище, чтобы передать ее после разрыва соединения для возобновления передачи событий с места разрыва (например, в случае внезапной перезагрузки сервера). Формат и содержимое метки в протоколе не регламентируется, и зависит от реализации сервиса push_events. Удаленный сервер не должен как-то анализировать или изменять метку.

2.4.4. **Подтверждение получения событий**

Высылается удаленным сервером в ответ на посылку с событиями.

Таблица 13 Посылка подтверждения получения событий (версия 2.0)

Номер байта	Описание
0-1	Длина посылки (Big Endian), должна быть равна “00 03”
2	Байт со значением 04 (подтверждение получения посылки)
3-4	Количество полученных инициативных сообщений в посылке (Big Endian)

Количество полученных инициативных сообщений должно совпадать с количеством отосланных сообщений, иначе посылка считается неполученной и посылается еще раз. Контроллер может разорвать соединение, если несовпадения количества происходят более 3 раз подряд.

При получении корректного подтверждения посылки удаленным сервером контроллер может послать следующие инициативные сообщения, если они присутствуют в очереди.

Приложение №2

Протокол обмена данными ISaGRAF через задачу связи isacom.

Версия 1.3.

1. Общие положения.

Сервер -- процесс isacom (ISaGRAF Communication), работающий на контроллере. Позволяет клиенту получать текущие значения переменных ISaGRAF и записывать новые значения.

Клиент -- процесс, подключающийся к задаче связи isacom. Получает от нее текущие значения переменных ISaGRAF и записывает новые значения.

Клиент получает переменные в виде области памяти, в которой целевая задача ISaGRAF хранит текущие значения всех переменных. Используя символьную таблицу можно затем получить из этой области значение любой переменной, зная ее адрес и размер (тип). Такой способ позволяет максимально быстро передавать большое количество переменных.

Клиент передает переменные для записи в виде списка структур, каждая из которых содержит адрес переменной, ее размер и значение.

Обмен происходит в виде посылок запросов и получения ответов. На каждый запрос обязательно должен быть получен ответ. И запрос, и ответ представляют собой сообщения, состоящие из двух частей: заголовка и данных.

Структура заголовка:

```
struct msg_hdr {
        uint32_t msg_type;      /* тип сообщения */
        uint32_t msg_param;     /* необязательный параметр */
        uint32_t msg_len; /* длина сообщения */
        uint16_t msg_flags;     /* флаги */
        uint16_t __reserved;    /* не используется */
};
```

Поле msg_flags представляет собой комбинацию следующих битовых полей:

0x0001	режим резервирования
0x0002	основной контроллер (в режиме резервирования)

Непосредственно после заголовка идут данные, если они есть (msg_len > 0). Структура данных зависит от типа сообщения.

Ответ может содержать либо запрошенные данные, либо подтверждение удачной обработки запроса, либо сообщение об ошибке.

2. **Коды ошибок.**

   0x01 неправильное сообщение
   0x02 внутренняя ошибка сервера
   0x03 данные отсутствуют
   0x04 таблица символов отсутствует

3. **Запрос клиентом текущих значений переменных.**

   Клиент посылает сообщение

   заголовок:

   msg_type = 0x03
   msg_param = номер ресурса ISaGRAF
   msg_len = 0 (нет данных)
   __reserved = 0 (не используется)
   данных нет.

   В случае ошибки сервер посылает сообщение

   заголовок:

   msg_type = 0x02
   msg_param = код ошибки
   msg_len = 0 (нет данных)
   __reserved = 0 (не используется)
   данных нет.

   В случае успешного выполнения запроса сервер посылает сообщение

   заголовок:

   msg_type = 0x01
   msg_param = 0 (не используется)
   msg_len = размер данных
   __reserved = 0 (не используется)
   Данные представляют собой область памяти переменных ISaGRAF.

4. **Запись клиентом новых значений переменных.**

Клиент посылает сообщение

заголовок:

msg_type = 0x04
msg_param = номер ресурса ISaGRAF
msg_len = размер данных
__reserved = 0 (не используется)

Данные представляют собой последовательность структур

```
struct var_entry {
        uint32_t var_addr; /* адрес переменной */
        uint32_t var_size;  /* размер переменной */
        uint8_t  var_data[1];       /* значение переменной */
};
```

Размер каждой структуры варьируется в зависимости от значения поля var_size.

В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)

данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = 0 (не используется)
msg_len = 0 (нет данных)
__reserved = 0 (не используется)

данных нет.

5. **Запрос клиентом версии протокола.**

Клиент посылает сообщение

заголовок:

msg_type = 0x05
msg_param = номер версии протокола клиента

msg_len = 0 (не используется)
__reserved = 0 (не используется)

Номер версии представляет собой двухбайтное число, младший байт содержит младший номер версии, старший байт - старший номер. При добавлении новых команд в протокол младший номер увеличивается на единицу. При любых изменениях в протоколе, приводящих к его несовместимости с предыдущими версиями (например, удаление команд), старший номер версии увеличивается на единицу.

В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = номер версии протокола сервера
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

6. Создание списка переменных.

Существует возможность задать до 16 списков переменных, которые затем можно использовать для выборочного чтения и записи переменных. Информация о заданных списках сохраняется только в рамках текущей сессии.

Чтобы задать список клиент посылает сообщение

заголовок:

msg_type = 0x06
msg_param = биты 0-5 - номер списка, биты 6-7 - тип списка,
биты 8-15 - номер ресурса
msg_len = размер данных списка
__reserved = 0 (не используется)

Номер списка может быть от 0 до 15. Если список с заданным номером уже существует, его данные заменяются новыми.

Типы списков:

0 переменные задаются по адресам
1 переменные задаются по именам

Для списков типа 0 данные представляют собой последовательность следующих структур:

4 байта адрес переменной 4 байта размер переменной

Адрес и размер определяют положение переменной в области памяти ISaGRAF.

Для списков типа 1 данные представляют собой последовательность следующих

структур:

имя переменной, 0x00, имя типа, 0x00, 4 байта размер переменной

Для неглобальных переменных имя переменной имеет вид variable@pou, где pou -имя области видимости.

В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = 0 (не используется)
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

7. Чтение переменных по списку.

Клиент посылает сообщение

заголовок:

msg_type = 0x07
msg_param = номер списка
msg_len = 0 (не используется)
__reserved = 0 (не используется)
В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = 0 (не используется)
msg_len = размер данных
__reserved = 0 (не используется)
Данные представляют собой последовательность следующих структур

байт флагов, данные переменной

Если байт флагов равен нулю, следом за ним идут данные переменной из списка.

Если байт флагов равен 0x01, данные переменной недоступны, и следом идет следующая структура.

8. **Запись переменных по списку.**

Клиент посылает сообщение

заголовок:

msg_type = 0x08
msg_param = номер списка
msg_len = размер данных
__reserved = 0 (не используется)
Данные представляют собой последовательность структур значений переменных из списка.

В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = 0 (не используется)
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

9. Расширенная запись переменных.

Клиент посылает сообщение

заголовок:

msg_type = 0x09
msg_param = номер ресурса
msg_len = размер данных
__reserved = 0 (не используется)

Данные представляют собой последовательность структур значений переменных.

Каждая структура имеет следующий вид:

vname, 0x00, tname, 0x00, vsize, doff, dsize, data

Поле vname задает имя переменной, оканчивающееся нулевым байтом. Для не глобальных переменных имя переменной имеет вид variable@pou, где pou - имя области видимости. Поле tname задает имя типа переменной, заканчивающееся нулевым байтом. Поле vsize имеет размер 4 байта и задает полный размер переменной. Поле doff имеет размер 4 байта и задает смещение записываемых данных внутри данных переменной. Поле dsize имеет размер 4 байта и задает размер записываемых данных. Непосредственно за полем dsize расположены записываемые данные. Размер записываемых данных должен быть в точности равен dsize.

В случае ошибки сервер посылает сообщение

заголовок:

msg_type = 0x02
msg_param = код ошибки
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.

В случае успешного выполнения запроса сервер посылает сообщение

заголовок:

msg_type = 0x01
msg_param = 0 (не используется)
msg_len = 0 (нет данных)
__reserved = 0 (не используется)
данных нет.