

Документация

REST API

artamonov.rest

Версия 4.19.0 от 13.10.2026

Содержание

Страниц: 73 · Слов: 8 218 · Время чтения: ≈ 42 мин

Обзор модуля	3
Обновления	8
Обновления	
Обновление: 4.19.0	24
Проверка работы	30
Начало работы	
Первые шаги	31
Схема работы	32
Многосайтовость	34
Карты роутов	35
Функции и методы	
Список функций	44
Журналы	49
Кэширование	50
Контроллеры	51
Токены авторизации	58
REST-документация	60
Примеры запросов	
Javascript-запрос	62
PHP-запрос	64
Роут-провайдер	66
Используемые ресурсы	73

Обзор модуля

Название	Описание
Наименование	REST API
Код модуля	artamonov.rest
Дата релиза	06.08.2018
Дата обновления	13.10.2026
Версия	4.19.0

Модуль помогает организовать программный интерфейс для внешних и внутренних приложений.

Ранее нами уже был разработан и опубликован на **бесплатной** основе модуль **REST API**. Данный модуль хорошо себя зарекомендовал для решения задач межплатформенных обменов, показателями этого факта являются отзывы и количество скачиваний. Со временем стало появляться множество запросов по доработке модуля, и как следствие, мы приняли решение о написании **новой, усовершенствованной** версии уже любимившегося модуля.

В данной версии мы **учли все недостатки и пожелания** из прошлой версии, очень хорошо оптимизировали ядро модуля, теперь оно **работает гораздо быстрее**, и на этом мы делаем свой основной упор.

Помимо переработки ядра немаловажным фактором является и добавление карты роутов. За счет карты появилась **возможность очень гибко настраивать** параметры для любого отдельного роута.

Что получите от приобретения модуля

- **Уменьшение стоимости** разработки API за счет **сокращения времени** на разработку собственного интерфейса.
- Поддержка от профессионалов, имеющих большой опыт создания API.
- Готовый RESTful API интерфейс, с библиотекой встроенных методов, а также с возможностью создавать собственные, кастомные методы, реализующие любую, необходимую логику. И конечно, с автогенерируемой документацией, которую можно предоставить сторонним разработчикам.

Процесс

1. Ознакомьтесь с документацией по модулю.
2. Ознакомьтесь с библиотекой встроенных роутов.
3. Добавьте собственную карту роутов со списком необходимых роутов.
Примечание: карты можно хранить в любой папке сайта.
4. Укажите контроллеры для роутов, которые будут обрабатывать входящие запросы. Контроллеры могут быть, как классами, так и обычными файлами (скриптами). Примечание: контроллеры можно хранить в любой папке сайта.
5. При возникновении вопросов по модулю обратитесь в техподдержку через контакты.

Возможности

- Загрузка интерфейса до запуска публичной части сайта.
- Отсутствие необходимости в дополнительной настройке платформы.
- Виртуальный путь к интерфейсу, в том числе и от корня домена (<https://domain.com/> или <https://domain.com/{api}/> или <https://api.domain.com/>).
- Настройка доступа в административной части сайта к разделам модуля на уровне групп пользователей.
- Автогенерируемая документация в административном разделе.
- Автогенерируемая документация для клиента в JSON-формате.
- Поддержка карт роутов.
- Создание произвольных роутов (`{api}/user/`, `{api}/user/get/`, `{api}/user/v1/`, ...).
- Установка собственных контроллеров для роутов (классы, внешние классы, php-файлы).
- Настройка уровня доступа к роутам.
- Валидация параметров входящего запроса.
- Авторизация по логину и паролю для конкретного роута.
- Авторизация по токену для конкретного роута.
- Генерация токенов для пользователей.
- Поддержка списка “белых” логинов/токенов для конкретного роута.

- Проверка групп пользователя при авторизации для конкретного роута.
- Возможность ограничения количества запросов в единицу времени на уровне групп пользователей.
- Возможность передачи дополнительных заголовков при ответе.
- Возможность деактивировать конкретный роут.
- Кэширование входящих запросов и ответов средствами платформы.
- Журналирование запросов и ответов.
- Настройка CORS из административной части сайта.
- Наличие функций упрощающих разработку интерфейса.
- Наличие библиотеки встроенных роутов ускоряющих разработку интерфейса.
- Наличие примеров и руководств для разработчика.

FAQ

Чем данный модуль отличается от REST API Битрикса?

- В REST API Битрикса **мало гибкости**, есть **необходимость использовать именно предоставленные роуты**. Данный модуль позволяет формировать **свои роуты**, писать **свою логику** при обработке запроса, то есть предоставляет **больше гибкости**.
- REST API Битрикса реализовано не совсем согласно архитектурному стилю REST.
- В REST API Битрикса, при обработке запроса порой происходят «лишние» действия, которые могут создавать дополнительную задержку при ответе клиенту, может это и не сильно критично, но API должно быть настолько быстрым, насколько это возможно.
- API не всегда должно быть доступно всем, иногда оно должно быть уникально (роуты, параметры, ...) для проекта, но за счет общего стиля REST API Битрикса, документация которого в общем доступе, эта уникальность нарушается и создает дополнительную лайзеку для атак на сайт. Используя данный модуль, можно указывать собственное имя роута, где будет работать API. Например: domain.com/rest/ или domain.com/api/, или domain.com/rest-api/, или api.domain.com, и так далее.
- В REST API Битрикса нет возможности как-то управлять интерфейсом из административной части сайта.

- В REST API Битрикса нет возможности регулировать доступы к роутам, ограничивать доступ по количеству запросов от клиента, журналировать запросы и ответы, и некоторые другие функции.
- На текущий момент, в REST API Битрикса, например, нет возможности создать заказ через API. Используя данный модуль вы можете реализовать этот функционал.
- В целом, данный модуль пишется и развивается на основе запросов пользователей, но REST API Битрикса пишется на усмотрение Битрикса. Плюс это или минус - решать вам.

Чем данный модуль отличается от бесплатной версии? [архив]

- Переписано ядро - увеличена скорость.
- Изменена архитектура модуля - увеличена скорость.
- Работа с роутами переведена на карты - увеличена скорость, больше гибкости в настройке и обработке каждого роута.
- Введены функции - увеличение скорости разработки. Пример функций: `route()`, `request()`, `response()` и прочие.
- Добавлена возможность не указывать имя роута для API. То есть, теперь API может запутаться относительно корня домена.
- Добавлена автогенерируемая документация, то есть теперь описание всех роутов можно увидеть прямо из административной или публичной части сайта.
- Добавлено кэширование. Теперь можно кэшировать запросы и ответы на них, тем самым увеличивая скорость ответа и снижая нагрузку на базу данных.
- Добавлена возможность ограничивать доступ к роуту в зависимости от количества запросов со стороны клиента.
- Автоматическое подключение модуля в файле `init.php`.

Примечание

- Минимальная версия PHP **>= 8.0** (согласно требованиям платформы). Использование более ранней версии возможно, но крайне не рекомендуется, ввиду возникновения ошибок.
- **Обязательно** наличие файла для обработки 404 ошибки (обычно имеется в корне сайта - `404.php`) со всеми "директивами" 1С-Битрикса.

- Если используется веб-сервер **Apache**: обязательно прописать `ErrorDocument 404 /404.php` в `.htaccess`, расположенный в корне сайта, или прямо в конфигурационном файле `Apache`.
- Если используется веб-сервер **Nginx**: обязательно прописать `error_page 404 = /404.php?$query_string;` в секцию `server` конфигурационного файла (обычно `nginx.conf`).
- Для работы **CORS**-запросов необходимо чтобы на веб-сервере были **разрешены** запросы типа **OPTIONS**.

Спасибо Вам за ознакомление с нашим модулем. Если у Вас имеются какие-либо вопросы/предложения, то мы всегда готовы обсудить их через [контакты](#).

Обновления

Релизы: 2026

4.19.0

28.06.2026

✓ Текущая версия

Тип	Описание
	SQL-выборки журнала запросов переписаны на штатное D7 ORM — устранён риск SQL-инъекции через поля фильтра и значение токена.
	IP клиента определяется только по <code>REMOTE_ADDR</code> — подделка заголовков <code>X-Forwarded-For</code> / <code>Client-IP</code> больше не влияет на лимиты запросов и журнал.
	Токены и уникальные коды теперь генерируются криптостойко (<code>random_int</code> / <code>random_bytes</code> вместо <code>rand</code> и <code>md5</code>); ранее выданные токены остаются рабочими.
	При удалении пользователя идентификатор приводится к целому числу.
	Демонстрационный контроллер больше не возвращает полный массив <code>\$_SERVER</code> .
	В заголовки ответа добавлена очистка переводов строки (защита от подмены HTTP-заголовков).
	Для нативного провайдера добавлен контроль доступа к вызываемым классам и методам. По умолчанию действует чёрный список опасных операций (произвольный SQL, файловые операции и заливка файлов, управление пользователями и группами, внешние HTTP-запросы, рассылки и т.п.) — обычные read-вызовы не затронуты. Дополнительно поддерживается режим белого списка: перекрыв роут своим классом, можно разрешить только нужные классы и методы.
	На роуты управления токеном (создание, получение, удаление) добавлена защита от перебора учётных данных — при превышении лимита неудачных попыток с одного IP возвращается ответ 429.
	При создании пользователя через API системные поля (группы, токен, служебные атрибуты) больше нельзя задать напрямую — защита от mass assignment и повышения привилегий. Набором полей можно управлять чёрным и белым списком, перекрыв роут своим классом.
	Административный AJAX генерации токенов теперь требует авторизации в админке, прав на раздел «Безопасность» и защищён от CSRF — массовая генерация токенов внешним запросом больше невозможна.
	Код поля токена из настроек проверяется как корректный идентификатор перед подстановкой в SQL — устранён риск инъекции через административную настройку.

[Подробнее о релизе](#)

4.18.2

04.02.2026

Тип	Описание
	Поправлен сброс значений параметров.

4.18.1

04.02.2026

Тип	Описание
	Исправлено получение значений параметров. Причина: после обновления платформы, названия параметров приводились к нижнему регистру.
	Обновлена ссылка на Документацию в меню модуля.
	Доработано описание параметра Отложенный запуск.

4.18.0

04.01.2026

Тип	Описание
	Техническое обновление.
	Улучшена поддержка версии PHP выше 8.1.

Релизы: 2024

4.17.0

30.06.2024

Тип	Описание
	Добавлена поддержка отключения передачи данных ответа при использовании <code>json()</code> . Например: <code>response()->json(null, 404)</code> ;
	Добавлена проверка существования пути при установке пути к собственным картам роутов.
	Добавлена возможность дополнять заголовок <code>Access-Control-Allow-Headers</code> . Смотрите раздел: Безопасность - Фильтры.

4.16.0

02.03.2024

Тип	Описание
	Улучшено определение кастомных заголовков <code>Content-Type</code> (<code>+json</code>) при выполнении запросов к модулю.

Релизы: 2023

4.15.0

18.11.2023

Тип	Описание
	Добавлена возможность указывать кастомные заголовки <code>Content-Type</code> (<code>+json</code>) при выполнении запросов к модулю. Например: <code>Content-Type: application/vnd.eda.picker.errors.v1+json</code> или <code>application/problem+json</code> , и так далее. Подробнее: https://www.iana.org/assignments/media-types/media-types.xhtml
	Добавлена возможность передачи собственного ID клиента в функцию <code>journal()->add()</code> .
	Добавлены функции-полифилы для корректной работы модуля на версии PHP меньше 8.0: <code>str_contains</code> , <code>str_starts_with</code> , <code>str_ends_with</code> .

4.14.0

22.07.2023

Тип	Описание
	Добавлена возможность указывать собственный заголовок <code>Content-Type</code> при ответе клиенту. Внимание: при проверке используется функция <code>str_contains</code> , которая доступна только с версии PHP 8.0.

4.13.2

31.05.2023

Тип	Описание
	Улучшена совместимость с PHP 8 и выше.

4.13.1

19.05.2023

Тип	Описание
	Обновлена ссылка на документацию к модулю.

4.13.0 05.05.2023

Тип	Описание
	Поправлена логика проверки входных параметров.

4.12.0 22.02.2023

Тип	Описание
	Улучшена совместимость с PHP 8 и выше.

4.11.0 13.01.2023

Тип	Описание
	Улучшена совместимость с PHP 8 и выше.

Релизы: 2022

4.10.0 30.08.2022

Тип	Описание
	На уровне ядра добавлен заголовок <code>Access-Control-Allow-Credentials</code> со значением <code>true</code> .
	Отключена проверка модуля безопасности на фишинг при редиректе на страницу документации из административного раздела.

4.9.0 02.08.2022

Тип	Описание

	Работа с журналом Входящие запросы - переведено на штатное ORM.
	Повышена производительность Монитора поддержки.
	Доработана вёрстка Монитора поддержки.
	Рефакторинг и оптимизация.

Релизы: 2021

4.8.1 19.07.2021

Тип	Описание
	Добавлена обратная совместимость для старых версий модуля, где еще отсутствует класс <code>Password</code> с методом <code>equals()</code> .
	Мелкие правки.

4.8.0 03.07.2021

Тип	Описание
	В новых версиях главного модуля, поле <code>PASSWORD</code> объявлено приватным. Поэтому, при авторизации по логину и паролю, могли наблюдаться проблемы.
	Незначительные улучшения.

4.7.0 02.06.2021

Тип	Описание
	Теперь учитывается формат даты и времени для не RU локализаций.
	Кодировка копирайта в заголовке ответа.
	Мелкие правки и рефакторинг.

Релизы: 2020

4.6.0

19.12.2020

Тип	Описание
	Добавлена возможность выбора сайтов, на которых необходимо запускать модуль.
	Улучшена работа модуля в режиме многосайтовости.
	Добавлено игнорирование всех запросов, которые выполняются на адрес <code>/bitrix/</code> .

4.5.0

03.12.2020

Тип	Описание
	Авторизация по логину и паролю. Теперь проверка пароля выполняется с помощью метода <code>Bitrix\Main\Security\Password::equals()</code> .
	В связи с устаревшим, начиная с версии PHP 7.2.0, функционалом механизма перегрузки функций (<code>mbstring.func_overload</code>), и отказом от использования его в платформе, строковые функции модуля были переведены в многобайтовый режим работы.
	Проблема при подключении модуля в файле <code>init.php</code> , в случае, если файл имелся в директории, но был абсолютно пустым.
	Теперь резервная копия файла <code>init.php</code> создаётся до установки и до удаления модуля. Ранее копия создавалась лишь при установке модуля.
	Незначительные технические изменения.

4.4.0

17.10.2020

Тип	Описание
	Авторизация по логину и паролю.
	Функция возврата ответа клиенту: <code>response()->json()</code> . Теперь в данную функцию можно передавать значения заголовков не только в виде строки, но и в виде массива. В котором могут быть указаны дополнительные параметры для заголовка (в соответствии с функцией <code>header(\$header, \$replace, \$http_response_code)</code>). Подробнее смотрите в роуте-примере (<code>example</code>).

	Класс <code>Artamonov\Rest\Foundation\Token</code> . Теперь можно генерировать токены пользователей с помощью данного класса, при любой необходимости. Класс поддерживает как генерацию токенов, так и другие действия. Генерация токенов возможна для списка пользователей, плюс поддерживаются параметры (<code>\$parameters</code>) от метода <code>UserTable::getList()</code> . Для удобства добавлена функция <code>token()</code> . Подробнее смотрите в документации.
	В разделе Безопасность, пошаговое создание и обновление токенов для пользователей. Полезная функция, если в базе сайта зарегистрировано большое количество пользователей.
	Незначительные технические изменения.

4.3.1 18.09.2020

Тип	Описание
	Техническое обновление.

4.3.0 11.09.2020

Тип	Описание
	Проблема возникающая после истечения демонстрационного периода.

4.2.0 06.09.2020

Тип	Описание
	Метод получения параметров модуля из базы. Теперь получаемые параметры кэшируются. Время кэширования зависит от настроек параметра <code>cache_flags</code> в файле <code>/bitrix/.settings.php</code> .
	Метод получения данных пользователя, если была активна авторизация для роута. Теперь метод умеет десериализовывать данные в пользовательских полях.
	Ключ карты роутов <code>example</code> был добавлен в роуте <code>{api}/docs</code> .
	Если в карте роутов отсутствует какой-либо контроллер (файл или класс), тогда в документации будет выведено сообщение об этом.

	Параметр для регулирования глобальной проверки сроков годности токенов. Внимание: по умолчанию данный параметр отключен. Подробнее в разделе Безопасность.
	Автогенерация токена при добавлении нового пользователя. Подробнее в разделе Безопасность.
	Сообщения возвращаемые клиенту при возникновении ошибок на этапе запуска интерфейса.

4.1.0

10.08.2020

Тип	Описание
	Проверка многоуровневых параметров при передаче запроса. Подробнее структуру карты можно посмотреть в документации к модулю.
	Раздел документации с картой роутов - теперь поддерживается отображение неограниченного количества уровней параметров роута.
	Поддержка разделителя параметров в документации карты роутов - для лучшего визуального представления списка параметров. Подробнее структуру карты можно посмотреть в примере карты роутов.
	При проверке параметра на <code>integer</code> , в опции <code>max_range</code> теперь используется константа <code>PHP_INT_MAX</code> .
	Теперь числа переданные в виде строки, будут автоматически преобразованы в нужный тип, при проверке входящих параметров. Поддерживаемые типы для преобразования: <code>integer</code> , <code>float</code> .

4.0.2

07.07.2020

Тип	Описание
	Верстка в разделе Документация - Карта роутов.

4.0.1

22.05.2020

Тип	Описание
	Проверка типа входящих параметров.

4.0.0

22.04.2020

Тип	Описание
	Настройка видимости меню модуля в общем меню админ. раздела.
	Нативный роут <code>{{api}}/provider</code> . Роут является провайдером к внутреннему api платформы 1С-Битрикс. Поддерживает <code>get</code> и <code>post</code> запросы. Подробнее про использование роута можно узнать в документации.

3.5.0

15.01.2020

Тип	Описание
	В <code>Helper</code> модуля добавлен метод <code>isBool</code> .
	Техническое обновление.

Релизы: 2019

3.4.0

26.10.2019

Тип	Описание
	Возможность использовать переменные в роутах. Теперь в карте роутов можно указывать шаблон строки с использованием переменных. Например: <code>/example/check/section/{{sectionId}}/{{productId}}/view/</code> . Значения переменных <code>sectionId</code> и <code>productId</code> можно будет получить так же через <code>request()->get()</code> . Примеры шаблонов строк можно посмотреть в документации по картам роутов.
	Константы отключающие сбор статистики, проверки агентов и другие.
	Мелкие правки по дизайну модуля в админ.разделе.
	Мелкие правки по коду.

3.3.2

31.08.2019

Тип	Описание



Решена проблема отображения роутов в документации, если контроллеры роутов были прописаны через путь к файлу, а не через неймспейс.

3.3.1

30.07.2019

Тип	Описание
	Решена проблема отображения технического сообщения на страницах Настройки и Безопасность при активации режима "Установка для разработки".

3.3.0

24.06.2019

Тип	Описание
	Отложенный запуск интерфейса. Теперь можно регулировать момент запуска модуля с помощью параметра в Настройках. Стало возможным запускать модуль либо при старте <code>init.php</code> (1.5), либо перед запуском визуальной части, в событии <code>OnProlog</code> (1.15). Подробнее: https://dev.1c-bitrix.ru/api_help/main/general/pageplan.php
	Документация.

3.2.0

07.05.2019

Тип	Описание
	Техническое обновление.

3.1.0

16.02.2019

Тип	Описание
	Для большего удобства документация по модулю вынесена на сторонний ресурс.
	Документация по модулю.
	Мелкие улучшения.

3.0.0

15.02.2019

Тип	Описание
	Метод создания таблиц журналов.
	Языковые фразы.

2.9.3 09.02.2019

Тип	Описание
	Выборка данных по дате в журнале Входящие запросы.
	Ошибка возникающая в случае удаления таблиц при удалении модуля.

2.9.2 09.02.2019

Тип	Описание
	Логика внутренних функций.

2.9.1 09.02.2019

Тип	Описание
	Незначительные ошибки.

2.9.0 09.02.2019

Тип	Описание
	Если роут отключен, тогда он будет исключен из документации в публичной части (см. описание версии 2.5.0).
	Теперь функции модуля не доступны из публичной части сайта.
	Оптимизация и увеличение производительности.

2.8.0

08.02.2019

Тип	Описание
	В Журнал Запрос/Ответ добавлено поле Метод, а также добавлен фильтр по данному полю.
	При авторизации через логин и пароль теперь еще происходит проверка на наличие <code>\$_SERVER['PHP_AUTH_USER']</code> и <code>\$_SERVER['PHP_AUTH_PW']</code> , для поддержки HTTP Authorization.
	При несовпадении типа контента запроса с настройками роута, теперь возвращается название типа контента, который необходим при запросе.
	Если запрос производится с участием авторизации, через токен или логин/пароль, тогда в параметры запроса будет добавлена информация о пользователе, которому принадлежат авторизационные данные. Данные пользователя сохраняются в ключ <code>_user</code> . Напоминание: в контроллере, параметры входящего запроса, можно получить с помощью функции <code>request()->get()</code> .
	При отключении нативных роутов они деактивируются и в документации.
	Добавлен индикатор активности нативных роутов в раздел Поддержка.
	Расширена поддержка типов контента при запросе. Например, ранее, метод <code>PUT</code> или <code>DELETE</code> поддерживали только <code>application/json</code> , теперь поддерживают и <code>application/x-www-form-urlencoded</code> , и другие. Напоминание: не забывайте указывать/ограничивать тип контента запроса через карту роутов, с помощью ключа <code>contentType</code> .
	Расширена поддержка типов запросов: <code>HEAD</code> , <code>PATCH</code> .
	Название Журнала Запрос/Ответ → Входящие запросы.
	Актуализирована документация по модулю.
	Описание модуля из раздела Документация - Основные сведения.
	Незначительные исправления.

2.7.0

01.02.2019

Тип	Описание
	Приведение названий входящих заголовков к нижнему регистру.

	Нативный роут для получения ID пользователя по логину или токenu. При запросе требуется авторизация по токenu. Уточнение: любой роут может быть перекрыт через собственную карту роутов. То есть, если нативный роут вам чем-то не подходит, то вы можете скопировать его в собственную карту и модифицировать под необходимую логику.
	Возможность включать/отключать использование нативных роутов.
	Возможность указывать путь к собственным картам роутов. Теперь, нет необходимости хранить рабочие файлы в папке модуля. А именно, карты роутов и контроллеры. Напоминаем о том, что контроллером для роута может выступать, как класс доступный по неймспейсу, так и обычный скрипт. Которые могут располагаться за пределами директории модуля.
	<code>lang</code> -папки для языков: <code>en</code> , <code>de</code> , <code>fr</code> . Внимание, все фразы имеют русский перевод. Папки добавлены для корректной работы модуля на мультязычных сайтах.

2.6.0

13.01.2019

Тип	Описание
	Добавлен параметр Срок годности токена. Теперь можно указывать срок годности токена при его генерации.
	В карте роутов, в секции <code>security</code> , в ключ <code>token</code> , добавлен ключ <code>checkExpire</code> , регулирующий необходимость проверки срока годности токена, при авторизации через токен.
	В карте роутов, в секции <code>security</code> , ключ <code>authType</code> переименован в <code>type</code> , и перенесен в ключ <code>auth</code> .

2.5.0

13.01.2019

Тип	Описание
	Добавлен встроенный роут <code>/api/docs</code> для получения документации в публичной части. Теперь клиенты могут получать автоматически генерируемую документацию в JSON-формате, которая создается на основе карт роутов.
	Папка <code>{module_id}/lib/controllers/example/</code> удалена, а класс <code>Example</code> перенесен в <code>{module_id}/lib/controllers/native/</code> .

2.4.0

12.01.2019

Тип	Описание
-----	----------



Возможность управлять CORS (кросс-доменными запросами) из административного раздела сайта.

2.3.0

12.01.2019

Тип	Описание
	Вернули поддержку файлового режима работы. Теперь в карте роута, вместо контроллера, можно указать путь к php-скрипту. Подробнее смотрите в карте-примере: <code>/routes/_example.php</code> .

2.2.0

12.01.2019

Тип	Описание
	Функция <code>json_encode</code> заменена на <code>Bitrix\Main\Web\Json::encode</code> для принудительного кодирования ответа в <code>utf-8</code> .
	Заголовки модуля отсылаются только в том случае, если был запущен интерфейс.
	В карте роутов, в секции <code>security</code> , убран ключ <code>required</code> .
	Возможность авторизации по логину/паролю. Тип авторизации настраивается в карте роута - ключ <code>authType</code> - подробнее смотрите в карте-примере: <code>/routes/_example.php</code> .
	Все настройки и функционал присущий авторизации через токен, так же действует и для авторизации через логин/пароль.

2.1.1

11.01.2019

Тип	Описание
	Добавлен принудительный перевод в нижний регистр названия класса при его автоматическом подключении

2.1.0

10.01.2019

Тип	Описание
	В карту роутов, в секцию <code>security</code> , добавлен ключ <code>required</code> , который указывает на необходимость активации параметра Авторизация по токену. Подробнее смотрите в карте-примере: <code>/routes/_example.php</code>

2.0.0

05.01.2019

Тип	Описание
	Отключение использования функции <code>GetModuleID</code> , так как из-за неё возникала проблема в работе модуля на windows-серверах, в виду разности разделителей директорий (" / ", " \ "). Проблема описана в Отзыве.
	Увеличен размер поля IP в таблице <code>artamonov_rest_request_response</code> . Для избежания проблемы, когда запросы посылаются из REST-клиентов.
	Добавлены роуты для работы с токенами (создание, получение, удаление).
	Добавлены роуты для работы с пользователями (создание, получение, обновление, удаление).
	Небольшая оптимизация и ускорение работы ядра.

Релизы: 2018

1.2.0

11.11.2018

Тип	Описание
	Добавлена возможность передавать сообщение при ответе <code>badRequest</code> , <code>internalServerError</code> .
	Добавлена валидация входящих параметров при запросе. Происходит проверка на наличие параметра в запросе, а также на соответствие типу. Валидация параметров реализована на два уровня. Поддерживаемые типы параметров при валидации: <code>string</code> , <code>integer</code> , <code>float</code> , <code>email</code> , <code>ip</code> , <code>domain</code> , <code>url</code> . Подробнее смотрите в карте роутов.
	Добавлены человеко-понятные ответы при неудачных запросах типа <code>badRequest</code> .
	Проблема при получении GET-параметров при запросах типа <code>PUT</code> и <code>DELETE</code> .
	Проблема при получении заголовков запроса на Nginx (без Apache).
	Стили модуля в административном разделе сайта.

1.1.0

24.10.2018

Тип	Описание
	Языковые фразы.
	Условие при установке кэша для избежания получения Исключения.
	Добавлен параметр в разделе Настройки для включения/отключения тестового роута - <code>http https://{domain}/{api}/example/check</code> .
	В карте роутов добавлен ключ <code>possibleValue</code> , для указания возможных значений параметра запроса. Подробнее смотрите в карте роутов.
	В карте роутов добавлена возможность добавлять пример ответа на запрос. Подробнее смотрите в карте роутов.
	В карте роутов добавлена возможность исключать описание роута. Подробнее смотрите в карте роутов.
	Вывод JSON в журнале Запрос/Ответ.

1.0.2 11.10.2018

Тип	Описание
	Параметры для возможности работы демо-режима

1.0.1 08.09.2018

Тип	Описание
	Работа CORS-запросов
	Интерфейсные ошибки

Обновление: 4.19.0

Это обновление безопасности. Рекомендуется установить его всем.

Если вы покупали модуль в Маркетплейсе — обновите его на [странице модуля в Маркетплейсе](#).

Эта инструкция нужна только тем, у кого нет доступа к официальному обновлению и кто хочет применить исправления вручную.

Обычная работа модуля не меняется: существующие роуты, токены и настройки остаются рабочими.

Что исправлено

1. Роут provider

Нативный роут provider — добавлен запрет на вызов опасных классов и методов (чёрный список), есть режим белого списка.

2. Журнал запросов

Выборки переведены на безопасное ORM, закрыта возможность SQL-инъекции.

3. Генерация токенов

Генерация стала криптостойкой; ранее выданные токены продолжают работать.

4. Определение IP-адреса

Адрес берётся только из доверенного источника, подделать заголовком нельзя.

5. Защита от перебора

На роуты управления токеном добавлено ограничение числа неудачных попыток с одного IP — защита от подбора логина и пароля.

6. Создание пользователя

При создании пользователя через API системные поля (группы, токен, служебные атрибуты) больше нельзя задать напрямую — защита от повышения привилегий.

7. Генерация токенов в админке

Административный метод массовой генерации токенов теперь требует авторизации в админке и защищён от подделки запросов (CSRF).

8. Код поля токена

Имя поля токена из настроек теперь проверяется перед подстановкой в запросы к базе — закрыта возможность инъекции через эту настройку.

9. Прочее

Удаление пользователя, демо-контроллер и заголовки ответа получили мелкие усиления.

Установка патча

1. Загрузка патча

[Скачать patch.zip](#)

2. Резервная копия

Перед установкой скопируйте папку модуля — на случай отката:

Текст

```
1 /bitrix/modules/artamonov.rest/
```

3. Распаковка архива

Внутри архива файлы уже разложены по своим местам — начиная с папки **bitrix**. Распакуйте архив в корневую папку сайта и согласитесь заменить файлы.

Текст

```

1  patch.zip
2  └─ bitrix/
3      └─ modules/
4          └─ artamonov.rest/
5              └─ admin/
6                  └─ ajax/security.php
7                      └─ pages/rest-api-security.php
8              └─ install/version.php
9              └─ lib/
10                  └─ controllers/native/ (example.php, provider.php, token.php, user.php)
11                  └─ foundation/ (helper.php, journal.php, request.php,
requestlimittable.php, response.php, settings.php)

```

Путь распаковки должен совпасть с уже установленным модулем — файлы просто заменятся новыми.

Ничего удалять вручную не нужно.

4. Проверка версии

Откройте Marketplace → Установленные решения и убедитесь, что указана версия 4.19.0.

Если версия обновилась — патч применён.

Подробнее об изменениях

Ниже — что именно поменялось в файлах, если хочется понять суть правок.

1. Роут provider

Перед вызовом класса и метода теперь выполняется проверка доступа.

По умолчанию работает чёрный список: он запрещает потенциально опасные операции — управление пользователями и группами, файловые операции и заливку файлов, прямые запросы к базе, внешние HTTP-запросы и т.п.

Обычные «читающие» вызовы вроде `getList` не затрагиваются.

Если нужно ограничить доступ жёстче, можно включить режим белого списка — разрешить только нужные классы и методы. Как это настроить, описано в статье [Роут-провайдер](#), раздел «Безопасность».

2. Генерация токенов

Токен больше не строится на основе `md5()` и предсказуемого `rand()` — вместо этого используется криптостойкий генератор случайных байт.

Формат токена не изменился, ранее выданные токены остаются рабочими.

PHP

```

1  // До
2  public function generateToken($userId = '', $userLogin = '')
3  {
4      $token = md5($userId . '-' . $userLogin . '=' . date('Y-m-dH:i:s') . $this-
>generateUniqueCode());
5      $token = str_split($token, 8);
6      $token = implode('-', $token);
7      return $token;
8  }
9
10 // После
11 public function generateToken($userId = '', $userLogin = '')
12 {
13     $token = bin2hex(random_bytes(16));
14     $token = str_split($token, 8);
15     $token = implode('-', $token);
16     return $token;
17 }
```

Аналогично уникальные коды теперь берутся из `random_int()` вместо `rand()` :

PHP

```

1  // До
2  $code .= $characters[rand(0, $charactersLength - 1)];
3
4  // После
5  $code .= $characters[random_int(0, $charactersLength - 1)];
```

3. Определение IP-адреса

IP-адрес клиента теперь берётся только из `REMOTE_ADDR` .

Раньше учитывались заголовки, которые клиент может подставить сам, — это позволяло подменить IP в логах и обойти ограничение частоты запросов.

PHP

```

1  // До
2  public function ip()
3  {
4      if ($this->ip === null) {
5          if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
6              $this->ip = $_SERVER['HTTP_CLIENT_IP'];
7          } else {if (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
8              $this->ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
9          } else {
10             $this->ip = $_SERVER['REMOTE_ADDR'];
11         }}
12     }
13     return $this->ip;
14 }
15
16 // После
17 public function ip()
18 {
19     if ($this->ip === null) {
20         $this->ip = $_SERVER['REMOTE_ADDR'] ?? '';
21     }
22     return $this->ip;
23 }

```

4. Защита от перебора

Роуты управления токеном (создание, получение, удаление) аутентифицируют пользователя по логину и паролю.

Теперь на них действует ограничение: если с одного IP за короткий промежуток времени накапливается слишком много неудачных попыток, дальнейшие запросы временно отклоняются с ответом 429. Это защищает от подбора учётных данных.

Привязка к IP, а не к логину, выбрана специально — чтобы перебором нельзя было заблокировать чужую учётную запись.

5. Создание пользователя

При создании пользователя через API теперь нельзя напрямую задать системные поля — группы (`GROUP_ID`), поля токена и служебные атрибуты.

Раньше такие поля принимались наравне с обычными, что позволяло, например, создать пользователя с правами администратора. Обычные поля (логин, e-mail, пароль, имена, контакты, собственные `UF_*`-поля) работают как прежде.

Список системных полей можно изменить, перекрыв роут.

6. Генерация токенов в админке

В разделе «Безопасность» есть инструмент массовой генерации токенов для пользователей. Он работает через служебный AJAX-запрос.

Раньше этот запрос не проверял, кто его отправляет. Теперь он доступен только авторизованным в админке сотрудникам с правом на раздел «Безопасность» и защищён от подделки запросов (CSRF). Внешний запрос со стороны больше не может запустить генерацию токенов.

На обычную работу модуля это не влияет — генерация из интерфейса работает как прежде.

7. Код поля токена

В настройках можно задать имя пользовательского поля, в котором хранится токен. Это значение подставлялось в запросы к базе как есть.

Теперь оно проверяется: допускается только корректное имя поля (буквы, цифры, подчёркивание), иначе используется значение по умолчанию. Это исключает возможность подставить через настройку посторонний фрагмент SQL.

Эксплуатировать это мог только администратор с доступом к настройкам, поэтому риск был невысоким — но место укреплено.

8. Прочее

Файл	Что изменилось
<code>journal.php</code>	Выборки переписаны на ORM, значения экранируются автоматически
<code>user.php</code>	ID пользователя приводится к числу перед удалением
<code>example.php</code>	Демо-роут отдаёт только безопасные поля вместо всего <code>\$_SERVER</code>
<code>response.php</code>	Из заголовков ответа вырезаются переводы строки

Проверка работы

Для проверки работы интерфейса можно воспользоваться тестовым роутом.

Роут активируется в Настройках модуля

`{http || https}://{domain}/{api}/example/check`

- `{http || https}` - используемый протокол
- `{domain}` - домен сайта
- `{api}` - путь к api-интерфейсу указанный в настройках модуля

Внимание: после проверки тестовый роут лучше отключить.

GET `http://artamonov.rest.dev:8888/api/example/check` Send **200 OK** TIME 185 ms SIZE 1950 B

Body Auth Query **Header** Docs

URL PREVIEW
`http://artamonov.rest.dev:8888/api/example/check?parameter1=value1¶meter2=value2`

parameter1	value1
parameter2	value2
New name	New value

Preview Header **8** Cookie Timeline

```

1 {
2   "message": "Запрос выполнен успешно",
3   "date": "2019-02-16 16:38:12",
4   "controller": "Artamonov\\Rest\\Controllers\\Native\\Example",
5   "action": "_get",
6   "method": "GET",
7   "header": {
8     "accept": "*/*",
9     "user-agent": "insomnia/6.3.2",
10    "host": "artamonov.rest.dev:8888"
11  },
12  "request": {
13    "parameter1": "value1",
14    "parameter2": "value2"
15  },
16  "server": {
17    "PHP_FCGI_CHILDREN": "4",
18    "PWD": "\\Applications\\MAMP\\fcgi-bin",
19    "SHLVL": "0",
20    "PHP_FCGI_MAX_REQUESTS": "200",
21    "_CF_USER_TEXT_ENCODING": "0x1F5:0x7:0x31",
22    "ORIG_SCRIPT_NAME": "\\fcgi-bin\\php7.1.22.fcgi",
23    "ORIG_PATH_TRANSLATED": "\\Users\\artamonov\\Projects\\artamonov.rest.loc\\bitrix\\urlrewrite.php",
24    "ORIG_PATH_INFO": "\\bitrix\\urlrewrite.php",
25    "ORIG_SCRIPT_FILENAME": "\\Applications\\MAMP\\fcgi-bin\\php7.1.22.fcgi",
26    "SCRIPT_NAME": "\\bitrix\\urlrewrite.php",
27    "REQUEST_URI": "\\api/example/check?parameter1=value1&parameter2=value2",
28    "QUERY_STRING": "parameter1=value1&parameter2=value2",
29    "REQUEST_METHOD": "GET",
30    "SERVER_PROTOCOL": "HTTP/1.1",
31    "GATEWAY_INTERFACE": "CGI/1.1",
32    "REDIRECT_URL": "\\bitrix\\urlrewrite.php",
33    "REDIRECT_QUERY_STRING": "parameter1=value1&parameter2=value2",
34    "REMOTE_PORT": "55078",
35    "SCRIPT_FILENAME": "\\Users\\artamonov\\Projects\\artamonov.rest.loc\\bitrix\\urlrewrite.php",
36    "SERVER_ADMIN": "you@example.com",
37    "DOCUMENT_ROOT": "\\Users\\artamonov\\Projects\\artamonov.rest.loc",
38    "REMOTE_ADDR": "127.0.0.1",
39    "SERVER_PORT": "8888",
40    "SERVER_ADDR": "127.0.0.1",
41    "SERVER_NAME": "artamonov.rest.dev",
42    "SERVER_SOFTWARE": "Apache",
43    "SERVER_SIGNATURE": "",
44    "PATH": "\\usr\\bin\\bin\\usr\\sbin\\sbin",
45    "HTTP_ACCEPT": "*/*",
46    "HTTP_USER_AGENT": "insomnia/6.3.2",
47    "HTTP_HOST": "artamonov.rest.dev:8888",
48    "REDIRECT_STATUS": "200",
49    "REDIRECT_HANDLER": "php-fastcgi",
50    "REDIRECT_REMOTE_USER": "",
51    "REDIRECT_REDIRECT_STATUS": "200",
52    "FCGI_ROLE": "RESPONDER",
53    "PHP_SELF": "\\bitrix\\urlrewrite.php",
54    "REQUEST_TIME_FLOAT": 1550309892.6537868976593017578125,
55    "REQUEST_TIME": 1550309892,
56    "argv": [
57      "parameter1=value1&parameter2=value2"
58    ],
59    "argc": 1
60  }
61 }
```

Первые шаги

Для начала работы необходимо:

1. Активировать интерфейс.
2. Указать путь интерфейса.
3. Для тестирования интерфейса необходимо активировать тестовый роут.
4. Для использования нативных роутов необходимо активировать нативные роуты.

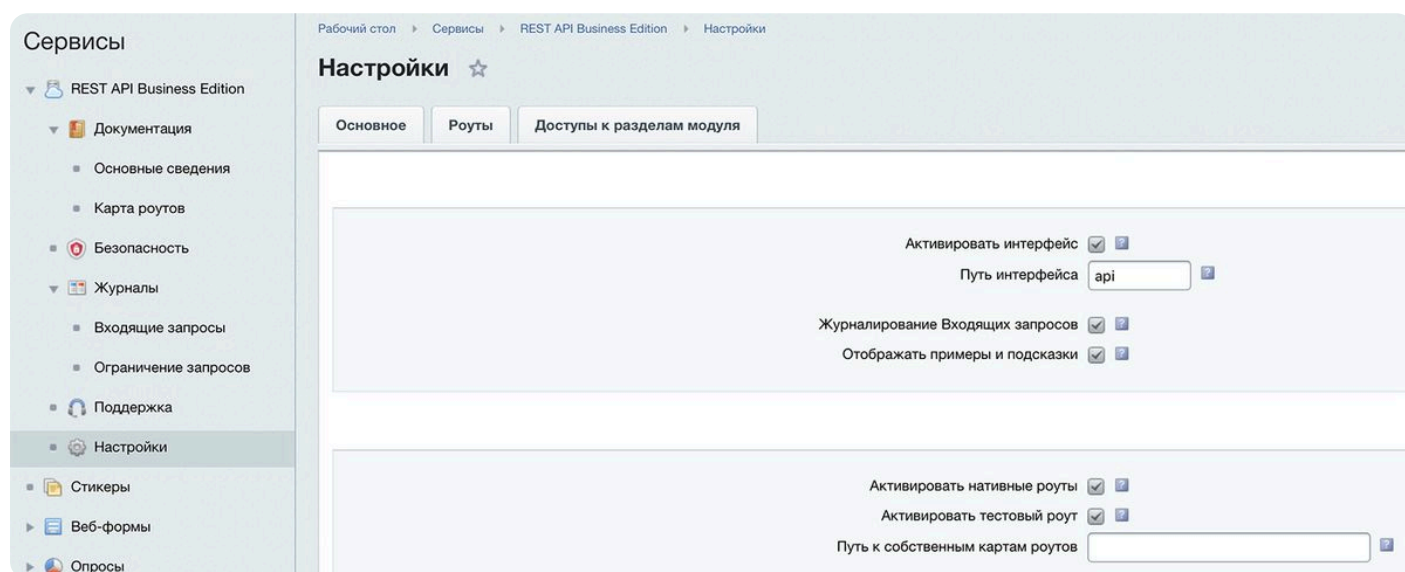


Схема работы

Для примера рассмотрим путь: **https://{domain}/api/example/check**

Общий принцип работы интерфейса следующий:

1. При входящем запросе веб-сервер пытается направить запрос по пути **/api/example/check**.
2. Когда веб-сервер понимает, что такого пути **нет**, он инициирует ошибку **404**.
3. Когда срабатывает ошибка 404, то происходит **запуск** файла **404.php** (указывается в **настройках веб-сервера**).
4. При запуске файла 404.php, учитывая, что в нем прописаны все **директивы Битрикса**, начинает обрабатывать файл **init.php**.
5. В случае, если файл init.php запускается **успешно**, то происходит **запуск модуля**.
6. При запуске, модуль анализирует настройки, правила и входящий запрос.
7. В случае, если для входящего запроса прописаны правила в карте роутов, тогда срабатывает **контроллер** указанный для роута.
8. В контроллере отрабатывает некоторая логика, которую **пропишет** разработчик.
9. В **конце отработки метода** контроллера **необходимо вызвать** функцию `response()→json()` для возврата результата клиенту.

Примечание

- Минимальная версия PHP **>= 7.2** (согласно требованиям платформы). Использование более ранней версии возможно, но крайне не рекомендуется, ввиду возникновения ошибок.
- **Обязательно** наличие файла для обработки 404 ошибки (обычно имеется в корне сайта - 404.php) со всеми “директивами” 1С-Битрикса.
- Если используется веб-сервер **Apache**: обязательно прописать `ErrorDocument 404 /404.php` в `.htaccess`, расположенный в корне сайта, или прямо в конфигурационном файле Apache.
- Если используется веб-сервер **Nginx**: обязательно прописать `error_page 404 = /404.php?$query_string;` в секцию `server` конфигурационного файла (обычно `nginx.conf`).

- Для работы **CORS**-запросов необходимо чтобы на веб-сервере были **разрешены** запросы типа **OPTIONS**.

Многосайтовость

Запуск модуля происходит в файле `init.php`, в папке `bitrix/php_interface` или `local/php_interface` (зависит от структуры проекта).

В многосайтовом режиме работы имеется возможность создавать файлы `init.php` для каждого сайта.

Например:

- `local/php_interface/s1/init.php`
- `local/php_interface/s2/init.php`
- ...
- `local/php_interface/{siteId}/init.php`

В связи с этим, вы можете вынести подключение модуля в нужный файл `init.php`, и запускать модуль только на определенном сайте.

Либо, если файл `init.php` используется один для всех сайтов, то можно ограничить запуск модуля на конкретных сайтах, с помощью соответствующего параметра в настройках модуля (доступно с версии 4.6.0).

Карты роутов

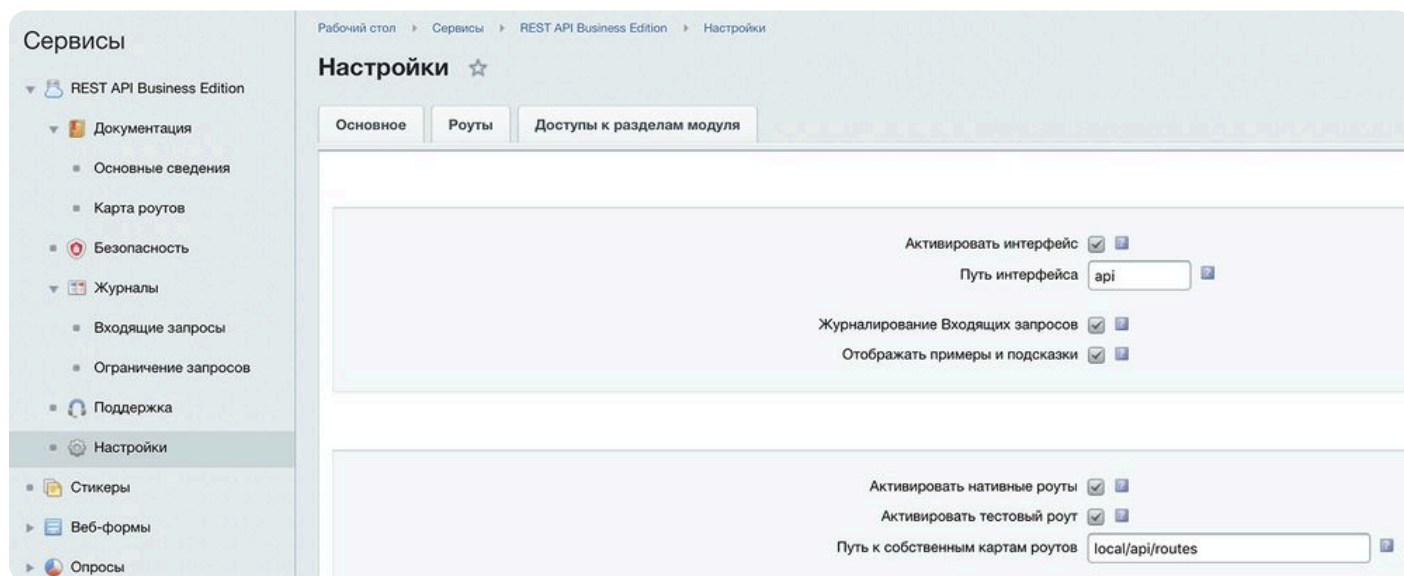
Карта роутов - массив описывающий правила обработки для входящего запроса.

По умолчанию, карты роутов располагаются в папке модуля **/bitrix/modules/artamonov.rest/routes/**.

Начиная с версии 2.7.0 в настройках модуля можно указать собственный путь к картам роутов.

Для добавления карты нужно создать PHP-файл, возвращающий массив, описывающий необходимые роуты. В качестве примера, можно ознакомиться с картой нативных роутов, расположенной по пути **/bitrix/modules/artamonov.rest/routes/_native.php**.

Количество карт роутов не ограничено.



PHP

```

1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 1/13/19 4:57 PM
4   * Author: Denis Artamonov
5   *
6   * Пример карты роутов
7   */
8
9  /**
10 | -----
11 | Пример карты роутов
12 | -----
13 |
14 | Внимание!
15 |
16 | - Данный файл может быть перезаписан при обновлении модуля.
17 | - Данный файл исключен из публичной документации.
18 |
19 | Для добавления своих роутов, создайте собственную карту.
20 | Например:
21 | /routes/sale.php
22 | /routes/iblock.php
23 | /local/api/routes/iblock.php
24 | И так далее, количество карт неограниченно.
25 |
26 | Карты могут располагаться вне папки модуля. Для этого необходимо указать путь к собственной
папке в настройках.
27 | Карты будут подгружены автоматически.
28 |
29 | Совет: не желательно разбивать карту на множество файлов,
30 |       так как, чем больше файлов, тем больше будет происходить их подключений,
31 |       соответственно, затрачивается дополнительное время при запуске интерфейса.
32 |
33 | А также, не забудьте указать контроллеры для обработки роутов.
34 | Контроллеры могут располагаться где угодно, главное чтобы они были доступны через пространство
имён.
35 | Вместо класса можно указать любой php-файл, который будет подключен и отработан при запросе.
36 |
37 | Для получения карты из контроллера можно воспользоваться методом request()->map()
38 |
39 | Поддерживаемые типы параметров: string, integer, float, email, ip, domain, url
40 |
41 */
42
43 return [
44     // Тип запроса
45     'GET' => [
46         // Роут
47         // Для получения всех параметров request()->get()
48         // Для получения конкретного параметра request()->get({parameter_name})
49         'example/check' => [
50             // Описание роута
51             // 'description' => 'GET',
52
53             // Активность роутра
54             // Необязательный параметр (по умолчанию роут активен)
55             //'active' => false,
56
57             // Контроллер обрабатывающий роут

```

```

58 // Данный ключ исключен из публичной документации
59 // Контроллер может располагаться за пределами модуля, главное чтобы он был доступен
через пространство имён
60 // Где Example - класс, _get - метод класса
61 'controller' => '\Artamonov\Rest\Controllers\Native\Example@_get',
62 // Вместо класса можно указать любой php-файл, который будет подключен и отработан
при запросе
63 // 'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/rest/controllers/file.php',
64
65 // Безопасность участвует при запуске интерфейса
66 'security' => [
67
68     // Настройки авторизации при запросе
69     'auth' => [
70         // Обязательная авторизация для роута
71         // Происходит проверка на наличие заголовков
72         // 'required' => true, // true || false
73
74         // Тип авторизации используемый для роута
75         // Если login, тогда должны быть переданы заголовки Authorization-Login и
Authorization-Password
76         // Если token, тогда должен быть передан заголовок Authorization-Token
77         // 'type' => 'login', // login || token
78     ],
79
80     // Настройки безопасности для логинов
81     // Данный ключ исключен из публичной документации
82     'login' => [
83         // Доступ к методу будет доступен только для логинов из списка
84         'whitelist' => [
85             // 'admin',
86         ]
87     ],
88
89     // Настройки безопасности для токенов
90     // Данный ключ исключен из публичной документации
91     'token' => [
92         // Проверять срок годности токена
93         // Если токен находится в белом списке, тогда параметр игнорируется
94         'checkExpire' => false, // true || false
95
96         // Доступ к методу будет доступен только для токенов из списка
97         'whitelist' => [
98             // 'b3bfb6b8-82dca90f-6049641b-76d957d6',
99             // 'bc95d11b-f2fdf7f4-15e869d3-882e72b5',
100             // '408f4f2e-d5a6e4a7-06930a16-8301b343',
101         ]
102     ],
103
104     // Настройки безопасности для групп
105     // Данный ключ исключен из публичной документации
106     'group' => [
107         // Доступ к методу будет доступен только для групп из списка.
108         // Указывается ID группы.
109         // Если имеется белый список логинов или токенов, тогда параметр
игнорируется.
110         'whitelist' => [
111             // 1,
112             // 6,
113             // 7,
114         ]
115     ]

```

```

116     ],
117     // Параметры запроса
118     // Напоминание: для получения текущих параметров в контроллере, можно воспользоваться
методом request()->map()
119     'parameters' => [
120         'iblock_id' => [
121             // Обязательный параметр
122             // 'required' => true, // true || false
123             'type' => 'integer',
124             // Возможные значения параметра
125             'possibleValue' => [
126                 '1',
127                 '...',
128                 '50'
129             ],
130             'description' => 'ID инфоблока'
131         ],
132         'active' => [
133             'type' => 'string',
134             'possibleValue' => [
135                 'Y',
136                 'N'
137             ],
138             'description' => 'Активность'
139         ],
140         'name' => [
141             'type' => 'string',
142             'description' => 'Имя элемента'
143         ],
144         'color' => [
145             'type' => 'string',
146             'possibleValue' => [
147                 'Белый',
148                 'Красный'
149             ],
150             'description' => 'Свойство: Цвет'
151         ],
152         'form' => [
153             'type' => 'string',
154             'possibleValue' => [
155                 'Круглый',
156                 'Квадрат'
157             ],
158             'description' => 'Свойство: Форма'
159         ],
160         'fields' => [
161             'type' => 'string',
162             'possibleValue' => [
163                 'preview_text',
164                 'detail_text',
165                 'preview_picture',
166                 'form'
167             ],
168             'description' => 'Дополнительные поля'
169         ],
170         'sort' => [
171             'type' => 'string',
172             'description' => 'Сортировка (пример: sort=id:asc,name:asc)'
173         ],
174         'limit' => [
175             'type' => 'integer',
176             'possibleValue' => [

```

```

177         '1',
178         '...',
179         '150'
180     ],
181     'description' => 'Количество (по умолчанию: 10)'
182 ],
183 'page' => [
184     'type' => 'integer',
185     'description' => 'Страница'
186 ]
187 ],
188 // Пример ответа
189 // Необходим лишь для документации
190 // #DOMAIN# и #API# будут автоматически заменены на реальные данные при выводе на
экран
191 // Данный ключ исключен из публичной документации
192 // 'example' => [
193 //     'request' => [
194 //         'url' => 'https://#DOMAIN#/#API#/example/check/?iblock_id=1&sort=id:asc',
195 //         'response' => [
196 //             'json' => '{"page":1,"total":3,"items":[{"ID":1,"NAME":"item1"},
{"ID":2,"NAME":"item2"}, {"ID":3,"NAME":"item3"}]}'
197 //         ]
198 //     ]
199 // ],
200 // Настройки для поведения роута в документации
201 // Данный ключ исключен из публичной документации
202 'documentation' => [
203     // Исключить роут
204     'exclude' => [
205         // Из документации в административной части сайта
206         'admin' => false, // true || false
207         // Из документации в публичной части сайта
208         'public' => false, // true || false
209     ]
210 ]
211 ],
212
213 // Пример роута с использованием переменных в строке
214 // Доступно начиная с версии 3.4.0
215 '/example/check/iblock/{{iblockId}}/' => [
216     'controller' => '\Artamonov\Rest\Controllers\Native\Example@_get',
217     'parameters' => [
218         'iblockId' => [
219             'required' => true,
220             'type' => 'integer',
221             'description' => 'ID инфоблока',
222         ],
223     ]
224 ],
225 'example/check/section/{{sectionId}}' => [
226     'controller' => '\Artamonov\Rest\Controllers\Native\Example@_get',
227     'parameters' => [
228         'sectionId' => [
229             'required' => true,
230             'type' => 'integer',
231             'description' => 'ID раздела',
232         ]
233     ]
234 ],
235 'example/check/section/{{sectionId}}/{{productId}}/view' => [
236     'controller' => '\Artamonov\Rest\Controllers\Native\Example@_get',

```

```

237     'parameters' => [
238         'sectionId' => [
239             'required' => true,
240             'type' => 'integer',
241             'description' => 'ID раздела',
242         ],
243         'productId' => [
244             'required' => true,
245             'type' => 'integer',
246             'description' => 'ID товара',
247         ],
248     ],
249 ],
250 ],
251 // Запросы поддерживают все возможные параметры: из строки запроса (GET) и из тела запроса
252 'POST' => [
253     'example/check' => [
254         // 'description' => 'POST',
255         // Роут отключен
256         // Клиент получит ответ со статусом: 434 Requested host unavailable
257         // 'active' => false,
258         'controller' => '\Artamonov\Rest\Controllers\Native\Example@_post',
259         // Сервер ожидает запрос с типом контента application/json
260         // Если тип будет отличаться, запрос будет отклонен
261         'contentType' => 'application/json',
262         'security' => [
263             'auth' => [
264                 // 'required' => true,
265                 // 'type' => 'login',
266             ],
267             'token' => [
268                 'whitelist' => [
269                     // '408f4f2e-d5a6e4a7-06930a16-8301b343'
270                 ]
271             ]
272         ],
273         'parameters' => [
274             // Параметры: уровень 1
275             'iblock_id' => [
276                 'required' => true,
277                 'type' => 'integer',
278                 'description' => 'ID инфоблока'
279             ],
280
281             'user' => [
282                 'required' => true,
283                 'type' => 'array',
284                 'description' => 'Автор',
285                 // Параметры: уровень 2
286                 'parameters' => [
287                     'id' => [
288                         'required' => true,
289                         'type' => 'integer',
290                         'description' => 'ID'
291                     ],
292                     'name' => [
293                         'required' => true,
294                         'type' => 'string',
295                         'description' => 'Имя'
296                     ]
297                 ]
298             ],

```

```

299     'items' => [
300         'required' => true,
301         'type' => 'array',
302         'description' => 'Массив объектов (элементов)',
303         'parameters' => [
304             // Параметры для элемента типа массив
305             // Правила прописываются только для одного item, но проверка происходит
для каждого объекта/массива
306             //
307             // Пример items при запросе:
308             // "items": [
309                 //
310                 //     {
311                 //         "name": "test",
312                 //         "color": "blue",
313                 //         "preview_text": "Описание анонса"
314                 //     },
315                 //     {
316                 //         "name": "test2",
317                 //         "color": "blue",
318                 //         "detail_text": "Детальное описание"
319                 //     },
320                 //     {
321                 //         "name": "test3",
322                 //         "color": "blue"
323                 //     }
324             ],
325             'name' => [
326                 'required' => true,
327                 'type' => 'string',
328                 'description' => 'Имя элемента'
329             ],
330             'preview_text' => [
331                 'type' => 'string',
332                 'description' => 'Описание анонса'
333             ],
334             'detail_text' => [
335                 'type' => 'string',
336                 'description' => 'Детальное описание'
337             ],
338             'preview_picture_url' => [
339                 'type' => 'url',
340                 'description' => 'Url изображения анонса'
341             ],
342             'detail_picture_url' => [
343                 'type' => 'url',
344                 'description' => 'Url детального изображения'
345             ],
346             'color' => [
347                 'required' => true,
348                 'type' => 'string',
349                 'description' => 'Свойство: Цвет'
350             ],
351             'form' => [
352                 'type' => 'string',
353                 'description' => 'Свойство: Форма'
354             ]
355         ]
356     ]
357 ]
358 ]
359 ]

```

```

360     ]
361 ],
362 'PUT' => [
363     'example/check' => [
364         // 'description' => 'PUT',
365         'controller' => '\Artamonov\Rest\Controllers\Native\Example@_put',
366         'contentType' => 'application/json',
367         'security' => [
368             'auth' => [
369                 // 'required' => true,
370                 // 'type' => 'login',
371             ],
372         ],
373         'parameters' => [
374             'element_id' => [
375                 'required' => true,
376                 'type' => 'integer',
377                 'description' => 'ID элемента'
378             ],
379             'name' => [
380                 'type' => 'string',
381                 'description' => 'Имя элемента'
382             ],
383             'preview_text' => [
384                 'type' => 'string',
385                 'description' => 'Описание анонса'
386             ],
387             'detail_text' => [
388                 'type' => 'string',
389                 'description' => 'Детальное описание'
390             ],
391         ],
392     ],
393 ],
394 'PATCH' => [
395     'example/check' => [
396         // 'description' => 'PATCH',
397         'controller' => '\Artamonov\Rest\Controllers\Native\Example@_put',
398         'contentType' => 'application/json',
399         'security' => [
400             'auth' => [
401                 // 'required' => true,
402                 // 'type' => 'login',
403             ],
404         ],
405         'parameters' => [
406             'element_id' => [
407                 'required' => true,
408                 'type' => 'integer',
409                 'description' => 'ID элемента'
410             ],
411             'name' => [
412                 'type' => 'string',
413                 'description' => 'Имя элемента'
414             ],
415             'preview_text' => [
416                 'type' => 'string',
417                 'description' => 'Описание анонса'
418             ],
419             'detail_text' => [
420                 'type' => 'string',
421                 'description' => 'Детальное описание'

```

```
422         ],
423     ],
424 ],
425 ],
426 'DELETE' => [
427     'example/check' => [
428         // 'description' => 'DELETE',
429         'controller' => '\Artamonov\Rest\Controllers\Native\Example@_delete',
430         'security' => [
431             'auth' => [
432                 // 'required' => true,
433                 // 'type' => 'login',
434             ],
435         ],
436         'parameters' => [
437             'element_ids' => [
438                 'required' => true,
439                 'type' => 'array',
440                 'description' => 'ID элементов'
441             ]
442         ]
443     ],
444 ],
445 'HEAD' => [
446     'example/check' => [
447         // 'description' => 'HEAD',
448         'controller' => '\Artamonov\Rest\Controllers\Native\Example@_head'
449     ]
450 ],
451 ];
```

Список функций

request()

Функция возвращает данные по текущему запросу.

Метод	Параметры	Описание
get(\$code = "")	\$code - код параметра	Параметры запроса
header(\$code = "")	\$header - код заголовка	Заголовки запроса
method()		Тип запроса
ip()		IP-адрес клиента
path()		Путь запроса
map()		Карта роута для обработки запроса

PHP

```
1 // Пример возврата данных по текущему запросу
2 request()->get();
```

response()

Функция возвращает ответ клиенту. Имеется набор предустановленных ответов, которые реализованы согласно стандартам HTTP.

Метод	Параметры	Описание
json(\$data = [], \$statusCode = 200, \$options = [], \$headers = [])	\$data - данные ответа \$statusCode - код ответа \$options - параметры ответа (аналогичны функции json_encode) \$headers - дополнительные заголовки ответа	Возвращает ответ клиенту в JSON-формате
accepted()		
badGateway()		
badRequest(\$message = "")		
conflict()		

created(\$data = [])		
failedDependency()		
forbidden()		
gone()		
httpVersionNotSupported()		
internalServerError(\$message = ")")		
invalidSslCertificate()		
lengthRequired()		
locked()		
methodNotAllowed()		
misdirectedRequest()		
movedPermanently(\$location = ")")		
multipleChoices()		
noContent()		
nonAuthoritativeInformation()		
notAcceptable()		
notFound()		
notImplemented()		
notModified()		
ok(\$data = [])		
partialContent(\$contentRange = ")")		
payloadTooLarge(\$retryAfterSeconds = 60)		
paymentRequired()		
permanentRedirect(\$location = ")")		
preconditionFailed()		

processing()		
requestedHostUnavailable()		
resetContent()		
seeOther(\$location = "")		
serviceUnavailable()		
sslHandshakeFailed()		
switchingProtocols(\$upgrade = "")		
temporaryRedirect(\$location = "")		
tooManyRequests()		
unprocessableEntity()		
unauthorized()		
unavailableForLegalReasons()		
unknownError()		
unsupportedMediaType()		
upgradeRequired()		
uriTooLong()		
useProxy(\$location = "")		
webServerIsDown()		

PHP

```

1  // Пример возврата json-ответа с произвольными данными
2  response()->json($data);
3
4  // Пример возврата json-ответа = OK
5  response()->ok();

```

cache()

Функция для работы с системным кэшем. Является обёрткой над системным классом Cache. Используется кэш установленный в настройках системы.

Метод	Параметры	Описание
get(\$id, \$ttl = 86400, \$dir = false)	\$id - идентификатор \$ttl - время жизни \$dir - директория	Получение кэша по ID, если кэша нет, тогда инициализирует его
set(\$data)	\$data - данные для кэша	Сохранение кэша в ранее инициализированный кэш
clear(\$dir)	\$dir - директория	Очистка кэша

PHP

```

1 // Пример получения кеша
2 cache()->get($cacheId, $cacheTtl, $cacheDir);

```

journal()

Функция для работы с журналами.

Метод	Параметры	Описание
add(\$type, \$data = [], \$clientId = "")	\$type - тип журнала \$data - данные \$clientId - ID клиента	Запись данных в журнал

PHP

```

1 // Пример добавления записи в журнал
2 // Тип журнала Входящие запросы: request-response
3 journal()->add('request-response', \['request' => request()->get(), 'response' => $response]);

```

db()

Функция возвращает текущее подключение к базе данных. Является обёрткой над системным методом getConnection() класса Application.

Метод	Параметры	Описание
query(\$sql)	\$sql - строка SQL-запроса	Выполнение запроса

PHP

```

1 // Пример выполнения запроса
2 db()->query($sql);

```

token()

Функция для работы с токенами.

Метод	Параметры	Описание
getData(\$token)	\$token - токен клиента	Возвращает информацию по токenu
getDataByUserId(\$userId)	\$userId - ID пользователя	Возвращает данные токена по ID пользователя
getExpirationDate(\$token)	\$token - токен клиента	Возвращает срок годности токена
updateExpirationDate(\$token)	\$token - токен клиента	Обновляет срок годности токена, согласно настройкам Безопасности модуля
delete(\$token)	\$token - токен клиента	Удаляет токен пользователя
generate(\$parameters = [], \$update = false)	\$parameters - все те же ключи, что и для метода UserTable::getList() \$update - флаг для регулирования - обновлять уже созданные токены или же создать новые токены	Генерация токенов для пользователей

PHP

```

1 // Пример получения данных о токене
2 token()->getData('8f757146-097eb57b-b07d669e-112463ff');
3
4 // Пример генерации токенов
5 token()->generate(['filter' => \['>ID' => 30]], true);

```

Журналы

Взаимодействие с журналами осуществляется через функцию `journal()` .

PHP

```
1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 2/16/19 5:57 PM
4   * Author: Denis Artamonov
5   *
6   * Пример записи в журнал
7   */
8
9  namespace Artamonov\Rest\Controllers\Example;
10
11  class Journal
12  {
13      public function save()
14      {
15          // Запишем информацию в Журнал: Входящие запросы
16          journal()->add(
17              // Тип журнала
18              'request-response',
19              // Данные
20              // Для журнала Входящие запросы поддерживаются ключи: request и response
21              [
22                  'request' => request()->get(),
23                  'response' => $response
24              ]
25          );
26      }
27  }
```

Кэширование

Модуль умеет взаимодействовать с кэшем установленным по умолчанию в платформе. По большей части кэширование необходимо использовать тогда, когда происходят выборки большого объема данных.

При работе с кэшем необходимо придерживаться следующей логики: сначала проверяем наличие необходимого кэша, если его нет, тогда создаем его. В следующий раз, данные уже будут подгружаться из кэша.

Взаимодействие с системным кэшем происходит за счет функции `cache()`.

Внимание: если используется работа с кэшем, тогда не нужно забывать про его очистку, в случае изменения данных.

PHP

```

1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 2/16/19 5:23 PM
4   * Author: Denis Artamonov
5   *
6   * Пример контроллера с методом
7   * В котором выполняется некоторая логика выборки данных и проверки кэша
8   */
9
10 namespace Artamonov\Rest\Controllers\Example;
11
12 class Cache
13 {
14     public function check()
15     {
16         $response = [];
17         // Отдадим данные из кэша если они в нем имеются
18         // Иначе, получим данные из базы и запишем в кэш
19         // $cacheId - уникальный идентификатор кэша
20         // Срок кэша - 7 дней
21         // Место хранения /bitrix/cache/example/check
22         if (!$response = cache()->get($cacheId, 604800, 'example/check')) {
23             // Какие-то данные из базы
24             $arResult = [1, 2, 3];
25             // Массив ответа для клиента
26             $response = &$arResult;
27             // Сохраняем данные в кэш чтобы при следующем запросе уже не делать запросы в базу
28             if ($response) {
29                 cache()->set($response);
30             }
31         }
32         // Возвращаем ответ клиенту
33         response()->json($response);
34     }
35 }
```

Контроллеры

Контроллер - это класс с методами или обычный скрипт, который будет отработан при входящем запросе.

Контроллеры указываются в карте роутов при описании роута. По умолчанию, контроллеры располагаются в папке модуля **/bitrix/modules/artamonov.rest/lib/controllers/**. Но, также контроллер может располагаться за пределами модуля, главное чтобы он был доступен через пространство имён или по указанному пути.

Начиная с версии 2.3.0 можно указать путь к PHP-скрипту, вместо указания неймспейса контроллера.

В качестве примера, можно ознакомиться с нативными контроллерами, расположенными по пути **/bitrix/modules/artamonov.rest/lib/controllers/native**.

PHP

```

1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 1/13/19 5:23 PM
4   * Author: Denis Artamonov
5   *
6   * Пример контроллера
7   */
8
9  namespace Artamonov\Rest\Controllers\Native;
10
11
12  class Example
13  {
14      public function __construct()
15      {
16          if (!config()->get('useExampleRoute')) {
17              response()->json('Showing examples is disabled in the settings');
18          }
19      }
20
21      public function _get()
22      {
23          $this->response(__FUNCTION__);
24      }
25
26      public function _post()
27      {
28          $this->response(__FUNCTION__);
29      }
30
31      public function _put()
32      {
33          $this->response(__FUNCTION__);
34      }
35
36      public function _delete()
37      {
38          $this->response(__FUNCTION__);
39      }
40
41      public function _head()
42      {
43          $this->response(__FUNCTION__);
44      }
45
46      private function response($action)
47      {
48          $response = [
49              'message' => 'Запрос выполнен успешно',
50              'date' => date('Y-m-d H:i:s'),
51              'controller' => __CLASS__,
52              'action' => $action,
53              'method' => request()->method(),
54              'header' => request()->header(),
55              'request' => request()->get(),
56              'server' => $_SERVER
57          ];
58          journal()->add('request-response', ['request' => request()->get(), 'response' =>
$response]);

```

```
59         response()->json($response);  
60     }  
61 }
```

Namespace контроллеры

Для использования контроллеров из собственной директории, необходимо:

1. Создать директорию, например, **/local/api/controllers/**;
2. Создать файл с классом, например, **/local/api/controllers/product.php**;
3. Указать для класса корректный **namespace** (согласно PSR);

PHP

```
1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 3/18/19 12:13 PM
4   * Author: Denis Artamonov
5   */
6
7  namespace Local\Api\Controllers;
8
9
10 class Product
11 {
12     public function __construct()
13     {
14         // code ...
15     }
16
17     public function _get()
18     {
19         // code ...
20     }
21
22     public function _post()
23     {
24         // code ...
25     }
26
27     public function _put()
28     {
29         // code ...
30     }
31
32     public function _patch()
33     {
34         // code ...
35     }
36
37     public function _delete()
38     {
39         // code ...
40     }
41
42     public function _head()
43     {
44         // code ...
45     }
46 }
```

4. Указать в карте роутов, в качестве **контроллера**, собственный класс через его **namespace**.

PHP

```

1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 3/18/19 12:19 PM
4   * Author: Denis Artamonov
5   */
6
7  return [
8      'GET' => [
9          'product/get' => [
10             'controller' => '\\Local\\Api\\Controllers\\Product@_get'
11         ]
12     ],
13     'POST' => [
14         'product/post' => [
15             'controller' => '\\Local\\Api\\Controllers\\Product@_post'
16         ]
17     ],
18     'PUT' => [
19         'product/put' => [
20             'controller' => '\\Local\\Api\\Controllers\\Product@_put'
21         ]
22     ],
23     'PATCH' => [
24         'product/patch' => [
25             'controller' => '\\Local\\Api\\Controllers\\Product@_patch'
26         ]
27     ],
28     'DELETE' => [
29         'product/delete' => [
30             'controller' => '\\Local\\Api\\Controllers\\Product@_delete'
31         ]
32     ],
33     'HEAD' => [
34         'product/head' => [
35             'controller' => '\\Local\\Api\\Controllers\\Product@_head'
36         ]
37     ]
38 ];

```

Файловые контроллеры

Для использования контроллеров из файлов, необходимо:

1. Создать директорию, например, **/local/api/controllers/**;
2. Создать файл с необходимым кодом, например, **/local/api/controllers/product.php**;

PHP

```
1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 3/18/19 12:37 PM
4   * Author: Denis Artamonov
5   */
6
7  // Массив в который формируем ответ
8  $response = [];
9
10 // code ...
11
12 // Возвращаем ответ клиенту
13 response->json($response);
```

3. Указать в карте роутов, в качестве контроллера, путь к PHP-скрипту.

PHP

```
1  /**
2   * Copyright (c) 2019 Denis Artamonov
3   * Created: 3/18/19 12:43 PM
4   * Author: Denis Artamonov
5   */
6
7  return [
8      'GET' => [
9          'product/get' => [
10             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
11         ]
12     ],
13     'POST' => [
14         'product/post' => [
15             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
16         ]
17     ],
18     'PUT' => [
19         'product/put' => [
20             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
21         ]
22     ],
23     'PATCH' => [
24         'product/patch' => [
25             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
26         ]
27     ],
28     'DELETE' => [
29         'product/delete' => [
30             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
31         ]
32     ],
33     'HEAD' => [
34         'product/head' => [
35             'controller' => $_SERVER['DOCUMENT_ROOT'] . '/local/api/controllers/product.php'
36         ]
37     ]
38 ];
```

Токены авторизации

В модуле предусмотрено два вида авторизации:

- с помощью логина и пароля пользователя
- с помощью токена

В данном случае будет рассмотрена авторизация с помощью токена.

Для возможности авторизации по токену, необходимо выполнить условия:

1. Включить Авторизацию по токену в настройках модуля - раздел Безопасность;
2. Сгенерировать токены для пользователей - раздел Безопасность;
3. Рекомендуется задать общую Ключевую фразу для токена - раздел Безопасность;
4. Настроить авторизацию в нужных роутах - Карта роутов. Ключи `['security']['auth']['required']` и `['security']['auth']['type']` - пример карты;
5. Со стороны клиента, при запросе, передать заголовок Authorization-Token со значением токена, а именно {ключевая фраза}:{токен}. Ключевая фраза указывается при наличии.

После такой настройки, на нужных роутах будет доступна авторизация по токену. И доступ к данным будет предоставлен только после проверки токена.

Также в модуле имеется класс для работы с токенами, ниже представлено описание методов класса.

PHP

```

1  use \Bitrix\Main\Loader;
2  use \Bitrix\Main\LoaderException;
3  use \Artamonov\Rest\Foundation\Token;
4
5  try {
6
7      Loader::includeModule('artamonov.rest');
8
9      $token = Token::getInstance();
10     $userId = 1;
11     $clientToken = '8f757146-097eb57b-b07d669e-112463ff';
12
13     // Возвращает информацию по токену.
14     // То есть, данные пользователя, которому принадлежит токен.
15     // Используется метод UserTable::getList().
16     $result = $token->getData($clientToken);
17
18     // Возвращает данные токена по ID пользователя
19     $result = $token->getDataByUserId($userId);
20
21     // Возвращает срок годности токена
22     $result = $token->getExpirationDate($clientToken);
23
24     // Обновляет срок годности токена, согласно настройкам Безопасности модуля
25     $result = $token->updateExpirationDate($clientToken);
26
27     // Удаляет токен
28     $result = $token->delete($clientToken);
29
30     // Генерация токенов для пользователей.
31     // Без переданных параметров, будут сгенерированы токены для тех пользователей, у кого токены
32     // отсутствуют.
33     // Поддерживается параметр $parameters - все те же ключи, что и для метода
34     // UserTable::getList().
35     // И параметр $update со значениями true | false - указывается на то, обновлять уже созданные
36     // токены или же создать новые токены, для тех у кого их нет.
37     $result = $token->generate();
38
39     // Пример обновления токенов для пользователей, у которых ID больше 30
40     $parameters['filter'] = [
41         '>ID' => 30
42     ];
43     $update = true;
44     $result = $token->generate($parameters, $update);
45 } catch (LoaderException $e) {
46 }

```

REST-документация

В модуле поддерживается два вида документации:

1. В административном разделе.
2. В публичной части.

Настройка роутов

Каждый конкретный роут можно исключить из необходимой документации. Настраивается такое поведение через карту роутов.

PHP

```

1  // Пример ключа настройки документации для роута
2  // ...
3  'documentation' => [
4      // Исключить роут
5      'exclude' => [
6          // Из документации в административной части сайта
7          'admin' => false, // true || false
8          // Из документации в публичной части сайта
9          'public' => false, // true || false
10     ]
11 ]
12 // ...

```

Активация документации в публичной части сайта

1. Для документации зарезервирован нативный роут: `{api_path}/docs`
2. Поэтому, чтобы получить документацию, необходимо либо активировать нативные роуты в настройках модуля, либо скопировать код контроллера в собственный, необходимый контроллер. Код нативного контроллера документации расположен в папке модуля:
`/lib/controllers/native/documentation.php`
3. Важный момент: документация будет возвращена в json-формате. Такой подход был выбран для того, чтобы было больше гибкости. Например, в таком формате можно передать документацию клиентам, которые отобразят документацию в удобном для себя виде, используя полученные данные.

```

1 {
2   "GET": {
3     "example/check": {
4       "active": true,
5       "security": {
6         "auth": {
7           "required": false,
8           "type": "token"
9         }
10      },
11      "parameters": {
12        "iblock_id": {
13          "type": "integer",
14          "possibleValue": [
15            "1",
16            "...",
17            "50"
18          ],
19          "description": "ID инфоблока"
20        },
21        "active": {
22          "type": "string",
23          "possibleValue": [
24            "Y",
25            "N"
26          ],
27          "description": "Активность"
28        },
29        "name": {
30          "type": "string",
31          "description": "Имя элемента"
32        },
33        "color": {
34          "type": "string",
35          "possibleValue": [
36            "Белый",
37            "Красный"
38          ],
39          "description": "Свойство: Цвет"
40        },
41        "form": {
42          "type": "string",
43          "possibleValue": [
44            "Круглый",
45            "Квадрат"
46          ],
47          "description": "Свойство: Форма"
48        },
49        "fields": {
50          "type": "string",
51          "possibleValue": [
52            "preview_text",
53            "detail_text",
54            "preview_picture",
55            "form"
56          ],
57          "description": "Дополнительные поля"
58        },
59        "sort": {
60          "type": "string",
61          "description": "Сортировка (пример: sort=id:asc,name:asc)"
62        },
63        "limit": {
64          "type": "integer",
65          "possibleValue": [
66            "1",
67            "...",
68            "150"
69          ],
70          "description": "Количество (по умолчанию: 10)"
71        },
72        "page": {
73          "type": "integer",
74          "description": "Страница"
75        }
76      }
77    },
78    "/example/check/iblock/{{iblockId}}/": {
79      "parameters": {
80        "iblockId": {
81          "required": true,
82          "type": "integer",
83          "description": "ID инфоблока"
84        }
85      }
86    },
87    "example/check/section/{{sectionId}}": {
88      "parameters": {
89        "sectionId": {
90          "required": true,
91          "type": "integer",
92          "description": "ID раздела"
93        }
94      }
95    }
96  }
97 }

```

Javascript-запрос

JavaScript

```

1  // https://developer.mozilla.org/ru/docs/Web/API/Fetch_API/Using_Fetch
2
3  'use strict';
4
5  // Адрес API
6  let url = 'https://artamonov.rest:8888/api/example/check';
7  // Объект настроек запроса
8  let init;
9
10 // ----- GET
11 // Пример запроса без использования авторизации
12
13 fetch(url)
14   .then(function (response) {
15     return response.json();
16   })
17   .then(function (json) {
18     console.log(json);
19   })
20   .catch(function (error) {
21     console.error(error);
22   });
23
24 // ----- GET
25 // Пример запроса с использованием авторизации
26
27 init = {
28   method: 'GET',
29   headers: new Headers({
30     // Авторизация по токену
31     'Authorization-Token': 'TOKEN_KEYWORD:05516ee7-12fe7d60-f51286cc-76522088',
32     // Или
33     // Авторизация по логину и паролю
34     'Authorization-Login': 'artamonov',
35     'Authorization-Password': 'w@3d32@#!hh5'
36   }),
37   mode: 'cors',
38   cache: 'default'
39 };
40
41 fetch(url, init)
42   .then(function (response) {
43     return response.json();
44   })
45   .then(function (json) {
46     console.log(json);
47   })
48   .catch(function (error) {
49     console.error(error);
50   });
51
52 // ----- POST
53 // Пример запроса с использованием авторизации и отправка тела запроса в формате Json
54
55 init = {
56   method: 'POST',

```

```
57     headers: new Headers({
58         // Авторизация по токену
59         'Authorization-Token': 'TOKEN_KEYWORD:05516ee7-12fe7d60-f51286cc-76522088',
60         // Или
61         // Авторизация по логину и паролю
62         'Authorization-Login': 'artamonov',
63         'Authorization-Password': 'w@3d32@#!hh5',
64         // Json
65         'Content-Type': 'application/json'
66     }),
67     mode: 'cors',
68     cache: 'no-cache',
69     body: JSON.stringify({
70         'param1': 'value1',
71         'param2': 'value2',
72         'param3': 'value3',
73         'param4': {
74             'param5': 'value5',
75             'param6': 'value6',
76         },
77     })
78 };
79
80 fetch(url, init)
81     .then(function (response) {
82         return response.json();
83     })
84     .then(function (json) {
85         console.log(json);
86     })
87     .catch(function (error) {
88         console.error(error);
89     });
```

PHP-запрос

PHP

```

1  // Адрес API
2  $url = 'https://artamonov.rest:8888/api/example/check';
3
4  // ----- GET
5  // Пример запроса без использования авторизации
6
7  $response = file_get_contents($url);
8  $response = json_decode($response, true);
9
10 echo '<pre>' . print_r($response, true) . '</pre>';
11
12 // ----- GET
13 // Пример запроса с использованием авторизации
14
15 $curl = curl_init();
16 curl_setopt_array($curl, [
17     CURLOPT_URL => $url,
18     CURLOPT_RETURNTRANSFER => true,
19     CURLOPT_HTTPHEADER => [
20         // Авторизация по токену
21         'Authorization-Token: TOKEN_KEYWORD:05516ee7-12fe7d60-f51286cc-76522088',
22         // Или
23         // Авторизация по логину и паролю
24         'Authorization-Login: artamonov',
25         'Authorization-Password: w@3d32@#!hh5',
26     ]
27 ]);
28 $response = curl_exec($curl);
29 $info = curl_getinfo($curl);
30 curl_close($curl);
31
32 $response = json_decode($response, true);
33
34 echo '<pre>' . print_r($response, true) . '</pre>';
35 echo '<pre>' . print_r($info, true) . '</pre>';
36
37 // ----- POST
38 // Пример запроса с использованием авторизации и отправка тела запроса в формате Json
39
40 $curl = curl_init();
41 curl_setopt_array($curl, [
42     CURLOPT_URL => $url,
43     CURLOPT_RETURNTRANSFER => true,
44     CURLOPT_HTTPHEADER => [
45         // Авторизация по токену
46         'Authorization-Token: TOKEN_KEYWORD:05516ee7-12fe7d60-f51286cc-76522088',
47         // Или
48         // Авторизация по логину и паролю
49         'Authorization-Login: artamonov',
50         'Authorization-Password: w@3d32@#!hh5',
51         // Json
52         'Content-Type: application/json'
53     ],
54     CURLOPT_CUSTOMREQUEST => 'POST',
55     CURLOPT_POSTFIELDS => json_encode([
56         'param1' => 'value1',

```

```
57         'param2' => 'value2',
58         'param3' => 'value3',
59         'param4' => [
60             'param5' => 'value5',
61             'param6' => 'value6',
62         ],
63     ]),
64 ];
65 $response = curl_exec($curl);
66 $info = curl_getinfo($curl);
67 curl_close($curl);
68
69 $response = json_decode($response, true);
70
71 echo '<pre>' . print_r($response, true) . '</pre>';
72 echo '<pre>' . print_r($info, true) . '</pre>';
```

Роут-провайдер

Данный роут помогает получить доступ к штатному АПИ платформы.

1. Необходимо активировать нативные роуты.
2. Для выполнения запроса к нативному роуту-провайдеру, также необходимо использовать авторизацию по токену.
3. Либо, можно переопределить роут-провайдер через собственные карты роутов, и сконфигурировать его под конкретные нужды проекта.

При штатных настройках, для взаимодействия с роутом, необходимо послать GET-запрос с нужным классом и методом, чтобы увидеть какие параметры готов принять метод указанного класса.

Далее, будет рассмотрен пример по получению заказов.

Безопасность

Роут-провайдер вызывает произвольные классы и методы платформы, поэтому по умолчанию защищён несколькими уровнями:

- работает только при включённых нативных роутах (по умолчанию они выключены);
- требует авторизацию по токену;
- имеет встроенный чёрный список опасных операций (прямые запросы к базе, файловые операции и заливка файлов, управление пользователями и группами, внешние HTTP-запросы, рассылки и другие высокоимпактные векторы).

Важно: чёрный список для универсального вызова любого класса принципиально не может быть полным.

Для production-среды настоятельно рекомендуется включить режим белого списка — тогда доступны только явно разрешённые классы и методы, а всё остальное запрещается.

Белый список включается переопределением роута собственным классом-наследником — достаточно вернуть непустой список в `getAllowClasses()` и/или `getAllowMethods()` :

PHP

```
1 namespace App\Rest;
2
3 class Provider extends \Artamonov\Rest\Controllers\Native\Provider
4 {
5     protected function getAllowClasses()
6     {
7         return ['\Bitrix\Sale\Order'];
8     }
9
10    protected function getAllowMethods()
11    {
12        return ['CSaleOrder' => ['GetList']];
13    }
14 }
```

Как только любой из списков непустой, провайдер работает строго по белому списку (чёрный список при этом не применяется).

Если провайдер не нужен — держите нативные роуты выключенными или перекройте/удалите этот роут в своей карте.

Получение информации по классу и методу

SQL

```
1 [GET] https://site.com/api/provider?class=CSaleOrder&method=GetList
```

Name	Type	Description
class *	String	Класс модуля
method *	String	Метод класса

200: Информация по классу и методу

JSON

```

1  {
2      "class": {
3          "name": "CSaleOrder",
4          "file": "C:\\WebServer\\module.loc\\bitrix\\modules\\sale\\mysql\\order.php"
5      },
6      "method": {
7          "name": "GetList",
8          "doc": "/*\n\t * @param array $arOrder\n\t * @param array $arFilter\n\t * @param
bool|array $arGroupBy\n\t * @param bool|array $arNavStartParams\n\t * @param array $arSelectFields\n\t *
@param array $arOptions\n\t * @return bool|CDBResult\n\t */",
9          "deprecated": false,
10         "static": true,
11         "parameters": {
12             "arOrder": {
13                 "name": "arOrder",
14                 "defaultValue": {
15                     "ID": "DESC"
16                 },
17                 "isPassedByReference": false
18             },
19             "arFilter": {
20                 "name": "arFilter",
21                 "defaultValue": [],
22                 "isPassedByReference": false
23             },
24             "arGroupBy": {
25                 "name": "arGroupBy",
26                 "defaultValue": false,
27                 "isPassedByReference": false
28             },
29             "arNavStartParams": {
30                 "name": "arNavStartParams",
31                 "defaultValue": false,
32                 "isPassedByReference": false
33             },
34             "arSelectFields": {
35                 "name": "arSelectFields",
36                 "defaultValue": [],
37                 "isPassedByReference": false
38             },
39             "arOptions": {
40                 "name": "arOptions",
41                 "defaultValue": [],
42                 "isPassedByReference": false
43             }
44         }
45     }
46 }

```

Теперь, когда мы видим, какие параметры готов принять метод указанного класса, мы можем выполнить запрос для получения данных.

Для выполнения запроса необходимо отправлять POST-запрос, в теле которого будут переданы параметры запроса.

Тело запроса

JSON

```
1  {
2      "class": "CSaleOrder",
3      "method": "GetList",
4      "parameters": {
5          "arFilter": {
6              "LID": "s1"
7          }
8      }
9  }
```

Получение данных используя класс и метод

SQL

```
1  [POST] https://site.com/api/provider
```

Headers

Name	Type	Description
Content-Type *	String	application/json

Request Body

Name	Type	Description
class *	String	Класс модуля
method *	String	Метод класса
parameters *	Object	Параметры для метода

200: Список заказов

JSON

```

1  [
2      {
3          "ID": "2",
4          "LID": "s1",
5          "ACCOUNT_NUMBER": "2",
6          "TRACKING_NUMBER": "",
7          "PAY_SYSTEM_ID": "1",
8          "DELIVERY_ID": "new1:profile",
9          "DATE_INSERT": "13.07.2023 16:41:16",
10         "DATE_UPDATE": "13.07.2023 16:41:16",
11         "PERSON_TYPE_ID": "0",
12         "USER_ID": "1",
13         "PAYED": "N",
14         "IS_SYNC_B24": "N",
15         "DATE_PAYED": null,
16         "EMP_PAYED_ID": null,
17         "DEDUCTED": "N",
18         "DATE_DEDUCTED": null,
19         "EMP_DEDUCTED_ID": null,
20         "REASON_UNDO_DEDUCTED": null,
21         "STATUS_ID": "N",
22         "DATE_STATUS": "13.07.2023 16:41:15",
23         "EMP_STATUS_ID": "1",
24         "MARKED": "N",
25         "DATE_MARKED": null,
26         "EMP_MARKED_ID": null,
27         "REASON_MARKED": null,
28         "PRICE_DELIVERY": "0",
29         "ALLOW_DELIVERY": "N",
30         "DATE_ALLOW_DELIVERY": null,
31         "EMP_ALLOW_DELIVERY_ID": null,
32         "RESERVED": "N",
33         "PRICE": "19350",
34         "CURRENCY": "RUB",
35         "DISCOUNT_VALUE": "0",
36         "TAX_VALUE": "0.00",
37         "SUM_PAID": "0.00",
38         "USER_DESCRIPTION": "",
39         "PAY_VOUCHER_NUM": "",
40         "PAY_VOUCHER_DATE": null,
41         "ADDITIONAL_INFO": null,
42         "COMMENTS": "",
43         "COMPANY_ID": null,
44         "CREATED_BY": "1",
45         "RESPONSIBLE_ID": null,
46         "STAT_GID": null,
47         "DATE_PAY_BEFORE": null,
48         "DATE_BILL": null,
49         "IS_RECURRING": "N",
50         "RECURRING_ID": null,
51         "LOCKED_BY": null,
52         "DATE_LOCK": null,
53         "RECOUNT_FLAG": "Y",
54         "AFFILIATE_ID": null,
55         "DELIVERY_DOC_NUM": null,
56         "DELIVERY_DOC_DATE": null,
57         "UPDATED_1C": "N",
58         "ORDER_TOPIC": null,
59         "XML_ID": "bx_64affefbefbb1",

```

```

60      "ID_1C": null,
61      "VERSION_1C": null,
62      "VERSION": "0",
63      "EXTERNAL_ORDER": "N",
64      "STORE_ID": null,
65      "CANCELED": "N",
66      "EMP_CANCELED_ID": null,
67      "DATE_CANCELED": null,
68      "REASON_CANCELED": null,
69      "BX_USER_ID": "3f61d5bdbd4e0835463b75bf7cfe5e36",
70      "SEARCH_CONTENT": null,
71      "RUNNING": "N",
72      "DATE_INSERT_FORMAT": "13.07.2023 16:41:16",
73      "DATE_UPDATE_SHORT": "13.07.2023",
74      "DATE_STATUS_SHORT": "13.07.2023",
75      "DATE_CANCELED_SHORT": null,
76      "BY_RECOMMENDATION": "N",
77      "LOCK_STATUS": "green",
78      "LOCK_USER_NAME": null,
79      "RESPONSIBLE_LOGIN": null,
80      "RESPONSIBLE_NAME": null,
81      "RESPONSIBLE_LAST_NAME": null,
82      "RESPONSIBLE_SECOND_NAME": null,
83      "RESPONSIBLE_EMAIL": null,
84      "RESPONSIBLE_WORK_POSITION": null,
85      "RESPONSIBLE_PERSONAL_PHOTO": null,
86      "USER_LOGIN": "admin",
87      "USER_NAME": "Денис",
88      "USER_LAST_NAME": "Артамонов",
89      "USER_EMAIL": "test@site.com"
90    },
91    {
92      "ID": "1",
93      "LID": "s1",
94      "ACCOUNT_NUMBER": "1",
95      "TRACKING_NUMBER": "",
96      "PAY_SYSTEM_ID": "1",
97      "DELIVERY_ID": "new1:profile",
98      "DATE_INSERT": "13.07.2023 16:41:03",
99      "DATE_UPDATE": "13.07.2023 16:41:03",
100     "PERSON_TYPE_ID": "0",
101     "USER_ID": "1",
102     "PAYED": "N",
103     "IS_SYNC_B24": "N",
104     "DATE_PAYED": null,
105     "EMP_PAYED_ID": null,
106     "DEDUCTED": "N",
107     "DATE_DEDUCTED": null,
108     "EMP_DEDUCTED_ID": null,
109     "REASON_UNDO_DEDUCTED": null,
110     "STATUS_ID": "N",
111     "DATE_STATUS": "13.07.2023 16:41:02",
112     "EMP_STATUS_ID": "1",
113     "MARKED": "N",
114     "DATE_MARKED": null,
115     "EMP_MARKED_ID": null,
116     "REASON_MARKED": null,
117     "PRICE_DELIVERY": "0",
118     "ALLOW_DELIVERY": "N",
119     "DATE_ALLOW_DELIVERY": null,
120     "EMP_ALLOW_DELIVERY_ID": null,
121     "RESERVED": "N",

```

```

122     "PRICE": "19350",
123     "CURRENCY": "RUB",
124     "DISCOUNT_VALUE": "0",
125     "TAX_VALUE": "0.00",
126     "SUM_PAID": "0.00",
127     "USER_DESCRIPTION": "",
128     "PAY_VOUCHER_NUM": "",
129     "PAY_VOUCHER_DATE": null,
130     "ADDITIONAL_INFO": null,
131     "COMMENTS": "",
132     "COMPANY_ID": null,
133     "CREATED_BY": "1",
134     "RESPONSIBLE_ID": null,
135     "STAT_GID": null,
136     "DATE_PAY_BEFORE": null,
137     "DATE_BILL": null,
138     "IS_RECURRING": "N",
139     "RECURRING_ID": null,
140     "LOCKED_BY": null,
141     "DATE_LOCK": null,
142     "RECOUNT_FLAG": "Y",
143     "AFFILIATE_ID": null,
144     "DELIVERY_DOC_NUM": null,
145     "DELIVERY_DOC_DATE": null,
146     "UPDATED_1C": "N",
147     "ORDER_TOPIC": null,
148     "XML_ID": "bx_64affeeaa677",
149     "ID_1C": null,
150     "VERSION_1C": null,
151     "VERSION": "0",
152     "EXTERNAL_ORDER": "N",
153     "STORE_ID": null,
154     "CANCELED": "N",
155     "EMP_CANCELED_ID": null,
156     "DATE_CANCELED": null,
157     "REASON_CANCELED": null,
158     "BX_USER_ID": "3f61d5bdbd4e0835463b75bf7cfe5e36",
159     "SEARCH_CONTENT": null,
160     "RUNNING": "N",
161     "DATE_INSERT_FORMAT": "13.07.2023 16:41:03",
162     "DATE_UPDATE_SHORT": "13.07.2023",
163     "DATE_STATUS_SHORT": "13.07.2023",
164     "DATE_CANCELED_SHORT": null,
165     "BY_RECOMMENDATION": "N",
166     "LOCK_STATUS": "green",
167     "LOCK_USER_NAME": null,
168     "RESPONSIBLE_LOGIN": null,
169     "RESPONSIBLE_NAME": null,
170     "RESPONSIBLE_LAST_NAME": null,
171     "RESPONSIBLE_SECOND_NAME": null,
172     "RESPONSIBLE_EMAIL": null,
173     "RESPONSIBLE_WORK_POSITION": null,
174     "RESPONSIBLE_PERSONAL_PHOTO": null,
175     "USER_LOGIN": "admin",
176     "USER_NAME": "Денис",
177     "USER_LAST_NAME": "Артамонов",
178     "USER_EMAIL": "test@site.com"
179 }
180 ]

```

Используемые ресурсы

<https://marketplace.1c-bitrix.ru/solutions/artamonov.rest/#tab-log-link>