

# Панель действий

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

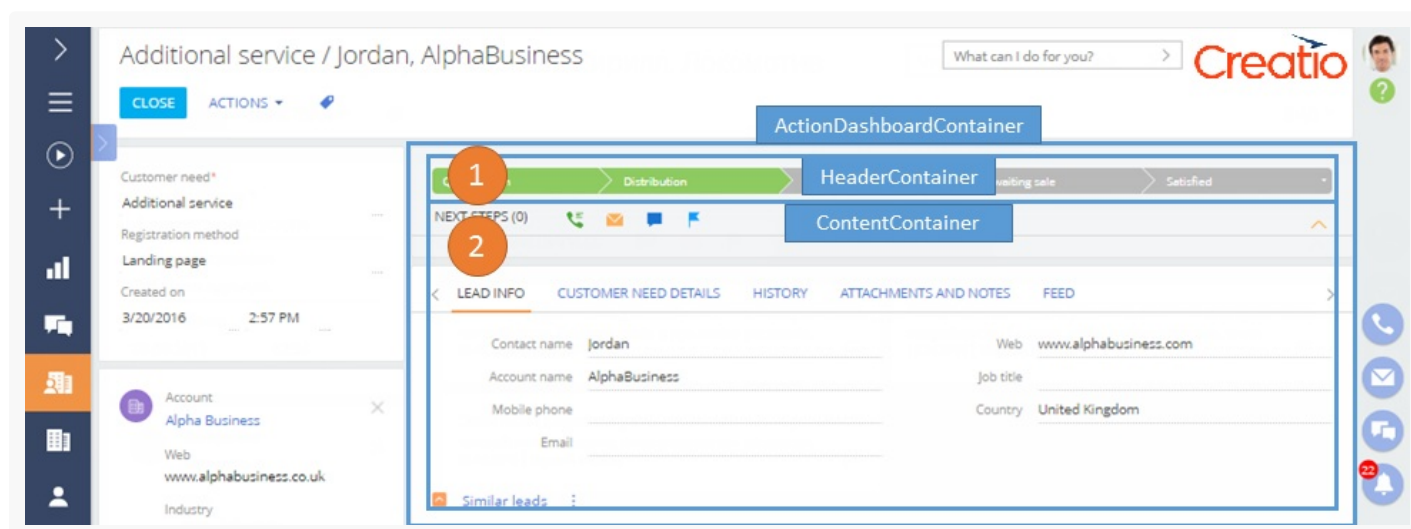
<b>Панель действий</b>	<b>4</b>
Добавить панель действий на страницу	4
Добавить новый канал на панель действий	5
Добавить пользовательское действие верификации	5
<b>Добавить панель действий</b>	<b>7</b>
Описание примера	7
Исходный код	7
Алгоритм реализации примера	7
<b>Добавить новый канал на панель действий</b>	<b>11</b>
Описание примера	11
Исходный код	11
Алгоритм выполнения примера	11
<b>Добавить мультиязычные шаблоны email-сообщений</b>	<b>18</b>
Описание примера	19
Исходный код	19
Предварительные настройки	19
Алгоритм реализации примера	20
<b>Создать пользовательскую страницу действия верификации</b>	<b>25</b>
1. Создать схему страницы действия верификации	26
2. Настроить представление страницы	27
2. Использовать созданную схему в бизнес-процессе	27
Результат выполнения примера	28

# Панель действий

## Оснoвы

Панель действий предназначена для отображения информации о текущем состоянии работы с записью. Она состоит из двух частей:

- **Индикатор стадий** (1) — показывает состояние этапов бизнес-процесса в тех разделах, где работа с записями выполняется с использованием бизнес-процесса.
- **Панель действий** (2):
  - Позволяет перейти к выполнению активности, работе с email-сообщениями или с лентой, не покидая раздел.
  - Отображает созданные по бизнес-процессу активности, которые находятся в неконечном состоянии и связаны с объектом раздела по соответствующему полю.
  - Может отображать автогенерируемую страницу, преднастроенную страницу, вопрос и страницу объекта в виде задач.



Панель действий расположена в контейнере `ActionDashboardContainer` страницы записи раздела. Индикатор стадий расположен во вложенном контейнере `HeaderContainer`, а панель действий — в `ContentContainer`.

Расположение элементов панели действий конфигурируется схемой модели представления `BaseActionsDashboard` и унаследованной схемой `SectionActionsDashboard` пакета `ActionsDashboard`.

## Добавить панель действий на страницу

1. Создайте схему модели представления, унаследованную от `SectionActionsDashboard`.
2. Создайте схему замещающей модели представления страницы.

3. В свойстве `modules` схемы замещающей модели представления страницы выполните настройку модуля.
4. В свойстве `diff` схемы замещающей модели представления страницы добавьте модуль на страницу.

## Добавить новый канал на панель действий

**Каналы в панели действий** — это способ коммуникации с контактом. Канал создается для каждого раздела, в котором он подключен, например, для обращения, контакта или лида.

Чтобы **добавить новый канал на панель действий**:

1. Создайте класс-наследник базового класса `BaseMessagePublisher`.
2. Создайте схему замещающей модели представления `SectionActionsDashboard`.
3. В свойстве `diff` схемы замещающей модели представления укажите операцию `insert` для вставки вкладки `CallsMessageTab` и контейнера сообщений, а также укажите модуль, который будет отрисовываться в данном канале на одной из вкладок.
4. Переопределите методы:
  - `getSectionPublishers()` — добавляет созданный канал в список издателей сообщений.
  - `getExtendedConfig()` — определяет параметры вкладки.
5. Создайте модуль-контейнер для прорисовки в панели действий страницы, в которой будет реализована логика добавляемого канала.
6. Создайте схему модели представления, в которой будет реализована логика канала. В качестве родительской схемы установите `BaseMessagePublisherPage`.

## Добавить пользовательское действие верификации

Элемент процесса [ *Действие верификации* ] используется в продуктах линейки Financial Services Creatio при верификации заявки сотрудником компании. С его помощью можно создать проверку данных в **кредитной заявке** — набор необходимых действий верификации, которые должен провести ответственный сотрудник. При помощи этого элемента можно реализовать процесс принятия решения по кредитной заявке. От результата выполнения действия верификации зависит дальнейшее ветвление бизнес-процесса.

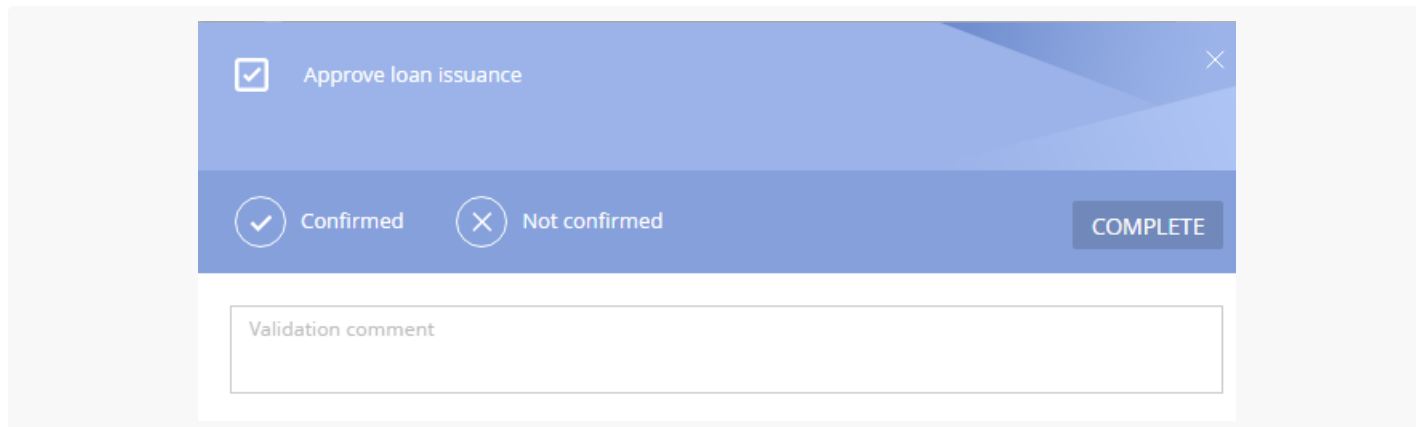
При создании настраиваемой страницы действия верификации, например, в бизнес-процессе [ *Подтверждение заявки* ] ( [ *Approve application* ]), есть возможность выбрать заранее созданную страницу с названием [ *Преднастроенная страница верификации* ] ( [ *Preconfigured verification page* ]).

Сама страница отображается, например, после нажатия на кнопку [ *Завершить* ] ([ *Complete* ]) активности [ *Согласовать выдачу кредита* ] ([ *Approve loan issuance* ]), которая создается при переходе заявки на стадию [ *Верификация* ] ([ *Validation* ]).

**Структурные элементы** преднастроенной страницы верификации:

- Кнопки выбора результата выполнения действия верификации.
- Поле [ *Комментарий* ] — содержит комментарий к действию верификации.
- Деталь [ *Сценарий разговора* ] — содержит скрипт-подсказку для верификатора. Доступна только в режиме чтения.
- Деталь [ *Файлы и ссылки* ] — содержит прикрепленные к действию верификации файлы и ссылки. Доступна только в режиме чтения.
- Деталь [ *Результаты проверок* ] — содержит контрольные вопросы и ответы на них.

**Важно.** Если деталь не содержит прикрепленные данные, то она не отображается на странице.



Creatio предоставляет возможность создавать пользовательские страницы верификации, наследующие преднастроенную.

Чтобы создать **пользовательскую страницу верификации**:

1. Создайте клиентскую схему страницы действия верификации.
2. Используйте созданную схему в бизнес-процессе.

## Добавить панель действий

 Средний

### Описание примера

Добавить инструментальную панель действий на страницу заказа.

### Исходный код

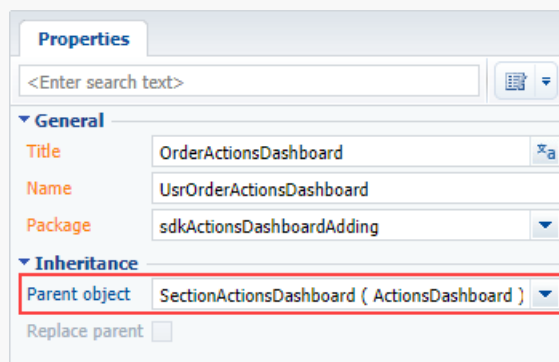
Пакет с реализацией примера можно скачать по [ссылке](#).

### Алгоритм реализации примера

#### 1. Создать схему модели представления OrderActionsDashboard

В качестве родительского объекта укажите схему `SectionActionsDashboard` (рис. 1).

Рис. 1. — Свойства схемы модели представления



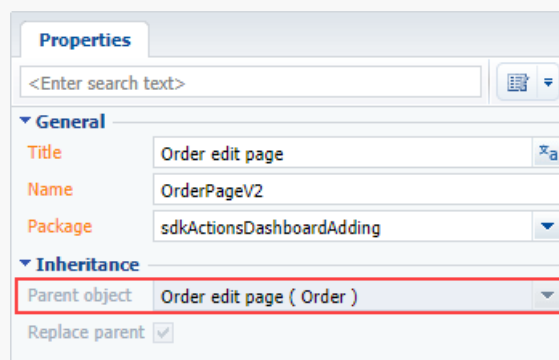
Исходный код схему модели представления:

```
define("UsrOrderActionsDashboard", [], function () {
    return {
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        methods: {},
        diff: /**SCHEMA_DIFF*/[ ]/**SCHEMA_DIFF*/
    };
});
```

## 2. Создать замещающую страницу заказа

Создайте схему замещающей модели представления, в которой в качестве родительского объекта укажите [ *Страница редактирования заказа* ] ([ *Order edit page* ], `OrderPageV2`) (рис. 2). Процесс создания схемы замещающей модели представления описан в статье ["Создать клиентскую схему"](#).

Рис. 2. — Свойства схему замещающей модели представления страницы записи



## 3. В коллекцию modules схемы страницы добавить конфигурационный объект с настройками модуля

На вкладку исходного кода добавьте код замещающего модуля страницы. В нем в коллекцию `modules` модели представления добавьте конфигурационный объект с настройками модуля.



## 4. В массив diff добавить конфигурационный объект с настройками расположения модуля на странице

Исходный код схемы замещающей модели представления:

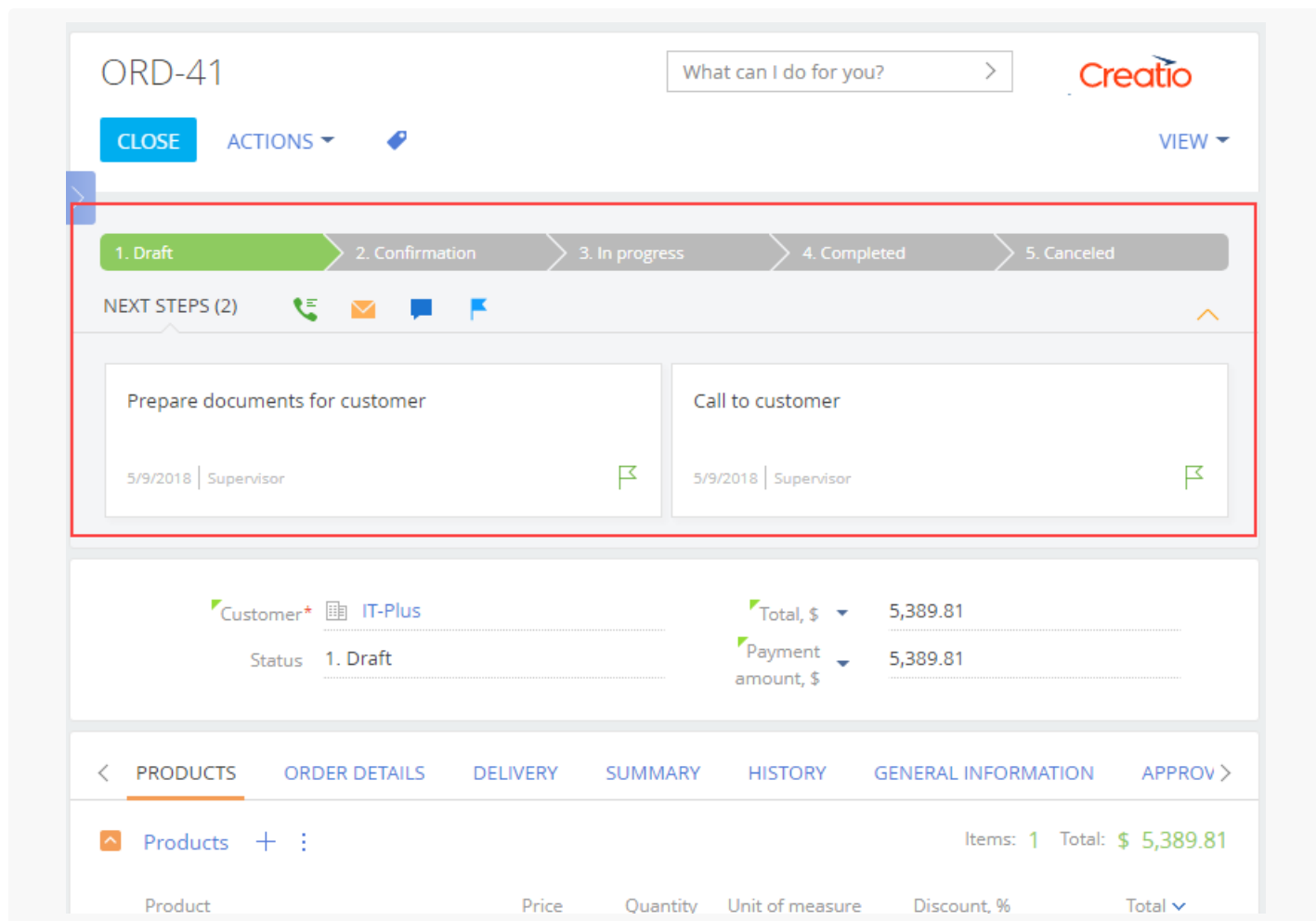
```
define("OrderPageV2", [],
function () {
return {
entitySchemaName: "Order",
attributes: {},
modules: /**SCHEMA_MODULES*/{
"ActionsDashboardModule": {
"config": {
"isSchemaConfigInitialized": true,
// Имя схемы.
"schemaName": "UsrOrderActionsDashboard",
"useHistoryState": false,
"parameters": {
// Конфигурационный объект модели представления.
"viewModelConfig": {
// Имя схемы сущности страницы.
"entitySchemaName": "Order",
// Конфигурационный объект блока Actions.
"actionsConfig": {
// Имя схемы для загрузки элементов в Actions.
"schemaName": "OrderStatus",
// Имя колонки в родительской схеме, ссылающейся на схему, с
// Если не указана, берет значение равное schemaName.
"columnName": "Status",
// Имя колонки для сортировки элементов.
"orderColumnName": "Position",
// Имя колонки для сортировки элементов в меню элемента.
"innerOrderColumnName": "Position"
},
// Отвечает за отображение модуля панели действий, значение [true] по умолчанию.
"useDashboard": true,
// Отвечает за отображение блока Content, значение [true] по умолчанию.
"contentVisible": true,
// Отвечает за отображение блока Header, значение [true] по умолчанию.
"headerVisible": true,
// Конфигурационный объект элементов панели.
"dashboardConfig": {
// Связь активностей с объектом страницы.
"Activity": {
// Имя колонки объекта страницы.
"masterColumnName": "Id",
// Имя колонки в объекте [Activity].
"referenceColumnName": "Order"
}
```

```

    }
}
}
}
}
}
/**SCHEMA_MODULES*/,
details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
methods: {},
diff: /**SCHEMA_DIFF*/[
    {
        "operation": "insert",
        "name": "ActionsDashboardModule",
        "parentName": "ActionDashboardContainer",
        "propertyName": "items",
        "values": {
            "classes": { wrapClassName: ["actions-dashboard-module"] },
            "itemType": Terrasoft.ViewItemType.MODULE
        }
    }
]/**SCHEMA_DIFF*/
};
});
```

После сохранения схемы и обновления страницы приложения на странице заказа появится инструментальная панель действий, которая будет отображать состояние заказа, а также связанные с ним незавершенные активности (рис. 3).

Рис. 3. — Демонстрация результата выполнения примера



# Добавить новый канал на панель действий

 Сложный

## Описание примера

Добавить новый пользовательский канал в инструментальную панель действий страницы контакта. Канал должен полностью повторять функциональность канала фиксации результатов звонка (канал `CallMessagePublisher`).

## Исходный код

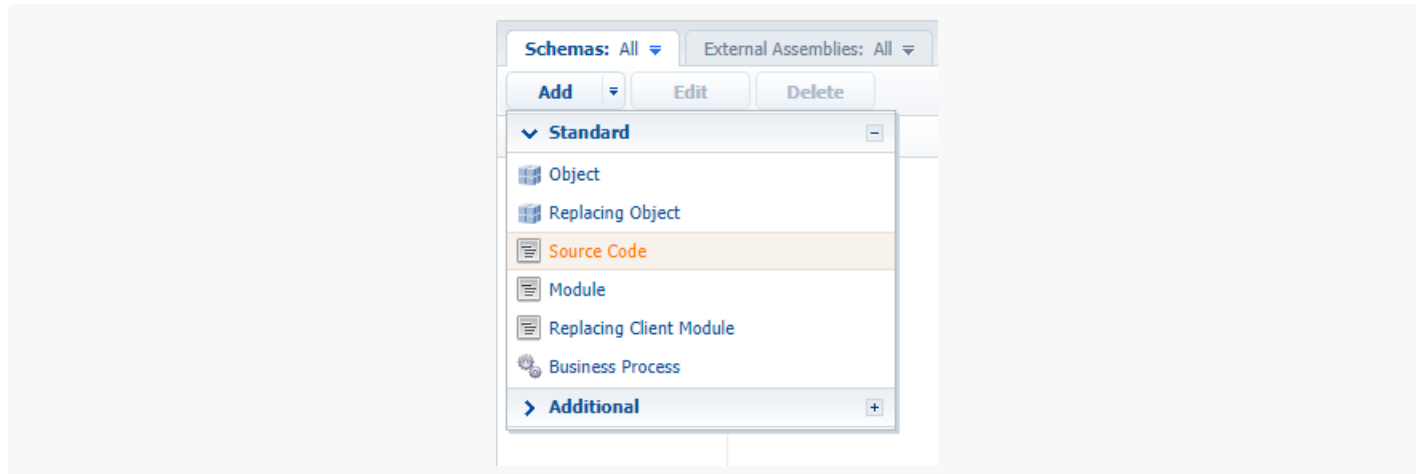
Пакет с реализацией примера можно скачать по [ссылке](#).

## Алгоритм выполнения примера

1. Добавить схему исходного кода `UsrCallsMessagePublisher`

Для создания схемы исходного кода в разделе [ Конфигурация ] на вкладке [ Схемы ] выполните пункт меню [ Добавить ] — [ Исходный код ] (рис. 1).

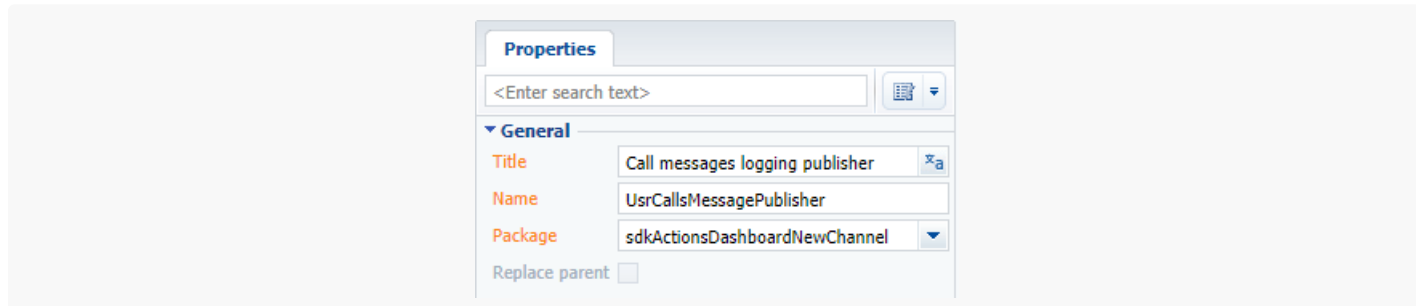
Рис. 1. — Добавление схемы исходного кода



Для созданной схемы укажите (рис. 2):

- [ Заголовок ] ([ Title ]) — "Издатель сообщений логирования звонка" (Call message logging publisher);
- [ Название ] ([ Name ]) — "UsrCallsMessagePublisher".

Рис. 2. — Свойства схемы исходного кода



В созданной схеме в пространстве имен `Terrasoft.Configuration` добавьте новый класс `CallsMessagePublisher`, наследуемый от класса `BaseMessagePublisher`. Класс `BaseMessagePublisher` содержит базовую логику сохранения объекта в базу данных и базовую логику обработчиков событий. Класс-наследник будет содержать логику для конкретного отправителя, например, заполнение колонок объекта `Activity` и последующую отправку сообщения.

Для реализации нового класса `CallsMessagePublisher` в созданную схему добавьте следующий исходный код:

```
using System.Collections.Generic;
using Terrasoft.Core;

namespace Terrasoft.Configuration
{
    // Класс-наследник BaseMessagePublisher.
```

```

public class CallsMessagePublisher : BaseMessagePublisher
{
    // Конструктор класса.
    public CallsMessagePublisher(UserConnection userConnection, Dictionary<string, string> entityFieldsData) : base(userConnection, entityFieldsData) {
        //Схема, с которой будет работать CallsMessagePublisher.
        EntitySchemaName = "Activity";
    }
}

```

После этого сохраните и опубликуйте схему.

## 2. Создать схему замещающей модели представления SectionActionsDashboard

Создайте схему замещающей модели представления, в которой в качестве родительского объекта укажите `SectionActionsDashboard` (рис. 3). Процесс создания схемы замещающей модели представления описан в статье ["Создать клиентскую схему"](#).

Рис. 3. — Свойства схемы замещающей модели представления

The image shows a 'Properties' window with a search bar at the top. Below it, the 'General' section contains fields for 'Title' (SectionActionsDashboard), 'Name' (SectionActionsDashboard), and 'Package' (sdkActionsDashboardNewChannel). The 'Inheritance' section is expanded, showing 'Parent object' set to 'SectionActionsDashboard ( ActionsDashboard )' and a checked 'Replace parent' checkbox. A red rectangle highlights the 'Parent object' field.

### На заметку.

Если нужно добавить канал только в одну страницу записи, то необходимо создать новый модуль с названием `имя_разделаSectionActionsDashboard` (например, `BooksSectionActionsDashboard`) и в качестве родительской схемы указать `SectionActionsDashboard` того раздела, в котором будет размещена страница записи.

В свойстве `diff` схемы замещающей модели представления установите операции вставки вкладки `CallsMessageTab` и контейнера сообщений, а также укажите модуль, который будет отрисовываться в данном канале на одной из вкладок. После этого новый канал будет виден на страницах записей тех разделов, в которых подключен `SectionActionsDashboard`.

В свойстве `methods` переопределите метод `getSectionPublishers()`, который добавит созданный канал в

список издателей сообщений, и метод `getExtendedConfig()`, в котором определяются параметры вкладки. Чтобы метод `getExtendedConfig()` отработал корректно, загрузите изображение иконки канала и укажите ее в параметре `ImageSrc`. Файл изображения иконки, используемый в данном примере, можно скачать [здесь](#).

Переопределите метод `onGetRecordInfoForPublisher()` и добавьте метод `getContactEntityParameterValue()`, определяющие значение контакта из страницы записи раздела, в котором находится панель действий.

Исходный код схемы замещающей модели представления:

```
define("SectionActionsDashboard", ["SectionActionsDashboardResources", "UsrCallsMessagePublisher"], function(resources) {
    return {
        attributes: {},
        messages: {},
        methods: {
            // Метод задает настройки отображения вкладки канала в панели действий.
            getExtendedConfig: function() {
                // Вызов родительского метода.
                var config = this.callParent(arguments);
                var lczImages = resources.localizableImages;
                config.CallsMessageTab = {
                    // Изображение вкладки.
                    "ImageSrc": this.Terrasoft.ImageUrlBuilder.getUrl(lczImages.CallsMessageTabImage),
                    // Значение маркера.
                    "MarkerValue": "calls-message-tab",
                    // Выравнивание.
                    "Align": this.Terrasoft.Align.RIGHT,
                    // Тэг.
                    "Tag": "UsrCalls"
                };
                return config;
            },
            // Переопределяет родительский и добавляет значение контакта из страницы записи
            // раздела, в котором находится панель действий.
            onGetRecordInfoForPublisher: function() {
                var info = this.callParent(arguments);
                info.additionalInfo.contact = this.getContactEntityParameterValue(info.relationSchemaName);
                return info;
            },
            // Определяет значение контакта из страницы записи раздела,
            // в котором находится панель действий.
            getContactEntityParameterValue: function(relationSchemaName) {
                var contact;
                if (relationSchemaName === "Contact") {
                    var id = this.getMasterEntityParameterValue("Id");
                    var name = this.getMasterEntityParameterValue("Name");
                    if (id && name) {
                        contact = {value: id, displayValue: name};
                    }
                }
                return contact;
            }
        }
    };
});
```

```

        }
    } else {
        contact = this.getMasterEntityParameterValue("Contact");
    }
    return contact;
},
//Добавляет созданный канал в список издателей сообщений.
getSectionPublishers: function() {
    var publishers = this.callParent(arguments);
    publishers.push("UsrCalls");
    return publishers;
}
},
// Массив модификаций, с помощью которых строится представление модуля в интерфейсе
diff: /**SCHEMA_DIFF*/[
    // Добавление вкладки CallsMessageTab.
    {
        // Тип операции – вставка.
        "operation": "insert",
        // Название вкладки.
        "name": "CallsMessageTab",
        // Название родительского элемента.
        "parentName": "Tabs",
        // Название свойства.
        "propertyName": "tabs",
        // Конфигурационный объект свойств.
        "values": {
            // Массив дочерних элементов.
            "items": []
        }
    },
    // Добавление контейнера сообщений.
    {
        "operation": "insert",
        "name": "CallsMessageTabContainer",
        "parentName": "CallsMessageTab",
        "propertyName": "items",
        "values": {
            // Тип элемента – контейнер.
            "itemType": this.Terrasoft.ViewItemType.CONTAINER,
            // CSS-класс для контейнера.
            "classes": {
                "wrapClassName": ["calls-message-content"]
            },
            "items": []
        }
    },
    // Добавление модуля UsrCallsMessageModule.

```

```

{
    "operation": "insert",
    "name": "UsrCallsMessageModule",
    "parentName": "CallsMessageTab",
    "propertyName": "items",
    "values": {
        // CSS-класс для модуля вкладок.
        "classes": {
            "wrapClassName": ["calls-message-module", "message-module"]
        },
        // Тип элемента – модуль.
        "itemType": this.Terrasoft.ViewItemType.MODULE,
        // Название модуля.
        "moduleName": "UsrCallsMessagePublisherModule",
        // Привязка метода, выполняемого после отрисовки элемента.
        "afterrender": {
            "bindTo": "onMessageModuleRendered"
        },
        // Привязка метода, выполняемого после перерисовки элемента.
        "afterrerender": {
            "bindTo": "onMessageModuleRendered"
        }
    }
}
]/**SCHEMA_DIFF*/
};
}
);

```

### 3. Создать модуль UsrCallsMessagePublisherModule

Модуль `UsrCallsMessagePublisherModule` служит контейнером для прорисовки в `SectionActionsDashboard` страницы `UsrCallsMessagePublisherPage`, в которой будет реализована логика добавляемого канала.

Для модуля установите следующие значения (рис. 4):

- [ *Заголовок* ] ([ *Title* ]) — "Модуль издателя сообщений логирования звонка" ("Call messages logging publisher module");
- [ *Название* ] ([ *Name* ]) — "UsrCallsMessagePublisherModule".

Рис. 4. — Свойства модуля



**Properties**

<Enter search text>

▼ **General**

Title: Call messages logging publisher module

Name: UsrCallsMessagePublisherModule

Package: sdkActionsDashboardNewChannel

▼ **Inheritance**

Parent object: [dropdown]

Forbid substitution: ☐

Replace parent: ☐

Исходный код модуля:

```
define("UsrCallsMessagePublisherModule", ["BaseMessagePublisherModule"],
function() {
    // Определение класса.
    Ext.define("Terrasoft.configuration.UsrCallsMessagePublisherModule", {
        // Базовый класс.
        extend: "Terrasoft.BaseMessagePublisherModule",
        // Сокращенное имя класса.
        alternateClassName: "Terrasoft.UsrCallsMessagePublisherModule",
        // Инициализация страницы, которая будет отрисовываться в данном модуле.
        initSchemaName: function() {
            this.schemaName = "UsrCallsMessagePublisherPage";
        }
    });
    // Возвращает объект класса, определенного в модуле.
    return Terrasoft.UsrCallsMessagePublisherModule;
});
```

## 4. Создать страницу UsrCallsMessagePublisherPage

Для создаваемой страницы установите в качестве родительского объекта схему `BaseMessagePublisherPage` пакета `MessagePublisher`. В качестве названия и заголовка укажите значение `"UsrCallsMessagePublisherPage"`.

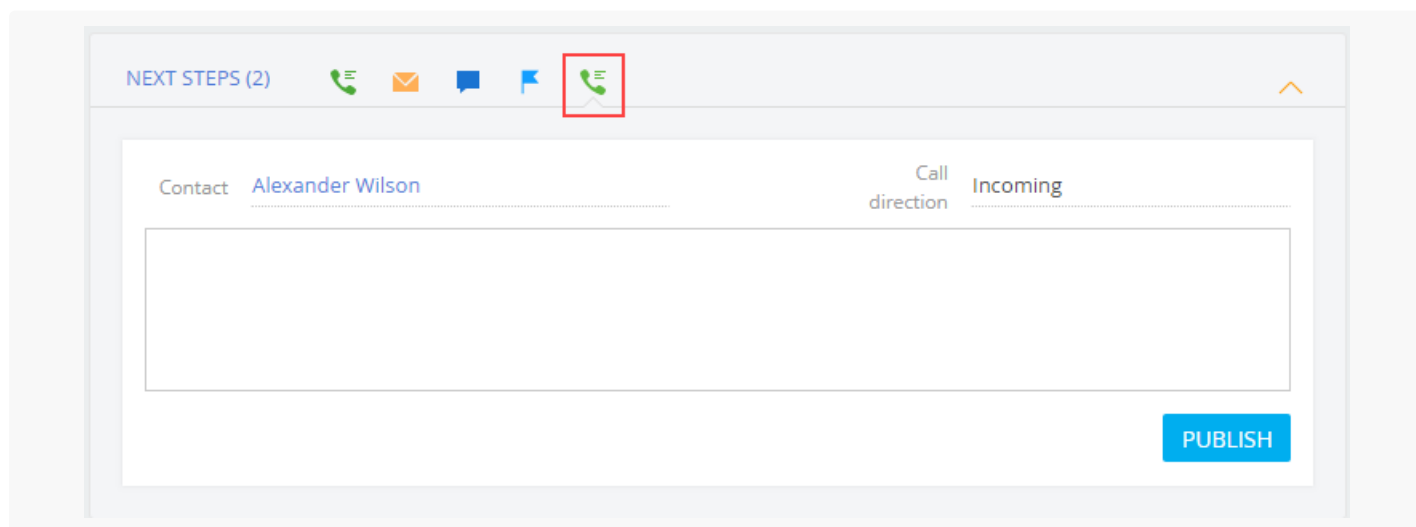
В исходном коде страницы укажите имя схемы объекта, с которым будет работать страница (в данном случае `Activity`), реализуйте логику публикации сообщения и переопределите метод `getServiceConfig`, в котором укажите имя класса из конфигурации.

```
// Задает класс, который будет работать с данной страницей.
```

```
getServiceConfig: function() {
    return {
        className: "Terrasoft.Configuration.CallsMessagePublisher"
    };
}
```

Реализация логики публикации сообщения содержит довольно большое количество методов, атрибутов и свойств. Полностью исходный код схемы `UsrCallsMessagePublisherPage` вы можете скачать по [ссылке](#). В исходном коде показана реализация рабочего канала `CallMessagePublisher`, который используется для логирования входящих и исходящих звонков. Результатом выполнения данного примера будет новый рабочий канал в `SectionActionsDashboard` (рис. 5).

Рис. 5. — Пример пользовательского канала `CallsMessagePublisherPage` в `SectionActionsDashboard` раздела [ *Контакты* ]



## Добавить мультиязычные шаблоны email-сообщений



Сложный

Creatio предоставляет возможность использовать собственную логику подбора шаблона email-сообщения по языку. На инструментальной панели действий (`ActionsDashboard`) записи раздела можно выбирать шаблоны email-сообщений на необходимом языке. Подбор выполняется на основании специализированных правил, которые могут быть определены в зависимости от раздела. Если специфические правила не определены, подбор ведется на основе контакта, связанного с редактируемой записью (колонок [ *Контакт* ]). Если колонки связи с контактом в объекте раздела нет, используется значение системной настройки `DefaultMessageLanguage`.

Для добавления собственной логики по подбору мультиязычных шаблонов:

1. Создайте класс или классы, унаследованные от `BaseLanguageRule` и определите правила подбора языка (один класс определяет одно правило).
2. Создайте класс, унаследованный от `BaseLanguageIterator`. В конструкторе класса определите свойство

`LanguageRules` как массив экземпляров классов, созданных на предыдущем шаге. Порядок следования соответствует приоритету правил.

3. Создайте класс-наследник от `AppEventListenerBase`, выполняющий привязку класса, определяющего правила подбора языка, к разделу.

4. В справочник [ *Шаблоны email сообщений* ] ([ *Email Templates* ]) добавьте нужные мультиязычные шаблоны.

## Описание примера

В пользовательский раздел добавить логику выбора языка email-сообщения на основе колонки `UsrContact` основного объекта раздела. Для примера использовать английский и испанский языки.

## Исходный код

Пакет с реализацией примера можно скачать по [ссылке](#).

### Важно.

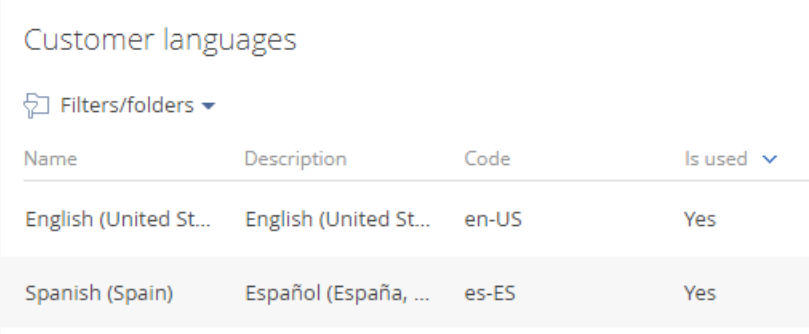
Пакет можно установить для продуктов Creatio, содержащих пакет `EmailTemplates`. После установки пакета убедитесь, что выполнены все предварительные настройки, описанные ниже.

## Предварительные настройки

Для корректной работы примера:

1. Убедитесь, что в справочнике [ *Языки общения* ] ([ *Customer languages* ]) используются английский и испанский языки (рис. 1).

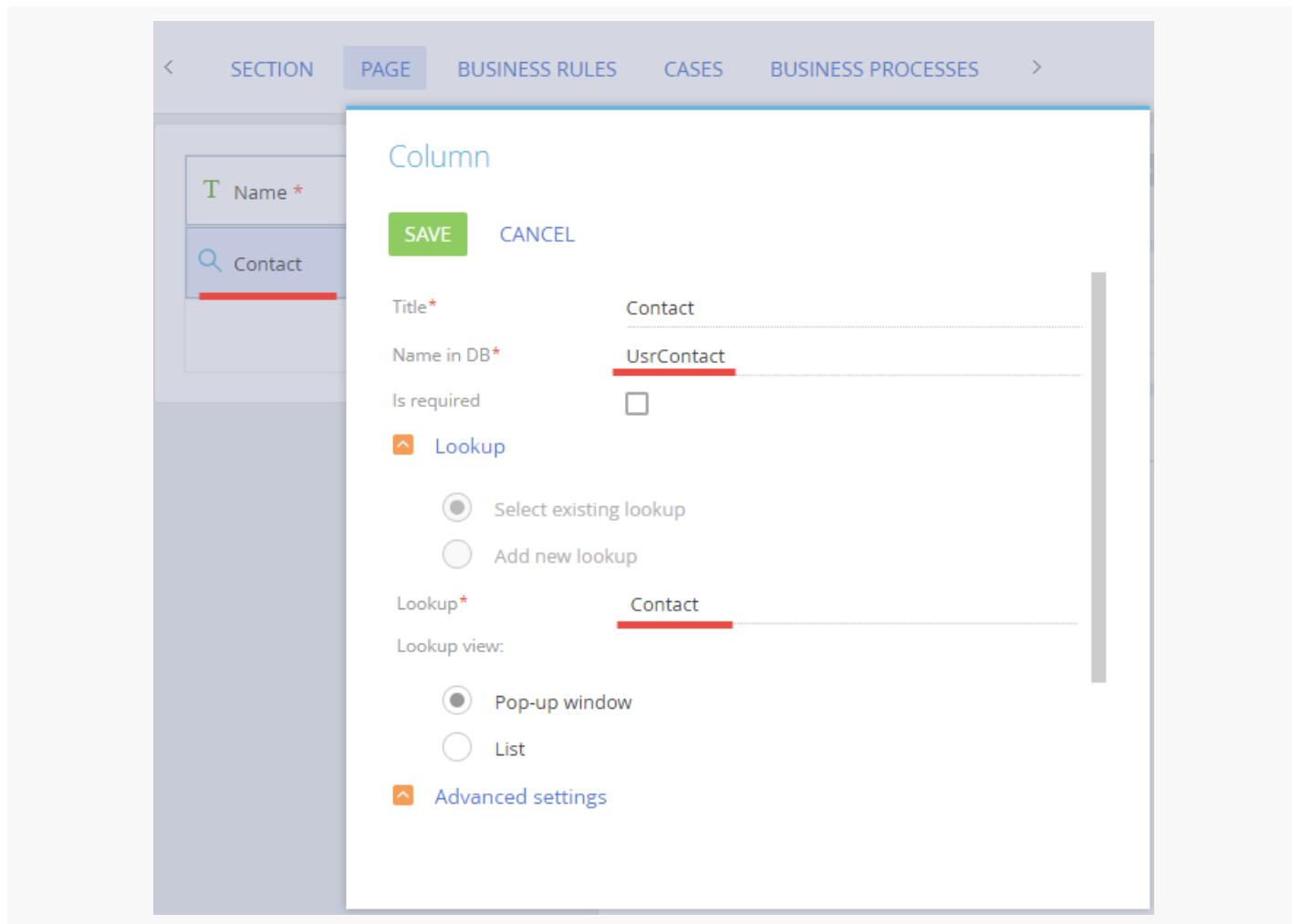
Рис. 1. — Справочник [ *Языки общения* ]



Filters/folders ▼			
Name	Description	Code	Is used ▼
English (United St...	English (United St...	en-US	Yes
Spanish (Spain)	Español (España, ...	es-ES	Yes

2. В мастере разделов проверьте, что на странице записи пользовательского раздела существует колонка `UsrContact`, связанная со справочником [ *Контакт* ] ([ *Contact* ]) (рис. 2).

Рис. 2. — Колонка `UsrContact`



## Алгоритм реализации примера

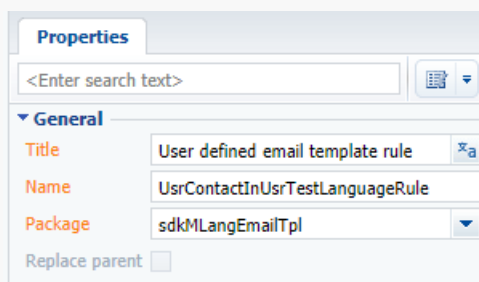
### 1. Добавить правило подбора языка

В пользовательском пакете создайте схему [ *Исходный код* ] (см. "[Создать схему \[ Исходный код \]](#)").

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrContactInUsrTestLanguageRule";
- [Заголовок] ([Title]) — "Пользовательское правило шаблона email-сообщений" ("User defined email template rule").

Рис. 3. — Свойства схемы [ *Исходный код* ]



Добавьте в схему следующий исходный код:

```
namespace Terrasoft.Configuration
{
    using System;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;
    public class ContactInUsrTestLanguageRule : BaseLanguageRule
    {
        public ContactInUsrTestLanguageRule (UserConnection userConnection) : base(userConnection)
        {
        }
        // Определяет идентификатор предпочитаемого языка пользователя.
        // recId – идентификатор текущей записи.
        public override Guid GetLanguageId(Guid recId)
        {
            // Создание экземпляра EntitySchemaQuery для основного объекта пользовательского раз
            var esq = new EntitySchemaQuery(UserConnection.EntitySchemaManager, "UsrMLangEmailTpl
            // Определение названия колонки языка контакта.
            var languageColumnName = esq.AddColumn("UsrContact.Language.Id").Name;
            // Получение экземпляра текущей записи.
            Entity usrRecEntity = esq.GetEntity(UserConnection, recId);
            // Получение значения идентификатора предпочитаемого языка пользователя.
            Guid languageId = usrRecEntity.GetTypedColumnValue<Guid>(languageColumnName);
            return languageId;
        }
    }
}
```

Опубликуйте схему.

## 2. Определить порядок правил подбора языка

В пользовательском пакете создайте схему [ Исходный код ] (см. ["Создать схему \[ Исходный код \]"](#)).

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrTestLanguageIterator";

- [Заголовок] ([Title]) — "Пользовательский итератор правил выбора языка" ("User defined language iterator").

Добавьте в схему исходный код, приведенный ниже:

```
namespace Terrasoft.Configuration
{
    using Terrasoft.Core;
    public class UsrTestLanguageIterator: BaseLanguageIterator
    {
        public UsrTestLanguageIterator(UserConnection userConnection): base(userConnection)
        {
            // Массив правил выбора языка.
            LanguageRules = new ILanguageRule[] {
                // Пользовательское правило.
                new ContactInUsrTestLanguageRule (UserConnection),
                // Правило по умолчанию.
                new DefaultLanguageRule(UserConnection),
            };
        }
    }
}
```

Вторым элементом массива является `DefaultLanguageRule`. Это правило использует для получения языка системную настройку `DefaultLanguage` и используется по умолчанию, если язык не был найден другими, более приоритетными правилами.

Выполните публикацию схемы.

### 3. Привязать итератор правил выбора языка к разделу

В пользовательском пакете создайте схему [ *Исходный код* ] (см. ["Создать схему \[ Исходный код \]"](#)).

Для созданной схемы укажите (рис. 3):

- [Название] ([Name]) — "UsrTestMLangBinder";
- [Заголовок] ([Title]) — "UsrTestMLangBinder".

Добавьте в схему следующий исходный код:

```
namespace Terrasoft.Configuration
{
    using Terrasoft.Core.Factories;
    using Terrasoft.Web.Common;
    public class UsrTestMLangBinder: AppEventListenerBase
    {
        public override void OnAppStart(AppEventContext context)
        {
```

```

        // Вызов базовой логики.
        base.OnAppStart(context);
        // Привязка итератора к пользовательскому разделу.
        // UsrMLangEmailTpl – название основного объекта раздела.
        ClassFactory.Bind<ILanguageIterator, UsrTestLanguageIterator>("UsrMLangEmailTpl");
    }
}
}

```

Опубликуйте схему.

## 4. Добавить необходимые мультиязычные шаблоны

В справочник [ *Шаблоны email сообщений* ] ([ *Email Templates* ]) добавьте новую запись (рис. 4), в которой определите шаблоны email-сообщений на требуемых языках (рис. 5).

Рис. 4. — Новая запись в справочнике [ *Шаблоны email сообщений* ]


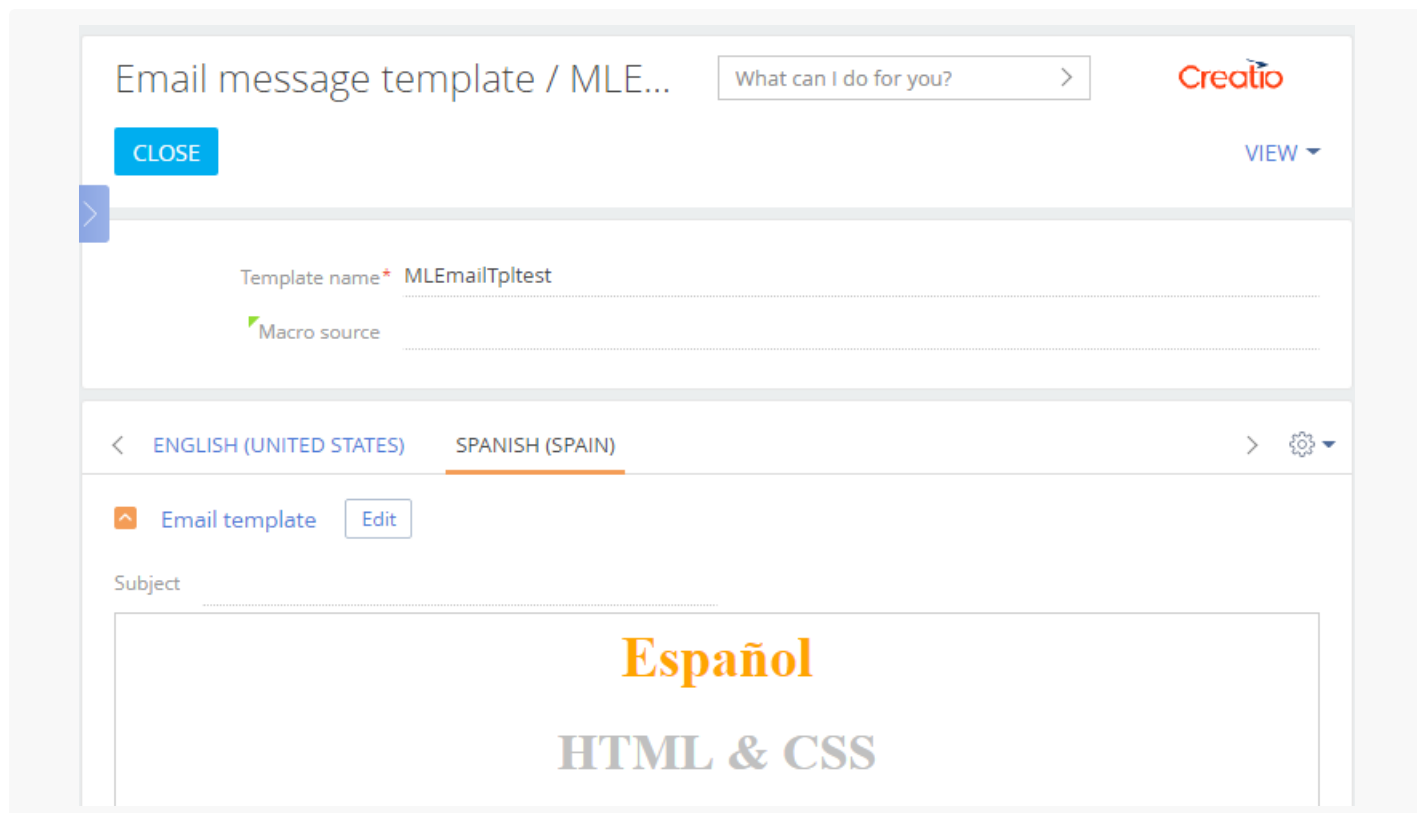
Email templates	
 Filters/folders ▼	
Case feedback request notification	Subject New message on case #[#Number#]
MLEmailTpltest (US, ES)	

Рис. 5. — Добавление шаблонов на требуемых языках



В результате выполнения примера на странице записи пользовательского раздела (рис. 6) в канале инструментальной панели действий (рис. 6, 1) шаблоны email сообщений (рис. 6, 2) будут выбираться автоматически на том языке контакта (рис. 6, 3), который установлен как предпочтительный (рис. 7).

Рис. 6. — Результат выполнения примера



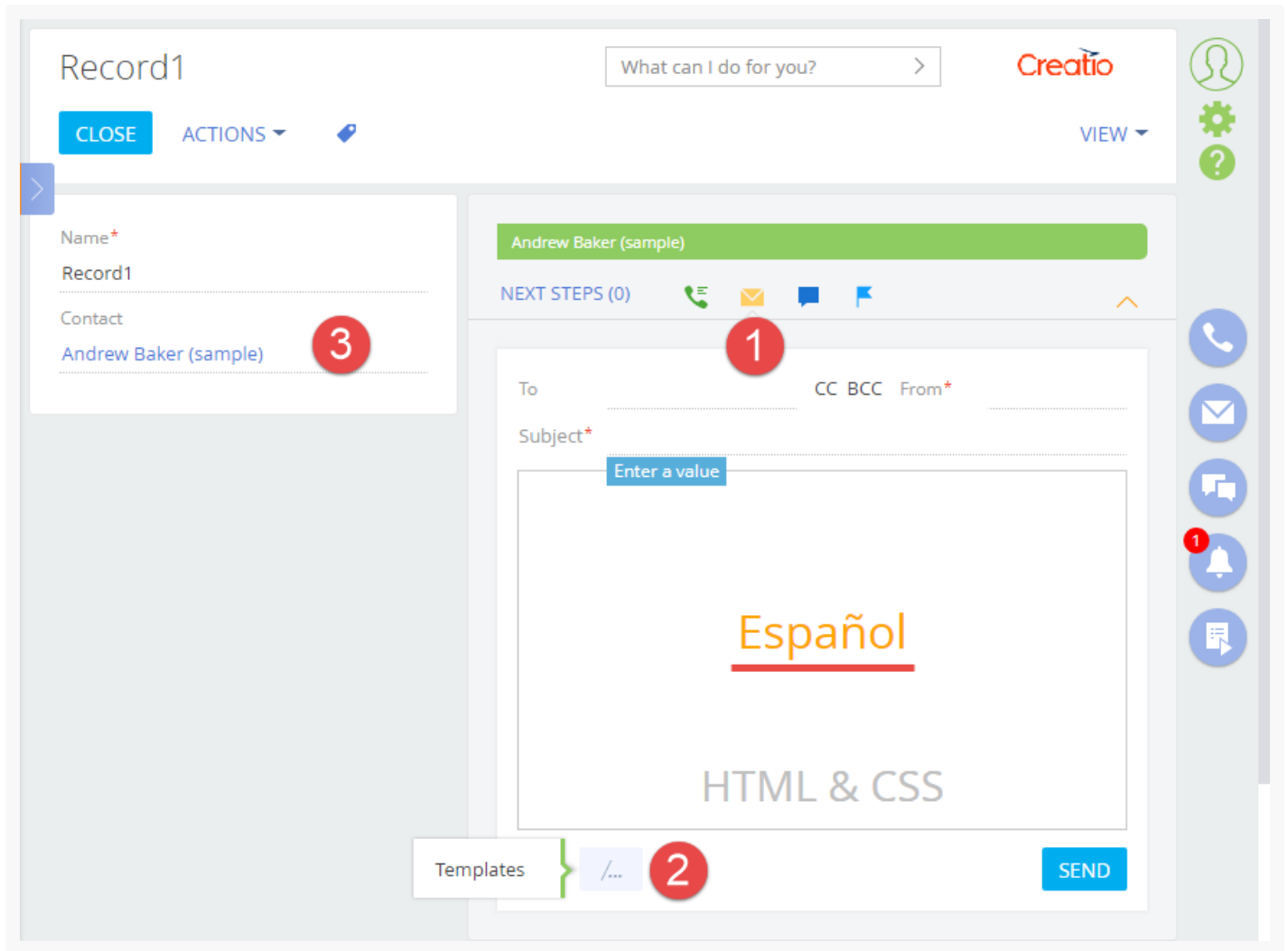
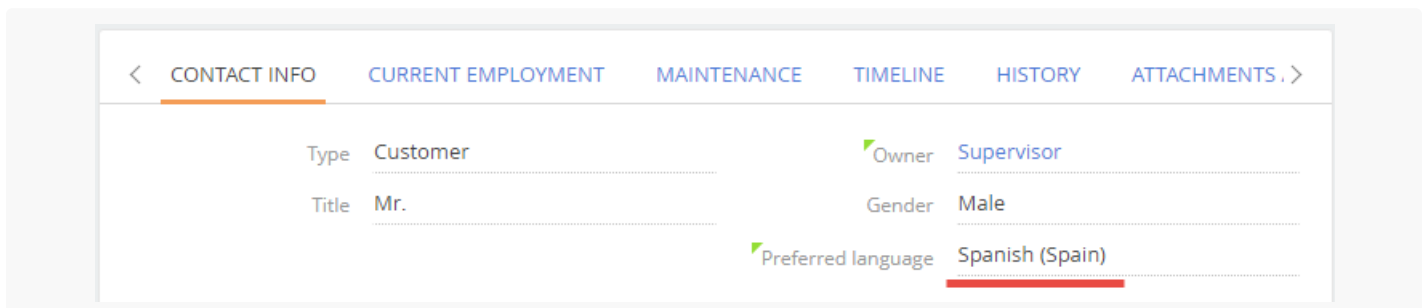


Рис. 7. — Предпочтительный язык контакта



## Создать пользовательскую страницу действия верификации

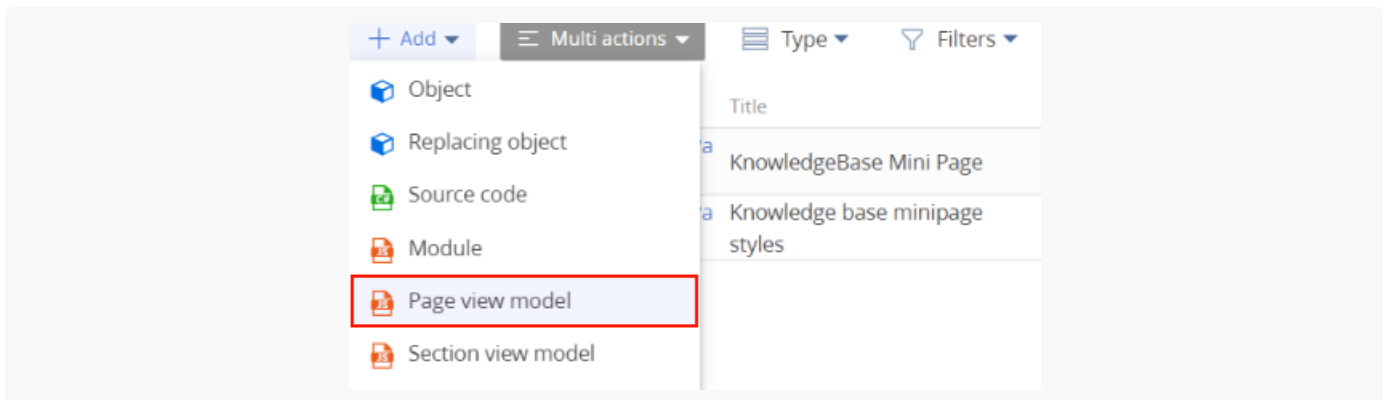
 Сложный

**Важно.** Пример может быть реализован только в продукте Financial Services Creatio.

**Пример.** Создать страницу действия верификации, на которой будет скрыто поле [ *Комментарий* ]. Страница действия верификации подробно описана в статье [Панель действий](#).

## 1. Создать схему страницы действия верификации

1. [Перейдите в раздел \[ Конфигурация \]](#) ([ *Configuration* ]) и выберите пользовательский [пакет](#), в который будет добавлена схема.
2. На панели инструментов реестра раздела нажмите [ *Добавить* ] —> [ *Модель представления страницы* ] ([ *Add* ] —> [ *Page view model* ]).



3. В дизайнере схем заполните свойства схемы:
  - [ *Код* ] ([ *Code* ]) — "UsrCommentlessAppValidationPage".
  - [ *Заголовок* ] ([ *Title* ]) — "Verification page without comments".
  - [ *Родительский объект* ] ([ *Parent object* ]) — выберите "Преднастроенная пользовательская страница" ("Preconfigured verification page").

**Module** [X]

Code \*  
UsrCommentlessAppValidationPage

Title \*  
Verification page without comments [X]

Parent object  
Preconfigured verification page [v]

Package  
sdkVerificationPage

Description [X]

CANCEL APPLY

Для применения заданных свойств нажмите [ *Применить* ] ([ *Apply* ]).

## 2. Настроить представление страницы

В дизайнере схем добавьте необходимый исходный код.

В массиве модификаций diff удалите из родительского элемента поле [ *Комментарий* ].

### UsrCommentlessAppValidationPage.js

```
define("UsrCommentlessAppValidationPage", [], function() {
  return {
    entitySchemaName: "AppValidation",
    diff: [{
      "operation": "remove",
      "name": "CommentContainer"
    }]
  };
});
```

## 2. Использовать созданную схему в бизнес-процессе

Чтобы использовать созданную схему, необходимо указать ее в поле [ *Выполнить на странице* ] ([ *Execute on page* ]) элемента [ *Действие верификации* ] ([ *Validation item* ]) бизнес-процесса. Эта схема может быть использована как в новых, так и в уже существующих бизнес-процессах, например, `Approve application`.

**Approve application**

SAVE RUN CANCEL ACTIONS

Validation item: Approve loan issuance

Which validation item to execute?

Approve loan issuance

Application\*  
[#Application#]

Execute on page\*  
**Verification page without comments**

How to perform validation?

Application approval

Who performs validation?

Group of employees

Not confirmed

Change application data

Add contract

Read contract id

Add contract parameters

Connect contract with application

Для применения изменений бизнес-процесс необходимо сохранить.

**Важно.** Чтобы изменения были окончательно применены, требуется перезапуск сайта приложения в IIS.

## Результат выполнения примера

После окончательного применения изменений прежняя страница верификации будет заменена пользовательской, не содержащей поля [ *Комментарий* ].

Approve loan issuance

Confirmed Not confirmed

COMPLETE