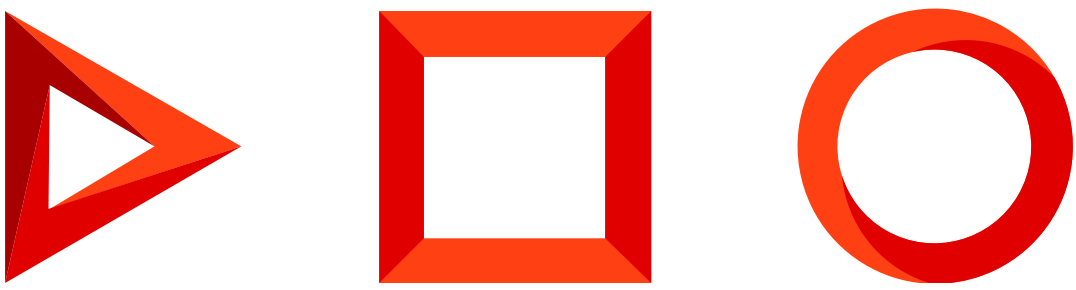
Creatio Academy

OData

Версия 8.0



Эта документация предоставляется с ограничениями на использование и защищена законами об интеллектуальной собственности. За исключением случаев, прямо разрешенных в вашем лицензионном соглашении или разрешенных законом, вы не можете использовать, копировать, воспроизводить, переводить, транслировать, изменять, лицензировать, передавать, распространять, демонстрировать, выполнять, публиковать или отображать любую часть в любой форме или посредством любые значения. Обратный инжиниринг, дизассемблирование или декомпиляция этой документации, если это не требуется по закону для взаимодействия, запрещены.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления и не может гарантировать отсутствие ошибок. Если вы обнаружите какие-либо ошибки, сообщите нам о них в письменной форме.

Содержание

OData

Сложный

OData (Open Data Protocol) — это утвержденный ISO/IEC стандарт OASIS, который определяет набор лучших практик для построения и использования REST API. Протокол позволяет создавать службы на основе REST, которые с помощью HTTP-запросов предоставляют веб-клиентам возможность публиковать и редактировать ресурсы, идентифицированные с использованием URL и определенные в модели данных.

Назначение протокола OData — выполнение запросов от внешних приложений к серверу баз данных Creatio.

Приложение Creatio поддерживает протоколы OData 4 и OData 3. По сравнению с OData 3, OData 4 предоставляет больше возможностей. Основное **отличие** протоколов — разный формат данных ответа на запрос, который возвращается сервером. Различия протоколов OData 3 и OData 4 описаны в официальной документации OData. При планировании интеграции с Creatio по протоколу OData необходимо использовать протокол версии 4.

Для выполнения внешних запросов к приложению Creatio необходимо выполнить **аутентификацию**. При интеграции по протоколу OData Creatio позволяет использовать Forms-аутентификации и OAuth-аутентификацию. Интеграция по протоколу OData не поддерживает аутентификацию по протоколу LDAP. Рекомендуемым способом аутентификации для интеграции с приложением по протоколу OData является **Forms-аутентификация**, которая реализована с помощью веб-сервиса `AuthService.svc`. Виды аутентификации, которые поддерживаются в Creatio, описаны в статье Аутентификация. Чтобы **настроить авторизацию по протоколу OAuth**, воспользуйтесь инструкцией, которая приведена в статье Настроить авторизацию интегрированных приложений по протоколу OAuth 2.0.

Протокол OData 4

Доступ к объектам Creatio по протоколу OData 4 предоставляет веб-сервис `odata`.

Адрес сервиса odata для .NET Framework

```
https://mycreatio.com/0/odata
```

Адрес сервиса odata для .NET Core

```
https://mycreatio.com/odata
```

Протокол OData 3

Доступ к объектам Creatio по протоколу OData 3 предоставляет веб-сервис `EntityDataService.svc`.

Адрес сервиса `EntityDataService.svc`

```
https://mycreatio.com/0/ServiceModel/EntityDataService.svc
```

Сервис `EntityDataService.svc` предоставляет возможность работы с объектами Creatio в WCF-клиенте.

Windows Communication Foundation (WCF) — программный фреймворк в составе .NET Framework, который используется для обмена данными между приложениями.

Организация **работы WCF-клиента** реализована путем получения метаданных сервиса и создания клиентских прокси-классов. Клиентское приложение использует эти классы-посредники для работы с сервисом `EntityDataService.svc` Creatio.

Чтобы **реализовать клиентское приложение**:

1. Создайте .NET-проект, в котором реализована интеграция с Creatio.
2. Сгенерируйте клиентские прокси-классы сервиса `EntityDataService.svc`.
3. Создайте экземпляр контекста среды выполнения сервиса `EntityDataService.svc`.
4. Реализуйте клиентскую бизнес-логику интеграции с использованием методов созданного экземпляра прокси-класса.

Способы генерации прокси-классов сервиса `EntityDataService.svc`:

- С использованием утилиты `DataServiceModel Metadata Utility Tool (DataSvcutil.exe)`.
- Из проекта клиентского приложения Visual Studio.

Генерировать прокси-классы с использованием утилиты DataServiceModel Metadata Utility Tool

`DataSvcUtil.exe` — программа командной строки, которая предоставлена сервисами `WCF Data Services`. Утилита использует канал OData для формирования клиентских классов службы данных, которые необходимы для доступа к службе данных из клиентского приложения .NET Framework.

Источники метаданных, которые утилита использует для формирования классов данных:

- `WSDL` — документ метаданных службы. Описывает модель данных, которая предоставлена службой данных.
- `CSDL` — файл модели данных. Используется язык определения концептуальной схемы (`CSDL`), который описан в спецификации [MC-CSDL]: формат файла определения концептуальной схемы.
- `EDMX` — *.xml-файл. Создается с помощью программ для работы с моделью `EDM`, которые входят в комплект `Entity Framework`. Дополнительные сведения описаны в спецификации [MC-EDMX]: модели EDM для формата упаковки служб данных.

Обычно утилита `DataSvcUtil.exe` установлена в каталоге `C:\Windows\Microsoft.NET\Framework\v4.0`. Для 64-разрядных версий систем это каталог `C:\Windows\Microsoft.NET\Framework64\v4.0`.

Формат вызова утилиты `DataSvcutil.exe`

```
datasvcutil /out:file [/in:file | /uri:serviceuri] [/dataservicecollection] [/language:devlang]
```

Детальная информация по утилите `DataSvcutil.exe` приведена в официальной документации MSDN.

Генерировать прокси-классы из проекта клиентского приложения Visual Studio

Чтобы сгенерировать прокси-классы из **проекта клиентского приложения Visual Studio**:

1. В контекстном меню проекта клиентского приложения Visual Studio, в котором планируется реализация интеграции с Creatio, выберите пункт [*Add Service Reference...*].
2. В поле [*Address*] введите полный адрес сервиса `EntityDataService.svc`.
3. Нажмите на кнопку [*Go*] и укажите имя и пароль пользователя Creatio. Если аутентификация прошла успешно, то в окне [*Services*] отобразятся поддерживаемые сервисом сущности.
4. В поле [*Namespace*] укажите имя пространства имен, в котором расположены сгенерированные прокси-классы (например, `CreatioServiceReference`). Ссылку на это пространство имен в дальнейшем добавьте в блок `using` кода проекта.
5. Для генерации прокси-классов нажмите кнопку [*OK*]. При этом в проект добавляется новый файл кода `Reference.cs`, содержащий описание прокси-классов, которые доступны к использованию для обращения и взаимодействия с ресурсами сервиса данных как с объектами.

Visual Studio предоставляет возможность генерации прокси-классы сервиса из сохраненного на диске **файла метаданных сервиса**. Для этого на шаге 2 в поле [*Address*] введите полный путь к файлу метаданных с префиксом `file://`.

```
file://C:/metadata.xml
```

После генерации прокси-классов сервиса в проект добавляется ссылка на сборку `Microsoft.Data.Services.Client.dll`, которая реализует поддержку протокола OData 3. Если в клиентском приложении необходимо использовать протокол более ранней версии, то ссылку на соответствующую сборку добавьте вручную. Данная клиентская библиотека позволяет выполнять запросы к сервису данных `EntityDataService.svc`, используя стандартные шаблоны программирования .NET Framework, включая использование языка запросов LINQ.

Чтобы **успешно компилировать проект**:

- В код проекта добавьте директивы `using`.

  Директивы `using`

  ```
  using System;
  using System.Data.Services.Client;
  ```

```
using System.Net;
using Terrasoft.Sdk.Examples.CreatioServiceReference;
using System.Linq;
```

- В код проекта добавьте объявление переменной адреса сервиса OData.

Объявление переменной адреса сервиса OData

```
private static Uri serverUri = new Uri("http://<имя_сервера>/<имя_приложения>/0/ServiceModel/
```

Особенности использования протокола OData

Особенности, которые необходимо учитывать при использовании протокола OData:

- Невозможно создать системных пользователей.
- Невозможно задать культуру возвращаемых данных. Культура определяется культурой пользователя от имени которого выполняется запрос.
- Тело ответа на запрос может содержать максимум 20 000 строк.
- Пакетный запрос может содержать максимум 100 подзапросов.
- Максимальный размер файла, который можно загрузить с помощью запроса, задается системной настройкой [*Максимальный размер загружаемого файла*] ([*Attachment max size*], код `MaxFileSize`) (по умолчанию — 10 Мб).

Решение типичных ошибок при интеграции по протоколу OData

При выполнении запросов по протоколу OData могут возникать ошибки, которые представлены в таблице ниже.

Типичные ошибки при интеграции по протоколу OData

Ошибка	Решение
Ошибка типа `{ "error": { "code": "", "message": "An error has occurred." } }`	Возникает при использовании в запросе операторов, которые не поддерживаются в Creatio. Для исправления ошибки переформулируйте запрос с использованием поддерживаемых операторов. Перечень доступных операторов описан в параметре `parameters` в статье Веб-сервис odata (OData 4).
Ошибка при выполнении запросов к отдельным полям объекта.	Возникает после внесения изменений в структуру объекта. Для исправления ошибки выполните компиляцию. После этого поля доступны к использованию.

Коды ошибок, которые возвращает Creatio в ответ на запрос по протоколу OData 4, описаны в статье Веб-сервис odata (OData 4).

Примеры интеграций по OData

Сложный

Creatio API документация, которая содержит примеры различных CRUD-операций к Creatio с использованием протоколов OData 3 и OData 4, доступна на сайте Postman.

Примеры запросов с типом данных Stream

Сложный

Элементы с типом данных Stream:

- Изображения.
- Файлы.
- Двоичные данные.

Для работы с типом данных Stream используются стандартные **методы**:

- `GET` — получение данных.
- `POST` — добавление данных.

- `PUT` — изменение данных.
- `DELETE` — удаление данных.

Для отображения результата выполнения запросов к Creatio при работе с типом данных Stream необходимо очистить кэш браузера.

Получить данные

Пример. Используя сервис работы с данными OData, получить фото контакта "New user".

Реализация примера

1. Получите идентификатор фото контакта "New user".

   Фото контакта содержится в колонке `[Data]` таблицы `[SysImage]` базы данных. Чтобы **получить идентификатор фото контакта** "New user", выполните следующий SQL-запрос.

   SQL-запрос

   ```
   select Id from SysImage where Id = (select PhotoId from Contact where Name = 'New user')
   ```

   Ответ на SQL-запрос

   ```
   29FE7EDF-4DB9-4E09-92B0-018047BA1F71
   ```

2. Получите фото контакта "New user".

   Чтобы **получить фото контакта** "New user", выполните следующий запрос.

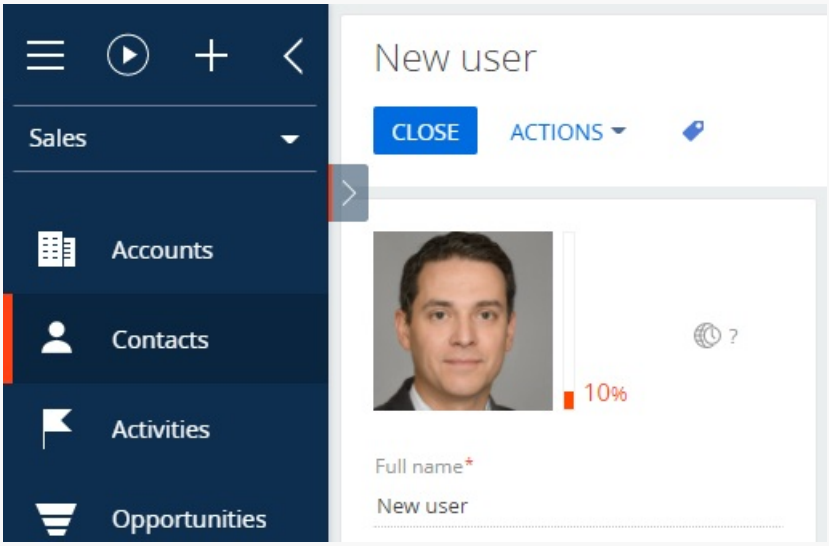
   Запрос

   ```
   // Получить значение поля [Data] экземпляра объекта с [Id] 29FE7EDF-4DB9-4E09-92B0-018047BA1F
   GET http://mycreatio.com/0/odata/SysImage(29FE7EDF-4DB9-4E09-92B0-018047BA1F71)/Data
   ```

   Ответ на запрос

   ```
   Status:  200 OK
   ```

Результат в приложении Creatio



Добавить данные

Пример. Используя сервис работы с данными OData, добавить контакт "New user". Затем добавить контакту фото.

Реализация примера

1. Добавьте контакт "New user".

   Все контакты содержатся в таблице `[Contact]` базы данных. Чтобы **добавить контакт** "New user", выполните следующий запрос.

Запрос

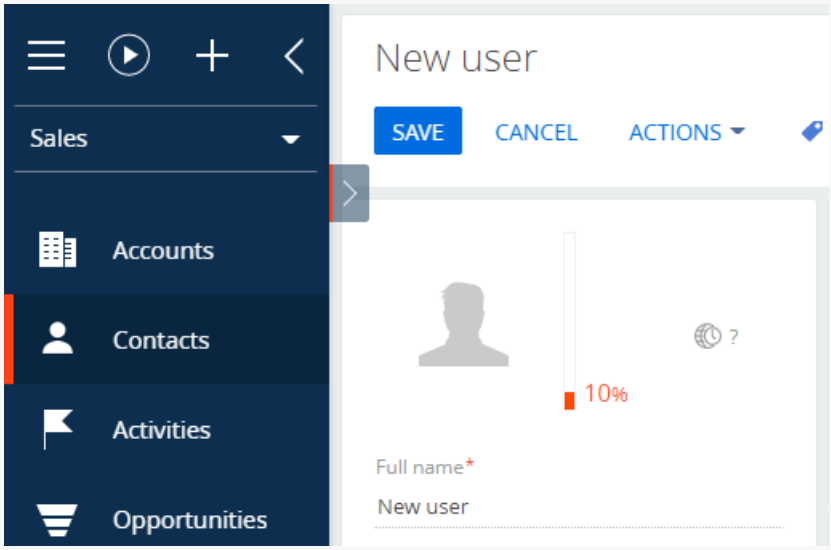
```
// Добавить экземпляр объекта коллекции [Contact].
POST http://mycreatio.com/0/odata/Contact

Accept: application/json; odata=verbose
Content-Type: application/json; odata=verbose; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO

{
    // В поле [Name] записать имя контакта "New user".
    "Name": "New user"
}
```

Ответ на запрос

```
Status: 201 Created

{
    "@odata.context": "http://mycreatio.com/0/odata/$metadata#Contact/$entity",
    "Id": "4c63c8fa-467b-48a6-973f-b2069298404f",
    "Name": "New user",
    "OwnerId": "410006e1-ca4e-4502-a9ec-e54d922d2c00",
    "CreatedOn": "2021-01-14T08:33:29.009023Z",
    "CreatedById": "410006e1-ca4e-4502-a9ec-e54d922d2c00",
    "ModifiedOn": "2021-01-14T08:33:29.009023Z",
    "ModifiedById": "410006e1-ca4e-4502-a9ec-e54d922d2c00",
    "ProcessListeners": 0,
    "Dear": "",
    "SalutationTypeId": "00000000-0000-0000-0000-000000000000",
    "GenderId": "00000000-0000-0000-0000-000000000000",
    "AccountId": "00000000-0000-0000-0000-000000000000",
    "DecisionRoleId": "00000000-0000-0000-0000-000000000000",
    "TypeId": "00000000-0000-0000-0000-000000000000",
    "JobId": "00000000-0000-0000-0000-000000000000",
    "JobTitle": "",
    "DepartmentId": "00000000-0000-0000-0000-000000000000",
    "BirthDate": "0001-01-01T00:00:00Z",
    "Phone": "",
    "MobilePhone": "",
    "HomePhone": "",
    "Skype": "",
    "Email": "",
    "AddressTypeId": "00000000-0000-0000-0000-000000000000",
    "Address": "",
```

```
    "CityId": "00000000-0000-0000-0000-000000000000",
    "RegionId": "00000000-0000-0000-0000-000000000000",
    "Zip": "",
    "CountryId": "00000000-0000-0000-0000-000000000000",
    "DoNotUseEmail": false,
    "DoNotUseCall": false,
    "DoNotUseFax": false,
    "DoNotUseSms": false,
    "DoNotUseMail": false,
    "Notes": "",
    "Facebook": "",
    "LinkedIn": "",
    "Twitter": "",
    "FacebookId": "",
    "LinkedInId": "",
    "TwitterId": "",
    "ContactPhoto@odata.mediaEditLink": "Contact(4c63c8fa-467b-48a6-973f-b2069298404f)/Contac
    "ContactPhoto@odata.mediaReadLink": "Contact(4c63c8fa-467b-48a6-973f-b2069298404f)/Contac
    "ContactPhoto@odata.mediaContentType": "application/octet-stream",
    "TwitterAFDAId": "00000000-0000-0000-0000-000000000000",
    "FacebookAFDAId": "00000000-0000-0000-0000-000000000000",
    "LinkedInAFDAId": "00000000-0000-0000-0000-000000000000",
    "PhotoId": "00000000-0000-0000-0000-000000000000",
    "GPSN": "",
    "GPSE": "",
    "Surname": "user",
    "GivenName": "New",
    "MiddleName": "",
    "Confirmed": true,
    "IsNonActualEmail": false,
    "Completeness": 0,
    "LanguageId": "6ebc31fa-ee6c-48e9-81bf-8003ac03b019",
    "Age": 0
}
```


Результат в приложении Creatio



Идентификатор контакта "New user" "4c63c8fa-467b-48a6-973f-b2069298404f".

2. Добавьте фото контакта "New user".

   Фото контакта должно содержаться в колонке `[Data]` таблицы `[SysImage]` базы данных. Для созданного контакта запись в таблице отсутствует, поэтому ее необходимо добавить. Чтобы **добавить запись в таблицу**, выполните следующий запрос.

Запрос

```
// Добавить экземпляр объекта коллекции [SysImage].
POST http://mycreatio.com/0/odata/SysImage

Accept: application/json; odata=verbose
Content-Type: application/json; odata=verbose; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO

{
    // В поле [Name] записать название файла с фото контакта.
    "Name": "scr_NewContactPhoto.png",
    // В поле [Id] записать произвольный идентификатор записи в таблице [SysImage].
    "Id": "410006E1-CA4E-4502-A9EC-E54D922D2C01",
    // В поле [MimeType] записать тип файла с фото контакта.
    "MimeType": "image/png"
}
```

Ответ на запрос

```
Status: 201 Created

{
```

```
    "@odata.context": "http://mycreatio.com/0/odata/$metadata#SysImage/$entity",
    "Id": "410006e1-ca4e-4502-a9ec-e54d922d2c01",
    "CreatedOn": "2021-01-14T08:52:47.7573789Z",
    "CreatedById": "410006e1-ca4e-4502-a9ec-e54d922d2c00",
    "ModifiedOn": "2021-01-14T08:52:47.7573789Z",
    "ModifiedById": "410006e1-ca4e-4502-a9ec-e54d922d2c00",
    "ProcessListeners": 0,
    "UploadedOn": "0001-01-01T00:00:00Z",
    "Name": "scr_NewContactPhoto.png",
    "Data@odata.mediaEditLink": "SysImage(410006e1-ca4e-4502-a9ec-e54d922d2c01)/Data",
    "Data@odata.mediaReadLink": "SysImage(410006e1-ca4e-4502-a9ec-e54d922d2c01)/Data",
    "Data@odata.mediaContentType": "application/octet-stream",
    "MimeType": "image/png",
    "HasRef": false,
    "PreviewData@odata.mediaEditLink": "SysImage(410006e1-ca4e-4502-a9ec-e54d922d2c01)/Previe
    "PreviewData@odata.mediaReadLink": "SysImage(410006e1-ca4e-4502-a9ec-e54d922d2c01)/Previe
    "PreviewData@odata.mediaContentType": "application/octet-stream"
}
```


В таблицу `[SysImage]` базы данных была добавлена запись, но колонка `[Data]` содержит значение "0x".

Изображение необходимо передать в теле запроса, а название изображения должно совпадать из значением поля `[Name]`. Чтобы **добавить фото контакта** в колонку `[Data]`, выполните следующий запрос.

Запрос

```
// Изменить значение поля [Data] экземпляра объекта с [Id] 410006e1-ca4e-4502-a9ec-e54d922d2c
PUT http://mycreatio.com/0/odata/SysImage(410006e1-ca4e-4502-a9ec-e54d922d2c01)/Data

Accept: application/json; text/plain; */*
Content-Type: application/octet-stream; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
```

Ответ на запрос

```
Status: 200 OK
```

3. Выполните привязку добавленного фото к контакту "New user".

   Для выполнения привязки фото к контакту "New user" необходимо установить связь между полем `[Data]` таблицы `[SysImage]` и полем `[PhotoId]` таблицы `[Contact]`. Чтобы **установить привязку**, выполните следующий запрос.

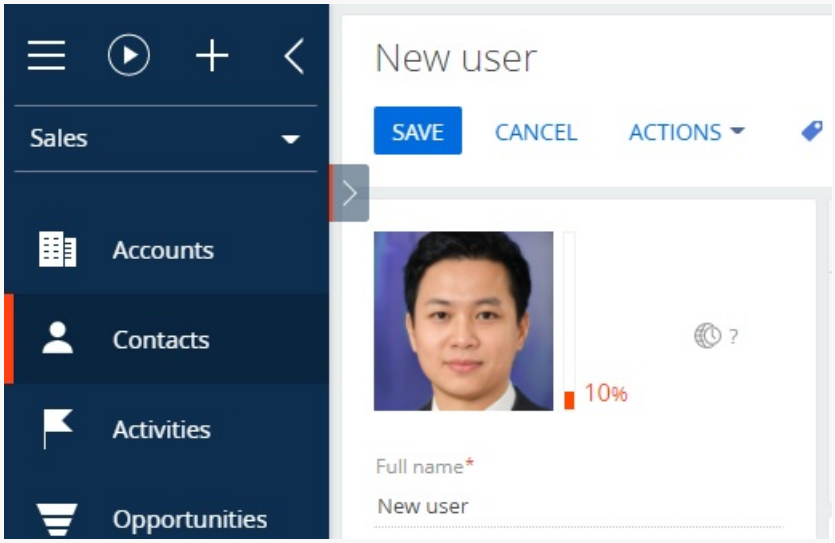
   Запрос

```
// Изменить поле [PhotoId] экземпляра объекта с [Id] 4c63c8fa-467b-48a6-973f-b2069298404f кол
PATCH http://mycreatio.com/0/odata/Contact(4c63c8fa-467b-48a6-973f-b2069298404f)

Accept: application/json;odata=verbose
Content-Type: application/json; odata=verbose; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO

{
    // В поле [PhotoId] записать идентификатор записи в таблице [SysImage].
    "PhotoId": "410006e1-ca4e-4502-a9ec-e54d922d2c01"
}
```

   Ответ на запрос

```
Status: 204 No Content
```

   Результат в приложении Creatio



Чтобы **добавить фото существующему контакту** выполните:

1. `POST`-запрос на добавление экземпляра объекта коллекции `[SysImage]`.
2. `PUT`-запрос на изменение значения поля `[Data]` экземпляра объекта коллекции `[SysImage]`.
3. `PATCH`-запрос на выполнение привязку добавленного фото к контакту "New user".

Изменить данные

Пример. Используя сервис работы с данными OData, изменить фото существующего контакта "New user".

Реализация примера

1. Получите идентификатор фото контакта "New user".

   Фото контакта содержится в колонке `[Data]` таблицы `[SysImage]` базы данных. Чтобы **получить идентификатор фото контакта** "New user", выполните следующий SQL-запрос.

   SQL-запрос

   ```
   select Id from SysImage where Id = (select PhotoId from Contact where Name = 'New user')
   ```

   Ответ на SQL-запрос

   ```
   29FE7EDF-4DB9-4E09-92B0-018047BA1F71
   ```

2. Измените фото контакта "`New user`".

   Чтобы **изменить фото контакта** "New user", выполните следующий запрос.

   Запрос

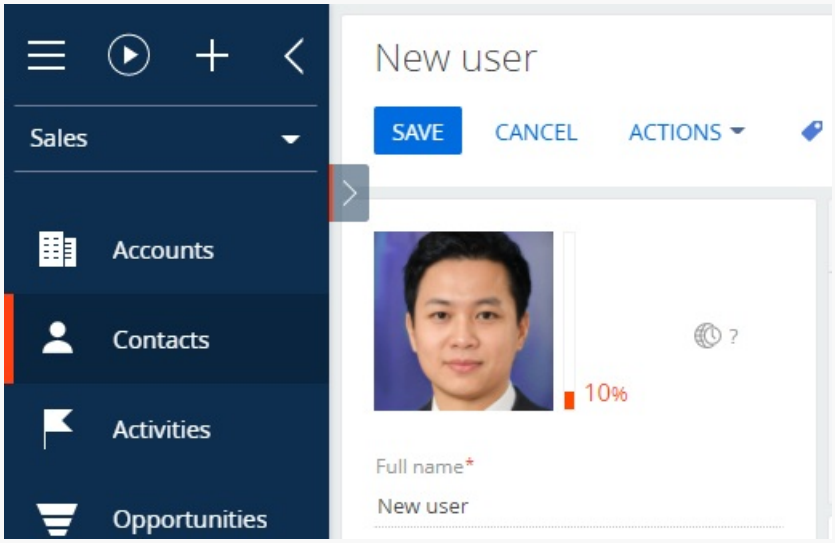
   ```
   // Изменить поле [Data] экземпляра объекта с [Id] 29FE7EDF-4DB9-4E09-92B0-018047BA1F71 коллек
   PUT http://mycreatio.com/0/odata/SysImage(29FE7EDF-4DB9-4E09-92B0-018047BA1F71)/Data
   ```

```
Accept: application/json; text/plain; */*
Content-Type: application/octet-stream; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
```

Ответ на запрос

```
Status: 200 OK
```

Результат в приложении Creatio



Удалить данные

Пример. Используя сервис работы с данными OData, удалить фото контакта "New user".

Реализация примера

1. Получите идентификатор фото контакта "New user".

Фото контакта содержится в колонке `[Data]` таблицы `[SysImage]` базы данных. Чтобы **получить идентификатор фото контакта** "New user", выполните следующий SQL-запрос.

SQL-запрос

```
select Id from SysImage where Id = (select PhotoId from Contact where Name = 'New user')
```

Ответ на SQL-запрос

```
29FE7EDF-4DB9-4E09-92B0-018047BA1F71
```

2. Удалить фото контакта "New user".

   Чтобы **удалить фото контакта** "New user", выполните следующий запрос.

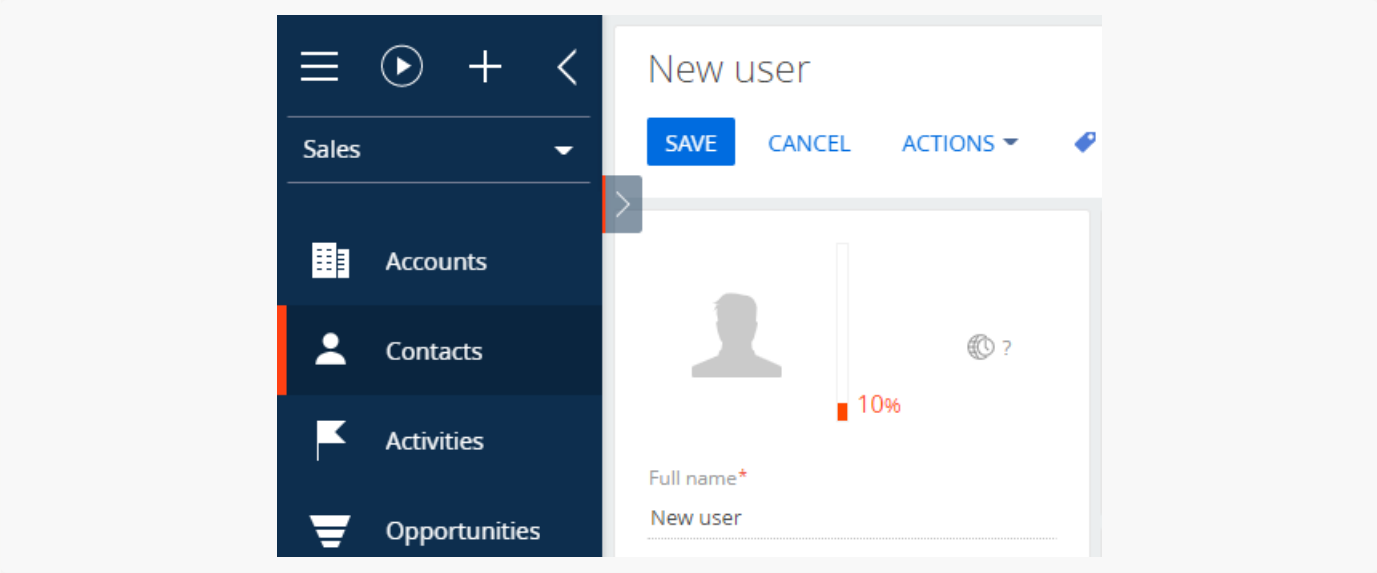
   Запрос

   ```
   // Удалить значение поля [Data] экземпляра объекта с [Id] 29FE7EDF-4DB9-4E09-92B0-018047BA1F7
   DELETE http://mycreatio.com/0/odata/SysImage(29FE7EDF-4DB9-4E09-92B0-018047BA1F71)/Data

   Accept: application/json; text/plain; */*
   Content-Type: application/json; charset=utf-8; IEEE754Compatible=true
   BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
   ```

   Ответ на запрос

   ```
   Status: 204 No Content
   ```

   Результат в приложении Creatio



Примеры пакетных запросов

Сложный

Пакетный запрос (batch-запрос) позволяет объединить множество HTTP-запросов в один, указав в теле каждый запрос как отдельный объект. Сервер баз данных Creatio возвращает один HTTP-ответ, внутри которого содержатся ответы на каждый переданный запрос. Использование пакетных запросов позволяет повысить производительность приложения.

Для пакетных запросов используются:

- HTTP-метод `POST`.
- Параметр $batch.
- Заголовок `Content-Type`.

Значения заголовка `Content-Type`:

- `application/json` — позволяет установить единый тип возвращаемого сервером контента для всех запросов пакетного запроса.
- `multipart/mixed` — позволяет в теле пакетного запроса установить свой заголовок `Content-Type` для каждого запроса.

Если один из запросов завершается с кодом ответа групп 4xx-5xx, то прерывается выполнение последующих запросов. Чтобы после неуспешного выполнения запроса продолжить выполнение следующих запросов, необходимо к основному HTTP-запросу добавить заголовок Prefer: `continue-on-error`.

Важно. Максимальное количество запросов в пакетном запросе — 100.

Пакетный запрос (Content-Type: application/json)

Пакетный запрос

```
POST http://mycreatio.com/0/odata/$batch

Content-Type: application/json; odata=verbose; IEEE754Compatible=true
Accept: application/json
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
ForceUseSession: true

{
    "requests": [
    {
        // Добавить экземпляр объекта коллекции City.
        "method": "POST",
        "url": "City",
        "id": "t3",
        "body": {
            // Добавить в поле Name значение Burbank.
            "Name": "Burbank"
        },
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    },
    {
        // Добавить экземпляр объекта коллекции City.
        "method": "POST",
        "atomicityGroup": "g1",
        "url": "City",
        "id": "t3",
        "body": {
            // Добавить в поле Id значение 62f9bc01-57cf-4cc7-90bf-8672acc922e3.
            "Id": "62f9bc01-57cf-4cc7-90bf-8672acc922e3",
            // Добавить в поле Name значение Spokane.
            "Name": "Spokane"
        },
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    },
    {
```

```
        // Изменить экземпляр объекта коллекции City с идентификатором 62f9bc01-57cf-4cc7-90bf-8
        "method": "PATCH",
        "atomicityGroup": "g1",
        "url": "City/62f9bc01-57cf-4cc7-90bf-8672acc922e3",
        "id": "t2",
        "body": {
            // Изменить значение поля Name на Texas.
            "Name": "Texas"
        },
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    }
    ]
}
```

Ответ на пакетный запрос

```
Status: 200 OK

{
    "responses": [
        {
            "id": "t3",
            "status": 201,
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(b6a05348-55b1-4314-a228-436ba305
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "b6a05348-55b1-4314-a228-436ba305d2f3",
                "CreatedOn": "2020-05-18T17:50:39.2838673Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T17:50:39.2838673Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Burbank",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        },
```

```
        {
            "id": "t3",
            "atomicityGroup": "c59e36b2-2aa9-44fa-86d3-e7d68eecbfa0",
            "status": 201,
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(62f9bc01-57cf-4cc7-90bf-8672acc9
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "62f9bc01-57cf-4cc7-90bf-8672acc922e3",
                "CreatedOn": "2020-05-18T17:50:39.361994Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T17:50:39.361994Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Spokane",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        },
        {
            "id": "t2",
            "atomicityGroup": "c59e36b2-2aa9-44fa-86d3-e7d68eecbfa0",
            "status": 204,
            "headers": {
                "odata-version": "4.0"
            }
        }
    ]
}
```


Пакетный запрос (Content-Type: application/json и Prefer: continue-on-error)

Пакетный запрос

```
POST http://mycreatio.com/0/odata/$batch

Content-Type: application/json; odata=verbose; IEEE754Compatible=true
Accept: application/json
```

```
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
ForceUseSession: true
Prefer: continue-on-error

{
    "requests": [
    {
        // Добавить экземпляр объекта коллекции City.
        "method": "POST",
        "url": "City",
        "id": "t3",
        "body": {
            // Добавить в поле Name значение Burbank.
            "Name": "Burbank"
        },
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    },
    {
        // Изменить экземпляр объекта коллекции City с идентификатором 62f9bc01-57cf-4cc7-90bf-8
        "method": "PATCH",
        "atomicityGroup": "g1",
        "url": "City/62f9bc01-57cf-4cc7-90bf-8672acc922e2",
        "id": "t2",
        "body": {
            // Изменить значение поля Name на Indiana.
            "Name": "Indiana"
        },
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    },
    {
        // Добавить экземпляр объекта коллекции City.
        "method": "POST",
        "atomicityGroup": "g1",
        "url": "City",
        "id": "t3",
        "body": {
            // Добавить в поле Id значение 62f9bc01-57cf-4cc7-90bf-8672acc922a1.
            "Id": "62f9bc01-57cf-4cc7-90bf-8672acc922a1",
            // Добавить в поле Name значение Iowa.
            "Name": "Iowa"
        },
```

```
        "headers": {
            "Content-Type": "application/json;odata=verbose",
            "Accept": "application/json;odata=verbose",
            "Prefer": "continue-on-error"
        }
    }
    ]
}
```

Ответ на пакетный запрос

```
Status: 200 OK

{
    "responses": [
        {
            "id": "t3",
            "status": 201,
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(2f5e68f8-38bd-43c1-8e15-a2f13b0a
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "2f5e68f8-38bd-43c1-8e15-a2f13b0aa56a",
                "CreatedOn": "2020-05-18T18:06:50.7329808Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T18:06:50.7329808Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Burbank",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        },
        {
            "id": "t2",
            "atomicityGroup": "0c1c4019-b9fb-4fb3-8642-2d0660c4551a",
            "status": 204,
            "headers": {
                "odata-version": "4.0"
            }
        },
```

```
        {
            "id": "t3",
            "atomicityGroup": "0c1c4019-b9fb-4fb3-8642-2d0660c4551a",
            "status": 201,
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(62f9bc01-57cf-4cc7-90bf-8672acc9
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "62f9bc01-57cf-4cc7-90bf-8672acc922a1",
                "CreatedOn": "2020-05-18T18:06:50.7954775Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T18:06:50.7954775Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Iowa",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        }
    ]
}
```

Пакетный запрос (Content-Type: multipart/mixed)

Пакетный запрос

```
POST http://mycreatio.com/0/odata/$batch

Content-Type: multipart/mixed;boundary=batch_a685-9724-d873; IEEE754Compatible=true
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
ForceUseSession: true

--batch_a685-9724-d873
Content-Type: multipart/mixed; boundary=changeset_06da-d998-8e7e

--changeset_06da-d998-8e7e
Content-Type: application/http
Content-Transfer-Encoding: binary

// Добавить экземпляр объекта коллекции City.
```

```
POST City HTTP/1.1
Content-ID: 1
Accept: application/atomsvc+xml;q=0.8, application/json;odata=verbose;q=0.5, */*;q=0.1
Content-Type: application/json;odata=verbose

// Добавить в поле Name значение Gilbert.
{"Name": "Gilbert"}

--changeset_06da-d998-8e7e
Content-Type: application/http
Content-Transfer-Encoding: binary

// Изменить экземпляр объекта коллекции City с идентификатором 62f9bc01-57cf-4cc7-90bf-8672acc92
PATCH City/62f9bc01-57cf-4cc7-90bf-8672acc922e2 HTTP/1.1
Content-ID: 2
Accept: application/atomsvc+xml;q=0.8, application/json;odata=verbose;q=0.5, */*;q=0.1
Content-Type: application/json;odata=verbose

// Изменить значение поля Name на Lincoln.
{"Name": "Lincoln"}

--changeset_06da-d998-8e7e
Content-Type: application/http
Content-Transfer-Encoding: binary

// Удалить экземпляр объекта коллекции City с идентификатором 62f9bc01-57cf-4cc7-90bf-8672acc922
DELETE City/62f9bc01-57cf-4cc7-90bf-8672acc922e2 HTTP/1.1
Content-ID: 3
Accept: application/atomsvc+xml;q=0.8, application/json;odata=verbose;q=0.5, */*;q=0.1
Content-Type: application/json;odata=verbose
```

Ответ на пакетный запрос

```
Status: 200 OK

--batchresponse_e17aace9-5cbb-49bd-b7ad-f1be8cc8c9d8
Content-Type: multipart/mixed; boundary=changesetresponse_a08c1df6-4b82-4a9b-be61-7ef4cc7b23ba

--changesetresponse_a08c1df6-4b82-4a9b-be61-7ef4cc7b23ba
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 1

HTTP/1.1 201 Created
Location: https://mycreatio.com/0/odata/City(fbd0565f-fa8a-4214-ae89-c976c5f3acb4)
Content-Type: application/json; odata.metadata=minimal
OData-Version: 4.0
```

```
{
    "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
    "Id": "fbd0565f-fa8a-4214-ae89-c976c5f3acb4",
    "CreatedOn": "2020-05-18T18:41:57.0917235Z",
    "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
    "ModifiedOn": "2020-05-18T18:41:57.0917235Z",
    "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
    "Name": "Gilbert",
    "Description": "",
    "CountryId": "00000000-0000-0000-0000-000000000000",
    "RegionId": "00000000-0000-0000-0000-000000000000",
    "TimeZoneId": "00000000-0000-0000-0000-000000000000",
    "ProcessListeners": 0
}
--changesetresponse_a08c1df6-4b82-4a9b-be61-7ef4cc7b23ba
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 2

HTTP/1.1 204 No Content
OData-Version: 4.0


--changesetresponse_a08c1df6-4b82-4a9b-be61-7ef4cc7b23ba
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 3

HTTP/1.1 204 No Content


--changesetresponse_a08c1df6-4b82-4a9b-be61-7ef4cc7b23ba--
--batchresponse_e17aace9-5cbb-49bd-b7ad-f1be8cc8c9d8--
```

Пакетный запрос (`Content-Type: multipart/mixed` и разными наборами запросов)

Пакетный запрос

```
POST http://mycreatio.com/0/odata/$batch

Content-Type: multipart/mixed;boundary=batch_a685-9724-d873; IEEE754Compatible=true
Accept: application/json
```

```
BPMCSRF: OpK/NuJJ1w/SQxmPvwNvfO
ForceUseSession: true

--batch_a685-9724-d873
Content-Type: multipart/mixed; boundary=changeset_06da-d998-8e7e

--changeset_06da-d998-8e7e
Content-Type: application/http
Content-Transfer-Encoding: binary

// Добавить экземпляр объекта коллекции City.
POST City HTTP/1.1
Content-ID: 1
Accept: application/atomsvc+xml;q=0.8, application/json;odata=verbose;q=0.5, */*;q=0.1
Content-Type: application/json;odata=verbose

// Добавить в поле Id значение d6bc67b1-9943-4e47-9aaf-91bf83e9c285.
// Добавить в поле Name значение Nebraska.
{"Id": "d6bc67b1-9943-4e47-9aaf-91bf83e9c285", "Name": "Nebraska"}

--batch_a685-9724-d873
Content-Type: multipart/mixed; boundary=changeset_06da-d998-8e71

--changeset_06da-d998-8e71
Content-Type: application/http
Content-Transfer-Encoding: binary

// Добавить экземпляр объекта коллекции City.
POST City HTTP/1.1
Content-ID: 2
Accept: application/atomsvc+xml;q=0.8, application/json;odata=verbose;q=0.5, */*;q=0.1
Content-Type: application/json;odata=verbose

// Добавить в поле Id значение d6bc67b1-9943-4e47-9aaf-91bf83e9c286.
// Добавить в поле Name значение Durham.
{"Id": "d6bc67b1-9943-4e47-9aaf-91bf83e9c286", "Name": "Durham"}
```

Ответ на пакетный запрос

```
Status: 200 OK

{
    "responses": [
        {
            "id": "1",
            "atomicityGroup": "e9621f72-42bd-47c1-b271-1027e4b68e3b",
            "status": 201,
```

```
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(d6bc67b1-9943-4e47-9aaf-91bf83e9
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "d6bc67b1-9943-4e47-9aaf-91bf83e9c285",
                "CreatedOn": "2020-05-18T18:49:16.3766324Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T18:49:16.3766324Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Nebraska",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        },
        {
            "id": "2",
            "atomicityGroup": "960e2272-d8cb-4b4d-827c-0181485dd71d",
            "status": 201,
            "headers": {
                "location": "https://mycreatio.com/0/odata/City(d6bc67b1-9943-4e47-9aaf-91bf83e9
                "content-type": "application/json; odata.metadata=minimal",
                "odata-version": "4.0"
            },
            "body": {
                "@odata.context": "https://mycreatio.com/0/odata/$metadata#City/$entity",
                "Id": "d6bc67b1-9943-4e47-9aaf-91bf83e9c286",
                "CreatedOn": "2020-05-18T18:49:16.4078852Z",
                "CreatedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "ModifiedOn": "2020-05-18T18:49:16.4078852Z",
                "ModifiedById": "dad159f3-6c2d-446a-98d2-0f4d26662bbe",
                "Name": "Durham",
                "Description": "",
                "CountryId": "00000000-0000-0000-0000-000000000000",
                "RegionId": "00000000-0000-0000-0000-000000000000",
                "TimeZoneId": "00000000-0000-0000-0000-000000000000",
                "ProcessListeners": 0
            }
        }
    ]
}
```

Примеры запросов в WCF-клиенте

Сложный

Для получения коллекции объектов сервиса используется универсальный класс DataServiceQuery, который представляет собой запрос к сервису, возвращающий коллекцию сущностей конкретного типа.

Чтобы выполнить запрос к сервису данных `EntityDataService.svc`, предварительно необходимо создать экземпляр объекта контекста среды приложения Creatio.

Далее в примерах будет использована forms-аутентификация.

Чтобы **реализовать forms-аутентификацию**:

1. Создайте класс `LoginClass`.
2. Реализуйте поля `authServiceUri` (строка запроса к методу `Login` аутентификационного сервиса `AuthService.svc`) и `AuthCookie` (Cookie аутентификации Creatio).
3. Реализуйте метод `TryLogin(string userName, string userPassword)`, который выполняет аутентификацию пользователя и сохраняет ответ сервера в поле `AuthCookie`.
4. Реализуйте метод `OnSendingRequestCookie(object sender, SendingRequestEventArgs e)`, который будет вызван в ответ на событие экземпляра контекста `SendingRequest` (создание нового экземпляра `HttpWebRequest`).

   В методе `OnSendingRequestCookie` выполняется аутентификация пользователя, а полученные в ответ cookies добавляются в запрос на получение данных.

   `OnSendingRequestCookie`

   ```
   static void OnSendingRequestCookie(object sender, SendingRequestEventArgs e)
   {
       // Вызов метода класса LoginClass, реализующего аутентификацию переданного в параметрах м
       LoginClass.TryLogin("CreatioUserName", "CreatioUserPassword");
       var req = e.Request as HttpWebRequest;
       // Добавление полученных аутентификационных cookie в запрос на получение данных.
       req.CookieContainer = LoginClass.AuthCookie;
       e.Request = req;
   }
   ```

Способы выполнения запроса к сервису:

- Выполнение LINQ-запроса к именованному объекту `DataServiceQuery`, который получен из контекста сервиса.
- Неявное перечисление объекта `DataServiceQuery`, который получен из контекста сервиса.
- Явный вызов метода Execute объекта `DataServiceQuery` или BeginExecute для асинхронного выполнения.

Получить коллекцию контактов через LINQ–запрос

Пример. Определить и выполнить запрос LINQ, возвращающий все сущности контактов сервиса `EntityDataService.svc`.

Реализация примера

LINQ-запрос

```
public static void GetOdataCollectioByLinqWcfExample()
{
    // Создание контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    try
    {
        // Построение запроса LINQ для получение коллекции контактов.
        var allContacts = from contacts in context.ContactCollection
            select contacts;
        foreach (Contact contact in allContacts)
        {
            // Выполнение действий с контактами.
        }
    }
    catch (Exception ex)
    {
        // Обработка ошибок.
    }
}
```

Получить коллекцию контактов неявным запросом

Пример. Использовать контекст для неявного выполнения запроса, возвращающего все сущности контактов сервиса `EntityDataService.svc`.

Реализация примера

`GetOdataCollectionByImplicitRequestExample()`

```
public static void GetOdataCollectionByImplicitRequestExample()
{
    // Создание объекта контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    try
    {
        // Определение неявного запроса к сервису для получения коллекции контактов.
        DataServiceQuery<Contact> allContacts = context.ContactCollection;
        foreach (Contact contact in allContacts)
        {
            // Выполнение действий с контактами.
        }
    }
    catch (Exception ex)
    {
        // Обработка ошибок.
    }
}
```

Получить коллекцию контактов явным запросом

Пример. Использовать контекст DataServiceContext для явного выполнения запроса к сервису `EntityDataService.svc`, который возвращает все сущности контактов.

Реализация примера

`GetOdataCollectionByExplicitRequestExample()`

```
public static void GetOdataCollectionByExplicitRequestExample()
{
    // Определение Uri запроса к сервису, который возвращает коллекцию контактов.
    Uri contactUri = new Uri(serverUri, "ContactCollection");
    // Создание объекта контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    try
    {
        // Выполнение явного запроса к сервису вызовом метода Execute<>().
        foreach (Contact contact in context.Execute<Contact>(contactUri))
        {
```

```
            // Выполнение действий с контактами.
        }
    }
    catch (Exception ex)
    {
        // Обработка ошибок.
    }
}
```

Примеры CRUD-операций

Получение объекта

```
public static void GetOdataObjectByWcfExample()
{
    // Создание контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    //
    var contact = context.ContactCollection.Where(c => c.Name.Contains("User")).First();
    // Выполнение действий над контактом.
}
```

Создание объекта

```
public static void CreateCreatioEntityByOdataWcfExample()
{
    // Создание нового контакта, инициализиция свойств.
    var contact = new Contact()
    {
        Id = Guid.NewGuid(),
        Name = "New Test User"
    };
    // Создание и инициализация свойств нового контрагента, к которому относится создаваемый кон
    var account = new Account()
    {
        Id = Guid.NewGuid(),
        Name = "Some Company"
    };
    contact.Account = account;
    // Создание контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
```

```
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>l(OnSendingRequestCookie)
    // Добавление созданного контакта в коллекцию контактов модели данных сервиса.
    context.AddToAccountCollection(account);
    // Добавление созданного контрагента в коллекцию контрагентов модели данных сервиса.
    context.AddToContactCollection(contact);
    // Установка связи между созданными контактом и контрагентом в модели данных сервиса.
    context.SetLink(contact, "Account", account);
    // Сохранение изменений данных в Creatio одним запросом.
    DataServiceResponse responces = context.SaveChanges(SaveChangesOptions.Batch);
    // Обработка ответов от сервера.
}
```

Изменение объекта

```
public static void UpdateCreatioEntityByOdatetWcfExample()
{
    // Создание контекста приложения Creatio.
    var context = new Creatio(serverUri);
    // Определение метода, который добавляет аутентификационные cookie при создании нового запро
    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    // Из колллекции контактов выбирается тот, по которому будет изменяться информация.
    var updateContact = context.ContactCollection.Where(c =< c.Name.Contains("Test")).First();
    // Изменение свойств выбранного контакта.
    updateContact.Notes = "New updated description for this contact.";
    updateContact.Phone = "123456789";
    // Сохранение изменений в модели данных сервиса.
    context.UpdateObject(updateContact);
    // Сохранение изменений данных в Creatio одним запросом.
    var responces = context.SaveChanges(SaveChangesOptions.Batch);
}
```

Удаление объекта

```
public static void DeleteCreatioEntityByOdataWcfExample()
{
    // Создание контекста приложения Creatio.
    var context = new Creatio(serverUri);

    context.SendingRequest += new EventHandler<SendingRequestEventArgs>(OnSendingRequestCookie);
    // Из коллекции контактов выбирается тот объект, который будет удален.
    var deleteContact = context.ContactCollection.Where(c => c.Name.Contains("Test")).First();
    // Удаление выбранного объекта из модели данных сервиса.
    context.DeleteObject(deleteContact);
    // Сохранение изменений данных в Creatio одним запросом.
```

```
    var responces = context.SaveChanges(SaveChangesOptions.Batch);
    // Обработка ответов от сервера.
}
```

Веб-сервис odata (OData 4) API

Сложный

В зависимости от используемого типа запроса протокол OData 4 возвращает различные данные.

Структура запроса

```
/* Строка запроса. */
method Creatio_application_address/0/odata/objects_collection(object_id)/object_field?$parameter

/* Заголовки запроса. */
Accept: application/json
Content-Type: application/json; charset=utf-8; IEEE754Compatible=true
ForceUseSession: true
BPMCSRF: authentication_cookie_value

/* Тело запроса (используется в POST и PATCH запросах). */
{
    "field1": "value1",
    "field2": "value2",
    ...
}
```

Структура ответа

```
/* Код состояния. */
Status: code

/* Тело ответа (присутствует в GET и POST запросах). */
{
    "@odata.context": "http://Creatio_application_address/0/odata/$metadata#data_resource",
    "value": [
    {
        "object1 field1": "object1 field_value1",
        "object1 field2": "object1 field_value2",
        ...
    },
    {
```

```
        "object2 field1": "object2 field_value1",
        "object2 field2": "object2 field_value2",
        ...
    },
    ...
    ]
}
```

Строка запроса

`method` **required**

Методы запроса, которые поддерживает приложение Creatio:

- `GET` — получение данных.
- `POST` — добавление данных.
- `PATCH` — изменение данных.
- `DELETE` — удаление данных.

`Creatio_application_address` **required**

Адрес приложения Creatio.

`odata` **required**

Адрес веб-сервиса протокола OData 4. Неизменяемая часть запроса.

`objects_collection` **required**

Имя таблицы базы данных (имя коллекции объектов). Чтобы получить перечень таблиц базы данных, выполните аутентификацию и один из запросов.

JSON-формат

```
/* Получает результат в формате JSON. */
GET Creatio_application_address/0/odata/

/* Заголовки запроса. */
ForceUseSession: true
BPMCSRF: authentication_cookie_value
```

XML-формат

```
/* Получает результат в формате XML. */
GET Creatio_application_address/0/odata/$metadata

/* Заголовки запроса. */
ForceUseSession: true
BPMCSRF: authentication_cookie_value
```

Данные таблицы [SysSchema]

```
/* Получает результат из таблицы [SysSchema] базы данных в формате JSON. */
GET Creatio_application_address/0/odata/SysSchema?$filter=ManagerName eq 'EntitySchemaManager

/* Заголовки запроса. */
ForceUseSession: true
BPMCSRF: authentication_cookie_value
```

`object_id` **optional**

Идентификатор строки записи таблицы базы данных (идентификатор экземпляра объекта коллекции).

`object_field` **optional**

Поле записи таблицы базы данных (поле экземпляра объекта коллекции).

`parameters` **optional**

Необязательные параметры OData 4, которые разрешены к использованию в строке `GET` запроса к Creatio. Для указания параметров используйте оператор `?`. Имя параметра записывается после оператора `$`. Чтобы использовать два и более параметров, воспользуйтесь оператором `&`.

Возможные параметры

$value		Значение поля.
$count	$count=true	Количество элементов, которые попали в выборку.
$skip	$skip=n	n первых элементов, которые не должны попасть в выборку.
$top	$top=n	n первых элементов, которые должны попасть в выборку.
$select	$select=field1,field2,...	Набор полей, которые должны попасть в выборку.
$orderby	$orderby=field asc или $orderby=field desc	Сортировка значений поля, которые попали в выборку.
$expand	$expand=field1,field2,...	Расширение связанных полей.
$filter	$filter=field template 'field_value'	Фильтрация полей, которые должны попасть в выборку.

Заголовки запроса

`Accept` `application/json` **required**

Тип данных, в котором ожидается ответ от сервера. Не обязательный к использованию в `GET` запросах.

`Content-Type` `application/json; charset=utf-8; IEEE754Compatible=true` **required**

Кодировка и тип ресурса, который передается в теле запроса. Не обязательный к использованию в `GET` запросах.

`ForceUseSession` `true` **required**

Принудительное использование существующей сессии.

`BPMCSRF` `authentication_cookie_value` **required**

Аутентификационный cookie.

Тело запроса

`field1, field2, ...` **required**

Имена полей, которые передаются в теле запроса.

`value1, value2, ...` **required**

Значения полей `field1, field2, ...`, которые передаются в теле запроса.

Коды состояний ответа

`code`

Код состояния ответа на запрос.

Возможные коды состояний

200 OK	Запрос, который не создает ресурс, успешно завершен и значение ресурса не равно нулю. Тело ответа содержит значение ресурса, указанного в URL-адресе запроса. Информация, которая возвращается с ответом, зависит от используемого метода запроса: `GET` — запрашиваемый ресурс найден и передан в теле ответа. `POST` — ресурс, который описывает результат действия сервера на запрос, передан в теле ответа.
201 Created	Запрос, который успешно создает ресурс. Тело ответа содержит созданный ресурс. Используется для `POST` запросов, которые создают коллекцию, создают объект мультимедиа (например, фотографию) или вызывают действие через его импорт.
202 Accepted	Запрос на работу с данными принят в обработку, но еще не завершен. Нет гарантий, что запрос успешно выполнится в процессе обработки данных (асинхронная обработка запроса).
204 No content	Запрос успешно обработан, но нет необходимости возвращать какие-либо данные. Запрашиваемый ресурс имеет нулевое значение. В ответе передаются только заголовки, тело ответа — пустое.
3xx Redirection	Перенаправление указывает что для выполнения запроса клиенту необходимо предпринять дальнейшие действия. Ответ включает заголовок `Location` с URL-адресом,

	который позволяет получить результат. Также ответ может включать заголовок `Retry-After`, который отображает время (в секундах). Это период, в течение которого клиент может подождать прежде чем повторить запрос к ресурсу, который возвращен в заголовке `Location`.
304 Not modified	Клиент выполняет `GET` запрос с заголовком `If-None-Match` и содержимое не изменилось. Ответ не содержит другие заголовки.
403 Forbidden	Сервер понял запрос, но отказывается его авторизировать. Это означает, что клиент не уполномочен совершать операции с запрошенным ресурсом. Причиной может быть некорректная кука `BPMCSRF`. Может возникать при отсутствии CSRF-токена. Для исправления ошибки добавьте передачу куки `BPMCSRF`.
404 Not Found	Сервер не может найти ресурс, который указанй в URL-адресе. Дополнительная информация может содержаться в теле ответа.
405 Method Not Allowed	Ресурс, который указан в URL-адресе, не поддерживает указанный метод запроса. В ответе необходимо получить заголовок `Allow`, который содержит перечень доступных методов запроса для ресурса. Может возникать для `PUT` и `DELETE` запросов при включенном расширении `WebDav HTTP`. Для исправления ошибки отключите в IIS расширение `WebDav HTTP`.
406 Not Acceptable	Ресурс, который указан в URL-адресе запроса, не имеет текущего представления, которое подходит для клиента в соответствии с `Accept`, `Accept-Charset` и `Accept-Language` заголовками запроса. Служба не желает предоставлять представление по умолчанию.
410 Gone	Запрошенный ресурс больше недоступен. Ресурс раньше был по указанному URL, но теперь удален и недоступен.
412 Prediction Failed	Клиент указал в заголовке запроса условие, которое ресурс не может выполнить.
424 Failed Dependency	Текущий запрос к ресурсу невозможно выполнить, потому что запрошенное действие зависит от другого действия, выполнить которое не удалось. Запрос не выполнен через сбой зависимости.
501 Not Implemented	Клиент использует метод запроса, который не реализован

	протоколом OData 4 и этот метод невозможно обработать. Тело ответа содержит описание нереализованного функционала.

Тело ответа

`@odata.context`

Информация о типе возвращаемых данных. Кроме пути к источнику данных, элемент `data_resource` может содержать параметр `$entity`, который показывает что в ответе был возвращен единственный экземпляр объекта коллекции. Параметр присутствует только для `GET` и `POST` запросов.

`value`

Содержит коллекцию объектов. Отсутствует, если в ответе возвращен один экземпляр объекта коллекции. Параметр присутствует только для `GET` запроса.

`[]`

Коллекция объектов. Параметр присутствует только для `GET` запроса.

`{}`

Экземпляры объектов коллекции. Параметр присутствует только для `GET` и `POST` запросов.

`object1 field1, object1 field2, ..., object2 field1, object2 field2, ...`

Имена полей `field1, field2, ...` экземпляров объектов `object1, object2, ...` коллекции. Параметр присутствует только для `GET` и `POST` запросов.

`object1 field_value1, object1 field_value2, ..., object2 field_value1, object2 field_value2, ...`

Значения полей `field1, field2, ...` экземпляров объектов `object1, object2, ...` коллекции. Параметр присутствует только для `GET` и `POST` запросов.

Веб-сервис EntityDataService.svc (OData 3) API

Сложный

В зависимости от используемого типа запроса протокол OData 3 может возвращать различные данные. Структура запроса и ответа рассмотрена ниже.

Структура запроса

```
// Строка запроса.
method Creatio_application_address/0/ServiceModel/EntityDataService.svc/objects_collectionCollec

// Заголовки запроса.
Accept: application/atom+xml; type=entry
Content-Type: application/json; odata=verbose
ForceUseSession: true
BPMCSRF: authentication_cookie_value

// Тело запроса (используется в POST и PATCH запросах).
{
    "field1": "value1",
    "field2": "value2",
    ...
}
```

Структура ответа

```
// Код состояния.
Status: code

// Тело ответа (присутствует для GET и POST запросов).
<?xml version="1.0" encoding="utf-8"?>
<feed xml:base="http://mycreatio.com/0/ServiceModel/EntityDataService.svc/" xmlns="http://www.w3
    <id>http://mycreatio.com/0/ServiceModel/EntityDataService.svc/data_resource</id>
    <title type="text">data_resource</title>
    <updated>date and time of request</updated>
    <link rel="self" title="data_resource" href="data_resource" />
    <entry>
        metadata_data
        <content type="application/xml">
            <m:properties>
                <d:object1 field1>object1 field_value1</d:object1 field1>
                <d:object1 field2>object1 field_value2</d:object1 field2>
                ...
            </m:properties>
        </content>
    </entry>
    <entry>
        metadata_data
        <content type="application/xml">
            <m:properties>
                <d:object2 field1>object2 field_value1</d:object2 field1>
```

```
                <d:object2 field2>object2 field_value2</d:object2 field2>
                ...
            </m:properties>
        </content>
    </entry>
    ...
</feed>
```

Строка запроса

`method` **required**

Приложение Creatio поддерживает следующие методы запроса:

- `GET` — получение данных.
- `POST` — добавление данных.
- `PATCH` — изменение данных.
- `DELETE` — удаление данных.

`Creatio_application_address` **required**

Адрес приложения Creatio.

`ServiceModel` **required**

Путь к веб-сервису протокола OData 3. Неизменяемая часть запроса.

`EntityDataService.svc` **required**

Адрес веб-сервиса протокола OData 3. Неизменяемая часть запроса.

`objects_collectionCollection` **required**

Имя таблицы базы данных (имя коллекции объектов). При использовании протокола OData 3 к первому имени коллекции объектов в строке запроса необходимо добавлять слово `Collection` (например, `ContactCollection`). Получить перечень таблиц базы данных можно выполнив запрос к базе данных.

MySQL

```
SELECT * FROM INFORMATION_SCHEMA.TABLES
```

Oracle

```
SELECT * FROM ALL_TABLES
```

PostgreSQL

```
SELECT table_name FROM information_schema.tables
```

`guid'object_id'` **optional**

Идентификатор строки записи таблицы базы данных (идентификатор экземпляра объекта коллекции). Например, `guid'00000000-0000-0000-0000-000000000000'`).

`object_field` **optional**

Поле записи таблицы базы данных (поле экземпляра объекта коллекции).

`parameters` **optional**

Необязательные параметры OData 3, которые разрешены к использованию в строке `GET`-запроса к Creatio. Для указания параметров необходимо использовать оператор `?`. Имя параметра должно записываться после оператора `$`. Чтобы использовать два и более параметров, необходимо воспользоваться оператором `&`.

Возможные параметры

$value		Значение поля.
$count	`$count=true`	Количество элементов, которые попали в выборку.
$skip	`$skip=n`	n первых элементов, которые не должны попасть в выборку.
$top	`$top=n`	n первых элементов, которые должны попасть в выборку.
$select	`$select=field1,field2,...`	Набор полей, которые должны попасть в выборку.
$orderby	`$orderby=field asc` или `$orderby=field desc`	Сортировка значений поля, которые попали в выборку.
$expand	`$expand=field1,field2,...`	Расширение связанных полей.
$filter	`$filter=field template 'field_value'`	Фильтрация полей, которые должны попасть в выборку.

Заголовки запроса

`Accept` `application/atom+xml; type=entry` **required**

Тип данных, который можно ожидать в ответе от сервера. Сервер возвращает ответ в формате XML. Не обязательный к использованию в `GET`-запросах.

`Content-Type` `application/json; odata=verbose` **required**

Кодировка и тип ресурса, который передается в теле запроса. Не обязательный к использованию в `GET`-запросах.

`ForceUseSession` `true` **required**

Заголовок `ForceUseSession` отвечает за принудительное использование уже существующей сессии. Отсутствует необходимость использования в запросе к сервису аутентификации `AuthService.svc`.

`BPMCSRF` `authentication_cookie_value` **required**

Аутентификационный cookie.

Тело запроса

`field1, field2, ...` **required**

Имена полей, которые передаются в теле запроса.

`value1, value2, ...` **required**

Значения полей `field1, field2, ...`, которые передаются в теле запроса.

Код состояния ответа

`code`

Код состояния ответа на запрос.

Возможные коды состояния

200 OK	Запрос `GET`, `PUT`, `MERGE` или `PATCH` успешно завершен. Тело ответа должно содержать значение объекта или свойства, указанного в URL-адресе запроса.
201 Created	Запрос `POST` успешно создал объект или ссылку. Тело ответа должно содержать обновленный объект.
202 Accepted	Запрос на изменение данных был принят в обработку, но еще не завершен. Тело ответа должно содержать заголовок `Location` в дополнение в заголовке `Retry-After`. Тело ответа должно быть пустым. Сервер должен вернуть код ответа 303 с заголовком `Location`, который содержит окончательный URL-адрес для получения результата запроса. Тело и заголовки окончательного URL-адреса должны быть отформатированы также, как и выполнение первоначального запроса на изменение данных.
204 No content	Запрос на изменение данных. Запрашиваемый ресурс имеет нулевое значение. Тело ответа должно быть пустым.
3xx Redirection	Запрос на изменение данных. Перенаправление указывает что клиент должен предпринимать дальнейшие действия для выполнения запроса. Ответ должен включать заголовок `Location` с URL-адресом, по которому можно получить результат.
4xx Client Error	Некорректные запросы. Сервер возвращает код в ответ на клиентские ошибки в дополнение к запросам на несуществующие ресурсы, такие как сущности, коллекции сущностей или свойства. Если тело ответа определено для кода ошибки, тело ошибки является таким, как определено для соответствующего формата.
404 Not Found	Объект или коллекция, указанные в URL-адресе, не существуют. Тело ответа должно быть пустым.

Тело ответа

`entry`

Экземпляр объекта коллекции.

`metadata_data`

Метаданные экземпляра объекта коллекции.

```
object1 field1, object1 field2, ..., object2 field1, object2 field2, ...
```

Имена полей `field1, field2, ...` экземпляров объектов `object1, object2, ...` коллекции.

`object1 field_value1, object1 field_value2, ..., object2 field_value1, object2 field_value2, ...`

Значения полей `field1, field2, ...` экземпляров объектов `object1, object2, ...` коллекции.