AMG99990

Tuesday, October 14, 2025 2:09 PM

Project AMG_99990

We need a new field on the customer: LoyaltyLevel. This field can have 4 values: None, Bronze, Silver and Gold.

1. The field should be visible and editable on the customer form.
2. The field should be added to the Data entity so it can be exported to BYOD.
3. Create code to fill this field for existing customers:
   If the turnover in the last three months was <= 10,000: None
   If the turnover in the last three months was > 10,000 and <= 20,0000: Bronze
   If the turnover in the last three months was > 20,000 and <= 30,0000: Silver
   If the turnover in the last three months was > 30,000: Gold

   You will have to create a class that is called by a Action Menu Item. Name of the menu item: Recalculate the Loyalty level.
   This class will have to go over each customer, check how much this customer has bought in the last 3 months and set the correct Loyalty Level.

   Add this Action Menu Item as an item in the Periodic menu of the Accounts receivable menu:

4. If the previous step works, we're going to make it a bit more complicated. Try if you can get this one to work as well:

   Besides adding the menu-item to the Periodic menu, also add it as a button on the customer form. In case the button is called from the customer table, only the current customer should be recalculated.

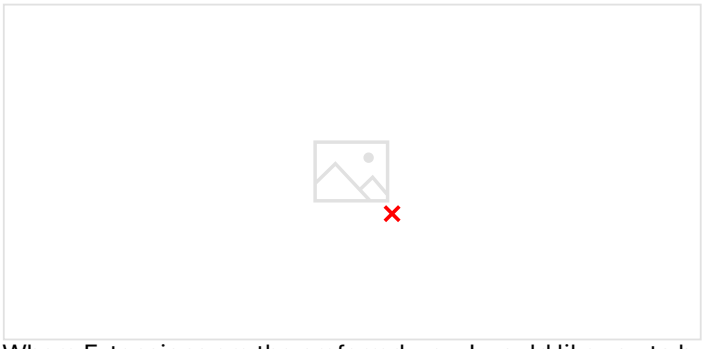
   As said: This makes it more complicated. Just see if you can get this to work, but do not spend to much time on this.

   Hint: The menu item on the form should pass the dataSource (= customer table) to the class. If you define that on the menu item on the form, the class main(args _args) method will receive the customer record in the _args argument.
5. Create code to check user input: If the customer is blocked, the Loyalty Level can not be changed to Gold. A customer in D365FO is blocked (in this scenario) when this field:

has the value All.

There are two ways to do this and in both cases you will need to create a class. One way is extending the table/form (??) like this:

The other way is using event handlers like this:



Where Extensions are the preferred way. I would like you to build both. The Extension and the Event handler, as there are situations in which you cannot use an extension.

Google for the stages what happens if you update/insert a record in Dynamics. You will find that it goes through several methods/events like validatWrite, write, insert, inserted etc. Try to figure out which of these methods/events you would need to extend or create an event handler for.

The screenshots above show examples from the Insert method. That's not what you need as the customer already exists, so no new record is inserted. An existing customer record is updated.

RESEARCH==========

updated.

RESEARCH==========

Summary of What the Screenshots Show

1. **Accounts Receivable > Periodic Tasks Menu (Image 1)**
   - This shows where your **Action Menu Item** ("Recalculate the Loyalty level") should be added. It confirms the correct navigation path:
     Modules > Accounts receivable > Periodic tasks.
2. **Customer Form – Credit and Collections Tab (Image 2)**
   - This is the customer form where the **LoyaltyLevel field** should be made visible and editable.
   - It also shows where a **button** could be added to trigger the recalculation for the current customer.
3. **Table Extension Example (Image 3)**
   - Demonstrates how to use [ExtensionOf(tableStr(...))] to extend a standard table.
   - This is relevant for extending the **CustTable** to add the LoyaltyLevel field and logic to restrict setting it to Gold if the customer is blocked.
4. **Event Handler Example (Image 4)**
   - Shows how to use [DataEventHandler(...)] for reacting to data events like Inserted.
   - You'll use a similar approach for handling updates to the LoyaltyLevel field via event handlers, especially for validation logic.