The Kentico E-commerce Solution marks shopping carts as abandoned with a scheduled task. The scheduled task creates an activity for every abandoned shopping cart. When you then set up a marketing automation process, the system sends an email reminder to customers with an abandoned shopping cart. In the email reminder, there can be, for example, links to the products added to their shopping carts or a link to their shopping cart to continue shopping.

Abandoned shopping carts do not have any property indicating whether they are abandoned or not. The process is used only to create the abandoned shopping cart activities for marketing automation.

How abandoned shopping cart reminders work

The scenario presumes the situation when a visitor (potentially a future customer) considers purchasing some products but then changes the mind and leaves. Store administrators then want to remind the visitor of the products. To be able to contact a visitor, the system saves the ID of the visitor's contact to the shopping cart (specifically to the *ShoppingCartContactID* column in the *COM_ShoppingCart* database table). In this scenario, visitors can be:

- A registered customer – the customer has a user assigned, the user has a contact assigned.
- An anonymous customer – the visitor has a contact assigned according to their activities. The contact can be:
    - With an email address – the email address is known, for example, from a subscribed newsletter.
    - Without an email address – the visitor did not provide any email address, therefore you cannot reach such visitor.

In the administration interface, you can set after how many hours shopping carts are considered as abandoned (see Walkthrough - Sending an automated reminder of an abandoned shopping cart). Besides that, you can customize the way the system recognizes abandoned shopping carts.

Customizing evaluation whether a shopping cart is abandoned

You can change the evaluation process when the system decides whether the shopping cart is abandoned.

When the scheduled task starts, the system initializes the **MarkCartAbandoned** class and triggers the **Execute** method. The **Execute** method retrieves all shopping carts fulfilling all of the following conditions:

- The shopping cart is non-empty (i.e., the shopping cart contains at least one product).
- The content of the shopping cart has not been changed for the time specified by the **Mark shopping cart as abandoned after (hours)** setting (the time is determined according to the *ShoppingCartLastUpdate* database column).
- The shopping cart has not been processed by previous runs of the scheduled task yet.

    If the shopping cart was changed after it was processed by the scheduled task's previous run (e.g., if the customer opened the shopping cart, changed the products and left again), the shopping cart is then again processed by the scheduled task.

Then you can filter the retrieved shopping carts with the **FilterAbandonedCarts** method:

```
protected virtual ObjectQuery<ShoppingCartInfo> FilterAbandonedCarts
(ObjectQuery<ShoppingCartInfo> abandonedCarts)
{
    return abandonedCarts;
}
```

The **Execute** method then loops through the filtered shopping carts which are labeled as abandoned with the **IsShoppingCartAbandoned** method:

```
protected virtual bool IsShoppingCartAbandoned(ShoppingCartInfo cart)
{
    return true;
}
```

The system then creates a **Shopping cart abandoned** activity for every looped abandoned shopping cart to its contact that ensures sending the reminder email to the contact.

Example – Creating a custom evaluation of abandoned shopping carts

Implement a class inheriting from the **MarkCartAbandoned** class to provide your custom logic of evaluating abandoned shopping carts.

1. Create a new class file in your project in Visual Studio, for example **CustomAbandonedCartEvaluation.cs**, that inherits from the **MarkCartAbandoned** class.
2. Add the **CMS**, **CMS.Ecommerce** and **CMS.DataEngine** namespaces to the *using* block.
3. Assign the inheritance from the **MarkCartAbandoned** class to the **CustomAbandonedCartEvaluation** class:

```
public class CustomAbandonedCartEvaluation : MarkCartAbandoned
{
}
```

4. Add the **RegisterCustomClass** assembly attribute above the class declaration.

```
[assembly: RegisterCustomClass("Custom.CustomMarkCartAbandoned", typeof
(CustomAbandonedCartEvaluation))]
public class CustomAbandonedCartEvaluation : MarkCartAbandoned
{
}
```

5. Override the methods you want to change.

   a. Override the **FilterAbandonedCarts** method. For example, if you do not want to send reminders to registered customers, you can exclude shopping carts containing user ID using the ObjectQuery API:

```
protected override ObjectQuery<ShoppingCartInfo> FilterAbandonedCarts
(ObjectQuery<ShoppingCartInfo> abandonedCarts)
{
    // Gets all abandoned carts where the user ID is not specified.
    ObjectQuery<ShoppingCartInfo> filteredCarts = abandonedCarts.WhereNull
("ShoppingCartUserID");

    return filteredCarts;
}
```

   b. Override the **IsShoppingCartAbandoned** method. For example, if you do not want to send reminders when the shopping cart is in euros, you can exclude such shopping carts:

```
protected override bool IsShoppingCartAbandoned(ShoppingCartInfo cart)
{
    // Checks whether the shopping cart is paid in euros.
    if (cart.Currency.CurrencyCode.EqualsCSafe("EUR"))
    {
        return false;
    }

    return true;
}
```

If you use the mentioned example, you need to add the **CMS.Base** namespace to the *using* block.

6. Create a new scheduled task in the **Scheduled tasks** application that will use the newly created *CustomAbandonedCartE valuation* class.

The system now excludes shopping carts based on your customization.

Learn more about scheduling custom tasks in Scheduling custom tasks.

```
using System;

using CMS;
using CMS.Ecommerce;
using CMS.DataEngine;
using CMS.Base;

/// <summary>
/// Customizes the process of evaluation abandoned shopping carts.
/// </summary>
[assembly: RegisterCustomClass("Custom.CustomMarkCartAbandoned", typeof
(CustomAbandonedCartEvaluation))]
public class CustomAbandonedCartEvaluation : MarkCartAbandoned
{
    protected override ObjectQuery<ShoppingCartInfo> FilterAbandonedCarts
(ObjectQuery<ShoppingCartInfo> abandonedCarts)
    {
        // Gets all abandoned carts where the user ID is not specified.
        ObjectQuery<ShoppingCartInfo> filteredCarts = abandonedCarts.WhereNull
("ShoppingCartUserID");

        return filteredCarts;
    }

    protected override bool IsShoppingCartAbandoned(ShoppingCartInfo cart)
    {
        // Checks whether the shopping cart is paid in euros.
        if (cart.Currency.CurrencyCode.EqualsCSafe("EUR"))
        {
            return false;
        }

        return true;
    }
}
```