

# Разработка приложения Marketplace

Защита исходного кода приложения Marketplace от  
плагиата

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>Защита исходного кода приложения Marketplace от плагиата</b>	<b>4</b>
Защита C#-кода от плагиата	4
Защита JavaScript-кода от плагиата	12

# Защита исходного кода приложения Marketplace от плагиата



Открытый исходный код приложения Marketplace содержится в [пакете](#), который заблокирован для изменения. При этом код доступен к просмотру пользователям с соответствующим уровнем доступа и не защищен от плагиата. Реализация защиты от плагиата выполняется отдельно для back-end и для front-end кода.

## Защита C#-кода от плагиата

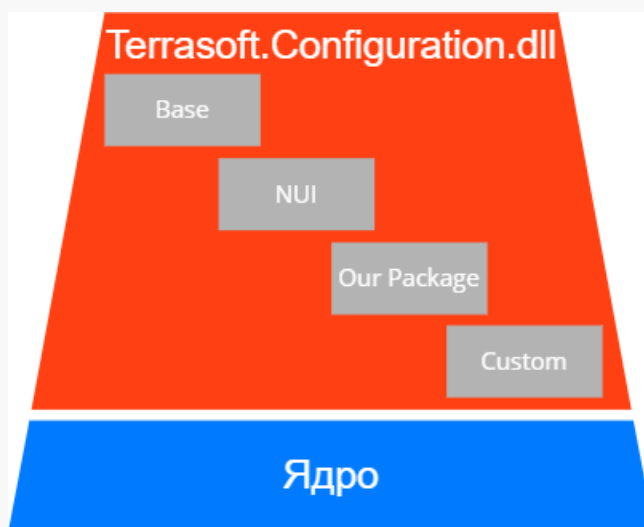
Использование [пакета-проекта](#) позволяет защитить C#-код приложения Marketplace от плагиата.

**Важно.** Настройку защиты исходного кода приложения Marketplace от плагиата разрешено выполнять только для исходного C#-кода собственной разработки.

**Способы** защиты C#-кода приложения Marketplace от плагиата:

- Для **нового приложения Marketplace** выполните разработку в пакете-проекте.
- Для **существующего приложения Marketplace** выполните конвертацию пакета в пакет-проект.

Разработка приложения Marketplace, как и других приложений, выполняется на уровне конфигурации, который содержит предустановленные пакеты приложения. В процессе публикации исходный C#-код пакетов компилируется в библиотеку `Terrasoft.Configuration.dll` и может взаимодействовать с ядром. Описание уровней кастомизации приложения Creatio содержится в статье [Разработка приложений на платформе Creatio](#). Детализированная схема уровней кастомизации приложения Creatio представлена на рисунке ниже.

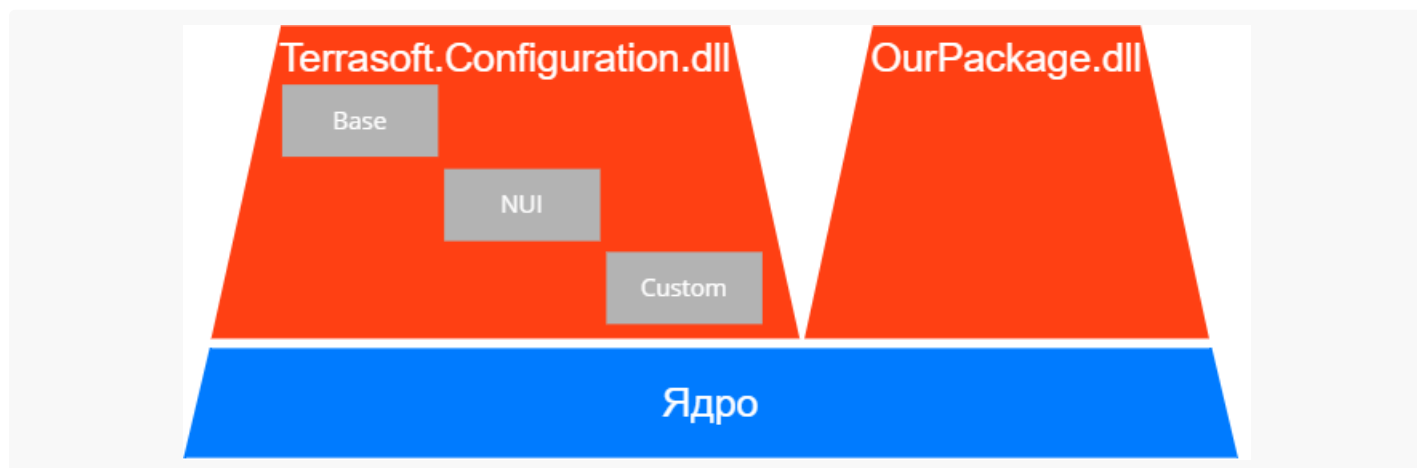


- `Base`, `NUI` — базовые пакеты приложения Creatio.
- `OurPackage` — пакет-проект с приложением Marketplace.
- `Custom` — специальный пакет приложения Creatio.

**Возможности**, которые предоставляет использование пакета-проекта с приложением Marketplace:

- Исключение C#-кода пользовательского приложения Marketplace из библиотеки `Terrasoft.Configuration.dll`.
- Выполнение установки приложения Marketplace, как отдельной \*.dll-библиотеки.

Схема уровней кастомизации приложения Creatio при наличии пакета-проекта, который содержит приложение Marketplace, представлена на рисунке ниже.



## Разработать приложение Marketplace в пакете-проекте

Разработку нового приложения Marketplace рекомендуется выполнять в пакете-проекте.

Чтобы **разработать приложение Marketplace в пакете-проекте**:

1. Настройте Creatio для работы в файловой системе.
2. Создайте пользовательский пакет.
3. Разработайте пользовательскую функциональность.
4. Выполните сборку пакета-проекта.

### 1. Настроить Creatio для работы в файловой системе

Для настройки Creatio для работы в файловой системе воспользуйтесь инструкцией, которая приведена в статье [Внешние IDE](#).

### 2. Создать пользовательский пакет

**Инструменты**, который позволяют создать пользовательский пакет:

- Creatio IDE. Создание пользовательского пакета с использованием Creatio IDE описано в статье [Создать пользовательский пакет](#).

- Утилита [Creatio command-line interface utility \(clio\)](#).

Чтобы **создать пользовательский пакет с использованием утилиты clio**:

1. Выполните установку clio (при необходимости).

#### Команда для установки clio

```
dotnet tool install clio -g
```

Установка clio подробно описана в официальной [документации утилиты на GitHub](#).

2. Перейдите в каталог `Pkg` приложения.

#### Команда для перехода в каталог `Pkg`

```
cd C:\inetpub\wwwroot\creatio\Terrasoft.WebApp\Terrasoft.Configuration\Pkg;
```

3. Создайте новый пакет.

#### Команда для создания нового пакета

```
clio init OurPackage;
```

4. Установите зависимости пакета. Для этого отредактируйте файл `descriptor.json`.

Ниже приведен пример установки зависимостей (свойство `DependsOn`) пакета `OurPackage` от пакета `ProductCore` и добавления описания (свойство `Descriptor`) пакета.

#### Пример установки зависимостей и добавления описания пакета

```
{
  "Descriptor": {
    "Uid": "45cc06b2-6448-4d9e-9f51-bee31a6dbc25",
    "PackageVersion": "7.8.0",
    "Name": "OurPackage",
    "ModifiedOnUtc": "/Date(1633420586000)/",
    "Maintainer": "Customer",
    "Description": "Payment calculator",
    "DependsOn": [
      {
        "Uid": "2fabaf6c-0f92-4530-aef8-40345c021da2",
        "PackageVersion": "7.8.0",
        "Name": "ProductCore"
      }
    ]
  }
}
```

```
}
}
```

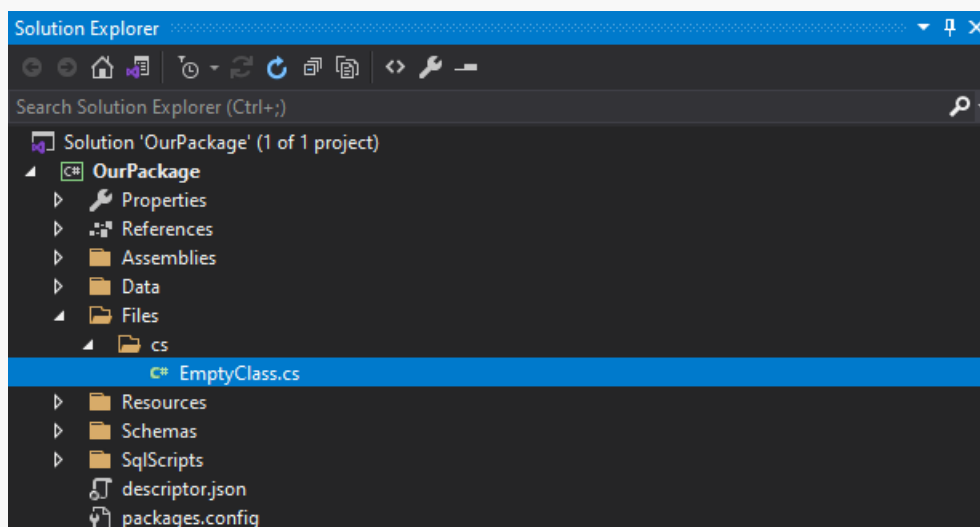
В результате создан пользовательский пакет `OurPackage`, который зависит от пакета `ProductCore`.

### 3. Разработать пользовательскую функциональность

Для разработки пользовательской функциональности можно использовать любую [внешнюю IDE](#). В нашем примере разработка выполняется в Microsoft Visual Studio Code.

Чтобы **разработать пользовательскую функциональность**:

1. Откройте проект `OurPackage.sln`.



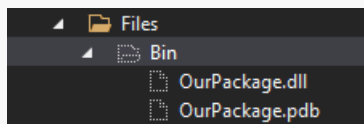
2. Установите NuGet-пакет CreatioSDK из репозитория, который доступен на официальном [сайте nuget](#). Выберите версию CreatioSDK, которая подходит для ваших целей.

3. Реализуйте пользовательскую функциональность в каталоге `Files\cs` приложения.

При разработке C#-кода можно создать приложение с помощью IDE. Для этого в Visual Studio нажмите комбинацию `Ctrl+Shift+B`.

4. Выполните сборку приложения.

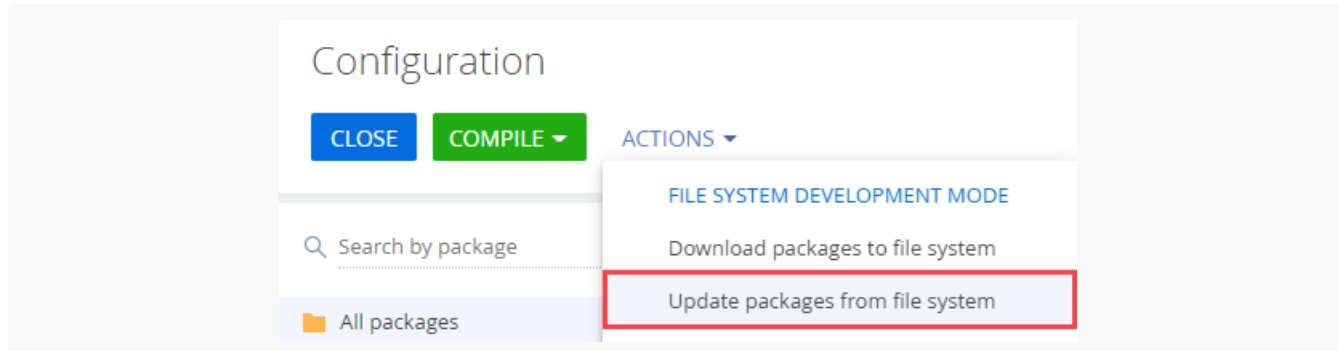
Если сборка приложения выполнится успешно, то \*.dll, \*.pdb и другие вспомогательные файлы помещаются в каталог `Files\Bin` приложения.



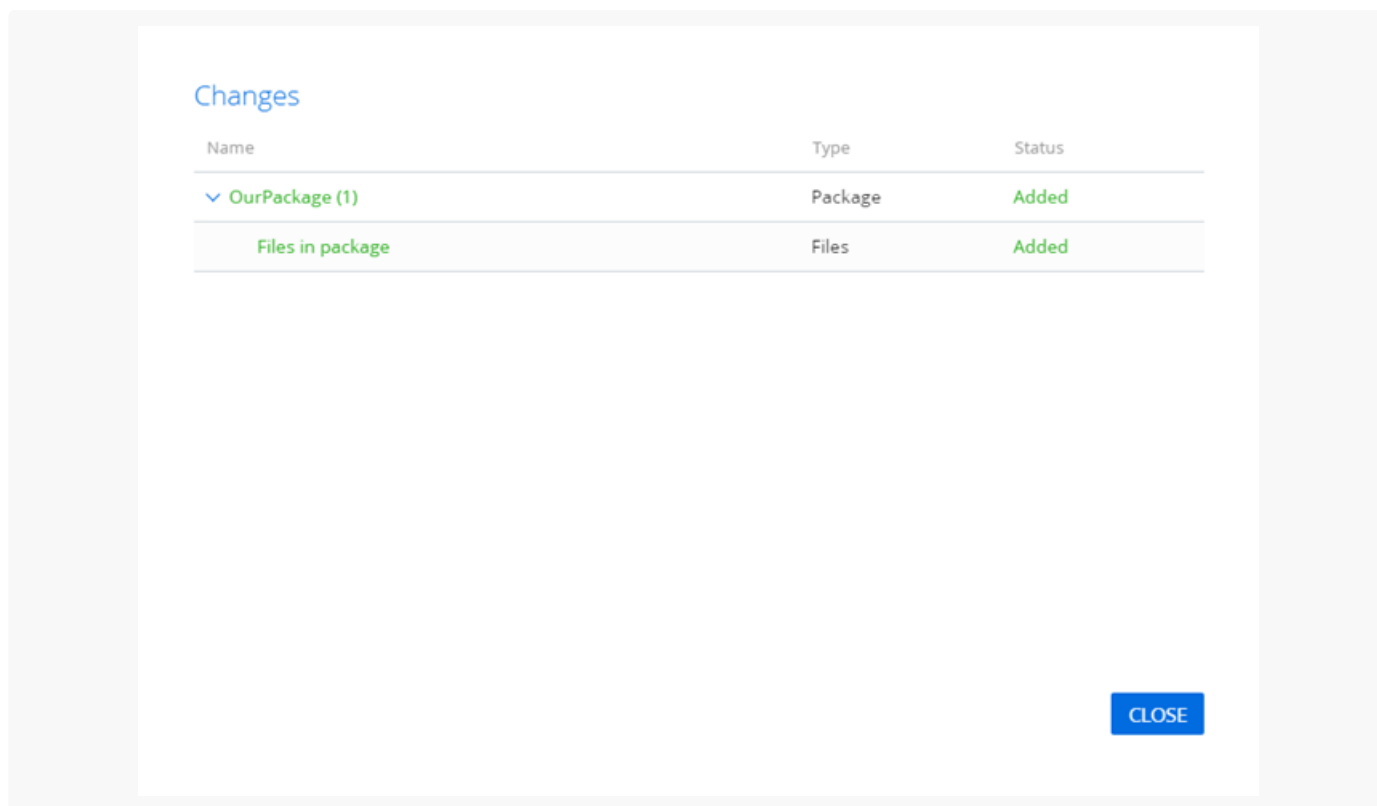
5. Загрузите пакет `OurPackage` из каталога `[Путь к приложению]\Terrasoft.WebApp\Terrasoft.Configuration\Pkg` в базу данных.

- а. Перейдите в дизайнер системы по кнопке .

- b. В блоке [ *Конфигурирование разработчиком* ] ([ *Admin area* ]) перейдите по ссылке [ *Управление конфигурацией* ] ([ *Advanced settings* ]).
- c. В группе [ *Разработка в файловой системе* ] ([ *File system development mode* ]) выпадающего списка [ *Действия* ] ([ *Actions* ]) панели инструментов нажмите [ *Обновить пакеты из файловой системы* ] ([ *Update packages from file system* ]).



В результате пакет `OurPackage` загружен в Creatio IDE.



6. Перезапустите приложение.

#### Команда для перезапуска приложения

```
clio restart
```



## 4. Выполните сборку пакета-проекта

**Назначение** выполнения сборки пакета-проекта — подготовка приложения Marketplace к публикации на онлайн-площадке Creatio Marketplace.

**Важно.** Если вы не хотите, чтобы исходный C#-код собственной разработки был включен в пакет-проект, то обязательно удалите его перед выполнением экспорта пакета.

Чтобы **выполнить сборку пакета-проекта**:

1. Удалите исходный C#-код собственной разработки из пакета-проекта (при необходимости).  
Выполнить, если вы не хотите, чтобы исходный C#-код был включен в пакет-проект.
2. Для автоматизации процесса выполнения сборки пакета-проекта создайте файл `PackagePublish.target`.
3. В файл `PackagePublish.target` добавьте код.

### Файл `PackagePublish.target`

```
<?xml version="1.0" encoding="utf-8" ?>
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <DestinationFolder>C:\PkgRelease\$(AssemblyName)</DestinationFolder>
  </PropertyGroup>

  <ItemGroup>
    <PkgAssemblies Include="Assemblies\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgData Include="Data\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgFiles Include="Files\Bin\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgProperties Include="Properties\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgResources Include="Resources\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgSchemas Include="Schemas\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgSqlScripts Include="SqlScripts\**"/>
  </ItemGroup>
  <ItemGroup>
    <PkgDescriptor Include="descriptor.json"/>
  </ItemGroup>
</Project>
```

```
</ItemGroup>

<Target Name="CopyFiles">
  <Copy
    SourceFiles="@(\PkgAssemblies)"
    DestinationFiles="@(\PkgAssemblies->'$(DestinationFolder)\Assemblies\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgData)"
    DestinationFiles="@(\PkgData->'$(DestinationFolder)\Data\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgFiles)"
    DestinationFiles="@(\PkgFiles->'$(DestinationFolder)\Files\Bin\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgProperties)"
    DestinationFiles="@(\PkgProperties->'$(DestinationFolder)\Properties\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgResources)"
    DestinationFiles="@(\PkgResources->'$(DestinationFolder)\Resources\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgSchemas)"
    DestinationFiles="@(\PkgSchemas->'$(DestinationFolder)\Schemas\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgSqlScripts)"
    DestinationFiles="@(\PkgSqlScripts->'$(DestinationFolder)\SqlScripts\%(RecursiveDir)%(Filename)%(Extension)')
  />
  <Copy
    SourceFiles="@(\PkgDescriptor)"
    DestinationFiles="@(\PkgDescriptor->'$(DestinationFolder)\%(RecursiveDir)%(Filename)%(Extension)')
  />
</Target>

<Target Name="CreateRelease" AfterTargets="CopyFiles">
  <Exec Command="clio generate-pkg-zip $(DestinationFolder) -d C:\PkgRelease\$(AssemblyName)$(AssemblyVersion)$(AssemblyConfiguration)$(AssemblyPlatform).zip" />
</Target>

</Project>
```

4. В файл `OurPackage.csproj` добавьте строку.

**Файл** OurPackage.csproj

```
<Import Project="PackagePublish.target" />
```

5. Откройте командную строку и выполните команду.

```
msbuild /t:CreateRelease
```

В результате пакет-проект будет выгружен в каталог `C:\PkgRelease`, который содержит вложенный каталог `OurPackage` и \*.gz-архив `OurPackage.gz`. Этот архив содержит подготовленное к публикации на онлайн-площадке Creatio Marketplace приложение Marketplace.

## Конвертировать пакет с приложением Marketplace в пакет-проект

Подготовка ранее разработанного приложения Marketplace к конвертации в пакет-проект может потребовать значительных модификаций.

Конвертация разработанного приложения Marketplace в пакет-проект выполняется с помощью команды `clio convert`, которая описана в официальной документации утилиты [на GitHub](#).

Невозможно корректно выполнить конвертацию некоторых файлов и схем, например, элемента [ *Действие процесса* ] ([ *User task* ]) бизнес-процесса. Независимо от значения признака [ *Partial* ], элемент [ *Действие процесса* ] ([ *User task* ]) является частичным классом и должен находиться в библиотеке `Terrasoft.Configuration.dll`. При выполнении конвертации код элемента [ *Действие процесса* ] ([ *User task* ]) по умолчанию сохраняется в каталог `AutoGenerated` пакета-проекта, а не в библиотеку `Terrasoft.Configuration.dll`.

Чтобы **конвертировать разработанное приложение Marketplace в пакет-проект**, выполните одну из команд:

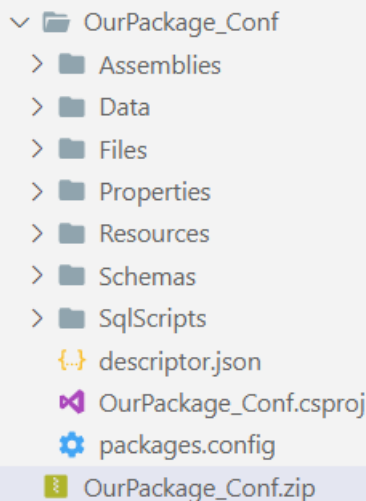
- `clio convert .\OurPackage_Conf\ -c false .`
- `clio convert .\OurPackage_Conf\ .`

Результат выполнения команд идентичен, поскольку для ключа `-c` (ConvertSourceCode) значение `false` установлено по умолчанию.

**Содержимое** C#-проекта после выполнения конвертации:

- Пакет-проект с конвертированным приложением Marketplace. Структура пакета-проекта описана в статье [Пакет-проект](#).
- \*.zip-архив с неконвертированным приложением Marketplace.

Структура C#-проекта после выполнения конвертации представлена на рисунке ниже.



В результате разработанное приложение Marketplace конвертировано в пакет-проект, который можно устанавливать в приложение Creatio.

**На заметку.** Формирование пакета-проекта является обязательным звеном CI/CD-конвейера. В репозитории рекомендуется хранить незащищенный от плагиата исходный C#-код приложения Marketplace.

## Защита JavaScript-кода от плагиата

**Способы** защиты JavaScript-кода приложения Marketplace от плагиата:

- Минификация.
- Обфускация.

**Важно.** Запрещено изменять структуру схемы клиентского модуля, поскольку дизайнеры приложения настроены на работу с определенной структурой схемы.

Лучшей практикой защиты JavaScript-кода от плагиата является использование миксинов для реализации логики, которую необходимо защитить. Не рекомендуется обфусцировать схемы клиентских модулей, которые используются мастерами (разделов, страниц, деталей) приложения.

Существует большое количество решений с открытым исходным кодом для обфускации JavaScript-кода. В нашем примере мы используем обфускатор JavaScript Obfuscator. Официальная документация обфускатора доступна [на GitHub](#).

Чтобы **защитить JavaScript-код от плагиата с использованием JavaScript Obfuscator**:

1. Выполните установку JavaScript Obfuscator.

**Команда для установки JavaScript Obfuscator**

```
npm install javascript-obfuscator -g
```

Установка JavaScript Obfuscator подробно описана в официальной [документации обфускатора на GitHub](#).

## 2. Подготовьте JavaScript-код к обфускации.

- a. Создайте миксин. В нашем примере это `MRKT_DemoMixin`. Миксины подробно описаны в статье [Клиентская схема](#).
- b. В миксине реализуйте JavaScript-код, который планируется обфусцировать.

**MRKT\_DemoMixin**

```
define("MRKT_DemoMixin", [], function() {
    Ext.define("Terrasoft.configuration.mixins.MRKT_DemoMixin", {
        alternateClassName: "Terrasoft.MRKT_DemoMixin",

        secretMethod: function(){
            console.log("MRKT_DemoMixin");
        },
    });
});
```

## 3. Укажите МИКСИН `MRKT_DemoMixin` в СВОЙСТВЕ `mixins` клиентской схемы (например, `ContactPageV2`).

**ContactPageV2**

```
define("ContactPageV2", ["MRKT_DemoMixin"],
function() {
    return {
        entitySchemaName: "Contact",
        mixins: {
            "MRKT_DemoMixin": "Terrasoft.MRKT_DemoMixin"
        },
        attributes: {},
        modules: /**SCHEMA_MODULES*/{}/**SCHEMA_MODULES*/,
        details: /**SCHEMA_DETAILS*/{}/**SCHEMA_DETAILS*/,
        businessRules: /**SCHEMA_BUSINESS_RULES*/{}/**SCHEMA_BUSINESS_RULES*/,
        methods: {
            onEntityInitialized: function() {
                this.callParent(arguments);

                /* Consume MRKT_DemoMixin. */
                this.secretMethod();
            },
        },
    };
});
```

```

    },
    dataModels: /**SCHEMA_DATA_MODELS*/{}/**SCHEMA_DATA_MODELS*/,
    diff: /**SCHEMA_DIFF*/[]/**SCHEMA_DIFF*/
  };
});

```

4. Скопируйте JavaScript-код, который планируется обфусцировать. Возможно, будет необходимость редактировать код в будущем.
5. Выполните обфускацию JavaScript-кода.

### Команда для обфускации JavaScript-кода

```
javascript-obfuscator MRKT_DemoMixin.js --output MRKT_DemoMixin.obfuscated.js
```

Выполнение обфускации в JavaScript Obfuscator подробно описана в официальной [документации обфускатора на GitHub](#).

В результате получим обфусцированный файл. Пример обфусцированного файла приведен ниже.

### Пример обфусцированного файла

```

function a0_0x2c69() {
  var _0x272852 = ['Terrasoft.MRKT_DemoMixin', '0k\x20-\x20MRKT_DemoMixin', '636527SjITzq', '4
  a0_0x2c69 = function() {
    return _0x272852;
  };
  return a0_0x2c69();
}
var a0_0x31e3ab = a0_0x3f9a;

function a0_0x3f9a(_0x104961, _0x4f9883) {
  var _0x2c694d = a0_0x2c69();
  return a0_0x3f9a = function(_0x3f9a09, _0x1d247d) {
    _0x3f9a09 = _0x3f9a09 - 0xa7;
    var _0x481962 = _0x2c694d[_0x3f9a09];
    return _0x481962;
  }, a0_0x3f9a(_0x104961, _0x4f9883);
}(function(_0x479332, _0x464f89) {
  var _0x279ddd = a0_0x3f9a,
    _0x3c692e = _0x479332();
  while (!![]) {
    try {
      var _0xae3ae0 = -parseInt(_0x279ddd(0xac)) / 0x1 + parseInt(_0x279ddd(0xa7)) / 0x2 +
      if (_0xae3ae0 === _0x464f89) break;
      else _0x3c692e['push'](_0x3c692e['shift']());
    } catch (_0x44cb38) {

```

```

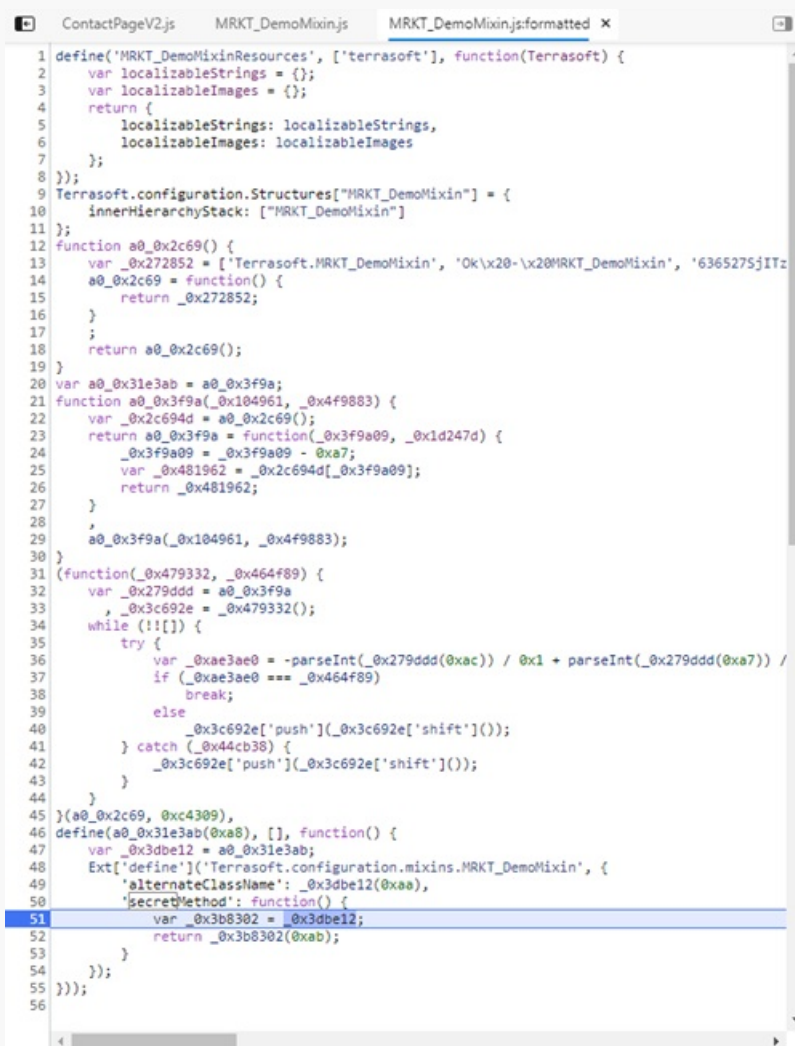
        _0x3c692e['push'](_0x3c692e['shift']());
    }
}
}(a0_0x2c69, 0xc4309), define(a0_0x31e3ab(0xa8), [], function() {
    var _0x3dbe12 = a0_0x31e3ab;
    Ext['define']('Terrasoft.configuration.mixins.MRKT_DemoMixin', {
        'alternateClassName': _0x3dbe12(0xaa),
        'secretMethod': function() {
            var _0x3b8302 = _0x3dbe12;
            return _0x3b8302(0xab);
        }
    });
}));

```

Чтобы **посмотреть отображение JavaScript-кода на странице браузера**:

1. Очистите кэш браузера.
2. Обновите страницу.
3. Запустите [инструменты отладки](#).

Пример отображения JavaScript-кода на странице браузера приведен на рисунке ниже.



```

1  define('MRKT_DemoMixinsResources', ['terrasoft'], function(Terrasoft) {
2    var localizableStrings = {};
3    var localizableImages = {};
4    return {
5      localizableStrings: localizableStrings,
6      localizableImages: localizableImages
7    };
8  });
9  Terrasoft.configuration.Structures["MRKT_DemoMixins"] = {
10   innerHierarchyStack: ["MRKT_DemoMixins"]
11 };
12 function a0_0x2c69() {
13   var _0x272852 = ['Terrasoft.MRKT_DemoMixins', '0k\\x20\\x20MRKT_DemoMixins', '6365275jITz
14   a0_0x2c69 = function() {
15     return _0x272852;
16   }
17   ;
18   return a0_0x2c69();
19 }
20 var a0_0x31e3ab = a0_0x3f9a;
21 function a0_0x3f9a(_0x104961, _0x4f9883) {
22   var _0x2c694d = a0_0x2c69();
23   return a0_0x3f9a = function(_0x3f9a09, _0x1d247d) {
24     _0x3f9a09 = _0x3f9a09 - 0xa7;
25     var _0x481962 = _0x2c694d[_0x3f9a09];
26     return _0x481962;
27   }
28   ,
29   a0_0x3f9a(_0x104961, _0x4f9883);
30 }
31 (function(_0x479332, _0x464f89) {
32   var _0x279ddd = a0_0x3f9a
33   , _0x3c692e = _0x479332();
34   while (!![]) {
35     try {
36       var _0xae3ae0 = -parseInt(_0x279ddd(0xac)) / 0x1 + parseInt(_0x279ddd(0xa7)) /
37       if (_0xae3ae0 === _0x464f89)
38         break;
39       else
40         _0x3c692e['push'](_0x3c692e['shift']());
41     } catch (_0x44cb38) {
42       _0x3c692e['push'](_0x3c692e['shift']());
43     }
44   }
45 }(a0_0x2c69, 0xc4309),
46 define(a0_0x31e3ab(0xa8), [], function() {
47   var _0x3dbe12 = a0_0x31e3ab;
48   Ext['define']('Terrasoft.configuration.mixins.MRKT_DemoMixins', {
49     'alternateClassName': _0x3dbe12(0xaa),
50     'secretMethod': function() {
51       var _0x3b8302 = _0x3dbe12;
52       return _0x3b8302(0xab);
53     }
54   });
55 });
56

```

**На заметку.** Обфусцирование JavaScript-кода является обязательным звеном CI/CD-конвейера. В репозитории рекомендуется хранить незащищенный от плагиата исходный JavaScript-код приложения Marketplace.