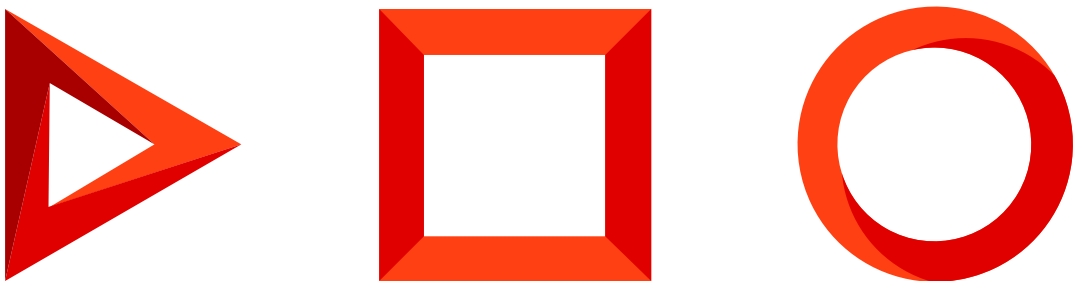
Creatio Academy

Сервис запуска бизнес-процессов

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

Сервис запуска бизнес-процессов

Сложный

Одной из задач интеграции внешнего приложения с Creatio является запуск бизнес-процессов. С этой целью в сервисной модели Creatio реализован веб-сервис `ProcessEngineService.svc`. **Назначение** `ProcessEngineService.svc` — запуск бизнес-процессов.

Запуск бизнес-процессов из внешнего приложения

Адрес сервиса `ProcessEngineService.svc`

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

Основные **методы** сервиса `ProcessEngineService.svc`:

- `Execute()` — запускает бизнес-процесс. Позволяет передать набор входящих параметров бизнес-процесса и вернуть результат его выполнения.
- `ExecProcElByUId()` — запускает отдельный элемент бизнес-процесса. Запускать на выполнение можно только элемент выполняющегося процесса.

На заметку. Полный перечень методов веб-сервиса доступен в Библиотеке .NET классов.

Запуск бизнес-процесса из front-end части

Для запуска бизнес-процесса из клиентской схемы используется модуль `ProcessModuleUtilities` пакета `NUI`. Этот модуль предоставляет удобный интерфейс для выполнения запросов к сервису `ProcessEngineService.svc`.

Чтобы **запустить бизнес-процесс из клиентской схемы**:

1. Подключите в качестве зависимости модуль `ProcessModuleUtilities` в модуль страницы, из которой вызывается сервис.
2. Вызовите метод `executeProcess(args)` модуля `ProcessModuleUtilities`, передав ему в качестве параметра объект `args` с такими свойствами:

Свойства объекта `args`

Свойство	Описание
`sysProcessName`	Имя вызываемого процесса (необязательное свойство в случае, если определено свойство `sysProcessId`).
`sysProcessId`	Уникальный идентификатор вызываемого процесса (необязательное свойство, если определено свойство `sysProcessName`).
`parameters`	Объект, свойствами которого являются входящие параметры вызываемого процесса.

Запустить бизнес-процесс через веб-сервис

Сложный

Пример. Используя веб-сервис `ProcessEngineService.svc`, из строки навигации браузера или из консольного приложения запустить демонстрационные бизнес-процессы создания и считывания контактов Creatio, .

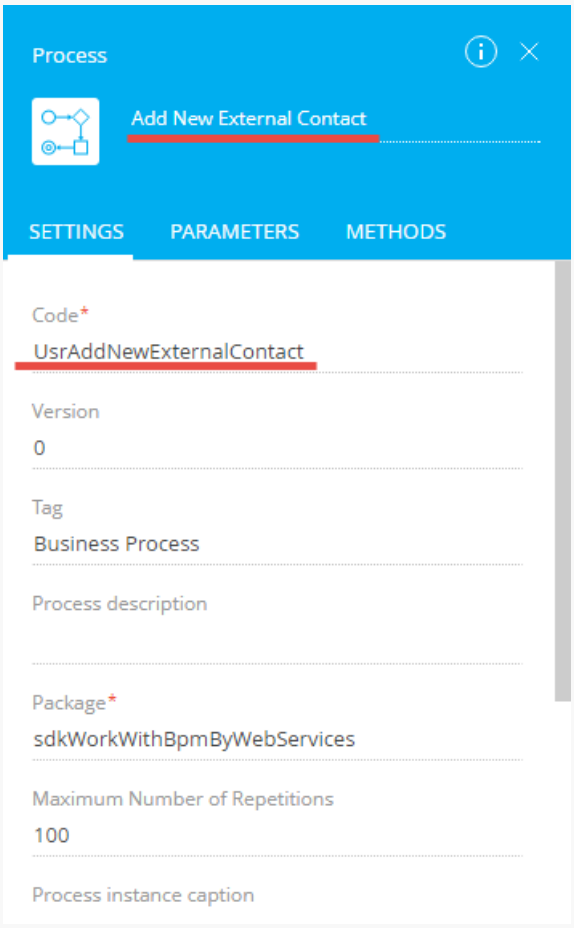
1. Создать процесс добавления контакта

На заметку. Особенности и лучшие практики создания бизнес-процессов в Creatio подробно описаны в документации по настройке процессов.

1. Создайте пользовательский пакет и установите его в качестве текущего.
2. Перейдите в дизайнер процессов.
3. Заполните значения **свойств бизнес-процесса**:

   - [*Название*] ([*Name*]) — "`Add New External Contact`".
   - [*Код*] ([*Code*]) — "`UsrAddNewExternalContact`".

   Для остальных свойств оставьте значения по умолчанию.



4. Добавьте параметры бизнес-процесса.

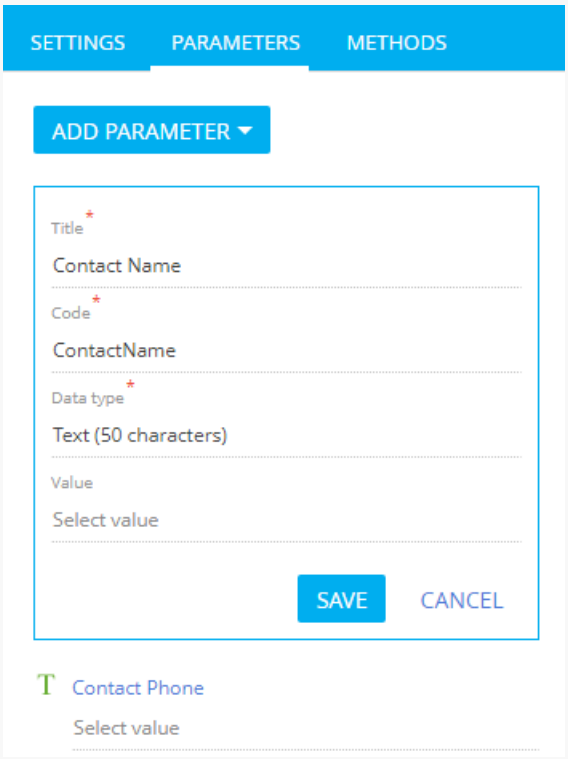
   С помощью параметров в процесс передаются реквизиты добавляемого контакта — имя и телефон.

   Значения свойств параметра `ContactName`:

   - [*Заголовок*] ([*Title*]) — "Имя контакта" ("Contact Name").
   - [*Код*] ([*Code*]) — "`ContactName`".
   - [*Тип данных*] ([*Data type*]) — "Текст (50 символов)" ("Text (50 characters)").

   Значения свойств параметра `ContactPhone`:

   - [*Заголовок*] ([*Title*]) — "Телефон контакта" ("Contact Phone").
   - [*Код*] ([*Code*]) — "`ContactPhone`".
   - [*Тип данных*] ([*Data type*]) — "Текст (50 символов)" ("Text (50 characters)").



5. Добавьте элемент [*Задание-сценарий*] ([*ScriptTask*]).

   Значения **свойств элемента**:

   - [*Название*] ([*Name*]) — "`Add contact`".
   - [*Код*] ([*Code*]) — "`ScriptTaskAddContact`".

6. Реализуйте логику добавления нового контакта.

   Чтобы редактировать код сценария, дважды щелкните по элементу на диаграмме. На панели настройки элемента откроется окно для ввода и редактирования программного кода.

`ScriptTaskAddContact`

```
/* Создание экземпляра схемы объекта [Контакт]. */
var schema = UserConnection.EntitySchemaManager.GetInstanceByName("Contact");
/* Создание экземпляра нового объекта. */
var entity = schema.CreateEntity(UserConnection);
/* Установка значений по умолчанию для колонок объекта. */
entity.SetDefColumnValues();
string contactName = Get<string>("ContactName");
string contactPhone = Get<string>("ContactPhone");
/* Установка значения колонки [Name] из параметра процесса. */
entity.SetColumnValue("Name", contactName);
/* Установка значения колонки [Phone] из параметра процесса. */
entity.SetColumnValue("Phone", contactPhone);
/* Сохранение нового контакта. */
entity.Save();
return true;
```

7. После внесения изменений сохраните бизнес-процесс, нажав на кнопку [*Сохранить*] ([*Save*]) на панели инструментов дизайнера процессов.

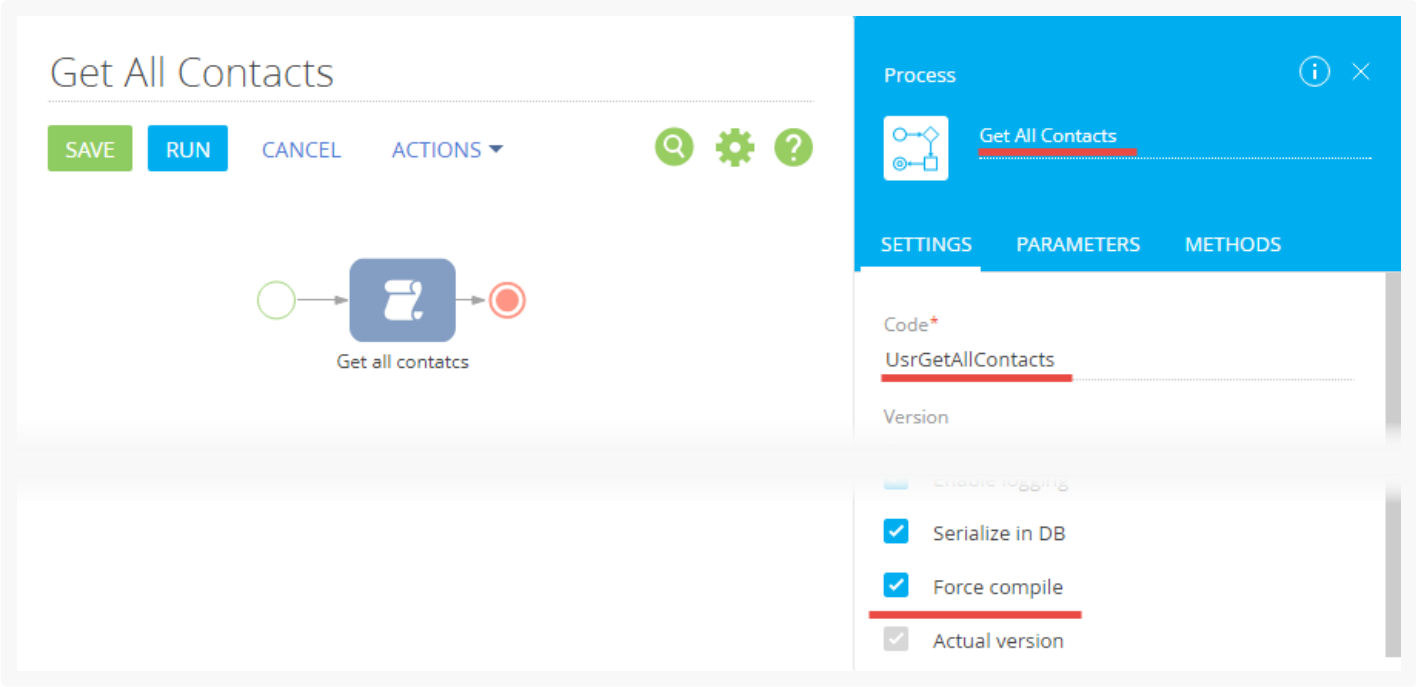
2. Создать процесс чтения контактов

Бизнес-процесс, формирующий список всех контактов, также содержит один элемент [*Задание-сценарий*] ([*ScriptTask*]), в котором реализуется необходимая логика.

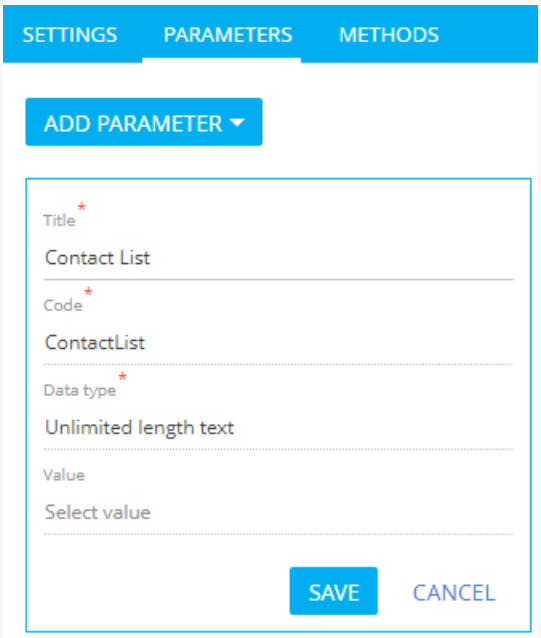
Значения **свойств бизнес-процесса**:

- [*Название*] ([*Name*]) — "`Get All Contacts`".
- [*Код*] ([*Code*]) — "`UsrGetAllContacts`".

Для остальных свойств оставьте значения по умолчанию.



Процесс `UsrGetAllContacts` содержит единственный параметр `ContactList`, через который процесс будет возвращать список всех контактов системы в виде JSON-объекта. Тип параметра — **строка неограниченной длины**.



Логику выборки контактов реализуйте в элементе процесса [*Задание-сценарий*] ([*ScriptTask*]).

Значения **свойств элемента**:

- [*Название*] ([*Name*]) — "`Get all contacts`".
- [*Код*] ([*Code*]) — "`ScriptTaskGetAllContacts`".

`ScriptTaskGetAllContacts`

```
/* Создание экземпляра EntitySchemaQuery. */
EntitySchemaQuery query = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "Contact");
/* Признак для обязательного выбора первичной колонки [Id]. */
query.PrimaryQueryColumn.IsAlwaysSelect = true;
/* Добавление в запрос колонок. */
query.AddColumn("Name");
query.AddColumn("Phone");
/* Получение результирующей коллекции. */
var entities = query.GetEntityCollection(UserConnection);
/* Формирование списка контактов для сериализации в JSON. */
List<object> contacts = new List<object>();
foreach (var item in entities)
{
    var contact = new
    {
        Id = item.GetTypedColumnValue<Guid>("Id"),
        Name = item.GetTypedColumnValue<string>("Name"),
        Phone = item.GetTypedColumnValue<string>("Phone")
    };
    contacts.Add(contact);
}
```

```
/* Сохранение сериализованной в JSON коллекции контактов в параметр ContactList. */
string contactList = JsonConvert.SerializeObject(contacts);
Set<string>("ContactList", contactList);
return true;
```

После внесения изменений сохраните и опубликуйте бизнес-процесс.

3. Запустить выполнение бизнес-процессов из строки навигации браузера

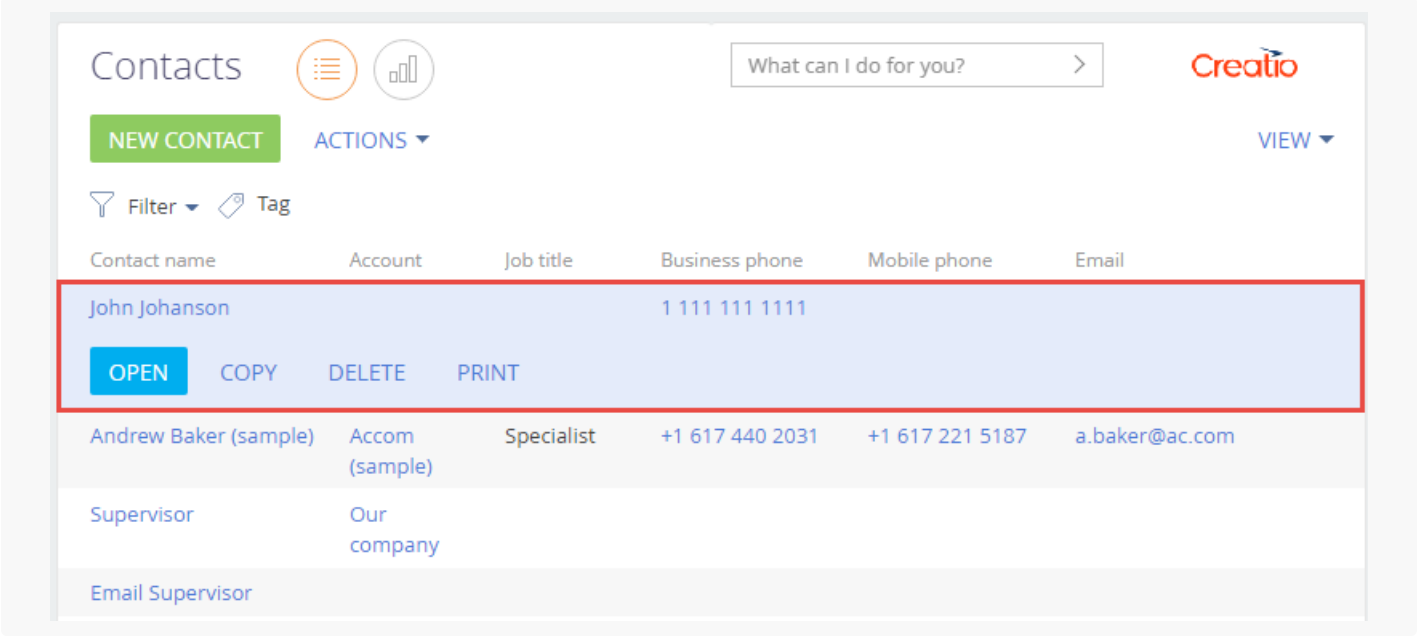
Поскольку вызов метода сервиса возможен с помощью `GET`-запроса, то для выполнения запуска бизнес-процесса можно использовать браузер.

Для запуска процесса создания нового контакта в строку навигации браузера введите URL:

URL для запуска процесса создания нового контакта

```
http[s]://[Адрес приложения Creatio]/0/ServiceModel/ProcessEngineService.svc/UsrAddNewExternalCo
```

После перехода по указанному URL, в приложении будет добавлен новый контакт.



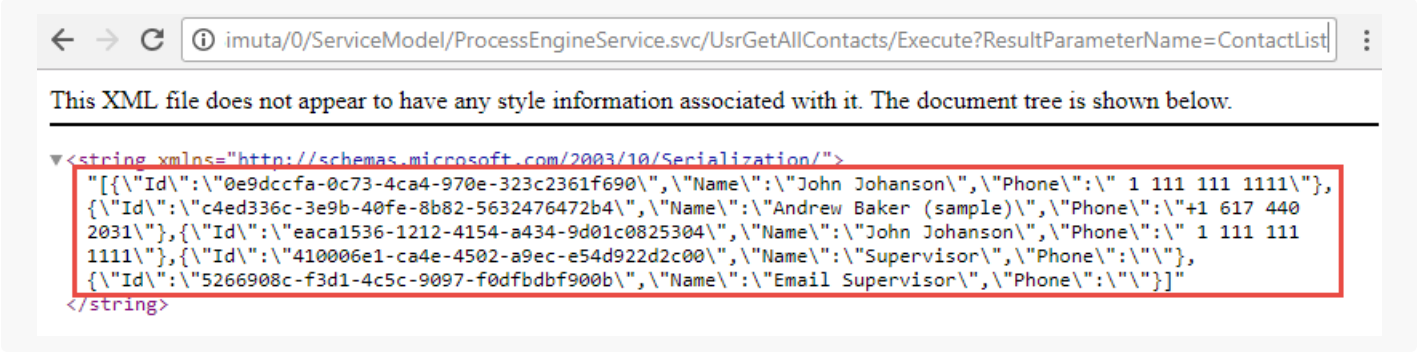
Важно. Новый контакт будет создан при каждом успешном запросе к сервису. Если выполнить несколько запросов с одинаковыми параметрами, будет создано несколько контактов-дублей.

Для запуска процесса чтения всех контактов в строку навигации браузера введите URL:

URL для запуска процесса чтения всех контактов

```
http[s]://[Адрес приложения Creatio]/0/ServiceModel/ProcessEngineService.svc/UsrGetAllContacts/E
```

После выполнения перехода по указанному URL, в окне браузера будет отображен JSON-объект, содержащий коллекцию контактов.



4. Запустить выполнение бизнес-процессов из консольного приложения

Полностью исходный код консольного пользовательского приложения, предназначенного для запуска бизнес-процессов с помощью сервиса `ProcessEngineService.svc`, доступен на GitHub.

Чтобы **запустить выполнение** бизнес-процессов из консольного приложения:

1. Выполните **аутентификацию**. Для этого предназначен сервис аутентификации AuthService.svc. Консольное приложение для выполнения аутентификации можно взять за основу для примера, приведенного ниже.
2. Для формирования запросов к сервису `ProcessEngineService.svc` в исходный код класса `Program` добавьте строковое поле, содержащее базовый URL сервиса.

   URL для формирования запросов к сервису `ProcessEngineService.svc`

   ```
   private const string processServiceUri = baseUri + @"/0/ServiceModel/ProcessEngineService.svc
   ```

3. Для выполнения запуска бизнес-процесса добавления контакта в исходный код класса `Program` добавьте метод `GET`.

   Метод `GET` для выполнения запуска бизнес-процесса

   ```
   public static void AddContact(string contactName, string contactPhone)
   {
       /* Формирование URL запроса. */
       string requestString = string.Format(processServiceUri +
               "UsrAddNewExternalContact/Execute?ContactName={0}&ContactPhone={1}",
   ```

```
                                contactName, contactPhone);
    /* Создание Http-запроса. */
    HttpWebRequest request = HttpWebRequest.Create(requestString) as HttpWebRequest;
    request.Method = "GET";
    request.CookieContainer = AuthCookie;
    /* Выполнение запроса и анализ Http-ответа. */
    using (var response = request.GetResponse())
    {
        /* Поскольку сервис вернет пустую строку, то можно вывести свойства http-ответа. */
        Console.WriteLine(response.ContentLength);
        Console.WriteLine(response.Headers.Count);
    }
}
```

4. Добавьте метод запуска процесса чтения контактов.

Метод `GET` для выполнения запуска бизнес-процесса

```
public static void GetAllContacts()
{
    /* Формирование URL запроса. */
    string requestString = processServiceUri +
                        "UsrGetAllContacts/Execute?ResultParameterName=ContactList";
    HttpWebRequest request = HttpWebRequest.Create(requestString) as HttpWebRequest;
    request.Method = "GET";
    request.CookieContainer = AuthCookie;
    /* Создание Http-запроса. */
    using (var response = request.GetResponse())
    {
        /* Выполнение запроса и вывод результата. */
        using (var reader = new StreamReader(response.GetResponseStream()))
        {
            string responseText = reader.ReadToEnd();
            Console.WriteLine(responseText);
        }
    }
}
```

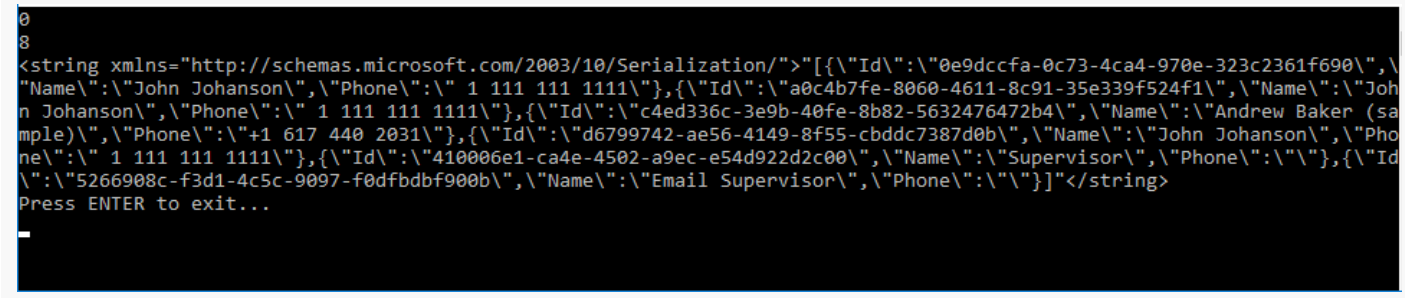
5. Вызов добавленных методов выполните в главном методе программы после успешной аутентификации.

Вызов методов

```
static void Main(string[] args)
{
    if (!TryLogin("Supervisor", "Supervisor"))
    {
```

```
            Console.WriteLine("Wrong login or password. Application will be terminated.");
        }
        else
        {
            try
            {
                /* Вызов методов запуска бизнес-процессов. */
                AddContact("John Johanson", "+1 111 111 1111");
                GetAllContacts();
            }
            catch (Exception)
            {
                /* Обработка исключений. */
                throw;
            }
        };
        Console.WriteLine("Press ENTER to exit...");
        Console.ReadLine();
    }
```

Результат выполнения пользовательского приложения



Запустить процесс из клиентского модуля

Средний

Чтобы запустить бизнес-процесс из JavaScript-кода клиентской схемы, необходимо:

1. В модуль страницы, из которой вызывается сервис, подключить в качестве зависимости модуль `ProcessModuleUtilities`. Этот модуль предоставляет удобный интерфейс для выполнения запросов к сервису ProcessEngineService.svc.
2. Вызвать метод `executeProcess(args)` модуля `ProcessModuleUtilities`, передав ему в качестве параметра объект `args` с такими свойствами:

Свойства объекта args

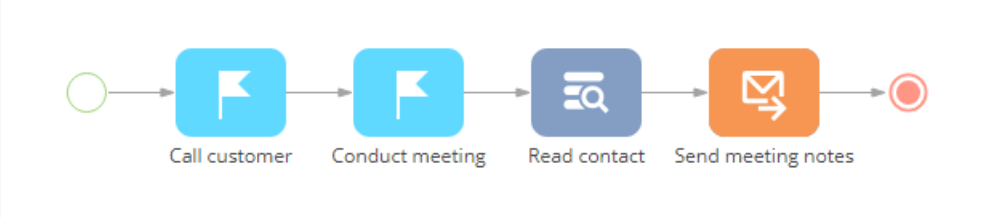
Свойство	Описание
`sysProcessName`	Имя вызываемого процесса (необязательное свойство в случае, если определено свойство `sysProcessId`).
`sysProcessId`	Уникальный идентификатор вызываемого процесса (необязательное свойство в случае, если определено свойство `sysProcessName`).
`parameters`	Объект, свойствами которого являются входящие параметры вызываемого процесса.

Пример. На страницу контрагента добавить действие, по которому будет запускаться бизнес-процесс "Проведение встречи". В бизнес-процесс передать в качестве параметра основной контакт контрагента.

1. Создать и настроить пользовательский бизнес-процесс "Проведение встречи"

1. Создайте бизнес-процесс.

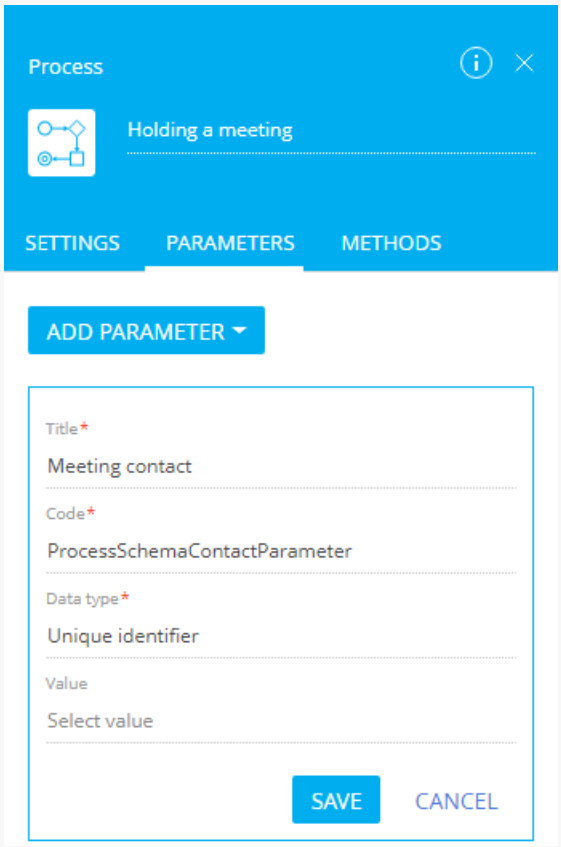
В примере используется бизнес-процесс "Проведение встречи", создание которого детально описано в разделе "Как работать с email" документации по настройке процессов .



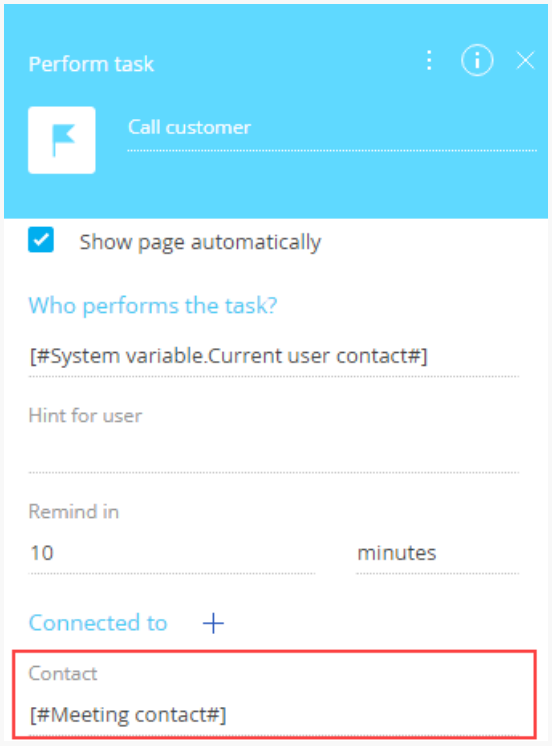
1. Добавьте в бизнес-процесс параметр.

Заполните **свойства параметра**:

- [*Заголовок*] ([*Title*]) — "Meeting contact".
- [*Код*] ([*Code*]) — "`ProcessSchemaContactParameter`".
- [*Тип данных*] ([*Data type*]) — "Уникальный идентификатор" ("Unique identifier").



1. В свойствах первого действия процесса [*Позвонить клиенту*] ([*Call customer*]) заполните поле [*Контакт*] ([*Contact*]) входящим параметром процесса.



2. Создать замещающую страницу контрагента и добавить на нее действие

Добавление действия на страницу записи подробно описано в статье Добавить действие на страницу записи.

В схему замещающего модуля страницы записи и схему раздела контрагента добавьте локализуемую строку `CallProcessCaption` с заголовком действия, например, "Назначить встречу" ("Schedule a meeting").

Дополнительно в объявлении модуля страницы записи подключите в качестве зависимости модуль `ProcessModuleUtilities`.

Исходные коды схемы раздела и страницы записи раздела [*Контрагенты*] ([*Accounts*]) приведены ниже.

3. Добавить необходимые методы в схемы

Для запуска процесса воспользуйтесь методом `executeProcess()` модуля `ProcessModuleUtilities`, в который в качестве параметра необходимо передать объект со следующими свойствами:

- имя созданного бизнес-процесса,
- объект с проинициализированными входящими параметрами для процесса.

В исходном коде, приведенном ниже, это реализовано в методе `callCustomProcess()`. Также реализованы методы проверки существования основного контакта `isAccountPrimaryContactSet()` и добавления элементов меню действий `getActions()`.

Исходный код замещающего модуля страницы записи:

```
define("AccountPageV2", ["ProcessModuleUtilities"], function(ProcessModuleUtilities) {
    return {
        // Название схемы объекта страницы записи.
        entitySchemaName: "Account",
        // Методы модели представления страницы записи.
        methods: {
            // Проверяет, заполнено ли поле [Основной контакт] страницы.
            isAccountPrimaryContactSet: function() {
                return this.get("PrimaryContact") ? true : false;
            },
            // Переопределение базового виртуального метода, возвращающего коллекцию действий ст
            getActions: function() {
                // Вызывается родительская реализация метода для получения
                // коллекции проинициализированных действий базовой страницы.
                var actionMenuItems = this.callParent(arguments);
                // Добавление линии-разделителя.
                actionMenuItems.addItem(this.getActionsMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                // Добавление пункта меню [Назначить встречу] в список действий страницы записи.
```

```
                actionMenuItems.addItem(this.getActionsMenuItem({
                    // Привязка заголовка пункта меню к локализуемой строке схемы.
                    "Caption": { bindTo: "Resources.Strings.CallProcessCaption" },
                    // Привязка метода-обработчика действия.
                    "Tag": "callCustomProcess",
                    // Привязка свойства видимости пункта меню к значению, которое возвращает ме
                    "Visible": { bindTo: "isAccountPrimaryContactSet" }
                }));
                return actionMenuItems;
            },
            // Метод-обработчик действия.
            callCustomProcess: function() {
                // Получение идентификатора основного контакта контрагента.
                var contactParameter = this.get("PrimaryContact");
                // Объект, который будет передан в качестве аргумента в метод executeProcess().
                var args = {
                    // Имя процесса, который необходимо запустить.
                    sysProcessName: "UsrCustomProcess",
                    // Объект со значением входящего параметра ContactParameter для процесса Cus
                    parameters: {
                        ProcessSchemaContactParameter: contactParameter.value
                    }
                };
                // Запуск пользовательского бизнес-процесса.
                ProcessModuleUtilities.executeProcess(args);
            }
        }
    };
});
```

Для корректного отображения действия в меню действий в совмещенном режиме отображения страницы с вертикальным реестром добавьте реализацию метода `isAccountPrimaryContactSet()` в схему раздела `.`

Исходный код замещающего модуля схемы раздела:

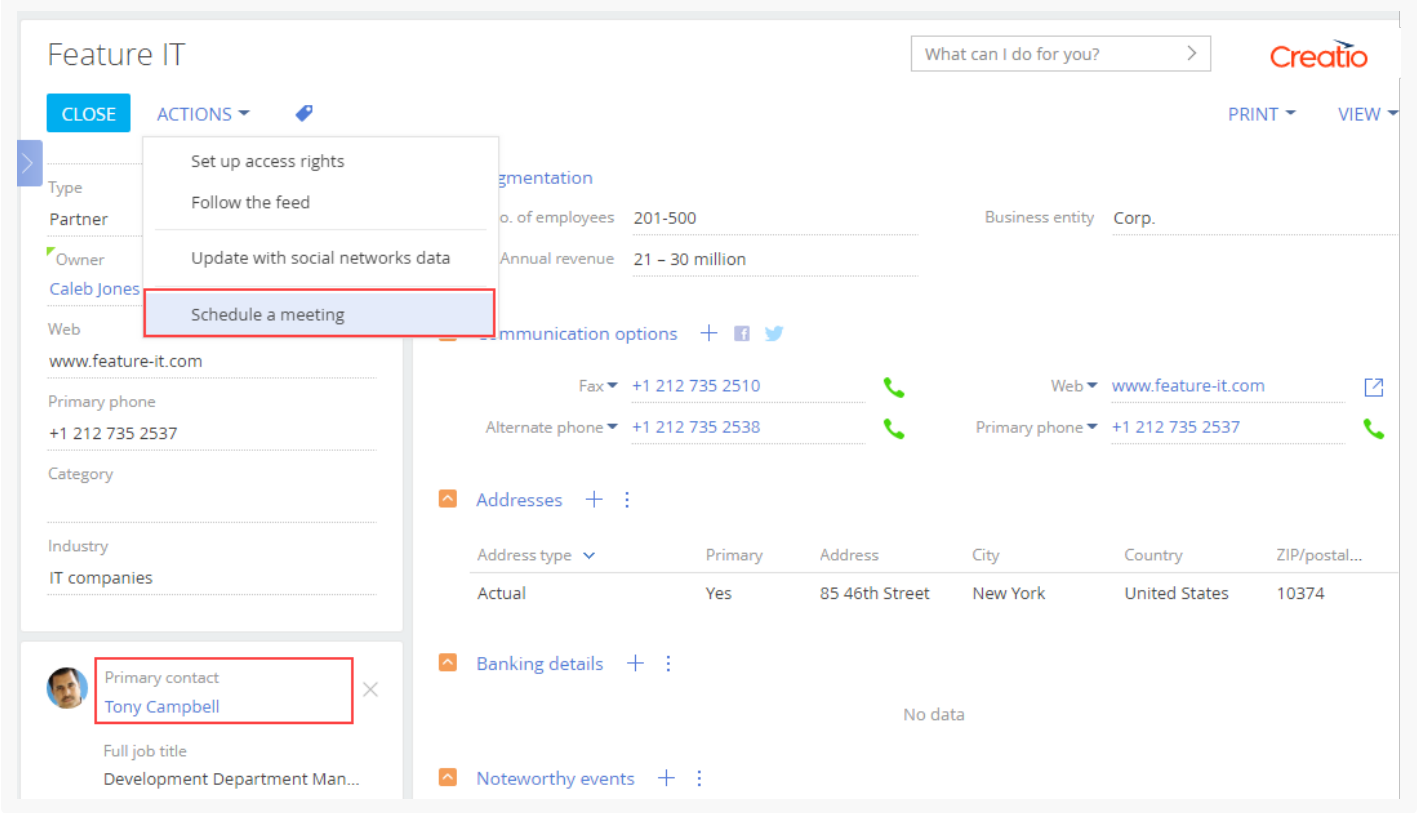
```
define("AccountSectionV2", [], function() {
    return {
        // Название схемы раздела.
        entitySchemaName: "Account",
        methods: {
            // Проверяет, заполнено ли поле [Основной контакт] выбранной записи.
            isAccountPrimaryContactSet: function() {
                // Определение активной записи.
                var activeRowId = this.get("ActiveRow");
                if (!activeRowId) {
                    return false;
```

```
                }
                // Получение коллекции данных списочного представления реестра раздела.
                // Получение модели выбранного контрагента по заданому значению первичной колонк
                var selectedAccount = this.get("GridData").get(activeRowId);
                if (selectedAccount) {
                    // Получение свойства модели — наличие основного контакта.
                    var selectedPrimaryContact = selectedAccount.get("PrimaryContact");
                    // Метод возвращает true, если основной контакт установлен. Иначе возвращает
                    return selectedPrimaryContact ? true : false;
                }
                return false;
            }
        }
    };
});
```

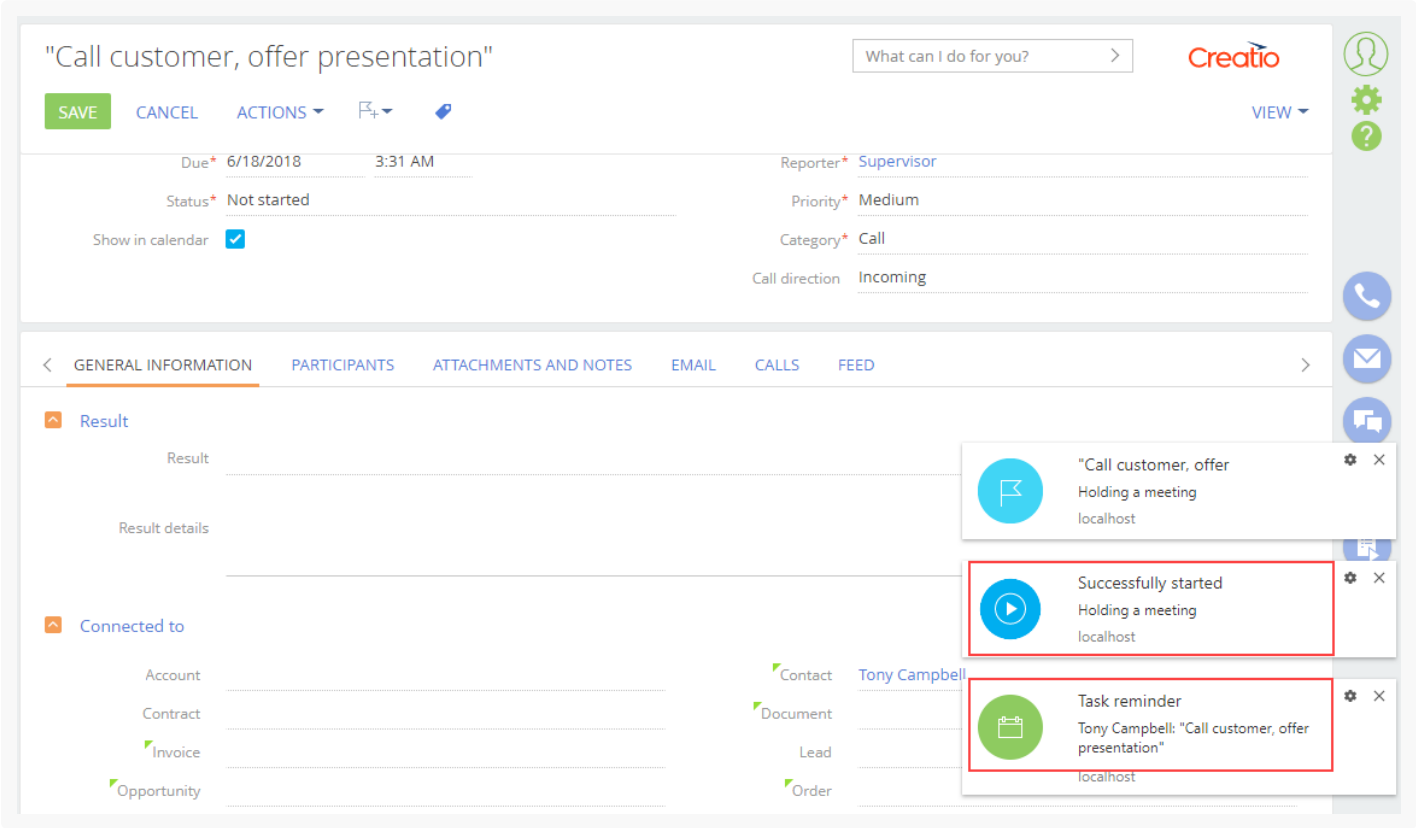
Результат выполнения примера

После сохранения схем и обновления страницы приложения в меню действий страницы контрагента появится новое действие [*Назначить встречу*] ([*Schedule a meeting*]) . Это действие будет доступным только в случае наличия основного контакта для активной записи реестра. При выполнении действия будет запущен пользовательский бизнес-процесс "Проведение встречи" ("Holding a meeting"). При этом в параметр бизнес-процесса будет передан основной контакт контрагента.

Вызов бизнес-процесса по действию на странице записи



Результат запуска бизнес-процесса. Передача параметра со страницы контрагента в бизнес-процесс



Веб-сервис ProcessEngineService.svc API

Сложный

Структура запроса

```
/* Строка запроса. */
GET Creatio_application_address/0/ServiceModel/ProcessEngineService.svc/[processSchemaName/]meth

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

Структура ответа для метода Execute()

```
/* Код состояния. */
Status: code

/* Тело ответа. */

    "[
```

```
        {\"Id\":\"object1_id\",\"object1 field1\":\"object1 field_value1\",\"object1 field2\":\"
        {\"Id\":\"object2_id\",\"object2 field1\":\"object2 field_value1\",\"object2 field2\":\"
  ...
  {},
    ]"
```

Строка запроса

`GET` **required**

Метод запроса на запуск бизнес-процессов.

`Creatio_application_address` **required**

Адрес приложения Creatio.

`ServiceModel` **required**

Путь к сервису запуска бизнес-процессов. Неизменяемая часть запроса.

`ProcessEngineService.svc` **required**

Адрес веб-сервиса запуска бизнес-процессов. Неизменяемая часть запроса.

`methodName` **required**

Метод веб-сервиса запуска бизнес-процессов.

Основные **методы**:

- `Execute()` — запуск бизнес-процесса. Позволяет передавать набор входящих параметров и возвращать результат выполнения веб-сервиса.
- `ExecProcElByUId()` — запуск отдельного элемента бизнес-процесса. Запускать на выполнение можно только элемент выполняющегося процесса. Если элемент процесса, запускаемый методом `ExecProcElByUId()`, уже выполнен на момент вызова метода, то повторно такой элемент выполняться не будет.

`ProcessSchemaName` **optional**

Используется для метода `Execute()`.

Название схемы бизнес-процесса. Название схемы бизнес-процесса можно узнать в разделе [*Конфигурация*] ([*Configuration*]).

`ResultParameterName` **optional**

Используется для метода `Execute()`.

Переменная для кода параметра процесса. Неизменяемая часть запроса.

`resultParameterName` **optional**

Используется для метода `Execute()`.

Код параметра процесса, который хранит результат выполнения процесса. Если этот параметр не задан, то веб-сервис запустит указанный бизнес-процесс без ожидания результата его выполнения. Если в вызываемом процессе отсутствует данный код параметра, веб-сервис вернет значение `null`.

`inputParameter` **optional**

Используется для метода `Execute()`.

Переменная для кода входящих параметров бизнес-процесса. Если передается несколько входящих параметров, то они должны быть объединены символом `&`.

`inputParameterValue` **optional**

Используется для метода `Execute()`.

Значения входящих параметров бизнес-процесса.

`ProcessElementUID` **optional**

Используется для метода `ExecProcElByUId()`.

Идентификатор запускаемого элемента процесса.

Заголовки запроса

`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 ответа

`[]`

Коллекция объектов.

`{}`

Экземпляры объектов коллекции.

`Id`

Название поля [*Id*].

`object1_id, object2_id, ...`

Идентификатор 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. Может присутствовать только для `GET` и `POST` запросов.