Оглавление

[Состав 2](#_Toc205555917)

[Порядок тестирования 3](#_Toc205555918)

[Примечания 7](#_Toc205555919)

## Состав

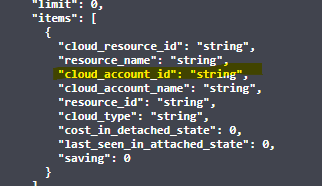
API Environment.json.postman\_environment - переменные окружения

Tests.postman\_collection - коллекция

## Порядок тестирования

* + Сначала выполнить Setup-тесты для получения валидных значений

Примечание:

Поскольку в ответе контракта отсутствует заявленное поле, а именно массив items с предоставленным cloud\_account\_id  


То выполняется GET-request к эндпоинту /restapi/v2/organizations/{{organization\_id}}/cloud\_accounts

При обращении к эндпоинту /restapi/v2/organizations/{{organization\_id}}/optimizations без параметров, ответ не содержит необходимый массив items, так как по дефолту параметр overview=false, для получения массива items overview == true, тогда получает необходимый массив данных.  
но так как используем дефолт, то логичнее будет обратиться к другому запросу, который отдаст полученные id

* + Затем выполнить позитивные тесты
  + После этого выполнить негативные тесты
  + В конце проверить аутентификацию и обработку ошибок

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Категория | Папка//Номер теста | Наименование теста | Цель | Эндпоинт | Ожидаемый результат | Критерий |
| Подготовка | SETUP-00 | Получение cloud\_accounts | Получить список облачных аккаунтов для использования в тестах | GET /cloud\_accounts | Статус 200, список cloud\_accounts | 1. Ответ содержит массив cloud\_accounts 2. Каждый аккаунт имеет id, name, type 3. Устанавливается переменная valid\_account\_id |
| Подготовка | SETUP-00 | Получение валидного типа оптимизации | Определить существующий тип оптимизации для использования в тестах | GET /optimizations | Статус 200, данные об оптимизациях | 1. Ответ содержит объект optimizations 2. Объект содержит хотя бы один тип оптимизации 3. Устанавливается переменная valid\_type |
| Позитивные тесты | POS-01 | Базовый запрос без параметров | Проверить успешное выполнение запроса без параметров | GET /optimizations | Статус 200, полные данные об оптимизациях | 1. Статус ответа 200 2. Ответ содержит все обязательные поля (total\_saving, optimizations, и т.д.) 3. Все поля имеют правильный тип данных 4. Оптимизации имеют корректную структуру (count, saving) 5. Временные метки логичны (last\_run < next\_run) 6. total\_saving приблизительно равен сумме всех saving |
| Позитивные тесты | POS-02 | Запрос с overview=true | Проверить формат ответа с overview | GET /optimizations?overview=true | Статус 200, данные в overview-формате | 1. Статус ответа 200 2. Проверить, что ответ в формате JSON  3. Проверить, что тело – json объект 4. Проверить структуру items 5. Проверить согласованность данных – items == null |
| Позитивные тесты | POS-03 | Запрос с валидными status и limit | Проверить фильтрацию по status и ограничение limit | GET /optimizations?type={valid\_type}&status={valid\_status}&limit=2 | Статус 200, ограниченное количество записей | 1. Статус ответа 200 2. Все оптимизации имеют указанный status 3. Количество записей соответствует limit |
| Позитивные тесты | POS-04 | Запрос с фильтрацией по cloud\_account\_id | Проверить фильтрацию по облачному аккаунту | GET /optimizations?cloud\_account\_id={valid\_account\_id}  UPD: корректнее для данной проверки использовать &overview=true с параметром для возврата массива | Статус 200, данные только для указанного аккаунта | 1. Статус ответа 200 2. Все записи относятся к указанному cloud\_account\_id |
| Негативные тесты | NEG-05 | Невалидный organization\_id | Проверить обработку невалидного UUID организации | GET /organizations/invalid-id/optimizations | Статус 404, ошибка формата | 1. Статус ответа 404 2. Сообщение об ошибке содержит "Invalid organization\_id format" |
| Негативные тесты | NEG-06 | Type без Status | Проверить обязательность параметра status при указании type | GET /optimizations?type={valid\_type} | Статус 400, ошибка валидации | 1. Статус ответа 400 2. Сообщение об ошибке содержит "status is required when type is specified"  UPD: параметр статуса необязательный, как следствие тест FAILED |
| Негативные тесты | NEG-07 | Limit=0 | Проверить обработку недопустимого значения limit | GET /optimizations?type={valid\_type}&status={valid\_status}&limit=0 | Статус 400, ошибка валидации | 1. Статус ответа 400 2. Код ошибки OE0224 3. Сообщение: "Value of "limit" should be between 1 and 2147483647" 4. Параметры ошибки: ["limit", 1, 2147483647] |
| Негативные тесты | NEG-08 | Невалидный Status | Проверить обработку недопустимого значения status | GET /optimizations?type={valid\_type}&status=invalid\_status | Статус 400, ошибка валидации | 1. Статус ответа 400 2. Код ошибки OE0460 3. Сообщение: "Status should be active, dismissed or excluded" |
| Негативные тесты | NEG-09 | Невалидный Overview | Проверить обработку недопустимого значения overview | GET /optimizations?overview=invalid | Статус 401, ошибка аутентификации | 1. Статус ответа 401 2. Код ошибки OE0235 3. Сообщение: "Unauthorized" |
| Аутентификация и граничные случаи | AUTH-10 | Запрос без токена | Проверить обработку отсутствующего токена | GET /optimizations (без Authorization) | Статус 401, ошибка аутентификации | 1. Статус ответа 401 2. Код ошибки OE0237 3. Сообщение: "This resource requires authorization" |
| Аутентификация и граничные случаи | EDGE-11 | Несуществующий тип оптимизации | Проверить обработку несуществующего типа | GET /optimizations?type=non\_existent\_type&status=active | Статус 200, пустой результат | 1. Статус ответа 200 2. Ответ содержит пустой массив items |
| Аутентификация и граничные случаи | EDGE-12 | Limit=1 (граничное значение) | Проверить обработку минимального значения limit |  |  |  |

Примечания **Тест POS-01: Базовый запрос без параметров**

1. Статус ответа 200
   * Проверка: код HTTP-ответа равен 200
2. Обязательные поля в ответе
   * Проверка: наличие полей: total\_saving, optimizations, dismissed\_optimizations, excluded\_optimizations, deleted\_at, id, created\_at, organization\_id, last\_run, next\_run, last\_completed
3. Типы данных
   * Проверка:
     + total\_saving - число
     + optimizations, dismissed\_optimizations, excluded\_optimizations - объекты
     + id, organization\_id - строки (UUID)
     + created\_at, last\_run, next\_run, last\_completed - числа (timestamp)
4. Структура оптимизаций
   * Проверка:
     + Для каждого типа: наличие count (число ≥ 0)
     + Для большинства типов (кроме inactive\_console\_users, inactive\_users, insecure\_security\_groups, s3\_public\_buckets): наличие saving (число ≥ 0)
     + При наличии error: строка
5. Временные метки
   * Проверка:
     + created\_at, last\_run, last\_completed < текущее время
     + next\_run > last\_run
6. Логическая проверка
   * Проверка: total\_saving приблизительно равен сумме всех saving из optimizations

**Тест POS-02: Запрос с overview=true**

1. Статус ответа 200
2. Проверить, что ответ в формате JSON
3. Проверить, что тело – json объект
4. Проверить структуру items

* Проверяется структуруа массива items, который содержит данные акк

1. Проверить согласованность данных

* Проверка, что items == null

**Тест NEG-03: Limit=0**

1. Структура ошибки
   * Проверка: наличие структуры error с полями status\_code, error\_code, reason, params
2. Код ошибки
   * Проверка: error\_code = "OE0224"
3. Сообщение об ошибке
   * Проверка: reason содержит "Value of "limit" should be between 1 and 2147483647"
4. Параметры ошибки
   * Проверка: params = ["limit", 1, 2147483647]

**Тест NEG-04: Невалидный Status**

1. Структура ошибки
   * Проверка: наличие структуры error
2. Код ошибки
   * Проверка: error\_code = "OE0460"
3. Сообщение об ошибке
   * Проверка: reason = "Status should be active, dismissed or excluded"

**Тест SETUP-01: Получение cloud\_accounts**

1. Структура ответа
   * Проверка: наличие массива cloud\_accounts
2. Структура аккаунта
   * Проверка: каждый аккаунт имеет id, name, type
3. Установка переменной
   * Проверка: установка переменной valid\_account\_id со значением первого аккаунта

**Тест SETUP-02: Получение валидного типа оптимизации**

1. Структура ответа
   * Проверка: наличие объекта optimizations
2. Валидность данных
   * Проверка: наличие поля saving (число) для первого типа оптимизации
3. Установка переменных
   * Проверка: установка переменных valid\_type и valid\_saving

**Рекомендации по выполнению тестов**

1. Порядок выполнения:
   * Сначала выполните тесты из категории "Подготовка" (SETUP)
   * Затем выполните позитивные тесты (POS)
   * После этого выполните негативные тесты (NEG)
   * В конце выполните тесты аутентификации и граничные случаи (AUTH/EDGE)
2. Переменные окружения:
   * Убедитесь, что в окружении заданы: base\_url, organization\_id, token
   * После выполнения SETUP-тестов должны быть установлены: valid\_account\_id, valid\_type, valid\_status
3. Отладка:
   * При возникновении ошибок проверьте:
     + Корректность токена аутентификации
     + Наличие данных в демо-организации
     + Соответствие структуры ответа ожидаемой
   * Используйте Postman Console для просмотра логов
4. Особые случаи:
   * Для типов оптимизаций inactive\_console\_users, inactive\_users, insecure\_security\_groups, s3\_public\_buckets не ожидайте наличие поля saving
   * При проверке total\_saving допускайте небольшую погрешность из-за округления чисел с плавающей точкой