Creatio Academy

Delivery

Customize delivery process

Version 8.0

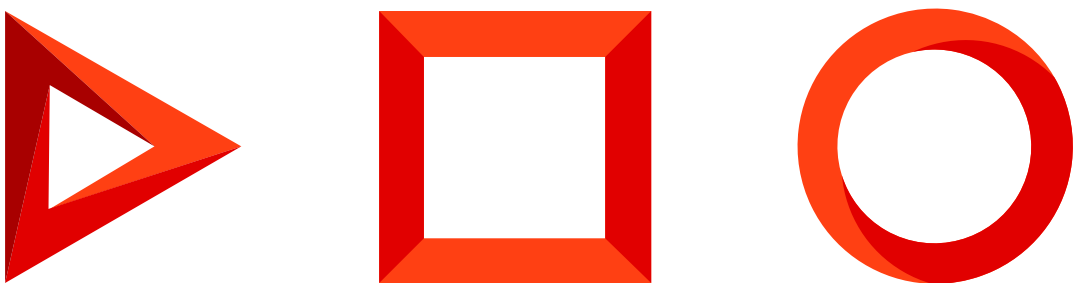


Table of Contents

Customize delivery process

Medium

You can customize delivery process in Creatio version 8.0.5 Atlas and later.

Creatio version 8.0.4 Atlas and earlier lets you execute CRUD operations with data as part of package installation using SQL scripts. However, if you execute CRUD operations with data using SQL scripts, you will not be able to completely roll back the configuration, including any data changes. This is because SQL scripts make irreversible changes to the database as part of the package installation process. To prevent that, use the install step functionality instead. The install step functionality is designed to execute CRUD operations with data as part of package installation in a way that does not cause irreparable changes to the database.

Creatio version 8.0.3 Atlas and later lets you use backward compatible SQL scripts to completely back the configuration. Learn more in a separate article: Backward compatible SQL scripts.

Use the install step functionality to execute the following **actions** as part of package installation:

- execute the previous setup for the Creatio instance
- populate the values of new columns
- move data to a new object structure
- generate app demo data

Attention. Creatio lets you set up the install steps only for assembly packages. To set up the install steps for a simple package or project package, convert the package to an assembly package. Learn more about assembly packages and package conversion in separate articles: Assembly package, Package conversion.

Steps to customize delivery process

1. Implement an install script
2. Add the install script to the installation steps

1. Implement an install script

A **package install script** is a [*Source code*] package schema. The install script implements the `IInstallScriptExecutor` interface. Use the `Entity` class to implement CRUD operations with data. Do not use C# classes listed in a separate article: C# classes to avoid in install scripts. The **purpose** of install scripts is to execute C# code as part of package installation. Install script executes database queries using the ORM model. Use install scripts to ensure you can roll the configuration back completely. An install script corresponds to one install step.

To **implement an install script**:

1. Create a [*Source code*] schema. Learn more about creating a schema in a separate article: Source code

(C#).

2. Create a **class** that implements the install script.

   - Add the `Terrasoft.Core` namespace in the Schema Designer.
   - Implement the `Terrasoft.Core.IInstallScriptExecutor` interface.

   We recommend creating an object using the `EntitySchemaManager.GetEntityByName(string name, UserConnection userConnection)` method that lets you receive an object with new fields that are being delivered for the first time.

3. Click [*Publish*] on the Source Code Designer's toolbar to compile the schema.

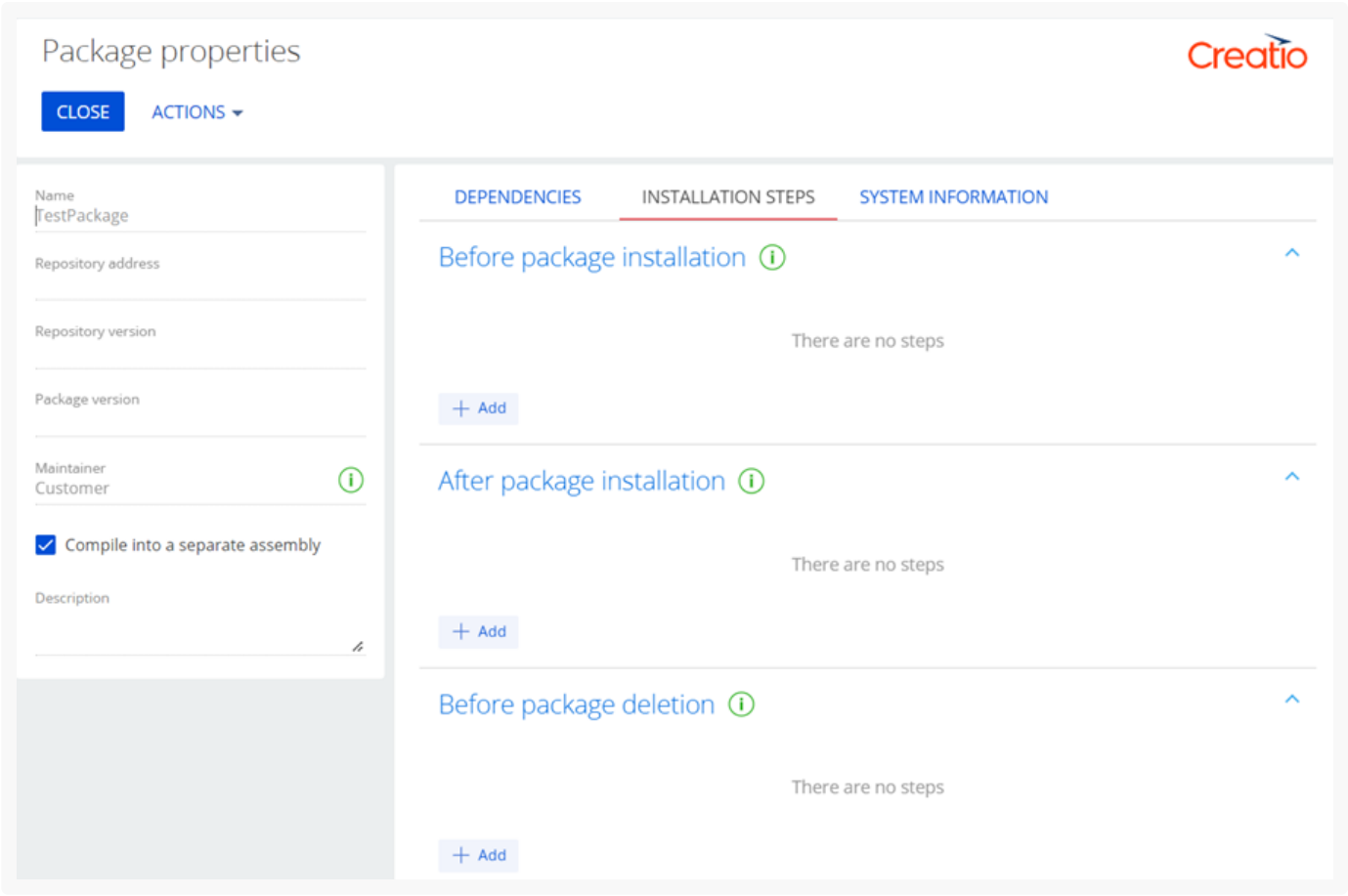
Click the [*Run as install script*] button on the Source Code Designer's toolbar to **debug the install script during development**.

2. Add the install script to the installation steps

1. Open the [*Configuration*] section and select a custom package.
2. Select [*Properties*] in the package menu.
3. Go to the [*Installation steps*] tab to set up the execution order of install steps as part of package installation.

   The [*Installation steps*] tab contains the following **start events**:

   - [*Before package installation*]
   - [*After package installation*]
   - [*Before package deletion*]



4. Add the install script to the start event.

   a. Click [+ *Add*] in the block of the start event.

   b. Select the schema that contains the install script.

   c. Click [*Select*].

   Creatio lets you execute the following **actions** on the [*Installation steps*] tab:

   - Add multiple install steps to a start event.
   - Drag install scripts within the block of a start event to change the execution order.
   - Click the button to delete an install step.
   - Click [*Run manually*] on an install step record to execute the script for the current Creatio instance without installing the package.

   If you use an install script to add a new record to an object that is being delivered for the first time, add the script to the [*After package installation*] start event. Otherwise, the install script execution will result in an error.

5. Click [*Apply*].

As a result, Creatio will execute the added install scripts as part of package installation.

Creatio executes the `Execute()` method of the class that implements the `Terrasoft.Core.IInstallScriptExecutor` interface as part of package installation. The method is called from `Source code` schemas specified on the [*Installation steps*] tab. If the `Source code` schema contains more than 1 interface of a class, Creatio executes

the `Execute()` method for each class that inherits from the `Terrasoft.Core.IInstallScriptExecutor` interface.

Work with big data

The execution time of install scripts is comparable with SQL scripts when working with small data sets. When working with large data sets, for example, 10,000 records, the script execution time increases the installation time significantly.

The effect of running the install script on the total installation time can differ depending on various **factors**, such as:

- the size of the data set with which the script is working
- computationally expensive operations within the script
- the resources allocated to the application server and the database, etc.

For example, you need to operate on a large data set, and the execution time of the install script is critical. Maximum number of records that you can obtain by EntitySchemaQuery request is specified in the [*MaxEntityRowCount*] (the default value is 20000). When working with large data sets, all records are loaded into memory, which can cauze the app to freeze. To prevent that, use pagination in EntitySchemaQuery requests.

When you work with large data sets, we recommend following these **recommendations**:

1. Use SQL scripts rather than install scripts.
2. Clear the [*Backward compatible*] checkbox for the used SQL script.
3. If the package installation ends with an error, restore the configuration from a database backup.

Examples of package install scripts

 Medium

Install scripts let you execute CRUD operations with data using `Entity` and `EntitySchemaQuery` classes. Learn more about the `Entity` class in the .NET class library. Learn more about the `EntitySchemaQuery` class in the .NET class library.

Retrieve data

Retrieve a contact that has specified data

Example. Retrieve a contact record that has the "`SomeContactName`" name.

Example that retrieves a contact record by name

```
public static Entity GetContact(UserConnection userConnection, string name) {
```

```
    EntitySchemaManager entitySchemaManager = userConnection.EntitySchemaManager;
    Entity contact = entitySchemaManager.GetEntityByName("Contact", userConnection);
    var contactConditions = new Dictionary<string, object> {
        { "SomeContactName", name }
    };
    return !contact.FetchFromDB(contactConditions) ? null : contact;
}
```

Retrieve data using pagination

Example. Fill out the type of delivery. Retrieve data using pagination.

Example that retrieves data using pagination

```
namespace Terrasoft.Configuration
{
    using System;
    using System.Collections.Generic;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;

    public class FillDeliveryType : IInstallScriptExecutor {

        public void Execute(UserConnection userConnection)
        {
            /* Create an ESQ that contains the [Id] column of the product and [Name] column from
            EntitySchemaQuery esqResult = new EntitySchemaQuery(userConnection.EntitySchemaManag
            EntitySchemaQueryColumn orderIdColumn = esqResult.AddColumn("Order.Id");
            EntitySchemaQueryColumn productCategoryColumn = esqResult.AddColumn("Product.Categor
            /* Implement pagination. */
            var options = new EntitySchemaQueryOptions {
                PageableDirection = Core.DB.PageableSelectDirection.First,
                RowsOffset = 0,
                PageableRowCount = userConnection.AppConnection.MaxEntityRowCount,
                PageableConditionValues = new Dictionary<string, object>()
            };
            /* Retrieve the collection of the orders. */
            EntityCollection productsInOrder = esqResult.GetEntityCollection(userConnection, opt
            EntitySchemaManager entitySchemaManager = userConnection.EntitySchemaManager;
            Entity orderEntity = entitySchemaManager.GetEntityByName("Order", userConnection);
            /* The first page. */
            EditOrders(productsInOrder, orderIdColumn, productCategoryColumn, orderEntity);
            int rowsRead = productsInOrder.Count;
            /* Use the pagination. */
            while (productsInOrder.Count > 0) {
```

```
                options.RowsOffset = rowsRead;
                /* Scroll forward. */
                options.PageableDirection = Core.DB.PageableSelectDirection.Next;
                /* Retrieve next page. */
                esqResult.ResetSelectQuery();
                productsInOrder = esqResult.GetEntityCollection(userConnection, options);
                rowsRead += productsInOrder.Count;
                EditOrders(productsInOrder, orderIdColumn, productCategoryColumn, orderEntity);
            }
        }

        private void EditOrders(EntityCollection productsInOrder, EntitySchemaQueryColumn orderI
                EntitySchemaQueryColumn productCategoryColumn, Entity orderEntity) {
                    /* Implement the modifying logic. */
        }
    }
}
```

To work with big data, follow the recommendations listed in a separate article: Customize delivery process.

Add data

Add a contact that has specified data

Example. Add a contact that has the specified name and account.

Example that adds a contact that has the specified name and account

```
public static void CreateContact(UserConnection userConnection, string contactName, string accou
    if (GetContact(userConnection, contactName) != null){
        return;
    }
    EntitySchemaManager entitySchemaManager = userConnection.EntitySchemaManager;
    Entity contact = entitySchemaManager.GetEntityByName("Contact", userConnection);
    contact.SetDefColumnValues();
    contact.SetColumnValue("Id", Guid.NewGuid());
    /* The AccountHelper class is a helper class. */
    contact.SetColumnValue("AccountId", AccountHelper.GetAccountId(userConnection, accountName))
    contact.SetColumnValue("Name", contactName);
    contact.Save();
}
```

Add a new job title that supports different cultures

Example. Add a new job title that supports Spanish and English cultures.

Example that adds a new job title that supports different cultures

```
namespace Terrasoft.Configuration {
    using System;
    using System.Globalization;
    using Terrasoft.Common;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;

    public class CreateJobTitle : IInstallScriptExecutor {

        public void Execute(UserConnection userConnection) {
            EntitySchemaManager entitySchemaManager = userConnection.EntitySchemaManager;
            Entity job = entitySchemaManager.GetEntityByName("Job", userConnection);
            job.SetDefColumnValues();
            job.SetColumnValue("Id", Guid.NewGuid());
            var jobName = new LocalizableString();
            var esCulture = new CultureInfo("es-ES");
            var enCulture = new CultureInfo("en-US");
            jobName.SetCultureValue(esCulture, "Mentor");
            jobName.SetCultureValue(enCulture, "Master");
            job.SetColumnValue("Name", jobName);
            job.Save();
        }
    }
}
```

Modify data

Example. Modify a contact field.

Example that modifies a contact field

```
public static void SetContactField(UserConnection userConnection, string contactName, string col
    /* The ContactHelper class is a helper class. */
    var entity = ContactHelper.GetContact(userConnection, contactName);
    /* The LanguageHelper class is a helper class. */
    entity.SetColumnValue("ProgrammingLanguageId", LanguageHelper.GetLanguageId(userConnection,
```

```
    entity.Save();
}
```

Delete data

Example. Delete a contact record.

Example that deletes a contact record

```
public static void DeleteContact(UserConnection userConnection, string contactName) {
    /* The ContactHelper class is a helper class. */
    var contact = ContactHelper.GetContact(userConnection, contactName);
    contact.Delete();
}
```

Populate a field using an install script

Medium

The example is relevant to Sales Creatio products.

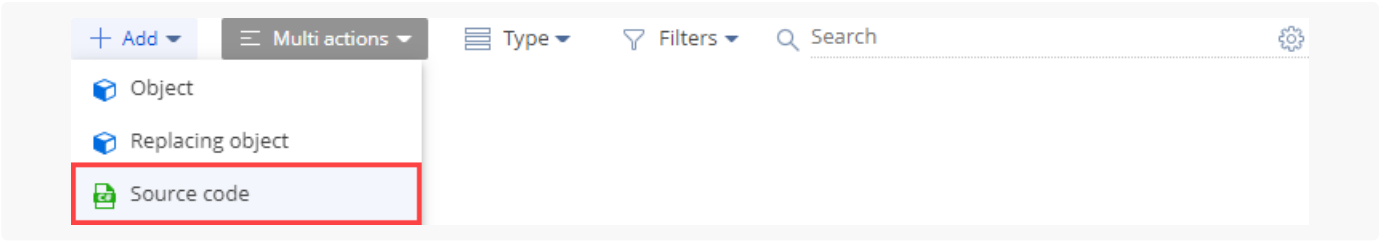
Example. Populate the [*Delivery type*] field on the order page using an install script.

- Set the field to "`Courier`" for products that have the "`Software`" category.
- Set the field to "`Customer pickup`" for products that have the "`Hardware`" and "`Services`" category.
- Leave the field empty for products that have other categories.

Populate the [*Delivery type*] field for orders that contain a single product.

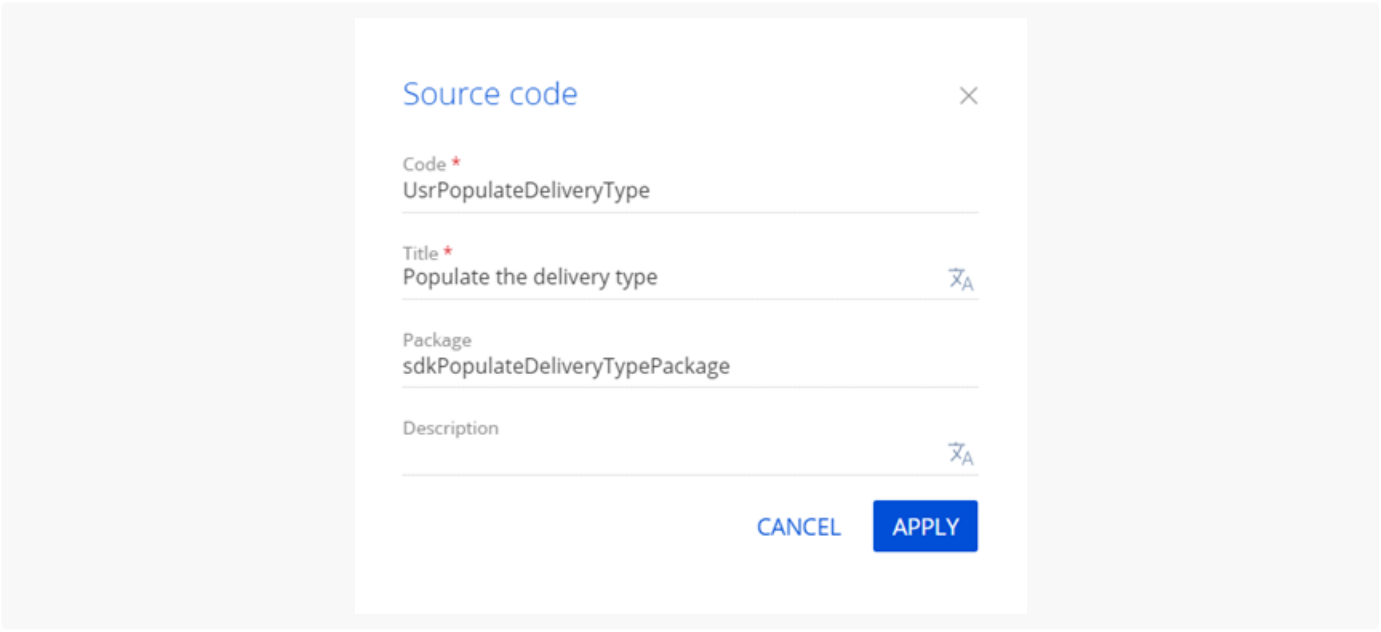
1. Implement an install script

1. Open the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Source code*] on the section list toolbar.



3. Fill out the **schema properties** in the Source Code Designer:

   - Set [*Code*] to "`UsrPopulateDeliveryType`."
   - Set [*Title*] to "`Populate the delivery type`."



   Click [*Apply*].

4. Implement the business logic of the delivery type filling.

`UsrFillDeliveryType`

```
namespace Terrasoft.Configuration
{
    using System;
    using System.Collections.Generic;
    using Terrasoft.Core;
    using Terrasoft.Core.Entities;

    public class UsrPopulateDeliveryType : IInstallScriptExecutor {

        private static object GetDeliveryIdByName(string deliveryName) {
            /* The unique ID of the delivery type. Delivery types are stored in the [Delivery
            switch (deliveryName) {
                case "Courier":
```

```
                    return new Guid("50df77d0-7b1f-4dbc-a02d-7b6ebb95dfd0");
                case "Customer pickup":
                    return new Guid("ab31305f-7c6d-4158-bd0a-760ac7897755");
                default:
                    return Guid.Empty;
            }
        }

        private static object GetDeliveryTypeByProductCategory(string category) {
            /* The category of the product. Product categories are stored in the [ProductCate
            switch (category) {
                case "Software":
                    return GetDeliveryIdByName("Courier");
                case "Hardware":
                    return GetDeliveryIdByName("Customer pickup");
                case "Services":
                    return GetDeliveryIdByName("Customer pickup");
                default:
                    return GetDeliveryIdByName("");
            }
        }

        public void Execute(UserConnection userConnection) {
            /* Create an ESQ that contains the [Id] column of the product and [Name] column f
            EntitySchemaQuery esqResult = new EntitySchemaQuery(userConnection.EntitySchemaMa
            EntitySchemaQueryColumn orderIdColumn = esqResult.AddColumn("Order.Id");
            EntitySchemaQueryColumn productCategoryColumn = esqResult.AddColumn("Product.Cate
            /* Retrieve the collection of products in the order. */
            EntityCollection productsInOrder = esqResult.GetEntityCollection(userConnection);
            EntitySchemaManager entitySchemaManager = userConnection.EntitySchemaManager;
            Entity orderEntity = entitySchemaManager.GetEntityByName("Order", userConnection)
            foreach (Entity entity in productsInOrder) {
                /* Retrieve the order object using the ID. */
                Guid id = entity.GetTypedColumnValue<Guid>(orderIdColumn.Name);
                string category = entity.GetTypedColumnValue<string>(productCategoryColumn.Na
                var conditions = new Dictionary<string, object> {
                    { "Id", id }
                };
                orderEntity.FetchFromDB(conditions);
                /* Populate the delivery type based on the product category. */
                orderEntity.SetColumnValue("DeliveryTypeId", GetDeliveryTypeByProductCategory
                /* Save the order. */
                orderEntity.Save();
            }
        }
    }
}
```

5. Click [*Publish*] on the Source Code Designer's toolbar to apply the changes to the database level.

2. Add the install script to the installation steps

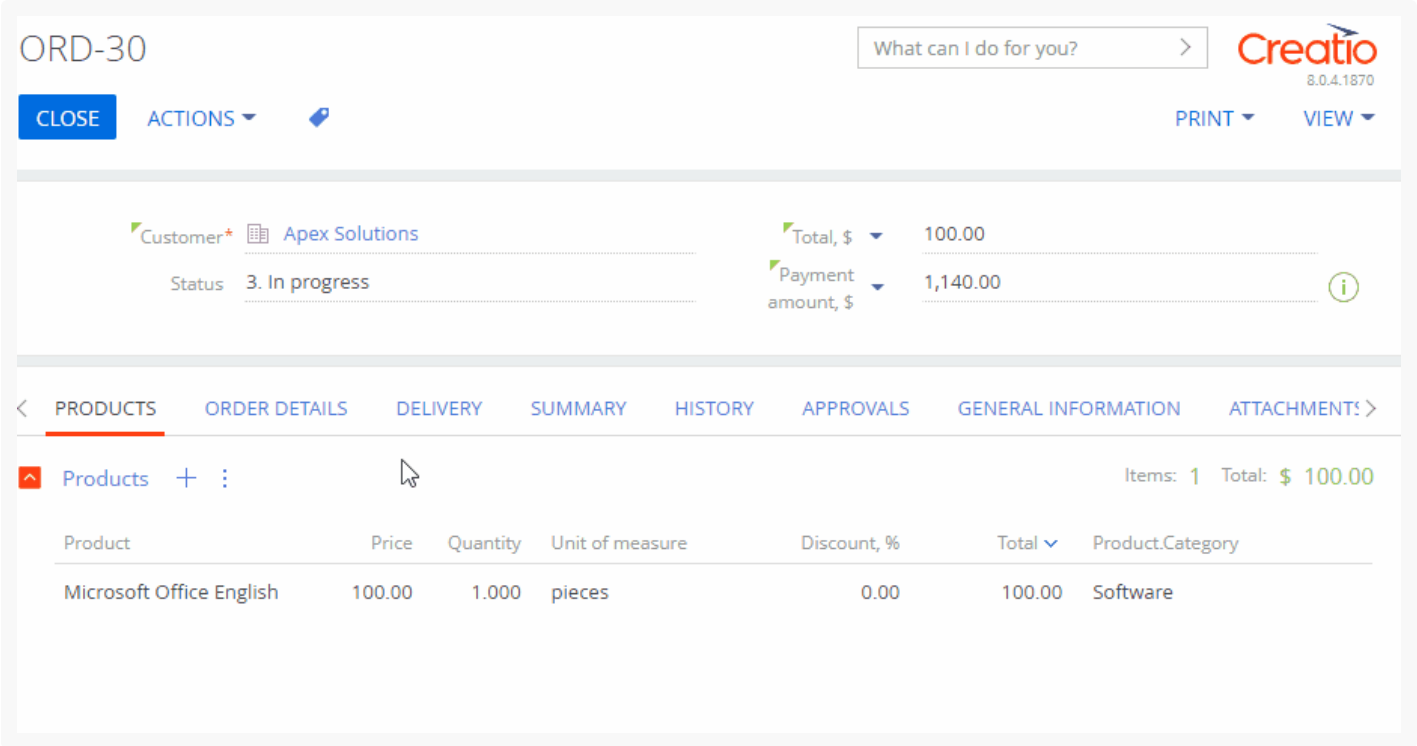
1. Open the [*Configuration*] section and select a custom package.
2. Select [*Properties*] in the package menu.
3. Go to the [*Installation steps*] tab to set up the execution order of steps when installing the package.
4. Add the `UsrFillDeliveryType` install script to the [*After package installation*] start event.
   a. Click [+ *Add*] in the [*After package installation*] block. This opens a window.
   b. Select the `UsrPopulateDeliveryType` schema.
   c. Click [*Select*].
5. Click [*Apply*].

Outcome of the example

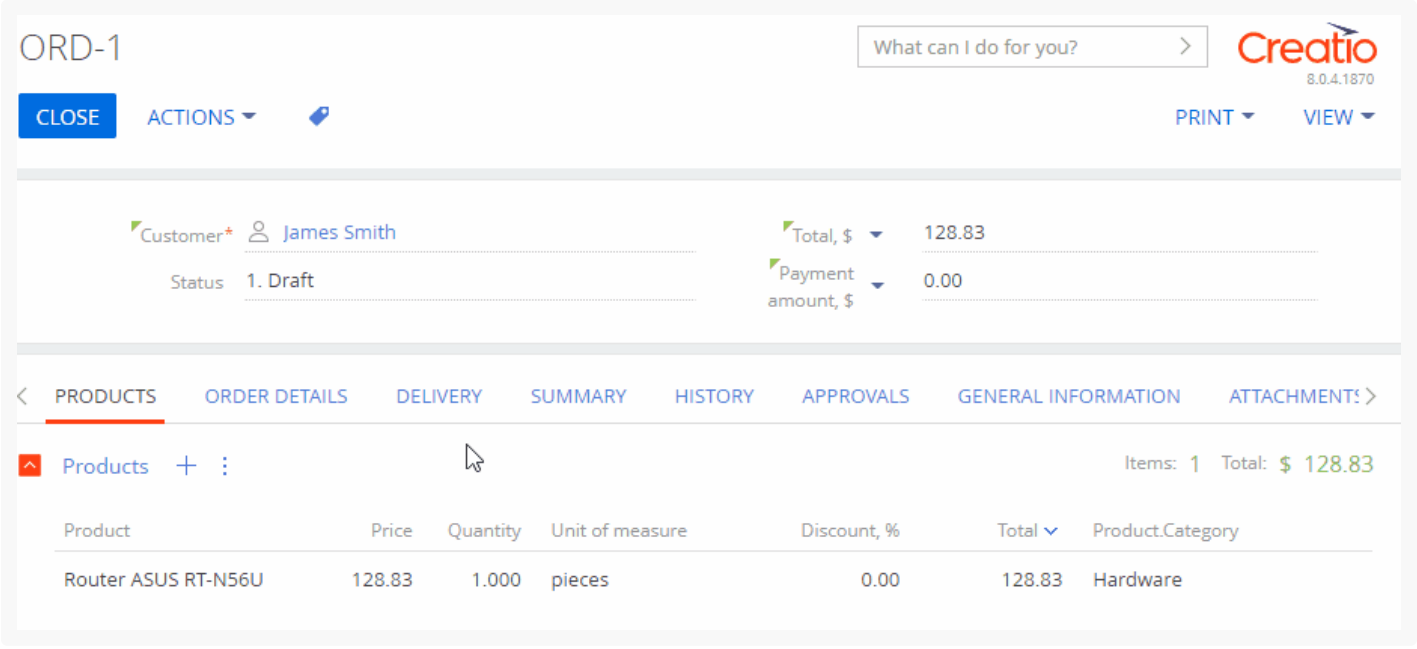
To **view the outcome of the example for the current Creatio instance**:

1. Click [*Actions*] → [*Run as install script*] on the Source Code Designer's toolbar to execute the [*Source code*] schema as install script.
2. Open the `Sales Enterprise` app page and click [*Run app*].
3. Open the `Orders` section.
4. Open the order page → [*Products*] tab to check the category of the product.
5. Go to the [*Delivery*] tab.

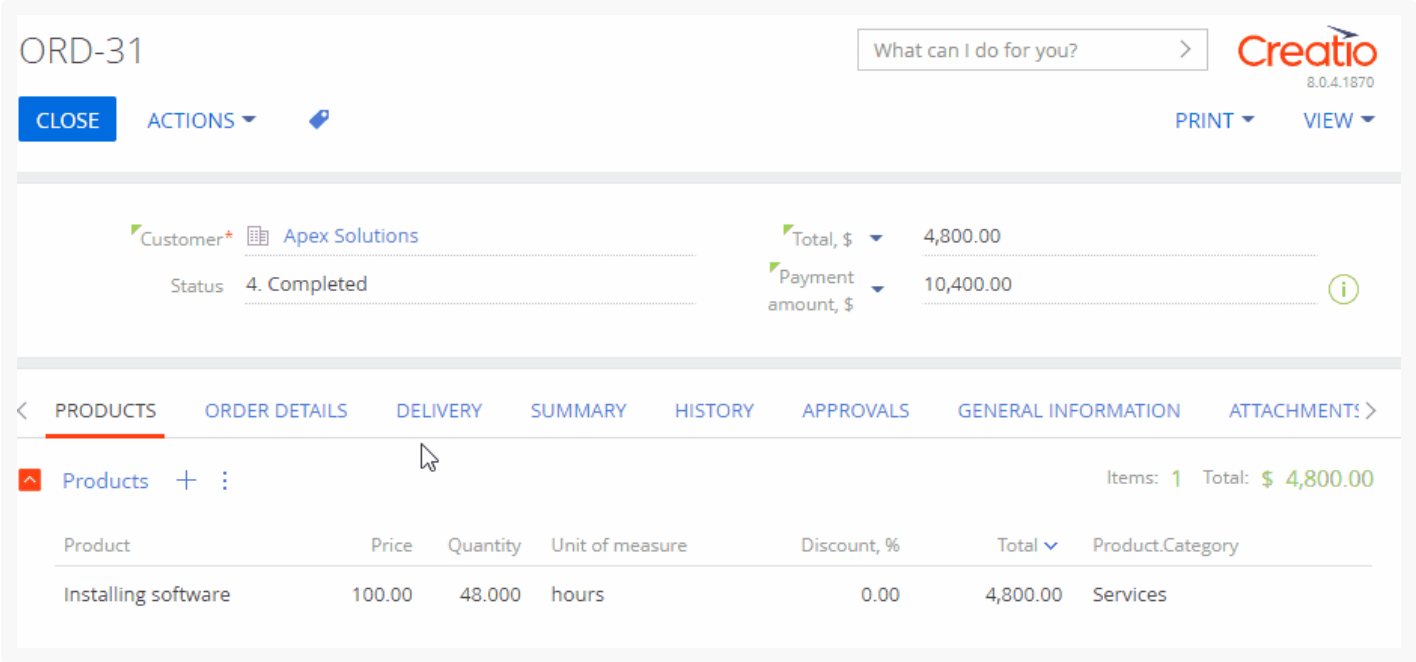
As a result, Creatio will set the "`Courier`" delivery type for products that have the "`Software`" category.



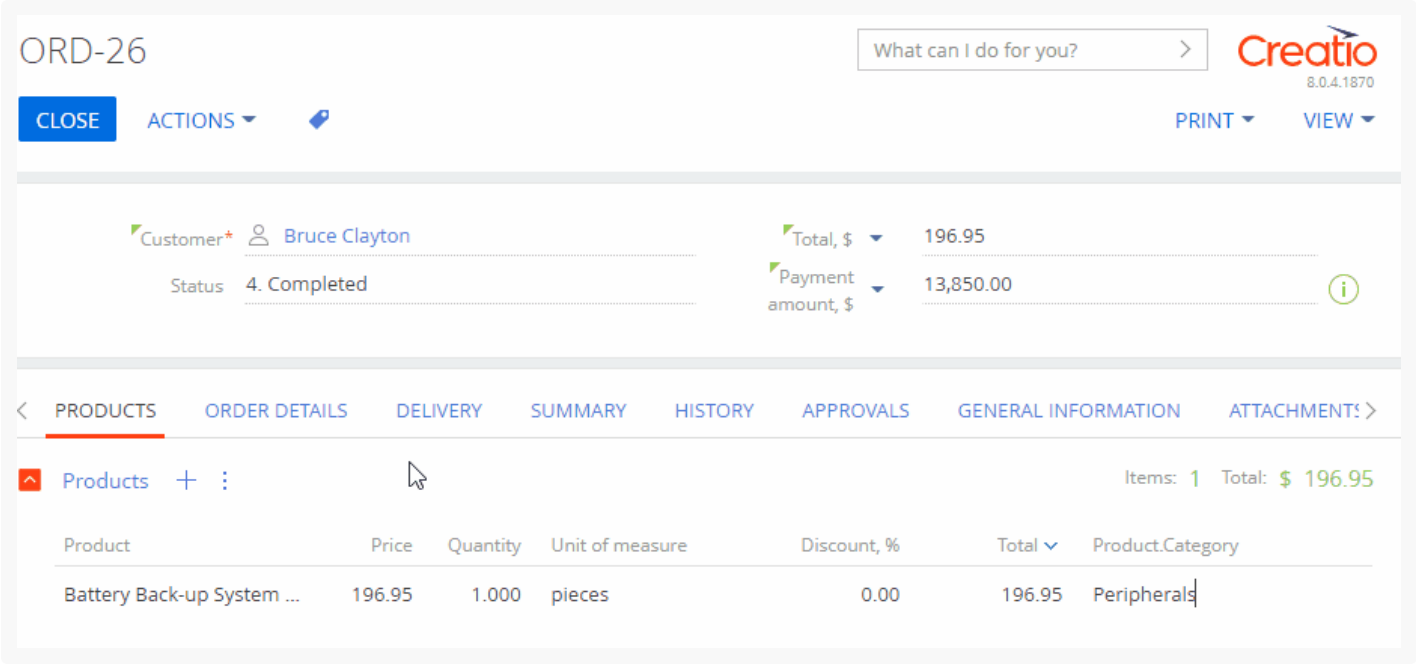
Creatio will set the "`Customer pickup`" delivery type for products that have "`Hardware`" category.



Creatio will set the "`Customer pickup`" delivery type for products that have "`Services`" category.



Creatio will leave the [*Delivery type*] field empty for products that have other categories.



To **view the outcome of the example for other Creatio instances**:

1. Select [*Export*] in the package menu.
2. Install the package to the needed Creatio instance. To do this, follow the guide in a separate article: Install apps from the Marketplace.
3. Repeat steps 2-5 of the procedure to view the outcome of the example for the current Creatio instance.

C# classes to avoid in install scripts C#

Medium

View the list of the C# classes and their methods to avoid in package install scripts in the table below.

C# classes to avoid in package install scripts

C# class to avoid	Methods to avoid
`Terrasoft.Core.DB.Insert`	All methods of the class
`Terrasoft.Core.DB.InsertSelect`	All methods of the class
`Terrasoft.Core.DB.CustomQuery`	All methods of the class
`Terrasoft.Core.DB.Delete`	All methods of the class
`Terrasoft.Core.DB.Update`	All methods of the class
`Terrasoft.Core.DB.UpdateSelect`	All methods of the class
`Terrasoft.Core.DB.DBMetaScript`	All methods of the class
`Terrasoft.Core.DB.DBExecutor.Execute(sqlText)`	All methods of the class
`Terrasoft.Core.DB.DBLobUtilities`	All methods of the class
`Terrasoft.Core.DB.UserDefinedFunction`	All methods of the class
`Terrasoft.Core.DB.StoredProcedure`	All methods of the class
`Terrasoft.Core.DB.DbCommentMetaEngine`	All methods of the class
`Terrasoft.Core.DB.EntitySchemaDbInfoActualizer`	All methods of the class
`Terrasoft.Core.DB.Sequence`	All methods of the class
`Terrasoft.Core.SchemaManager`	`RemoveSchema()`
	`SaveSchema()`
	`SaveSchemaInCustomPackage()`
	`SaveSchemaMetaData()`
	`MoveSchemas()`
	`MoveSchema()`

C# class to avoid	Methods to avoid
	`SaveSchemaPackageDifference()`
`Terrasoft.Core.Process.BaseProcessSchemaManager`	`RemoveRunningProcessDataByUId()`
	`RemoveRunningProcessData()`
	`EnableProcess()`
	`DisableProcess()`
`Terrasoft.Core.Process.ProcessSchemaManager`	`SaveDesignedSchema()`
	`Save()`
	`ConvertProcessSchema()`
	`CopyProcessSchema()`
`Terrasoft.Core.Campaign.CampaignSchemaManager`	`UnregisterStartSignalEvents()`
	`ActualizeCampaignSchemaInfo()`
	`SaveSchemaCopy()`
`Terrasoft.Core.Entities.EntitySchemaManager`	`SaveSchema()`
	`SaveEntitySchemaReferences()`
`Terrasoft.Services.ServiceSchemaManager`	`SaveSchema()`
`Terrasoft.Core.ImageAPI.ImageAPI`	All methods of the class
`Terrasoft.Core.ImageAPI.IImageAPI`	All methods of the class
`Terrasoft.Core.Packages.PackageStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageCompileDBStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageDBStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageFileStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageNonRemovableResourcesStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageRepositoryStorage`	All methods of the class

C# class to avoid	Methods to avoid
`Terrasoft.Core.Packages.PackageZipFileStorage`	All methods of the class
`Terrasoft.Core.Packages.SinglePackageDBStorage`	All methods of the class
`Terrasoft.Core.Packages.SinglePackageSvnStorage`	All methods of the class
`Terrasoft.Core.Packages.PackageInstallUtilities`	All methods of the class
`Terrasoft.Core.Packages.WorkspaceUtilities`	All methods of the class
`Terrasoft.Core.Process.ProcessUserTaskExtensions`	All methods of the class
`Terrasoft.Core.Configuration.SysSettings`	`SetValue()`
	`SetDefValue()`
`Terrasoft.Core.LicManager`	`DeleteLicenses()`
	`DeleteUsersLicenses()`
	`DeleteUserLicense()`
	`AddUsersAvailableLicences()`
	`AddLicensesToActiveUsers()`
	`AddUserLicense()`
`Terrasoft.Core.Scheduler.AppScheduler`	All methods of the class
`Terrasoft.Core.Process.ProcessEngineImpl`	All methods of the class
`Terrasoft.Core.Translation.IResourceProvider`	All methods of the class
`Terrasoft.Nui.ServiceModel.WebService`	All methods of the class
`Terrasoft.Nui.ServiceModel.BaseCrudDataService`	All methods of the class
`Terrasoft.Nui.ServiceModel.BaseSysSettingsService`	All methods of the class
`Terrasoft.Nui.ServiceModel.BaseSspCrudDataService`	All methods of the class
`Terrasoft.Nui.ServiceModel.Extensions`	All methods of the class
`Terrasoft.Web.Common.LoginUtilities`	All methods of the class

C# class to avoid	Methods to avoid
	`CreateUser()`
`Terrasoft.Web.Common.SysAdminUtilities`	`RemoveUserRoles()`
	`RunUpdateSsoContactProcess()`
	`SaveSsoContactValues()`
`Terrasoft.File.Abstractions.IFile`	`CopyAsync()`
	`MoveAsync()`
	`DeleteAsync()`
	`WriteAsync()`
	`SaveAsync()`
	`SetAttribute()`
`Terrasoft.IO.IDirectory` and `System.IO.Directory`	`CreateDirectory()`
	`Clear()`
	`Delete()`
	`SafeDelete()`
	`Move()`
	`CreateOrReplaceDirectory()`
	`CopyDirectory()`
`Terrasoft.IO.IFile` and `System.IO.File`	`Copy()`
	`Create()`
	`Delete()`
	`Move()`
	`SetAttributes()`
	`SetCreationTime()`

C# class to avoid	Methods to avoid
	`SetLastWriteTimeUtc()`
	`WriteAllBytes()`
	`WriteAllLines()`
	`WriteAllText()`
`System.Data.Common`	All methods of the class
`System.Data.SqlClient`	All methods of the class
`Microsoft.Data.SqlClient`	All methods of the class
`Oracle.DataAccess`	All methods of the class
`Oracle.ManagedDataAccess`	All methods of the class
`Npgsql`	All methods of the class
`NEventStore`	All methods of the class
`Quartz`	All methods of the class