Руководство системного администратора программного обеспечения

Data Quality Framework

Аннотация

Настоящий документ представляет собой руководство системного администратора программного обеспечения (далее – ПО) Data Quality Framework (далее – ПО DQF), в котором определяется порядок установки, настройки и администрирования ПО.

Исключительные права на ПО DQF принадлежат ООО «Клин Дейта».

Перед использованием необходимо ознакомиться с документом «Руководство по развертыванию и установке программного обеспечения Data Quality Framework».

Оглавление

1. Область применения 4
1.1. Требования к уровню подготовки персонала 4
1.2. Управление доступом 4
2. Общие сведения 5
3. Установка и настройка ПО DQF 6
3.1. Установка и настройка ПО DQF: Engine 6
3.1.1. Подготовка ИТ–инфраструктуры 6
3.1.1.1. Требования к аппаратной платформе 6
3.1.1.2. Требования к сетевой инфраструктуре 6
3.1.2. Установка ПО DQF: Engine 6
3.1.3. Обеспечение отказоустойчивости ПО DQF: Engine 6
3.1.4. Логирование в ПО DQF 7
3.1.5. Управление файлами конфигураций комплексных алгоритмов 7
3.1.6. Формирование сообщений-запросов и получение сообщений-ответов с использованием брокера очередей 8
3.2. Установка и настройка ПО DQF: Federation 12
3.2.1. Подготовка ИТ–инфраструктуры 12
3.2.1.1. Требования к аппаратной платформе 12
3.2.1.1.1. Конфигурация сервиса Узла 12
3.2.1.1.2. Конфигурация БД Узла 12
3.2.1.1.3. Конфигурация сервиса Центра 12
3.2.1.1.4. Конфигурация БД Центра 13
3.2.1.1.5. Конфигурация БД кэша Центра 13
3.2.1.2. Требования к сетевой инфраструктуре 13
3.2.2. Установка ПО DQF: Federation 13
3.2.3. Обеспечение отказоустойчивости ПО DQF: Federation 14
3.2.4. Логирование в ПО DQF: Federation 14
3.2.5. Формирование сообщений-запросов с использованием брокера очередей 14

1. Область применения

Настоящий документ предназначен для пользователей, в должностные обязанности которых входят функции по администрированию ПО DQF.

Настоящее Руководство системного администратора предназначено для:

- подготовки к установке и настройке дистрибутива ПО DQF;
- мониторинга событий в сбои работоспособности ПО DQF;
- управления настройками ПО DQF.

1.1. Требования к уровню подготовки персонала

К уровню подготовки администратора ПО DQF предъявляются следующие требования:

- знание технической и рабочей документации на ПО DQF;
- наличие опыта работы по установке, настройке и поддержке ПО DQF;
- базовые знания Linux, Java, работы с консолью в оболочке Bash (написание командных срок);
- базовые навыки администрирования ПО RabbitMQ, Kafka, Docker.

1.2. Управление доступом

Установка и администрирование ПО DQF осуществляются системным администратором, имеющим доступ к программно-аппаратной инфраструктуре, на которой будет произведена установка ПО DQF.

2. Общие сведения

Программное обеспечение Data Quality Framework (далее – ПО DQF) реализует функции контроля качества и сопоставления данных.

Компонент контроля качества (далее – ПО DQF: Engine) обеспечивает контроль данных, которые рассматриваются как целостное образование, состоящее из взаимосвязанных частей. ПО DQF: Engine нацелено на мониторинг качества данных на различных уровнях и обнаружение ошибок по сформированным правилам и критериям проверок.

Функциональные возможности ПО DQF: Engine позволяют с помощью специальных правил и алгоритмов производить верификацию проверяемого объекта и его атрибутов и выявлять ошибки, а также настраивать проверки данных в соответствии с предметной областью и заданной конфигурацией (бизнес-логики) исполняемой проверки без доработки программного обеспечения.

Компонент сопоставления данных (далее – ПО DQF: Federation) обеспечивает сопоставление данных объектов (физических, юридических и иных объектов) из разных источников.

Компоненты ПО DQF могут работать как комплексно, так и независимо друг от друга.

3. Установка и настройка ПО DQF

3.1. Установка и настройка ПО DQF: Engine

3.1.1. Подготовка ИТ–инфраструктуры

Для установки ПО DQF: Engine ИТ-инфраструктура в части аппаратной платформы и сетевой инфраструктуры должна быть подготовлена в соответствии с требованиями, приведенными в данном разделе.

3.1.1.1. Требования к аппаратной платформе

Для установки ПО DQF: Engine необходимо обеспечить настройку сервера приложения в следующей конфигурации:

- Процессор:
  - Intel Xeon Ice Lake или новее (Intel Xeon Silver 4310 и более старшие версии) или AMD EPYC второго поколения или новее (AMD EPYC 7502P и более старшие версии);
  - от 4 ядер.
- ОЗУ:
  - от 4 Гб.

3.1.1.2. Требования к сетевой инфраструктуре

Пропускная способность канала между сервером приложений и внешними информационными системами, направляющими запросы в ПО DQF: Engine, должна составлять не менее 1 Гбит/с.

Пропускная способность канала между сервером приложений и брокером сообщений должна составлять не менее 1 Гбит/с.

3.1.2. Установка ПО DQF: Engine

Правила и порядок установки ПО DQF: Engine приведены в документе «Руководство по развертыванию и установке программного обеспечения Data Quality Framework». Для перезапуска ПО DQF: Engine используется порядок действий, приведенный в разделе 2.1.6 документа «Руководство по развертыванию и установке программного обеспечения Data Quality Framework».

3.1.3. Обеспечение отказоустойчивости ПО DQF: Engine

Отказоустойчивая схема развертывания ПО DQF: Engine подразумевает установку двух экземпляров программного обеспечения, которые работают одновременно.

Поступающие запросы от внешних информационных систем в брокер очередей балансируются между экземплярами ПО DQF: Engine, а при отказе одного из экземпляров программного обеспечения все новые запросы направляются на второй. При восстановлении работоспособности первого экземпляра ПО DQF: Engine балансировщик распределяет нагрузку между обоими экземплярами.

Для реализации отказоустойчивой схемы требуется:

– приобретение дополнительной лицензии на установку экземпляра ПО DQF: Engine;

– два сервера приложений;

– установка и настройка балансировщик нагрузки - HAproxy версии 2.0.

3.1.4. Логирование в ПО DQF

Для получения логов при использовании ПО DQF: Engine в консоли сервера необходимо ввести команду: docker logs dqf-engine-standalone, где dqf-engine-standalone, является названием контейнера приложения.

Логирование ведется в стандартный поток ввода-вывода, просмотр которого доступен из запущенного контейнера приложения.

Пример журнала логов ПО DQF: Engine :

```
2022-09-09 19:18:25.326 DEBUG 1 --- [ntContainer#0-1]
r.c.e.e.algorithm.spi.AlgorithmFactory   : Опрашиваю, поддерживает ли фабрика
алгоритм: unknown_algorithm_id
2022-09-09 19:18:25.327 ERROR 1 --- [ntContainer#0-1]
c.d.e.s.c.ComplexAlgorithmFileRepository : Ошибка поиска алгоритма
unknown_algorithm_id при чтении директории /opt/dqf/engine/complex

java.nio.file.NoSuchFileException:
/opt/dqf/engine/complex/unknown_algorithm_id.ca.yaml
```

3.1.5. Управление файлами конфигураций комплексных алгоритмов

Администратору ПО DQF: Engine доступно управление файлами конфигураций комплексных алгоритмов ПО DQF: Engine, а именно:

– загрузка файлов;

– редактирование файлов;

– удаление файлов.

Управление файлами конфигураций комплексных алгоритмов осуществляется в директории, все вносимые изменения (загрузка/редактирование/удаление комплексных алгоритмов) отображаются в директории и в ПО DQF: Engine.

Для загрузки новых файлов конфигураций комплексных алгоритмов администратору ПО DQF: Engine необходимо скопировать их в директорию, указанную

при установке ПО DQF: Engine. Правила формирования настроек при установке ПО DQF: Engine приведены в документе «Руководство по развертыванию и установке программного обеспечения Data Quality Framework». Например, если смонтированная директория при установке ПО DQF: Engine была /etc/configurations и в данной директории доступны два файла: check_balance.ca.yaml, check_limits.ca.yaml, - соответственно в ПО DQF: Engine будут отображаться загруженные конфигурации комплексных алгоритмов check_balance и check_limits, определенные в соответствующих файлах. Так, при размещении нового файла конфигурации в директории, новый комплексный алгоритм будет доступен в ПО DQF: Engine.

Параметры файлов конфигураций комплексных алгоритмов следующие:

– формат файла – Yaml 1.2.2;
– кодировка - UTF-8;
– расширение - .ca.yaml.

3.1.6. Формирование сообщений-запросов и получение сообщений-ответов с использованием брокера очередей

Ниже приведено описание формата сообщения-запроса (для ПО DQF: Engine – входящего сообщения). Формат заголовка сообщения приведен в таблице (Таблица 1), формат тела сообщения приведен в таблице (Таблица 2). Тело является JSON-объектов в кодировке UTF-8.

Таблица 1 - Формат заголовка сообщения-запроса (для ПО DQF: Engine - входящего сообщения)

Название	Обязателен	Тип	Пример	Описание
message_id	нет	Строка	1	Используется для журналирования.
correlation_id	нет	Строка	1	Если передан в сообщении-запросе - будет возвращен в сообщении-ответе.

Таблица 2 - Формат тела сообщения-запроса (для ПО DQF: Engine - входящего сообщения)

Название	Обязателен	Тип	Пример(ы)	Описание
algorithmId	да	Строка	my_complex_algo	Идентификатор комплексного алгоритма для выполнения проверки.

Название	Обязателен	Тип	Пример(ы)	Описание
objects	да	Любой валидный JSON, не являющийся null	1. {"attr": "value"} 2. ["arrayValue"] 3. 42 4. 3.14 5. true 6. false	Объекты для проверки. Будут переданы в контекст исполнения.

Пример сообщения-запроса (входящего сообщения для ПО DQF):

```
{
  "algorithmId": "check_balance",
  "objects": {
    "balance": 42
  }
}
```

Ниже приведено описание формата сообщения-ответа (для ПО DQF: Engine – исходящего сообщения). Формат заголовка сообщения приведен в таблице (Таблица 3**Таблица *1***), формат тела сообщения приведен в таблице (Таблица 4**Таблица *2***). Тело является JSON-объектов в кодировке UTF-8.

Таблица 3 - Формат заголовка сообщения-ответа (для ПО DQF: Engine - исходящего сообщения)

Название	Обязателен	Тип	Описание
message_id	да	Строка	Идентификатор сообщения-ответа. Будет случайно сгенерирован перед отправкой.
correlation_id	нет	Строка	Будет равен соответствующему заголовку во сообщении-запросе.
content_encoding	да	Строка	Кодировка сообщения-ответа. Всегда UTF-8
content_type	да	Строка	Тип тела сообщения-ответа. Всегда application/json

Таблица 4 - Формат тела сообщения-ответа (для ПО DQF: Engine - исходящего сообщения)

Название	Обязателен	Тип	Описание
correlationId	да, может быть null	Строка	Идентификатор корреляции. Равен значению заголовка correlation_id, сообщения-запроса, если был передан, иначе null.
status	да	Логический литерал	Статус выполнения комплексного алгоритма. Значение и его

Название	Обязателен	Тип	Описание
			интерпретация зависит от конкретного комплексного алгоритма.
variables	да	json-объект	Переменные, которые были заполнены в рамках выполнения комплексного алгоритма. Значение и интерпретация зависят от конкретного комплексного алгоритма.

Пример сообщения-ответа (исходящего сообщения для ПО DQF: Engine):

```
{
  "correlationId": null,
  "status": true,
  "variables": {
    "limit": 50000,
    "limitCurrency": "RUB"
  }
}
```

При возникновении проблем при обработке запроса (например, был передан неизвестный идентификатор комплексного алгоритма или в файле-конфигурации комплексного алгоритма была допущена ошибка) в очередь для ответов будет отправлено сообщение с информацией об ошибке. Формат заголовка сообщения об ошибке приведен в таблице (Таблица 5**Таблица *1***), формат тела сообщения об ошибке приведен в таблице (Таблица 6**Таблица *2***). Тело является JSON-объектов в кодировке UTF-8.

Таблица 5 - Формат заголовка сообщения об ошибке

Название	Обязателен	Тип	Описание
message_id	да	Строка	Идентификатор сообщения-ответа. Будет случайно сгенерирован перед отправкой.
correlation_id	нет	Строка	Будет равен соответствующему заголовку в сообщении-запросе.
content_encoding	да	Строка	Кодировка сообщения-ответа. Всегда UTF-8
content_type	да	Строка	Тип тела сообщения-запроса. Всегда application/json

Таблица 6 - Формат тела сообщения об ошибке

Название	Обязателен	Тип	Описание
correlationId	да, может быть null	Строка	Идентификатор корреляции. Равен значению заголовка correlation_id, сообщения-запроса, если был передан, иначе null.

Название	Обязателен	Тип	Описание
exception	да	json-объект	Информация об ошибке с сообщением и трейсингом для поиска места возникновения.

Пример сообщения об ошибке (приведена только часть сообщения для сокращения объема документа (приведен первый и последний элемент массива stackTrace)):


```
{
  "correlationId": null,
  "exception": {
    "message": "Ошибка в позиции (3:17@37 - 3:41@61): Пропущена закрывающая
кавычка: `'`\n| '$.objects.balance > 0\n> ^----------------------^\n",
    "localizedMessage": "Ошибка в позиции (3:17@37 - 3:41@61): Пропущена
закрывающая кавычка: `'`\n| '$.objects.balance > 0\n> ^----------------------
^\n"
    "suppressed": [],
    "cause": {
      "cause": null,
      "stackTrace": [
         {
           "classLoaderName": null,
           "moduleName":null,
           "moduleVersion":null,
           "methodName":"parse",
           "fileName":"StringTextExpressionParser.java",
           "lineNumber":92,

"className":"ru.cleandata.execution.engine.impl.expression.text.parser",
           "nativeMethod":false
         },
         {
           "classLoaderName": null,
           "moduleName":java.base,
           "moduleVersion":"11.0.15",
           "methodName":"run",
           "fileName":"Thread.java",
           "lineNumber":829,
           "className":"java.lang.Thread",
           "nativeMethod":false
         }
       ],
       "configLocation":{
         "start":{
           "lineNumber":3,
           "columnNumber":17,
           "streamOffset":37
         },
         "end":{
           "lineNumber":3,
           "columnNumber":41,
           "streamOffset":61
         }
       }
    }
  }
}
```

3.2. Установка и настройка ПО DQF: Federation

3.2.1. Подготовка ИТ–инфраструктуры

Для установки ПО DQF: Federation ИТ-инфраструктура в части аппаратной платформы и сетевой инфраструктуры должна быть подготовлена в соответствии с требованиями, приведенными в данном разделе.

3.2.1.1. Требования к аппаратной платформе

Для установки ПО DQF: Federation необходимо обеспечить настройку сервера приложения в следующей конфигурации:

3.2.1.1.1. Конфигурация сервиса Узла

Процессор:

– Intel Xeon Ice Lake или новее (Intel Xeon Silver 4310 и более старшие версии) или AMD EPYC второго поколения или новее (AMD EPYC 7502P и более старшие версии);

– от 4 ядер.

ОЗУ:

– от 4 Гб.

Дисковое пространство:

– от 5 Гб.

3.2.1.1.2. Конфигурация БД Узла

Процессор:

– от 8 ядер.

ОЗУ:

– от 16 Гб.

Дисковое пространство:

– от 400 Гб.

3.2.1.1.3. Конфигурация сервиса Центра

Процессор:

– Intel Xeon Ice Lake или новее (Intel Xeon Silver 4310 и более старшие версии) или AMD EPYC второго поколения или новее (AMD EPYC 7502P и более старшие версии);

– от 8 ядер.

ОЗУ:

– от 8 Гб.

Дисковое пространство:

– от 5 Гб.

3.2.1.1.4. Конфигурация БД Центра

Процессор:

– от 16 ядер.

ОЗУ:

– от 32 Гб.

Дисковое пространство:

– от 200 Гб.

3.2.1.1.5. Конфигурация БД кэша Центра

Процессор:

– от 4 ядер.

ОЗУ:

– от 16 Гб.

Дисковое пространство:

– от 5 Гб.

3.2.1.2. Требования к сетевой инфраструктуре

Пропускная способность канала между сервером приложений и внешними информационными системами, направляющими запросы в ПО DQF: Federation, должна составлять не менее 1 Гбит/с.

Пропускная способность канала между сервером приложений и брокером сообщений должна составлять не менее 1 Гбит/с.

3.2.2. Установка ПО DQF: Federation

Правила и порядок установки ПО DQF: Federation приведены в документе «Руководство по развертыванию и установке программного обеспечения Data Quality Framework». Для перезапуска ПО DQF: Federation используется порядок действий, приведенный в разделе 2.2.6 документа «Руководство по развертыванию и установке программного обеспечения Data Quality Framework».

3.2.3. Обеспечение отказоустойчивости ПО DQF: Federation

Отказоустойчивая схема развертывания ПО DQF: Federation подразумевает установку двух экземпляров сервисов программного обеспечения (сервис Центра и сервис Узлов), которые работают одновременно. Поступающие запросы от внешних информационных систем в брокер очередей балансируются между экземплярами ПО DQF: Federation, а при отказе одного экземпляров программного обеспечения все новые запросы считываются вторым. При восстановлении работоспособности первого экземпляра ПО DQF: Federation считывание сообщений из брокера автоматически сбалансируется.

Для реализации отказоустойчивой схемы требуется:

– приобретение дополнительной лицензии на установку экземпляра ПО DQF: Federation;

– два сервера приложений;

3.2.4. Логирование в ПО DQF: Federation

Для получения логов при использовании ПО DQF: Federation в консоли сервера необходимо ввести команду: docker logs container-name, где container-name является названием контейнера одного из сервисов ПО DQF: Federation (Узел, Центр).

Логирование ведется в стандартный поток ввода-вывода, просмотр которого доступен из запущенного контейнера приложения.

Пример журнала логов ПО DQF: Federation:


```
{"@timestamp":"2024-04-
12T14:15:02.064Z","log.level":"DEBUG","message":"Обрабатываю агрегированное
обновление:
AggregatedUpdate(card=520e9a712ef7c69b3ff436eb6db5349e\u0001esia\u000120\u000
10000000000000001, nodeIds=[node-0], fragments=[ФИО/ИМЯ/0, ФИО/ОТЧЕСТВО/0,
ДР/ДАТА_РОЖДЕНИЯ/0, СНИЛС/0, ФИО/ФАМИЛИЯ/0], deleted=false)","ecs.version":
"1.2.0","service.name":"fmdm-center","service.version":"2024.02.19-
9e3d43fb","service.node.name":"fmdm-center-instance-0","event.dataset":"fmdm-
center","process.thread.name":"centerUpdate-0-C-
1","log.logger":"ru.cleandata.fmdm.center.handler.AggregatedUpdateHandlerAggr
egatingUntilFilterPassed","traceId":"1945ea08bc6cdcef4159b295f2fa843a","spanI
d":"f5cf6d80f27a75a1"}
```


3.2.5. Формирование сообщений-запросов с использованием брокера очередей

Ниже приведено описание формата сообщения-запроса на сопоставление (для ПО DQF: Federation – входящего сообщения). Заголовочная часть сообщения на сопоставление соответствует спецификации https://www.w3.org/TR/trace-context-1/, формат тела

сообщения приведен в таблице (Таблица 7). Тело является JSON-объектом в кодировке UTF-8.

Таблица 7 - Формат тела сообщения-запроса на сопоставление (для ПО DQF: Federation - входящего сообщения)

Название	Обязателен	Тип	Описание
sourceId	да	String	Идентификатор источника.
cardId	да	String	Идентификатор карточки в источнике.
version	да	Number	Версия обновления – монотонно возрастающая последовательность для истории изменений одной и той же карточки.
fragments	нет	Object	Ассоциативный массив: – имя фрагмента; – список его значений, упорядоченных в соответствии с индексами значений. Должен быть пуст, если карточка удалена.
deleted	да	Boolean	Флаг, показывающий, удалилась ли карточка.

Пример сообщения-запроса (входящего сообщения для ПО DQF):

```
{
  "datamartId": "витрина_1",
  "cardId": "карточка_1",
  "version": 1,
  "fragments": {
    "Свойство-Атрибут/Компонент-Атрибут/0": [
      "Значение-Атрибут"
    ],
    " Свойство-Реквизит/Компонент-Реквизит/0": [
      "Значение-Реквизит"
    ]
  },
  "deleted": false
}
```

При возникновении проблем в контейнере сервиса Узла (например, при передаче не подходящего для данного Узла фрагмента) будет залоггирована ошибка.

Проверка, что запрос обработан ПО DQF: Federation, осуществляется с помощью sql-запроса в БД Центра ниже:

```
TARGET_CARDS as (
                    select
                        ID as CARD_ID
                    from
                        CARD C
                    where 1=1
                      and (C.SOURCE_ID, C.SOURCE_CARD_ID) in (
                        select * from unnest(ARRAY['витрина_1'],
ARRAY['карточка_1']) as X(SOURCE_ID, SOURCE_CARD_ID)
                    )
                      and not C.DELETED
                      and not C.DEPRECATED
                ),
                INCLUDING_CONSOLIDATIONS as (
                    select CONSOLIDATION_ID
                    from
                        CONSOLIDATION_CARD CC
                            inner join CONSOLIDATION C
                                       on CC.CONSOLIDATION_ID = C.ID
                    where 1=1
                      and CC.CARD_ID in (select CARD_ID from TARGET_CARDS)
                      and C.TOP_LEVEL
                )
            select
                CC.CONSOLIDATION_ID as GROUP_ID,
                array_agg(C.SOURCE_ID) as SOURCE_IDS,
                array_agg(C.SOURCE_CARD_ID) as SOURCE_CARD_IDS
            from
                CONSOLIDATION_CARD CC
                    inner join CARD C
                               on CC.CARD_ID = C.ID
            where 1=1
              and CC.CONSOLIDATION_ID in (select CONSOLIDATION_ID from
INCLUDING_CONSOLIDATIONS)
              and (1<>1
                or C.ID in (select CARD_ID from TARGET_CARDS)
                or case when array_length(ARRAY['витрина_2'], 1)>0
                       then C.SOURCE_ID in (select
unnest(ARRAY['витрина_2']))
                       else true
                    end
                )
            group by
                CC.CONSOLIDATION_ID
```