Документация

Выпуск 3.4.1.12

XDAC - Data Access Connector

апр. 25, 2024

Содержание

ГЛАВА 1

Общие сведения

При подключении к базе данных приложение сканирует набор процедур и функций, которые расположены в схеме, указанной в настроечном параметре “parsingSchema”. Результат сканирования записывается в кэш приложения, который далее используется для сопоставления запрашиваемых процедур и параметров переданных в запросе.

Каждая процедура и функция, которая была получена в результате сканирования, публикуется как endpoint приложения xdac. Доступ осуществляется по наименованию процедуры или функции, который указывается в пути вызова сервиса. Параметры передаваемые в процедуру передаются в формате json, и сопоставляется по наименованию. Результат работы функции преобразуется в скалярное значение json и отдаётся конечному потребителю.

ГЛАВА 2

Архитектура и системные требования

2.1 Архитектура

Архитектура приложения представляет собой веб-сервер, который обрабатывает клиентские HTTP-запросы посредством формирования отчетов на основе документов-шаблонов и возвращает результат клиенту.

2.2 Среда исполнения

Поддерживаемые архитектуры:

- x86-64
- ARM (в том числе байкал)
- e2k Эльбрус

Сертифицировано со следующими отечественными ОС (https://xsquare.ru/o-nas/):

- Astra Linux
- RED OS
- Alt Linux
- ROSA

Поддерживаемые ОС:

- Ubuntu 20+
- Red Hat 8+
- Debian 10+

Поддерживаемые СУБД:

- PostgreSQL

- PostgresPRO
- Tantor
- Jatoba
- Pangolin

2.3 Системные требования

CPU - 1 Ядро RAM - 100 Мб HDD - 100 Мб + Логи

Установка системы виртуализации/контейнеризации, операционной системы, базы данных осуществляется на усмотрение Администратора исходя из потребностей Организации.

ГЛАВА 3

XDAC - Data Access Connector

3.1 Проблематика

Разработка API для приложения часто требует от разработчика дополнительных навыков, таких как понимание работы протокола http, знание дополнительного языка программирования и инструментария, который этот язык предоставляет, а также умения администрирования (установка и мониторинг) готового API.

Зачастую вся логика и данные, которые необходимо опубликовать через API уже существуют в том или ином виде на стороне базы данных. В этом случае разработчик вынужден разрабатывать точки доступа для этих данных, при этом процесс разработки сводится к рутинному и трудозатратному процессу написанию однотипного кода.

3.2 Решение

XDAC позволяет автоматизировать процесс публикации API, предоставляя доступ к процедурам и функциям хранимых на стороне Postgresql. Используя данное решение разработчику базы данных больше не потребуется писать отдельное приложение.

ГЛАВА 4

Вызов процедур и функций

Каждая процедура и функция, полученная в результате сканирования схемы, публикуется сервисом с префиксом `/rpc/`

```
curl "http://localhost:3000/xdac/rpc/function_name" -X POST
```

Для передачи параметров необходимо включить в тело запроса json, наименование полей которого будет соответствовать наименованию аргументов функции. Для примера создадим функцию следующего вида:

```
CREATE OR REPLACE FUNCTION api.add_numbers (
    first integer,
    second integer
)
RETURNS integer AS
$body$
BEGIN
    return first + second;
END;
$body$
LANGUAGE 'plpgsql'
```

Для вызова функции необходимо передать запрос следующего вида:

```
curl -d '{"first":10,"second":30}' 'http://localhost:3000/xdac/rpc/add_numbers'
```

```
40
```

Каждый вызов процедуры или функции осуществляется в своей транзакции. Наличие оператора `COMMIT` в теле функции приведет к ошибке.

4.1 Правила наименования

Все наименования (имена процедур и функций, наименования аргументов) - регистрозависимые.

Для вызова функции с именем `StRaNgeNamE` необходимо вызвать метод следующим образом:

```
curl "http://localhost:3000/xdac/rpc/StRaNgeNamE"
```

Вызов в другом формате приведет к ошибке:

```
{
    "code": "XDAC001",
    "hint": "Make sure that routine name is correct or try to reload applicaion",
    "message": "Routine strangename was not found in app cache",
    "details": ""
}
```

4.2 Вызов процедуры с JSON аргументом

Если необходимо передать json внутрь функции или процедуры, необходимо передать заголовки `Content-Type: application-jsonи Prefer:params=single-object`

```
curl -H «Content-Type: application/json» -H «Prefer: params=single-object» -d „
↪{«text»:»Test message»,»input»:[1,2,3]}“ «http://localhost:3000/xdac/rpc/jsonInput»
```

Сама процедура должна иметь только один аргумент с типом данных **json** или **jsonb**:

```
CREATE OR REPLACE FUNCTION api."jsonInput" (
    "myPar" json
)
RETURNS json AS
$body$
begin
return "myPar";
END;
$body$
LANGUAGE 'plpgsql'
```

4.3 Загрузка бинарного файла

Для того чтобы передать бинарный файл в процедуру или функцию необходимо, чтобы процедура имела один аргумент с типом bytea, а запрос содержал заголовок **“Content-Type: application/octet-stream”**

```
curl "http://localhost:3000/xdac/rpc/upload_binary" \
    -X POST -H "Content-Type: application/octet-stream" \
    --data-binary "@file_name.ext"
```

```
CREATE OR REPLACE PROCEDURE api."upload_binary" (
    bytea
)
AS
```

(continues on next page)

(продолжение с предыдущей страницы)

```
$body$
BEGIN
    insert into api.files(blob)
    values ($1);
END;
$body$
LANGUAGE 'plpgsql'
```

ГЛАВА 5

Работа с заголовками

Все заголовки передаваемые в сервис записываются в GUC параметры сессии postgresql. Следующие параметры доступны для доступа `request.headers`, `request.method` and `request.path`

```
-- Получить все заголовки. Они преобразуются в json
SELECT current_setting('request.headers', true)::json;

-- Метод с помощью которого был вызван сервис
SELECT current_setting('request.method', true); --GET/POST/PUT....

-- Путь по которому был вызван сервис
SELECT current_setting('request.path', true); --/xdac/rpc/test
```

Для того чтобы установить заголовки ответа и код ответа используются GUC `response.headers` и `response.status`

```
select set_config('response.headers','[{"Content-Type":"application/json"},{"Set-
↪Cookie":"foo=bar"}]',true);
--необходимо передать массив объектов json
select set_config('response.status','405',true); -- устанавливает код ответа в 405
```

ГЛАВА 6

Обработка ошибок

В случае возникновения ошибки сервис вернёт её в формате:

```
{
  "code": "",
  "hint": "",
  "message": "",
  "details": ""
}
```

Сервис предоставляет возможность обрабатывать пользовательские ошибки вызванные оператором raise

```
{
  "code": "SRV01",
  "hint": "Try GET method",
  "message": "This is postgresql error",
  "details": "This is detail"
}
```

При этом транзакция будет завершена и откачена с помощью оператора `ROLLBACK`.

Если необходимо передать пользовательскую ошибку в сервис, при этом сохранить результаты выполнения функции, можно использовать установку статуса ответа через GUC

```
perform set_config('response.status','405',true);
perform set_config('response.headers','[{"Content-Type":"application/json"}]',true);
return json_build_object('message','Only GET method is allowed');
```

ГЛАВА 7

Управление кэшом приложения

Все процедуры и функции доступные схеме сканируются при запуске приложения и помещаются в кэш. При изменении сигнатуры процедур или создании новых функций они не будут доступны для вызова через API. Для этого необходимо перезагрузить приложение, для того чтобы кэш обновился.

Для автоматизации перезагрузки кэша приложения, XDAC предоставляет возможность перезагрузить кэш с помощью механизма уведомлений `NOTIFY`.

Пользователь может отправить уведомление в канал xdac с уведомлением о перезагрузке кэша:

```
NOTIFY xdac, 'reload';
```

7.1 Автоматизация уведомлений

Следующий способ автоматизировать перезагрузку кэша - создание event trigger на операции ddl. Пользователь с правами SUPERUSER может создать следующий триггер:

```
CREATE OR REPLACE FUNCTION public.xdac_watch() RETURNS event_trigger
LANGUAGE plpgsql
AS $$
 BEGIN
   NOTIFY xdac, 'reload';
 END;
$$;

CREATE EVENT TRIGGER xdac_watch  ON ddl_command_end  EXECUTE PROCEDURE public.xdac_
↪watch();
```

При создании нового объекта базы данных, приложение автоматически получит уведомление о необходимости перезагрузить кэш.

ГЛАВА 8

Функциональные требования

XSQUARE-XDAC – это автономный веб сервер, который позволяет автоматизировать процесс создания REST API, предоставляя доступ к хранимым процедурам и функциям на стороне PostgreSQL.

Основные функциональные характеристики и возможности XSQUARE-XDAC:

- Автоматическое создание конечных точек подключения (API) на основе хранимых процедур и функций БД.
- Автоматическое преобразование сигнатур процедур и функций в параметры запросов к API.
- Возможность использования нескольких методов для работы с созданным API (GET/POST).
- Возможность настраивать доступ к конечным точкам, используя настройки прав пользователя БД.
- Предоставление доступа к дополнительным параметрам http запроса (заголовки,код ответа и пр.) внтури вызываемых процедур и фунций.
- Возможность обрабатывать запросы типа multipart/form-data.
- Загрузка и выгрузка бинарных данных.
- Обслуживание и балансировка HTTP запросов между Веб клиентом и базой данных.
- Поддержка пула соединений с базой данных.

ГЛАВА 9

Жизненный цикл

9.1 Общие сведения

Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения XSQUARE-XDAC, в том числе устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения, совершенствование программного обеспечения, а также информация о персонале, необходимом для такой поддержки.

Программа	XSQUARE-XDAC
Разработчик	ООО «Хи-Квадрат»
Пользователь	Юридическое лицо, использующее Программу согласно договора с Разработчиком
Сайт	https://lcdp.xsquare.ru

9.2 Поддержание жизненного цикла Программы

1. Жизненный цикл Программы включает в себя следующие этапы:
2. Проектирование и разработка Программы, осуществляемые Разработчиком;
3. Тестирование и выявление неисправностей в работе Программы Разработчиком;
4. Установка, использование и обновление Программы Пользователем согласно лицензионному соглашению с Разработчиком;
5. Модернизация программы Разработчиком согласно собственному плану доработок и улучшений, а также по заявкам Пользователя;
6. Осуществление технической поддержки Пользователя Разработчиком по вопросам установки, интеграции и эксплуатации Программы;
7. Выпуск Разработчиком обновленных сборок модернизированной Программы.

8. Разработчик регулирует проведение всех этапов жизненного цикла программы за исключением процессов установки, интеграции и использования Программы Пользователем.

9.3 Устранение неисправностей, выявленных в ходе эксплуатации Программы

Неисправности, выявленные в ходе эксплуатации Программы могут быть устранены следующими способами:

1. Внесение исправлений в код Программы Разработчиком согласно своей дорожной карте разработки Программы;
2. Внесение исправлений в код Программы на основе обращения Пользователя;

Пользователь может сформировать следующие запросы:

- Отчёт об инциденте с приложением информации об условиях возникновения бага с использованием графической информации, лог-файлов, информации о программном окружении и номерах версий используемого программного обеспечения, включая версию и редакцию Программы. Запрос также должен содержать информацию об ожидаемом и фактическом поведении Программы и любую другую информацию, которая поможет диагностировать и устранить неисправность Программы Разработчиком;
- Запрос на улучшение Программы в целях изменения её поведения для достижения нужных результатов в решениях Пользователя;
- Запрос на предоставление дополнительной информации о функционировании и возможностях Программы.

Запросы могут быть отправлены Пользователем только с помощью трекинговой системы Разработчика - https://tracker.xsquare.ru. Доступ к трекенговой системе предоставляется по факту приобретения Программы.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваиваится уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о новом функционале Программы, либо о добавлении задачи по развитию в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для устранения неисправностей в работе Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы.

9.4 Совершенствование Программы

Программа непрерывно улучшается и модернизируется, выпускаются регулярные обновления, публикуются информационные материалы на Сайте Программы, Пользователи информируются об изменениях в Программе.

Пользователь может инициировать запрос на изменение или улучшение работы Программы с помощью формирования запроса к Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о внесенных изменениях в код Программы, либо о добавлении задачи по модернизации в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для улучшения работы Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы

9.5 Техническая поддержка Программы

Техническая поддержка Программы осуществляется с помощью формирования запросов Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Техническая поддержка Пользователя включает в себя:

- Помощь в установке Программы;
- Помощь в установке базовых общесистемных компонентов (операционной системы, HTTP Сервер, сервер баз данных и)
- Помощь в интеграции Программы в существующие решения Пользователя;
- Помощь в устранении неисправностей, возникающих в работе Программы;
- Консультации по функционированию Программы;
- Сбор информации о некорректной работе Программы для последующего выпуска модернизации Программы согласно плану доработок;
- Информирование Пользователя об обновлениях Программы

9.6 Информация о персонале, необходимом для обеспечения поддержки

Персонал, который будет осуществлять поддержку Программы со стороны Пользователя, должен обладать:

1. Базовыми навыками администрирование операционных систем семейства Unix
2. Базовыми навыками работы с офисными пакетами
3. Пользователи Программы должны обладать навыками работы с персональным компьютером и веб браузером.
4. В случае возникновения вопросов у персонала, им следует обратиться к Разработчику за получением технической поддержки

ГЛАВА 10

Эксплуатация

Данный раздел описывает порядок поддержания работоспособности приложения и порядок загрузки компонентов.

10.1 XDAC

Для загрузки xsquare.xdac пользователю необходимо убедиться в наличие правильно настроенного конфигурационного файла.

```
cat /usr/local/xsquare.xdac/config.json
```

Команда должна отобразить правильный конфигурационный файл.

Далее необходимо запустить выполнив команду:

```
systemctl start xsquare.xdac
```

После проверить состояние сервера приложений:

```
systemctl status xsquare.xdac
```

В случае возникновения ошибок они будут записаны в журнал. Проверить сообщения об ошибках можно выполнив команду:

```
journalctl -u xsquare.xdac
```