GS LABS
GS GROUP

Публичный - L0

DRE Messaging Service

Руководство администратора

Индекс	DREMessagingService-AG
Конфиденциальность	Публичный - L0
Ревизия	1.0
Статус	Согласован

Содержание

1. Аннотация

Данный документ содержит настройки компонентов DRE Messaging Service (далее - Hermes или Системы) и рекомендуемые значения их параметров.

Документ предназначен для сотрудников отдела мониторинга и инсталляции, а также для других технических специалистов, в обязанности которых входит настройка системы Hermes и поддержание её работоспособности.

2. Термины и сокращения

Термин	Определение
ACM (Account Manager)	Продукт DRE Account Manager.
Okydrom	Продукт DRE Guaranteed Delivery System.
Приёмник (Set Top Box)	Приёмное оборудование с интегрированным ПО для обработки получаемых данных, установленное у абонента.

Сокращение	Расшифровка
ID	Identifier
IP	Internet Protocol
OTT	Over-The-Top
STB	Set Top Box

3. Обслуживание системы

Обслуживание Hermes заключается в выполнении следующих основных действий:

- Изменение настроек компонентов Hermes с помощью файлов *yaml*, содержащих настраиваемые параметры (**при необходимости**).
- Мониторинг работы компонентов Hermes с помощью Prometheus, устранение ошибок на основе метрик или логов системы.

4. Настройка Backend

Настраиваемые параметры описаны в таблице ниже.

Параметр	Описание
LOGGER_LEVEL	Степень логирования событий. Возможные значения: • trace, • debug, • info (значение по умолчанию), • warning, • error, • fatal. Для тестирования рекомендуется "debug" или "trace". Подробное описание уровней логирования приведено в разделе ниже.
SYSTEM_PULL_NOTIFICATIONS_ENABLED	Флаг активации взаимодействия с БД. Возможные значения: • false - Не работает функциональность pull+push нотификаций, на все запросы api/v1/pull /notifications* возвращается ошибка. Подключение к базе данных при этом отсутствует. • true - Включена функциональность pull+push, активировано подключение к БД. **ВНИМАНИЕ!** Обязательно указать в конфиге параметры реальной базы данных для подключения.
SYSTEM_PULL_NOTIFICATIONS_ERROR_CODE	Код ошибки, возвращаемой при вызове API pull нотификаций, но при выключенном режиме (т.е. при SYSTEM_PULL_NOTIFICATIONS_ENABLED: "false").

SYSTEM_TRACER_ENABLED	Настройки взаимодействия с Jaeger. Jaeger - система отслеживания запросов. Используется для мониторинга и устранения неполадок распределенных систем на основе микросервисов, в том числе: • Распределенное распространение контекста • Распределенный мониторинг транзакций • Анализ причин • Анализ сервисных зависимостей • Оптимизация производительности / задержек Флаг включения трассировки. Возможные значения: • false - трассировка выключена. • true - трассировка включена.
SYSTEM_TRACER_AGENT_HOST_PORT	Адрес и номер порта agent, т.е. сервера, на который идет отправка параметров трассировки.
SYSTEM_BACKEND_DB_MASTER_HOST	IP-адрес мастер реплики (инстанс кластера Hermes DB).
SYSTEM_BACKEND_DB_MASTER_PORT	Номер порта мастер реплики (инстанс кластера Hermes DB).
SYSTEM_BACKEND_DB_MASTER_USER	Логин пользователя, под которым осуществляется подключение к мастер реплике (инстансу кластера Hermes DB).
SYSTEM_BACKEND_DB_MASTER_PASSWORD	Пароль для подключения к мастер реплике (инстансу кластера Hermes DB).
SYSTEM_BACKEND_DB_MASTER_DB_NAME	Имя мастер реплики (инстанс кластера Hermes DB) в PostgreSQL.
SYSTEM_BACKEND_DB_MASTER_TIMEOUT	Время ожидания ответа (в сек.) от мастер реплики (инстанс кластера Hermes DB).
SYSTEM_BACKEND_DB_MASTER_MAX_CONNS	Максимальное количество соединений, создаваемых сервером с мастер репликой (инстансом кластера Hermes DB).
SYSTEM_BACKEND_DB_ASYNC_REPLICA_HOST	IP-адрес асинхронной реплики (инстанс кластера Hermes DB).
SYSTEM_BACKEND_DB_ASYNC_REPLICA_PORT	Номер порта асинхронной реплики (инстанс кластера Hermes DB).

SYSTEM_BACKEND_DB_ASYNC_REPLICA_USER	Логин пользователя, под которым осуществляется подключение к асинхронной реплике (инстансу кластера Hermes DB).
SYSTEM_BACKEND_DB_ASYNC_REPLICA_PASSWORD	Пароль для подключения к асинхронной реплике (инстансу кластера Hermes DB).
SYSTEM_BACKEND_DB_ASYNC_REPLICA_DB_NAME	Имя асинхронной реплики (инстанс кластера Hermes DB) в PostgreSQL.
SYSTEM_BACKEND_DB_ASYNC_REPLICA_TIMEOUT	Время ожидания ответа (в сек.) от асинхронной реплики (инстанс кластера Hermes DB).
SYSTEM_BACKEND_DB_ASYNC_REPLICA_MAX_CONNS	Максимальное количество соединений, создаваемых сервером с асинхронной репликой (инстансом кластера Hermes DB).
SYSTEM_GRPC_PORT	Номер порта для работы с gRPC (системой удаленного вызова процедур). **Примечание**. С gRPC работают на одном хосту, поэтому другие параметры не требуются.
SYSTEM_HTTP_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: <*IP-адрес*>:<*номер порта*> Используется протокол незащищенной сети.
SYSTEM_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_HTTP_IS_USED	Флаг включения взаимодействия по http. Возможные значения: • false - Взаимодействие выключено. • true - Сервис взаимодействует по http.
SYSTEM_PROMETHEUS_HTTP_ADDRESS	Адрес для запуска сервера Prometheus в контейнере. Формат записи: <*IP-адрес*>:<*номер порта*> Используется протокол незащищенной сети.

SYSTEM_PROMETHEUS_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_PROMETHEUS_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_REDIS_ADDRESS	Адрес и номер порта Redis.
SYSTEM_SERIALIZATION_EMIT_DEFAULTS	Параметр определяет на уровне сервера, возвращать ли значения false для булевых и 0 для целочисленных атрибутов.
SYSTEM_PPROF_ENABLE	Флаг включения взаимодействия с pprof. Возможные значения: • false - pprof выключен. Значение по умолчанию. • true - pprof включен. **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.
SYSTEM_PPROF_ACCESS_KEY	Значение ключа доступа, отправляемого в запросе к pprof. Запрос используется для профилирования сервисов (исследования CPU). **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.
SYSTEM_USE_OPERATOR	Флаг включения в параметры push и pull нотификаций кода оператора. Если флаг имеет значение false, то работа с операторами не происходит, даже если оператор был передан.

5. Настройка Broker

Настраиваемые параметры описаны в таблице ниже.

Параметр	Описание
broker.nodeSelector.key	Название label, который должен быть добавлен на ноду, чтобы broker "знал", на какой ноде ему необходимо запустить pod.
broker.nodeSelector.value	Значение label, который используется также для обозначения ноды, на которой необходимо запуститься. **Примечание**. В "DRE Messaging Service. Руководство по установке" данные параметры (*broker.nodeSelector.key* и *broker.nodeSelector.value*) имели дефолтное значение "ka: external". Эти параметры необходимы для установки и не используются при работе сервиса.
broker.NodePort	Номер node-порта при развертывании сервиса. **Примечание**. Параметр не используется сервером, используется только при развертке.
broker.config.logger.enable_health_checks	Параметр отвечает за включение логирования livenessprobe запросов. Если данный параметр находится в положении false, то логирование не происходит. Начальное значение - false.
LOGGER_LEVEL	Степень логирования событий. Возможные значения: • trace, • debug, • info (значение по умолчанию), • warning, • error, • fatal. Для тестирования рекомендуется "debug" или "trace". Подробное описание уровней логирования приведено в разделе ниже.

SUPPORTED_SCANNER_TYPES	Массив из типов сканеров (Scanner types), которые поддерживаются на данном экземпляре Broker_go. **Примечание**. Здесь и далее Scanner - условное наименование части внешней системы, которая собирает Уведомления, полученные от источника (Message Provider System), и передает их компоненту Backend. **Примечание**. Когда на Broker_go приходит запрос по websocket, то Broker_go проверяет, имеет ли запрос тип, описанный в данном параметре. Если да, то Broker_go разрешает соединение и подписывает на нотификации. Если нет, то соединение не устанавливается.
SUPPORTED_GROUPS	Типы subs-group, которые будут учитываться при формировании метрик по подсчету количества подписок, пришедших от устройств. Если устройство подписалось на канал с subs-group, которого нет в списке supported_groups, то эта подписка в метрике учитываться не будет.
OKYDROM_HOST	Адрес сервиса "DRE Guaranteed Delivery System" (далее по тексту - Okydrom) (без указания протокола), который может иметь следующие значения: IP-адрес:порт либо servicename (в K8s).
OKYDROM_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Okydrom.
OKYDROM_PROTOCOL	Протокол, по которому осуществляется взаимодействие с Okydrom. Значение по умолчанию: http
OKYDROM_HEALTH_CHECK	Флаг проверки доступности (health check) Okydrom при старте.
OKYDROM_NOTIFY_TIME	Время (в секундах) перед первой попыткой отправки нотификации из Okydrom.
OKYDROM_REPEAT_AMOUNT	Количество попыток отправки нотификации в Okydrom (в случае если отправка заканчилась ошибкой).
OKYDROM_REPEAT_INTERVAL	Время (в секундах), через которое нужно повторить попытку отправки нотификации в Okydrom (в случае если отправка заканчилась ошибкой).

TRIGGER_CONFIG_BUFFER_SIZE	Количество событий, хранимых сервисом trigger_config, перед отправкой нотификации в Okydrom. **Примечание**. Trigger (триггер событий) - приложение внутри Broker, которое отвечает за отправление нотификаций в сторонние системы через Okydrom. trigger отвечает за то, при выполнении каких действий с какими сканерами должны создаваться нотификации в Okydrom и какой при этом будет передаваться хост. Также сервис определяет количество хранимых событий перед отправкой нотификации в Okydrom и количество времени, через которое в Okydrom будет создана нотификация, если не будет достигнут предел количества событий.
TRIGGER_CONFIG_FLUSH_TIMEOUT	Временной промежуток (в миллисекундах), по истечении которого в Okydrom будет создана нотификация о хранимых событиях.
TRIGGER_CONFIG_ENABLED	Флаг включения взаимодействия с trigger. Возможные значения: • false - взаимодействие выключено. В этом случае не будет нотификаций через Okydrom. • true - взаимодействие включено.
TRIGGER_CONFIG_EVENTS_CHAN_SIZE	Длина внутренней очереди (количество нотификаций в канале) между Broker и триггером событий.
TRIGGER_CONFIG_NOTIFY_DELAY	Время жизни ключа broker_go (хранится в Redis), указывается в секундах. Является ограничением на отправку *event*. Значение по умолчанию - 60 (1 минута). **Примечание**. Параметр является частью механизма контроля повторных соединений в Hermes. При обратных нотификациях сохраняется кэш по устройству, которое уже ходило в Okydrom. Если кэш присутствует, то в Okydrom не идём.
TRIGGER_CONFIG_DELAY_TTL	Время кеширования обратной нотификации от устройства к серверу. Начальное значение - 1200, измеряется в секундах. Значение стандартного таймаута между pong'ами - 300 секунд.

TRIGGER_CONFIG_DELAY_PONG_LIMIT	Номер pong, после которого данные в кеше об обратной нотификации будут продлены. Начальное значение - 2.
TRIGGER_CONFIG_DELAY_PONG_LIMIT_DELTA	Предел отклонения от pong_limit. Отклонение формируется в интервале [0, TRIGGER_CONFIG_DELAY_PONG_LIMIT_DELTA]. Начальное значение - 1.

TRIGGER_CONFIG_ACTIONS_*EVENT_SCANNER*
_NOTIFY_HOST
 	
где:
 	
• *EVENT* - действие, при выполнении которого должна создаваться нотификация в Okydrom. Примеры:
  • CLIENT-SUBSCRIBED
  • CLIENT-UNSUBSCRIBED
• *SCANNER* - система, в состав которой входит Scanner, используемый для отправки нотификаций. Примеры:
  • SMS
  • DRM
	Мапа *EVENT* и соответствующих им подсистемам.
	
	Когда Broker раскидывает нотификации (перед отправкой клиенту), то смотрит на *EVENT*. Если такой *EVENT* есть в мапе в actions, то дальше Broker смотрит, нужный ли это тип сканнера. Если есть такой тип сканнера в мапе, то Broker добавляет его с таким хостом.
	
	Иными словами, параметр определяет, при выполнении каких действий *EVENT* с какими сканерами *SCANNER* должны создаваться нотификации в Okydrom и какой при этом будет передаваться хост.

TRIGGER_CONFIG_ACTIONS_*EVENT_SCANNER*
_DELAY_FILTER_SUB_GROUP
 	
где:
 	
• *EVENT* - действие, при выполнении которого должна создаваться нотификация в Okydrom. Примеры:
  • CLIENT-SUBSCRIBED
  • CLIENT-UNSUBSCRIBED
• *SCANNER* - система, в состав которой входит Scanner, используемый для отправки нотификаций. Примеры:
  • SMS
  • DRM
	Параметр показывает, подписки с каким sub_group будут закешированы. Начальное значение - "individual".

SYSTEM_TRACER_ENABLED	Настройки взаимодействия с Jaeger. Jaeger - система отслеживания запросов. Используется для мониторинга и устранения неполадок распределенных систем на основе микросервисов, в том числе: • Распределенное распространение контекста • Распределенный мониторинг транзакций • Анализ причин • Анализ сервисных зависимостей • Оптимизация производительности / задержек Флаг включения трассировки. Возможные значения: • false - трассировка выключена. • true - трассировка включена.
SYSTEM_TRACER_AGENT_HOST_PORT	Адрес и номер порта agent, т.е. сервера, на который идет отправка параметров трассировки.
SYSTEM_HTTP_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: <*IP-адрес*>:<*номер порта*> Используется протокол незащищенной сети.
SYSTEM_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_HTTPS_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: <*IP-адрес*>:<*номер порта*> Используется протокол **защищенной** сети.
SYSTEM_HTTPS_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных). Используется протокол **защищенной** сети.
SYSTEM_HTTPS_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных). Используется протокол **защищенной** сети.

SYSTEM_HTTPS_ENABLED	Флаг, показывающий, используется или нет протокол **защищенной** сети.
SYSTEM_HTTPS_CA_CERT	Путь к файлу с цепочкой сертификатов (ca-chain.cert.pem).
SYSTEM_HTTPS_SERVER_CERT	Путь к файлу с публичной частью server certificate (server.cert.pem).
SYSTEM_HTTPS_SERVER_KEY	Путь к файлу с приватной частью server certificate (server.key.pem).
SYSTEM_HTTPS_VERIFY_MODE	Параметр, по которому переключаются режимы работы с сертификатами. Возможные значения: • 0 - NoClientCert ClientAuthType = iota NoClientCert означает, что сертификат клиента не будет запрашиваться во время взаимодействия. Если какие-либо сертификаты будут отправлены, то они не будут подвергаться проверке. • 1 - RequestClientCert RequestClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, однако не требует, чтобы клиент отправлял какие-либо сертификаты. • 2 - RequireAnyClientCert (значение по умолчанию) RequireAnyClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, и что по крайней мере один сертификат должен быть отправлен клиентом, но этот сертификат не обязательно должен быть действительным (валидным). • 3 - VerifyClientCertIfGiven VerifyClientCertIfGiven означает, что сертификат клиента должен быть запрошен во время взаимодействия, но не требует, чтобы клиент отправлял сертификат. Если клиент отправляет сертификат, он должен быть действительным. • 4 - RequireAndVerifyClientCert RequireAndVerifyClientCert означает, что сертификат клиента должен быть запрошен во время взаимодействия, и что клиент должен отправить по крайней мере один действительный сертификат.

SYSTEM_PROMETHEUS_HTTP_ADDRESS	Адрес для запуска сервера Prometheus в контейнере. Формат записи: *<IP-адрес>:<номер порта>* Используется протокол незащищенной сети.
SYSTEM_PROMETHEUS_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_PROMETHEUS_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_WEBSOCKET_HANDLER_READ_BUFFER_SIZE	Размер буфера вывода данных. Если размер буфера равен 0, тогда они аллоцируются http-сервером стандартной библиотеки go. Размеры буферов не ограничивают размеры сообщений, которые могут быть приняты или отправлены.
SYSTEM_WEBSOCKET_HANDLER_WRITE_BUFFER_SIZE	Размер буфера ввода данных. Если размер буфера равен 0, тогда они аллоцируются http-сервером стандартной библиотеки go. Размеры буферов не ограничивают размеры сообщений, которые могут быть приняты или отправлены.
SYSTEM_WEBSOCKET_HANDLER_USE_WRITE_BUFFER_POOL	Пул буферов для операций записи. Если задан как false, тогда буферы для записи аллоцируются при создании websocket-соединения на время жизни соединения.
SYSTEM_WEBSOCKET_CLIENT_WRITE_WAIT	Таймаут (в секундах) ожидания записи для websocket-соединения. По истечению таймаута, соединение переходит в нерабочее состояние. Все будущие записи по такому соединению будут возвращать ошибку, что приведет к разрыву данного соединения.
SYSTEM_WEBSOCKET_CLIENT_PONG_WAIT	Таймаут (в секундах) ожидания pong сообщения от клиента для websocket-соединения. По истечению таймаута, соединение переходит в нерабочее состояние. Все будущие операции чтения по такому соединению будут возвращать ошибку, что приведет к разрыву данного соединения.

SYSTEM_WEBSOCKET_CLIENT_MAX_MESSAGE_SIZE	Максимальная длина сообщений (в байтах), принимаемых от клиента. Если длина сообщения превышена, то клиенту отправляется сообщение о закрытии соединения, а логе сервиса будет ошибка 'websocket: read limit exceeded'.
SYSTEM_REDIS_ADDRESS	Адрес и номер порта Redis.
SYSTEM_REDIS_SUBSCRIBER_WORKERS	Количество goroutine обработчиков (модуль по приему сообщений от Redis).
SYSTEM_REDIS_SUBSCRIBER_CHANNEL_SIZE	Размер канала для входящих сообщений от Redis.
SYSTEM_REDIS_SUBSCRIBER_HEALTH_CHECK_INTERVAL_SEC	Интервал (в сек.) health check'а соединения с Redis.
SYSTEM_REDIS_SUBSCRIBER_CHANNEL_SEND_TIMEOUT_SEC	Время (в сек.) ожидания доступности канала на запись (канал может быть заполнен), по истечению этого времени сообщение теряется.
SYSTEM_PPROF_ENABLE	Флаг включения взаимодействия с pprof. Возможные значения: • false - pprof выключен. Значение по умолчанию. • true - pprof включен. **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.
SYSTEM_PPROF_ACCESS_KEY	Значение ключа доступа, отправляемого в запросе к pprof. Запрос используется для профилирования сервисов (исследования CPU). **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.

6. Настройка Orchestrator

Настраиваемые параметры описаны в таблице ниже.

Параметр	Описание
LOGGER_LEVEL	Степень логирования событий. Возможные значения: • trace, • debug, • info (значение по умолчанию), • warning, • error, • fatal. Для тестирования рекомендуется "debug" или "trace". Подробное описание уровней логирования приведено в разделе ниже.
ERRORS_API_NOT_ALLOWED_CODE	Код ошибки для сценария ошибки "api not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер неизвестный URL request).
ERRORS_API_NOT_ALLOWED_TITLE	Заголовок ошибки для сценария ошибки "api not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер неизвестный URL request).
ERRORS_API_NOT_ALLOWED_DETAIL	Содержимое блока с ошибкой для сценария ошибки "api not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер неизвестный URL request).
ERRORS_METHOD_NOT_ALLOWED_CODE	Код ошибки для сценария ошибки "method not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер верный URL request, но при этом использует неподдерживаемый HTTP method).
ERRORS_METHOD_NOT_ALLOWED_TITLE	Заголовок ошибки для сценария ошибки "method not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер верный URL request, но при этом использует неподдерживаемый HTTP method).
ERRORS_METHOD_NOT_ALLOWED_DETAIL	Содержимое блока с ошибкой для сценария ошибки "method not allowed" (ошибки, отправляемой в случае, если клиент пытается отправить на сервер верный URL request, но при этом использует неподдерживаемый HTTP method).

ERRORS_INTERNAL_CODE	Код ошибки для обобщенной внутренней ошибки ("internal error").
ERRORS_INTERNAL_TITLE	Заголовок ошибки для обобщенной внутренней ошибки ("internal error").
ERRORS_INTERNAL_DETAIL	Содержимое блока с ошибкой для обобщенной внутренней ошибки ("internal error").
REDIRECT_ADDRESSES	Массив адресов (IP-адрес:порт) для переадресации (на указанные адреса Orchestrator распределяет трафик). Указываются либо адреса Broker'ов, либо Balancer, который будет взаимодействовать с Broker'ами.
SYSTEM_ROUTING_FILE_PATH	Путь к файлу ***routing.yaml***. В соответствии с настройками, заданными в routing.yaml, в orchestrator_go_server осуществляется перенаправление (проксирование) запроса в требуемый микросервис (backend_go). **ВНИМАНИЕ!** Администратор Системы должен: • Перед развёртыванием Hermes - открыть routing.yaml и проверить корректность ссылки на backend (параметр variables:backend). • В любое время - открыть routing.yaml и ознакомиться с остальными параметрами, в частности, с тем, какие функции проксируются (параметры incoming и methods).
SYSTEM_TRACER_ENABLED	Настройки взаимодействия с Jaeger. Jaeger - система отслеживания запросов. Используется для мониторинга и устранения неполадок распределенных систем на основе микросервисов, в том числе: • Распределенное распространение контекста • Распределенный мониторинг транзакций • Анализ причин • Анализ сервисных зависимостей • Оптимизация производительности / задержек Флаг включения трассировки. Возможные значения: • false - трассировка выключена. • true - трассировка включена.
SYSTEM_TRACER_AGENT_HOST_PORT	Адрес и номер порта agent, т.е. сервера, на который идет отправка параметров трассировки.

SYSTEM_HTTP_ADDRESS	Адрес для запуска сервера в контейнере. Формат записи: *<IP-адрес>:<номер порта>* Используется протокол незащищенной сети.
SYSTEM_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_HTTP_IS_USED	Флаг включения взаимодействия по http. Возможные значения: • false - Взаимодействие выключено. • true - Сервис взаимодействует по http.
SYSTEM_PROMETHEUS_HTTP_ADDRESS	Адрес для запуска сервера Prometheus в контейнере. Формат записи: *<IP-адрес>:<номер порта>* Используется протокол незащищенной сети.
SYSTEM_PROMETHEUS_HTTP_WRITE_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на запись данных). Используется протокол незащищенной сети.
SYSTEM_PROMETHEUS_HTTP_READ_TIMEOUT	Таймаут (в сек.), по которому будет разорвано соединение с Prometheus (при обращении на чтение данных). Используется протокол незащищенной сети.
SYSTEM_PPROF_ENABLED	Флаг включения взаимодействия с pprof. Возможные значения: • false - pprof выключен. Значение по умолчанию. • true - pprof включен. **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.
SYSTEM_PPROF_ACCESS_KEY	Значение ключа доступа, отправляемого в запросе к pprof. Запрос используется для профилирования сервисов (исследования CPU). **Примечание**. Подробное описание приведено в разделе Взаимодействие с pprof.

SYSTEM_RATE_LIMIT_RPS	Настройка для лимитирования количества входящих запросов, где rps - количество запросов в секунду. Если лимит будет превышен, то пользователю будет возвращена ошибка http 429 Too Many Requests.

7. Взаимодействие с pprof

Для анализа узких мест в реализации серверов и точечной настройки Golang серверов используется сторонний компонент pprof.

Для работы с ним в конфигурационном файле каждого Golang сервера, а также в default.yaml/production.yaml имеется секция pprof.

По умолчанию pprof выключен. Чтобы включить взаимодействие с pprof, необходимо:

- в production.yaml выставить pprof.enable равным true и заменить access_key на собственное значение:

```
...
pprof:
  enabled: true
  access_key: fc64a74a-51ca-4cb9-afea-5d5c633bc4d0
...
```

> ⚠ Значение pprof в production.yaml имеет более высокий приоритет, чем в конфигурационных файлах компонентов, поэтому менять значения в конфигурационном файле каждого отдельного сервиса не требуется.

- развернуть либо перезапустить все службы Hermes.

Если pprof включен, то он позволяет отправлять запрос вида:

Пример запроса

```
curl --request GET '192.168.11.86:30170/debug/pprof/profile?seconds=15' --header 'X-Api-Key: fc64a74a-51ca-4cb9-afea-5d5c633bc4d0' --output fasentry.bin
```

Наличие параметра *--header 'X-Api-Key: значение_access_key_из_конфига'* в запросе обязательно.

Запрос используется для профилирования сервисов (исследования CPU).

8. Настройка взаимодействия DRE Messaging Service (Hermes) с DRE Account Manager (Account Manager)

Для создания разветвленной системы прав необходимо прописать права пользователей (permissions) в сервисе "DRE Account Manager" (далее по тексту - Account Manager или ACM).

Процедура выполняется в следующих случаях:

- при установке системы Hermes "с нуля";
- в случае обновления/добавления/удаления прав (permissions).

8.1. Добавление сервиса 'internalserviceaccess' в Account Manager

Данная процедура выполняется однократно, до установки/обновления прав в Account Manager. В случае сбоя /переустановки Account Manager "с нуля" этот сервис может быть удалён - в этом случае его придётся создавать заново.

1. Войдите в web-интерфейс Account Manager.
2. В левой части окна выберите Администрирование -> Сервисы. Раздел Services доступен только суперпользователю Account Manager.
3. Проверьте, что в списке сервисов присутствует сервис 'internalserviceaccess'. Если его нет, добавьте его. Подробное описание работы с сервисами приведено в документе "Руководство пользователя" Account Manager в разделе "Сервисы".

8.2. Добавление permissions в Account Manager

Добавление permissions происходит с помощью скрипта. Доступ к репозиторию с необходимыми для загрузки данными предоставляется по запросу.

Загрузка происходил автоматически. Права берутся из файла permissions.json и загружаются в Account Manager с помощью скрипта load.sh, который, в свою очередь, вызывает скрипт create_acm_permissions. Адрес Account Manager задаётся в production.yaml в параметре account_manager.address

8.3. Проверка загруженных permissions

1. Войдите в web-интерфейс ACM.
2. В левой части окна выберите Администрирование -> Роли.
3. В появившейся форме, в поле "Название" введите имя административной роли (например, ACM).
4. На экране отобразится список разрешений, доступных для этой роли. Необходимо, чтобы в правый список были добавлены все разрешения для работы с Hermes (начинаются с internalserviceaccess). Если это не так, добавьте их к административной роли.
5. Далее токен данного пользователя можно использовать для взаимодействия с Backend API.

9. Scanner types

> Scanner - условное наименование части внешней системы, которая собирает Уведомления, полученные от источника (Message Provider System), и передает их компоненту Backend.

Для настройки scanner types, поддерживаемых Hermes, администратор должен учитывать следующее:

1. Для всех видов Уведомлений (*только push*, *push+pull*):
    a. Если создали новый тип сканера, то перед развёртыванием Hermes, в параметрах настроек Broker (в default.yaml, supported_scanner_types) нужно добавить новый тип сканера. Так же возможно добавление нового типа сканера в процессе эксплуатации через динамический конфиг broker_go.
2. Только для *push+pull* Уведомлений:
    a. В Hermes DB есть начальное наполнение таблицы hrm_scanner_types. Наполнение устанавливается автоматически, "из коробки".
    b. Для добавления нового типа сканера можно воспользоваться функцией БД scanner_type_create (функция из hermes_db_api).

10. Сбор метрик

Компоненты собирают статистику о своей работе. Показатели, по которым собирается статистика, задаются так называемыми "метриками".

Сбор метрик осуществляется средствами Prometheus, где собранная информация аккумулируется и отображается графически с помощью Grafana.

⚠ Инструменты Prometheus и Grafana являются сторонними по отношению к Hermes продуктами. Описание их структуры, принципов действия, процедур установки и т.п. выходит за рамки настоящего документа.

11. Ведение Логов

Параметры логирования (режим ведения логов и т.д.) настраиваются в **values.yaml.gotmpl**, с помощью параметров ***logger_xxx***.

11.1. Режимы Ведения Логов

Компонент ведет логи, информация из которых может быть использована для решения возникающих проблем. Логи могут вестись с разной степенью подробности.

Доступны следующие режимы ведения логов:

- trace: подробная информация по любым действиям;
- debug: конфигурационные данные (при запуске системы), другая информация, необходимая для отладки, + сообщения уровня Info;
- info (значение по умолчанию): базовая информация (сообщения о запуске, работе, выключении системы) + сообщения уровня Warning;
- warning: системные предупреждения + сообщения уровня Error;
- error: все ошибки, возникающие в процессе работы, в том числе ошибки уровня Fatal;
- fatal: критические ошибки, приводящие к сбоям системы.

11.2. Формат записей

ⓘ Приведенные ниже форматы логов применимы для backend_go, broker_go, orchestrator_go.

Все логи представлены в формате JSON. Каждое сообщение лога представлено в формате отдельной JSON-структуры с исчерпывающим набором полей.

В зависимости от уровня логирования изменяется набор данных внутри JSON либо добавляются новые элементы (записи лога).

11.2.1. Форматы логов

Логи для режима **debug** имеют структуру JSON и по формату делятся на следующие виды:

1. Лог для event: "request";
2. Лог для event: "response";
3. Лог для event: "http".

Логи для event: "request" и event: "response" содержат информацию по запросу к Broker и ответу на запрос соответственно.

Лог для event: "http" хранит меньше информации и предназначен скорее для статистики. Используется для логирования взаимодействия по http протоколу.

ВНИМАНИЕ! Настоятельно рекомендуется настраивать систему так, чтобы логи с event: "http" хранились значительно дольше, чем логи с event: "request" и event: "response".

Описание логов выполнено следующим образом:

- Пример лога указанного типа.
- Таблица с описанием параметров, используемых только в логе данного типа. Указываются только те параметры, которые являются более-менее частными (уникальными) для данного лога (остальные параметры приведены в разделе "Общие параметры"). Если у формата лога нет уникальных параметров, то таблица отсутствует.

11.2.1.1. Общие параметры

Общие параметры, которые встречаются во всех типах логов, приведены в таблице ниже.

Параметр	Описание
app	Наименование приложения (микросервиса Hermes)
app_ver	Версия приложения (микросервиса Hermes)
cid	Correlation id, используемый для трассировки запросов
"details":	Дополнительные передаваемые параметры (состав различается от одного формата лога к другому)
event	Тип логируемого события. От типа логируемого события зависит формат лога
level	Уровень логирования (см. Режимы Ведения Логов) **Примечание**. Структура лога регламентирована только для "level":"debug"
msg	Комментарий к логу
time	Дата и время (вплоть до миллисекунд) формирования записи (фиксации лога)

11.2.1.2. event: "request"

Пример лога для event: "request"


```
{
        "app":"broker_go",
        "app_ver":"1.0.17",
        "cid":"637b84c8-4cd4-40de-9d0d-33b9c4b56464",
        "details":{
                "body":null,
                "headers":{
                        "Accept-Encoding":["gzip"],
                        "Grpc-Metadata-Cid":["637b84c8-4cd4-40de-9d0d-33b9c4b56464"],
                        "User-Agent":["Go-http-client/1.1"],
                        "X-Correlation-Id":["637b84c8-4cd4-40de-9d0d-33b9c4b56464"]
                },
                "method":"GET",
                "path":"/api/v1/id",
                "protocol":"HTTP/1.1",
                "request_length":0
        },
        "event":"request",
        "level":"debug",
        "msg":"Request accepted",
        "time":"2021-11-30T13:51:02.207Z"
}
```


Параметр	Описание
details.body	Тело запроса Опционально может иметь null, "", {}
details.headers	Все headers (заголовки) в запросе (т.е. дополнительно может содержать cookies, X-correlation-id и т.п.)
details.method	Метод (тип) http-запроса
details.path	URL запроса
details.protocol	Протокол взаимодействия Значение по умолчанию: "HTTP/1.1"
details. request_length	Длина запроса (в Байтах)

11.2.1.3. event: "response"

Пример лога для event: "response"


```
{
        "app":"broker_go",
        "app_ver":"1.0.17",
        "cid":"637b84c8-4cd4-40de-9d0d-33b9c4b56464",
        "details":{
                "body":{
                        "id":"723bfe25-76f1-414c-baa1-6b967ea70cd1"},
                        "headers":{
                                "Content-Type":["application/json"],
                                "X-Correlation-Id":["637b84c8-4cd4-40de-9d0d-33b9c4b56464"]
                        },
                        "status":200},
                        "event":"response",
                        "level":"debug",
                        "msg":"Response was sent",
                        "time":"2021-11-30T13:51:02.207Z"
}
```


Параметр	Описание
details.body	Тело запроса Опционально может иметь null, "", {}
details.body. errors	Описание ошибки (код, заголовок, текст ошибки) **Примечание**. Передается, только если запрос завершился ошибкой
details.body. headers	Все headers (заголовки) в ответе (т.е. дополнительно может содержать cookies, X-correlation-id и т.п.)
details.status	Код ответа от сервера

11.2.1.4. event: "http"

Пример лога для event: "http"


```
{
        "app":"broker_go",
        "app_ver":"1.0.17",
        "cid":"637b84c8-4cd4-40de-9d0d-33b9c4b56464",
        "details":{
                "accept_language":"",
                "domain_id":"",
                "hwid":"",
                "method":"GET",
                "path":"/api/v1/id",
                "protocol":"HTTP/1.1",
                "remote_addr":"",
                "request_length":0,
                "request_time":0,
                "response_length":45,
                "status":200,
                "user_agent":
                "Go-http-client/1.1"
        },
        "event":"http",
        "level":"debug",
        "msg":"Completed",
        "time":"2021-11-30T13:51:02.207Z"
}
```


Параметр	Описание
details. accept_language	Код языка в формате ISO 639-1, используемый для создания описания ошибки, по умолчанию "ru" См. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Language
details.domain_id	Идентификатор домена
details.hwid	Идентификатор устройства пользователя, с которого пришел запрос
details.method	Метод (тип) http-запроса
details.path	URL запроса
details.protocol	Протокол взаимодействия Значение по умолчанию: "HTTP/1.1"
details.remote_addr	IP-адрес, с которого был сделан запрос
details. request_length	Длина запроса (в Байтах), включая строку запроса, заголовок и тело запроса
details.request_time	Время обработки запроса (в миллисекундах)
details. response_length	Длина ответа (в Байтах)

details.status	Код ответа от сервера
details.user_agent	Заголовок `User-Agent` запроса

11.2.1.5. Логи для других уровней логирования (не debug)

ВНИМАНИЕ! Логи для других уровней логирования (все кроме debug) не регламентированы.

Пример лога для logger_level: trace

```
{
        "app":"broker_go",
        "app_ver":"1.0.17",
        "cid":"637b84c8-4cd4-40de-9d0d-33b9c4b56464",
        "level":"trace",
        "msg":"unmarshal body failed: unexpected end of JSON input",
        "time":"2021-11-30T13:51:02.207Z"
}
```

Пример лога для logger_level: error

```
{
        "app":"broker_go",
        "app_ver":"1.0.17",
        "hw_id":"05f5d90a3ed35cf0a403a099a30cf0612d8f8529833e14aee7e1aa8f3fcaa8d1",
        "level":"error",
        "module":"client",
        "msg":"ReadMessage error: unexpected EOF, code: 1006",
        "time":"2021-11-30T13:52:48.562Z"
}
```