

Интеграция и внешний API

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	4
Запуск бизнес-процессов из внешнего приложения	4
Запуск бизнес-процесса из front-end части	4
Запустить бизнес-процесс через веб-сервис	5
1. Создать процесс добавления контакта	5
2. Создать процесс чтения контактов	8
3. Запустить выполнение бизнес-процессов из строки навигации браузера	10
4. Запустить выполнение бизнес-процессов из консольного приложения	11
Запустить процесс из клиентского модуля	13
1. Создать и настроить пользовательский бизнес-процесс "Проведение встречи"	14
2. Создать замещающую страницу контрагента и добавить на нее действие	16
3. Добавить необходимые методы в схемы	16
Результат выполнения примера	18
Веб-сервис ProcessEngineService.svc	19
Строка запроса	20
Заголовки запроса	21
Тело ответа	21

Сервис запуска бизнес-процессов



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
<code>sysProcessName</code>	Имя вызываемого процесса (необязательное свойство в случае, если определено свойство <code>sysProcessId</code>).
<code>sysProcessId</code>	Уникальный идентификатор вызываемого процесса (необязательное свойство, если определено свойство <code>sysProcessName</code>).
<code>parameters</code>	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.

Process

Add New External Contact

SETTINGS PARAMETERS METHODS

Code*
UsrAddNewExternalContact

Version
0

Tag
Business Process

Process description

Package*
sdkWorkWithBpmByWebServices

Maximum Number of Repetitions
100

Process instance caption

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)").

SETTINGS PARAMETERS METHODS

ADD PARAMETER ▼

Title *

Contact Name

Code *

ContactName

Data type *

Text (50 characters)

Value

Select value

SAVE CANCEL

T Contact Phone

Select value

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.

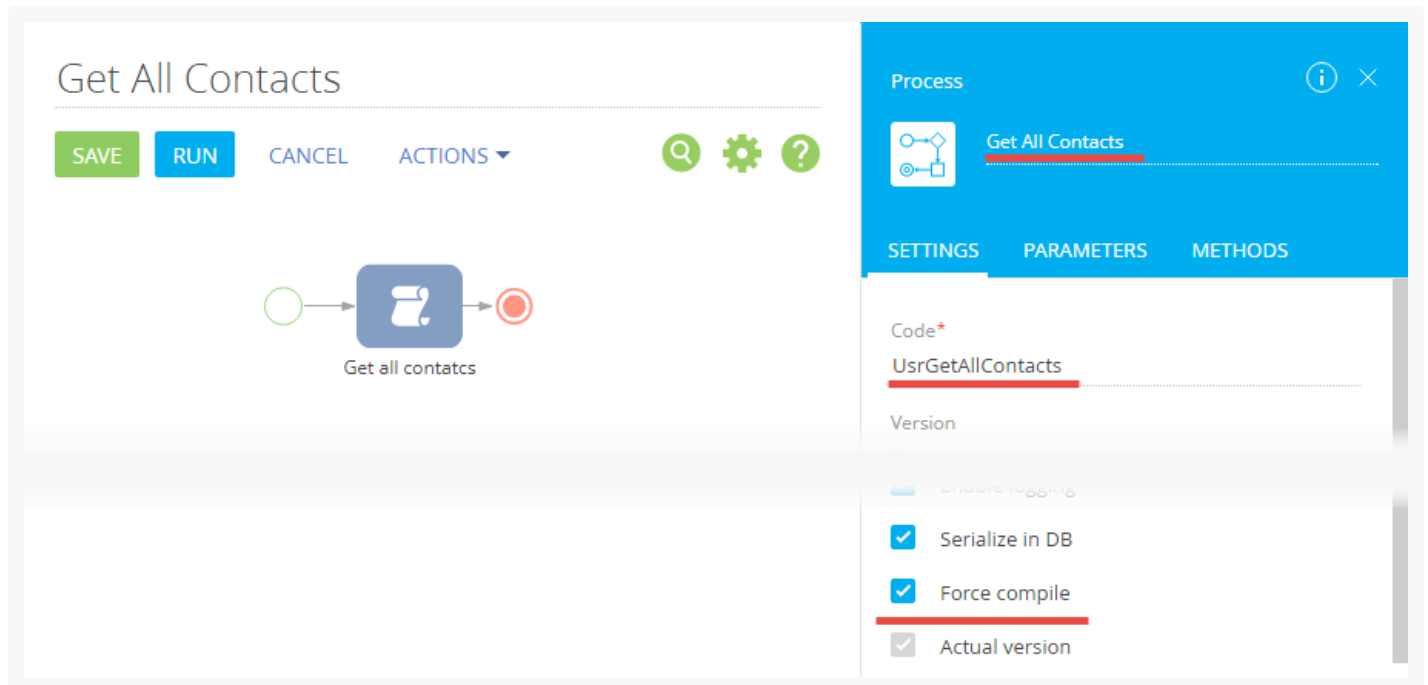
2. Создать процесс чтения контактов

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**.

SETTINGS
PARAMETERS
METHODS

ADD PARAMETER ▼

Title *

Contact List

Code *

ContactList

Data type *

Unlimited length text

Value

Select value

SAVE CANCEL

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

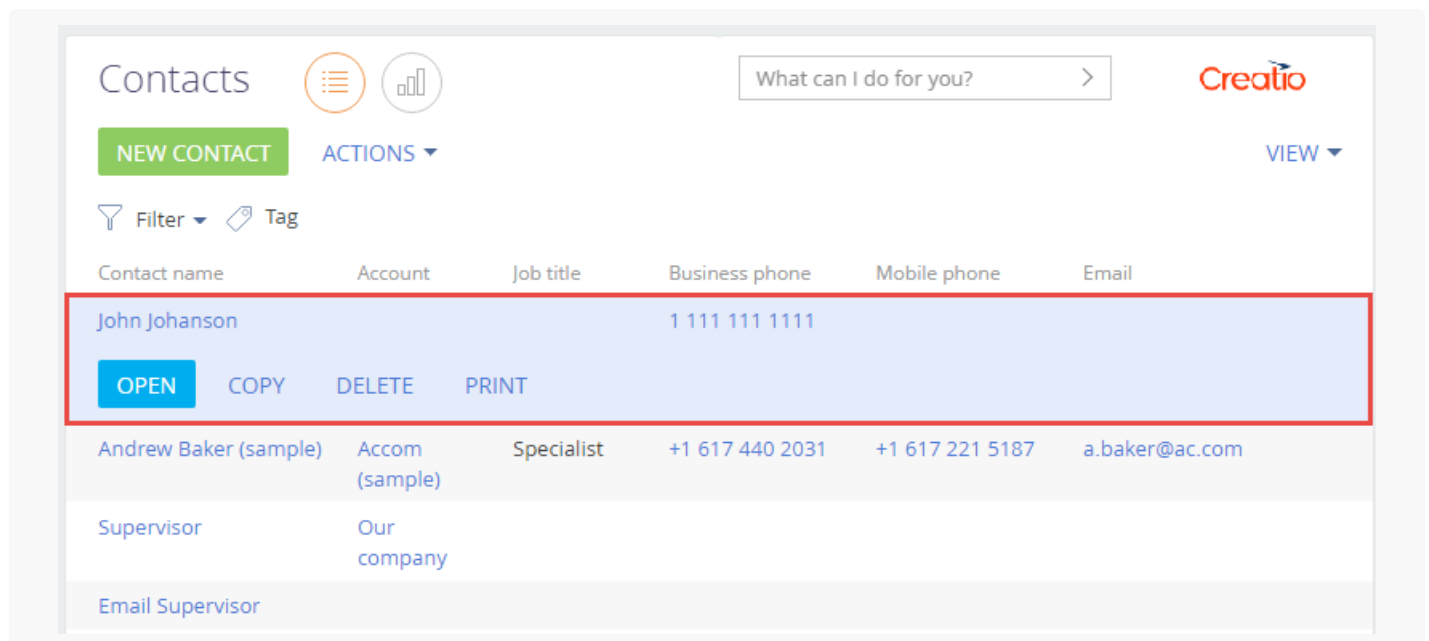
Поскольку вызов метода сервиса возможен с помощью `GET`-запроса, то для выполнения запуска бизнес-процесса можно использовать браузер.

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/UsrAddNewExternalCc`

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



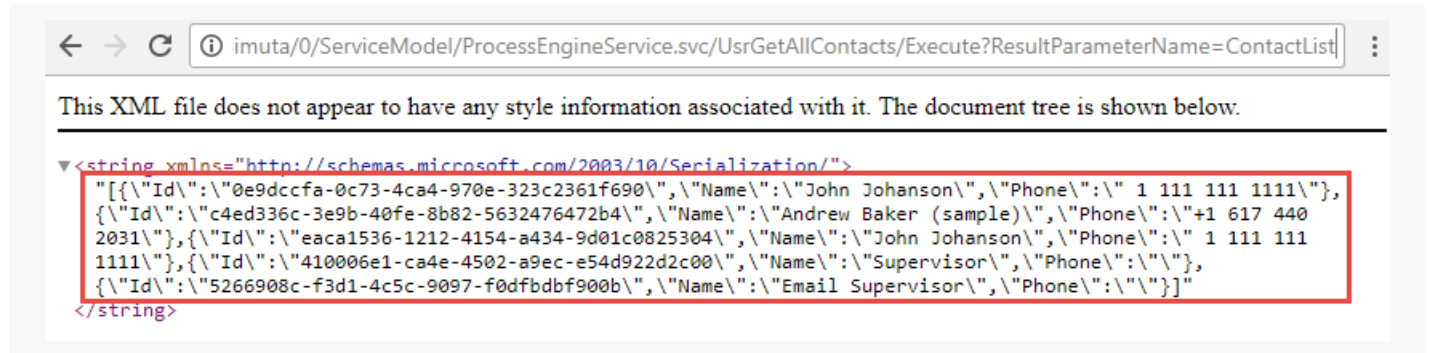
Важно. Новый контакт будет создан при каждом успешном запросе к сервису. Если выполнить несколько запросов с одинаковыми параметрами, будет создано несколько контактов-дублей.

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

```

0
8
<string xmlns="http://schemas.microsoft.com/2003/10/Serialization/">[{"Id":"0e9dccfa-0c73-4ca4-970e-323c2361f690",\
"Name":"John Johanson",\Phone":" 1 111 111 1111"}, {"Id":"a0c4b7fe-8060-4611-8c91-35e339f524f1",\Name":"Joh
n Johanson",\Phone":" 1 111 111 1111"}, {"Id":"c4ed336c-3e9b-40fe-8b82-5632476472b4",\Name":"Andrew Baker (sa
mple)",\Phone":"+1 617 440 2031"}, {"Id":"d6799742-ae56-4149-8f55-cbddc7387d0b",\Name":"John Johanson",\Pho
ne":" 1 111 111 1111"}, {"Id":"410006e1-ca4e-4502-a9ec-e54d922d2c00",\Name":"Supervisor",\Phone":"\"}, {"Id
\":"5266908c-f3d1-4c5c-9097-f0dfbdf900b",\Name":"Email Supervisor",\Phone":"\"}]</string>
Press ENTER to exit...
-

```

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

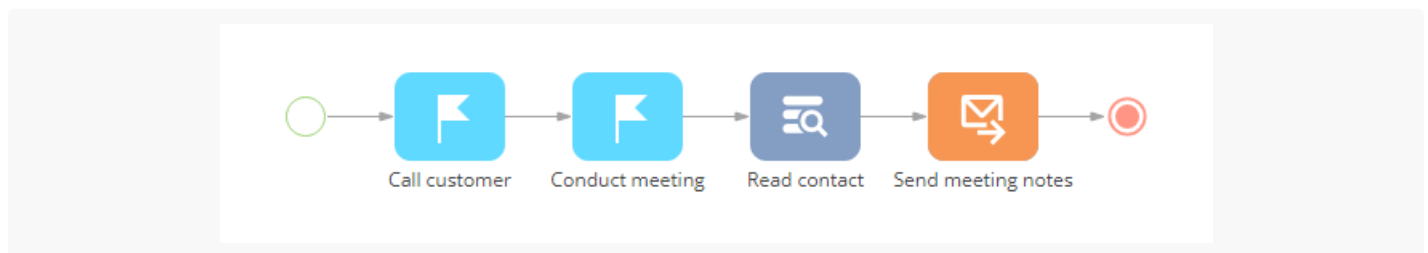
Свойство	Описание
sysProcessName	Имя вызываемого процесса (необязательное свойство в случае, если определено свойство sysProcessId).
sysProcessId	Уникальный идентификатор вызываемого процесса (необязательное свойство в случае, если определено свойство sysProcessName).
parameters	Объект, свойствами которого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

The screenshot shows a configuration window titled 'Process' with a close button (X) and an information icon (i). Below the title bar, there is a process icon and the name 'Holding a meeting'. The window has three tabs: 'SETTINGS', 'PARAMETERS' (which is selected), and 'METHODS'. Under the 'PARAMETERS' tab, there is a blue button labeled 'ADD PARAMETER' with a dropdown arrow. Below this button is a form with the following fields:

- Title***: Meeting contact
- Code***: ProcessSchemaContactParameter
- Data type***: Unique identifier
- Value**: Select value

 At the bottom right of the form are two buttons: 'SAVE' and 'CANCEL'.

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

The screenshot shows a configuration window titled 'Perform task' with a menu icon (three dots), an information icon (i), and a close button (X). Below the title bar, there is a task icon and the name 'Call customer'. The window contains the following settings:

- ☒ Show page automatically
- Who performs the task?**: [#System variable.Current user contact#]
- Hint for user**: (empty field)
- Remind in**: 10 minutes
- Connected to**: +
- Contact**: [#Meeting contact#] (this field is highlighted with a red rectangle)

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 заданному значению первичной колонки
    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

The screenshot displays the 'Feature IT' contact record in the Creatio system. The 'ACTIONS' menu is open, showing the following options: 'Set up access rights', 'Follow the feed', 'Update with social networks data', and 'Schedule a meeting'. The 'Schedule a meeting' option is highlighted with a red box. The contact details for Caleb Jones are visible, including his primary phone number (+1 212 735 2537) and the 'Schedule a meeting' action being highlighted in the ACTIONS menu. The 'Primary contact' field shows Tony Campbell. The 'Addresses' section shows a primary address in New York, and the 'Banking details' section shows 'No data'.

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

"Call customer, offer presentation"

What can I do for you?

SAVE CANCEL ACTIONS

VIEW

Due* 6/18/2018 3:31 AM

Reporter* Supervisor

Status* Not started

Priority* Medium

Category* Call

Call direction Incoming

Show in calendar ☒

GENERAL INFORMATION PARTICIPANTS ATTACHMENTS AND NOTES EMAIL CALLS FEED

Result

Result details

Connected to

Account

Contract

Invoice

Opportunity

Contact Tony Campbell

Document

Lead

Order

"Call customer, offer presentation"
Holding a meeting
localhost

Successfully started
Holding a meeting
localhost

Task reminder
Tony Campbell: "Call customer, offer presentation"
localhost

Веб-сервис ProcessEngineService.svc API

Сложный

Структура запроса

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

/* 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.
- `ExecProcElById()` — запуск отдельного элемента бизнес-процесса. Запускать на выполнение можно только элемент выполняющегося процесса. Если элемент процесса, запускаемый методом `ExecProcElById()`, 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.