



POWERED BY  DUG

DYNAMICS CON LIVE

MAY 2024

Give the User a Chance

Just because it works, doesn't mean it works ...



Henrik Helgesen – Aka TheDoubleH

- Originally from Denmark.
- Currently live in Los Angeles, California
- Started with Navigator 3.04 in 1995
- Have worked in all aspects from before life, to death – all in NAV / BC
- Stickler for rules – Previously known as the GUI Police
- Long Time Listener, First time Caller
- Hockey Fan (Los Angeles Kings)
- Magic Fan (Magic Castle Member)





Yes - I am a Dinosaur

Objective

Share some tips on what options you as a developer have to help guide the user in their day-to-day work



As Developers we tend to “make it Work”

**... but only if the user does EXACTLY
what we expect them to do ...**



What can we do?





Prevent Bad Data

Prevent <Blank> Records

→ ↻ | + New Edit List Delete | ↗ 🔍 ☰

Code ↑	Description	G/L Account No.
→	⋮	
ONE	One	
THREE	Three	
TWO	Two	



Prevent <Blank> Records

- As a Developer, You can ensure that the Primary Key fields is populated.
- Base App: Table 251 – Gen. Product Posting Group
- The `Code` field has NotBlank = true

```
1 table 251 "Gen. Product Posting Group"
2 {
3     Caption = 'Gen. Product Posting Group';
4     DataCaptionFields = "Code", Description;
5     LookupPageID = "Gen. Product Posting Groups";
6
7     fields
8     {
9         field(1; "Code"; Code[20])
10        {
11            Caption = 'Code';
12            NotBlank = true;
13        }
14        field(2; Description; Text[100])
15        {
16            Caption = 'Description';
17        }
18        field(3; "Def. VAT Prod. Posting Group"; Code[20])
19        {
20            Caption = 'Def. VAT Prod. Posting Group';
21            TableRelation = "VAT Product Posting Group";
22        }
23    }
24 }
```

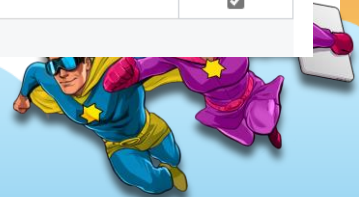
General Product Posting Groups Not saved

Search + New Edit List Delete Setup More options

✖ The page has an error. [Refresh \(F5\)](#) to undo the change, or correct the error.

Code ↑	Description	Def. VAT Prod. Posting Group	Auto Insert Default
MANUFACT	Capacities		<input checked="" type="checkbox"/>
NO TAX	Miscellaneous without tax		<input checked="" type="checkbox"/>
RAW MAT	Raw materials		<input checked="" type="checkbox"/>
RETAIL	Retail		<input checked="" type="checkbox"/>
SERVICES	Resources, etc.		<input checked="" type="checkbox"/>
	NotBlank is set to true		<input checked="" type="checkbox"/>
			<input checked="" type="checkbox"/>

✖ ✖ Validation Results
Code must be filled in. Enter a value.



Prevent <Blank> Records

Property: **NotBlank**

Sets a value that specifies whether users must enter a value in the selected field or text box.

Only “valid” for Primary Key Fields.

Only Primary Key Fields are enforced on OnInsert, but other fields are enforced – if you happen to enter / clear a value.



Indicate Mandatory fields for processing

On pages, You can set ShowMandatory to true. This will place a red asterisk in the text box, giving the user a visual indicator that a value is required.

```
54     field("No."; "No.")
55     {
56         ApplicationArea = Basic, Suite;
57         ShowMandatory = Type <> Type::" ";
58         Tooltip = 'Specifies the number of a general ledger account, item, resource, additional c
59
60     trigger OnValidate()
61     begin
62         NoOnAfterValidate();
63         UpdateEditableOnRow();
64         ShowShortcutDimCode(ShortcutDimCode);
65
66     trigger AfterValidate()
```

The screenshot shows the Dynamics 365 Business Central interface for a Sales Order. The header includes the order number 'S-ORD101001' and the company name 'Adatum Corporation'. The 'General' section contains fields for Customer Name, Contact, Posting Date, Order Date, Due Date, Requested Delivery Date, External Document No., and Status. Below this is a table with columns for Type, No., Item Reference No., Description, Location Code, and Quantity. A row is highlighted with a red border, showing a dropdown menu with 'New' selected, a red asterisk in the 'No.' column, and a red asterisk in the 'Description' column. The bottom of the screen shows 'Subtotal Excl. Tax (USD)' and 'Total Excl. Tax (USD)'.



Indicate Mandatory fields for processing

Property: **ShowMandatory**

Sets a value that specifies whether users must enter a value in the selected field or text box. The field is marked on the page with a red asterisk and does not enforce any validation. Once the field is filled, the red asterisk disappears. The ShowMandatory property only controls the UI and overrides any asterisk marking of the NotBlank Property.

Also, be aware that while it is possible to use an expression for the ShowMandatory property, the property cannot validate an AL function.

For an example, see the Purchase Invoice page where the Vendor Invoice No field is controlled by the VendorInvoiceNoMandatory expression.

VendorInvoiceNoMandatory is a variable that takes the value from the Ext. Doc Mandatory field on the Purchase page and the Payables page.



Indicate Mandatory fields for processing

Property: **ShowMandatory**

Sets a value that specifies whether users must enter a value in the selected field or text box. The field is marked on the page with a red asterisk and does not enforce any validation. Once the field is filled, the red asterisk disappears. The ShowMandatory property only controls the UI and overrides any asterisk marking of the NotBlank Property.

Also, be aware that while it is possible to use an expression for the ShowMandatory property, the property cannot validate an AL function.

For an example, see the Purchase Invoice page where the Vendor Invoice No field is controlled by the VendorInvoiceNoMandatory expression.

VendorInvoiceNoMandatory is a variable that takes the value from the Ext. Doc Mandatory field on the Purchase page and the Payables page.





Steve Endow

@steveendow

Why is the Vendor Invoice No field marked as required for a brand new Purchase Order? 🤔 #MSDyn365BC

Purchase Order 106009 · Fabrikam, Inc.

Home Prepare Print/Send Request Approval Order Actions Related Automate Fewer options

Post Release Create Whse. Receipt Create Inventory Put-away/Pick... Send Intercompany Purchase Order Archive Document

General Show less

Vendor No.	10000	Invoice Received Date	
Vendor Name	Fabrikam, Inc.	Posting Date	12/31/2023
Buy-from		Due Date	12/31/2023
Address	10 North Lake Avenue	Vendor Invoice No.	* The value for this field is required.
Address 2		Your Reference	
City	Atlanta	Purchaser Code	

12:09 PM · Apr 14, 2024



NotBlank

≠

ShowMandatory



ShowMandatory
Is not
Enforced

Enabled (true/false)

```
field("Reorder Quantity"; "Reorder Quantity")
{
  ApplicationArea = Planning;
  Enabled = ReorderQtyEnable;
  Tooltip = 'Specifies a standard lot size quantity to be used for all order proposals';
}
```

Manipulating the Enabled Property keeps the field on the page but enables control whether field can be edited or not.

This can be good when another field controls what information is needed.

The Item Card is a good example of it's use.

Planning

Show more

Reordering Policy	<input type="text" value="Order"/>
Order Tracking Policy	<input type="text" value="None"/>
Stockkeeping Unit Exists	<input checked="" type="checkbox"/> No
Critical	<input checked="" type="checkbox"/>
Safety Lead Time	<input type="text"/>
Safety Stock Quantity	<input type="text" value="0"/>
Lot-for-Lot Parameters	
Include Inventory	<input checked="" type="checkbox"/>
Lot Accumulation Period	<input type="text"/>
Rescheduling Period	<input type="text"/>

Reorder-Point Parameters

Reorder Point	<input type="text" value="25"/>
Reorder Quantity	<input type="text" value="0"/>
Maximum Inventory	<input type="text" value="0"/>

Order Modifiers

Minimum Order Quantity	<input type="text" value="0"/>
Maximum Order Quantity	<input type="text" value="0"/>
Order Multiple	<input type="text" value="0"/>



Enabled (true/false)

```
field("Reorder Quantity"; "Reorder Quantity")
{
  ApplicationArea = Planning;
  Enabled = ReorderQtyEnable;
  ToolTip = 'Specifies a standard lot size quantity to be used for all order proposals';
}
```

Manipulating the Enabled Property keeps the field on the page but enables control whether field can be edited or not.

This can be good when another field controls what information is needed.

The Item Card is a good example of it's use.

Planning

Show more

Reordering Policy	Maximum Qty.	Reorder-Point Parameters	
Order Tracking Policy	None	Reorder Point	25
Stockkeeping Unit Exists	No	Reorder Quantity	0
Critical	<input type="checkbox"/>	Maximum Inventory	0
Safety Lead Time		Order Modifiers	
Safety Stock Quantity	0	Minimum Order Quantity	0
Lot-for-Lot Parameters		Maximum Order Quantity	0
Include Inventory	<input type="checkbox"/>	Order Multiple	0
Lot Accumulation Period			
Rescheduling Period			



Visible (true/false)

Visible should only be set OnOpen.
Otherwise, the page can be jumbled
and confusing.

Works well for Card pages for tables
that require different fields populated
based on an already known value. i.e.:

- Credit Card Apps
- Transaction Type
- Card Issuer
- Gateway
- Shipping Solutions
- Carrier

```
131 trigger OnOpenPage()  
132 begin  
133     IsCountyVisible := FormatAddress.UseCounty("Country/Region Code");  
134 end;  
135  
136
```

```
36     group(Control15)  
37     {  
38         ShowCaption = false;  
39         Visible = IsCountyVisible;  
40         field(County; County)  
41         {  
42             ApplicationArea = Basic, Suite;  
43         }  
44     }
```





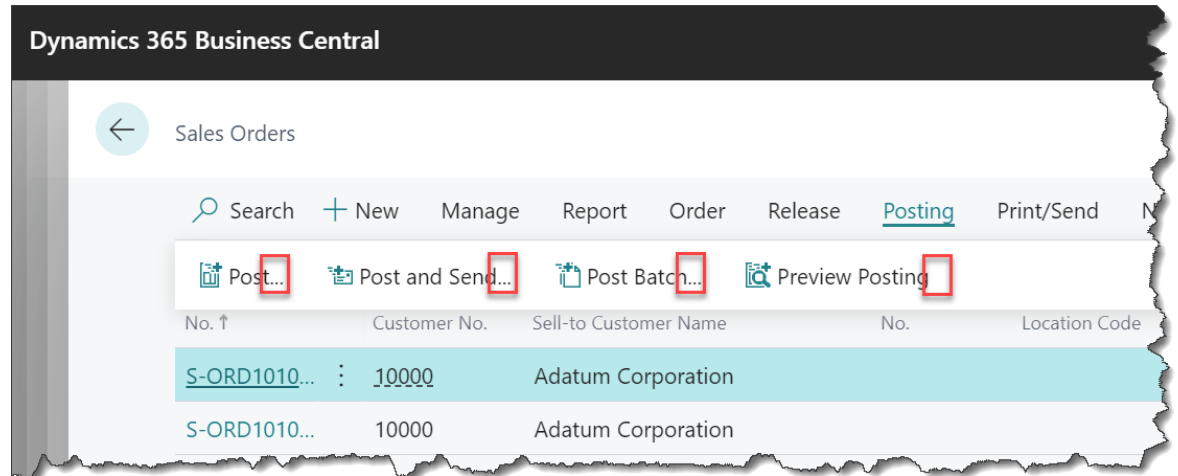
Indicators

Ellipsis (Three dots)

Official Documentation says: "Sets a value that specifies whether an ellipsis (...) is appended to the caption on a command button or menu item.

An ellipsis tells the user that other choices will appear if the command button or menu item are selected."

To me – It's an indicator that 'nothing bad' will happen if You press this action.

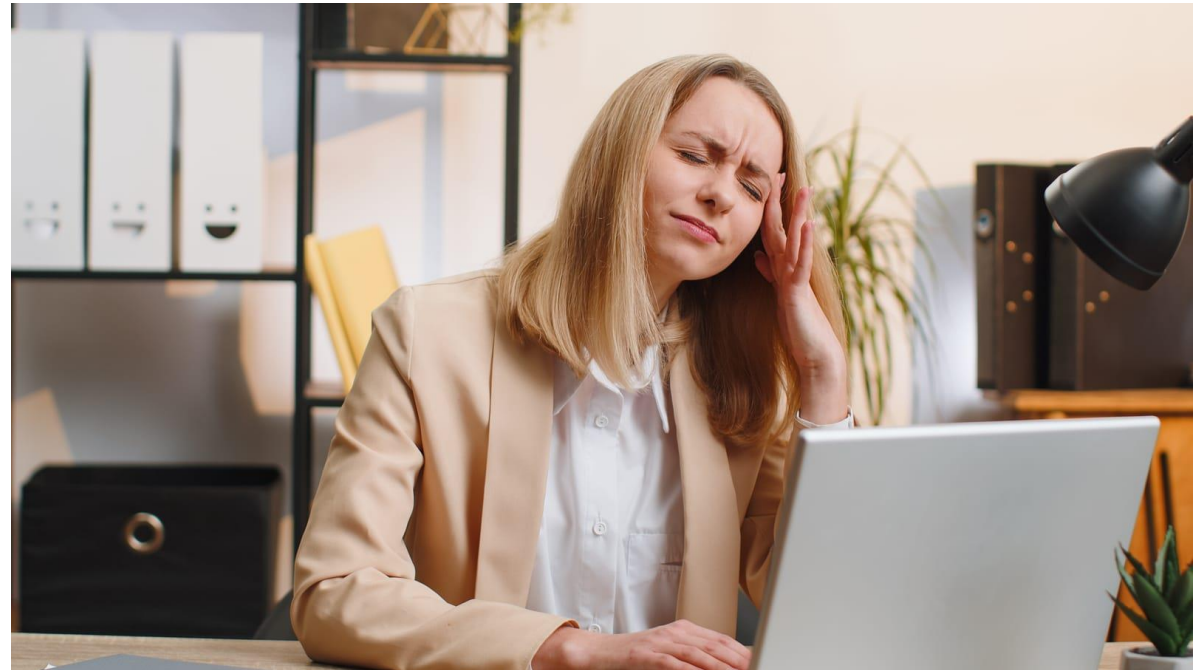


Show something is happening

Unless You, as a developer, do anything, Users are kept in the dark about what is happening.

All they are shown is this:

Working on it ...



Show something is happening

When processing data, that could take longer than one second, I always show the user, that something is happening.

Using the Dialog variable type, allows for 'communication' to the user – and more importantly – gives the user a 'Cancel' button.

Window.Update takes a lot of resources, so You will want to be cautious, and not update too rapidly.

You really shouldn't need to update more often than ever second.

```
if TempSalesLineGlobal.FindSet() then
  repeat
    ErrorMessageMgt.PushContext(ErrorContextElementPostLine, TempSalesLineGlobal.RecordId, 0, PostDocumentLinesMsg);
    ItemInRollRndg := false;
    LineCount := LineCount + 1;
    if not HideProgressWindow then
      Window.Update(2, LineCount);

  PostSalesLine(
    SalesHeader, TempSalesLineGlobal, EverythingInvoiced, TempVATAmountLine, TempVATAmountLineRemainder,
    TempItemLedgEntryNotInvoiced, HasATOShippedNotInvoiced, TempDropShptPostBuffer, ICGenInLineNo,
    TempServiceItem2, TempServiceItemComp2);

  if RoundingLineInserted then
    LastLineRetrieved := true
  else begin
    BiggestLineNo := MAX(BiggestLineNo, TempSalesLineGlobal."Line No.");
    LastLineRetrieved := TempSalesLineGlobal.Next() = 0;
    if LastLineRetrieved and SalesSetup."Invoice Rounding" then
      InvoiceRounding(SalesHeader, TempSalesLineGlobal, false, BiggestLineNo);
  end;
OnRunOnBeforePostSalesLineEndLoop(SalesHeader, TempSalesLineGlobal, LastLineRetrieved, SalesInvHeader, SalesCrMemoHeader, Rec, xSalesLine);
ErrorMessageMgt.PopContext(ErrorContextElementPostLine);
```

Working on it...

Order S-ORD101005 -> Invoice PS-INV103215
Posting lines 23
Posting sales and tax
Posting to customers
Posting to bal. account

Cancel



Show something is happening

... and show something when you are done.



Processing Complete.

OK



Processing Complete.

147 Lines inserted.

OK



My gift to you ...



Working on it...

Updating entries ...

Entry Description: Direct Cost 20000 on 09/23/21



Elapsed time :..... 1 Minutes, 14 Seconds

Estimated time left :..... 3 Minutes, 4 Seconds

Estimated end time :..... 22:45:52

Cancel

```
procedure UpdateWindow(Counter: Integer; NoOfRecords: Integer);
var
  ProgressBar: Codeunit "Progress Bar TBHLG";
  EndTime: DateTime;
  CurrDuration: Duration;
  EstimatedDuration: Duration;

begin
  if CurrentDateTime < LastUpdate + 1000 then
    exit;

  if Counter = 0 then
    exit;

  Window.Update(20, ProgressBar.ProgressBar(Counter, NoOfRecords));
  LastUpdate := CurrentDateTime;
```

```
var
  GLEntry: Record "G/L Entry";
  DialogHelperTBHLG: Codeunit "Dialog Helper TBHLG";
  NoOfRecords: Integer;
  OnRecordNo: Integer;
begin
  DialogHelperTBHLG.OpenWindow('Updating entries ...\\Entry Description: #1#####', true);
  NoOfRecords := GLEntry.Count();
  GLEntry.FindSet();
  repeat
    OnRecordNo += 1;
    DialogHelperTBHLG.UpdateWindow(1, GLEntry.Description, OnRecordNo, NoOfRecords);
    sleep(50)
  until GLEntry.next() = 0;
end;
```

In my BCALToolbox (Open Source <https://go.thedoubleh.dev/toolbox>) I have created function, that make it easy.

These functions makes sure I only update once every second.





Meaningful Errors

Give the user a reference



The Customer does not exist. Identification fields and values:
No.="

Share details ▾

Was this information helpful? Yes No

OK



The Payment Terms does not exist. Identification fields and values:
Code="

Share details ▾

Was this information helpful? Yes No

OK



The G/L Account does not exist. Identification fields and values:
No.="

Share details ▾

Was this information helpful? Yes No

OK

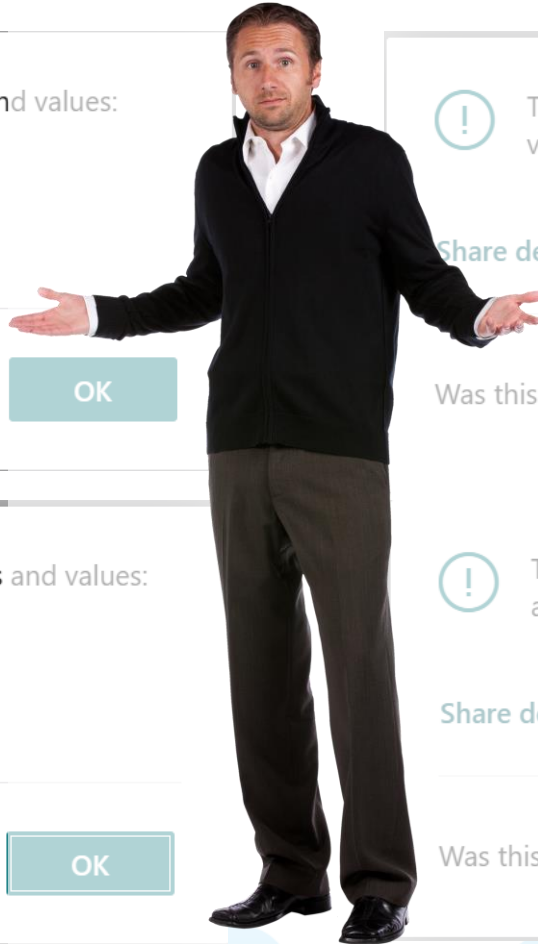


The Customer Posting Group does not exist. Identification fields and values:
Code="

Share details ▾

Was this information helpful? Yes No

OK



Give the user a reference

If you do not give the user a reference, how will they know what is wrong?

Using a testfield will make sure a value is there (or will make sure that the value is a specific value)

As a child, I was always taught:

“Always use a testfield, if you are dependent of the value”



Give the user a reference

Method: **testfield**

Tests that the content of the field is not zero or blank (empty string). If it is, an error message is displayed.





BUT ...

I use the testfield as a backup

Show multiple errors

← Error Messages ↗ ↘

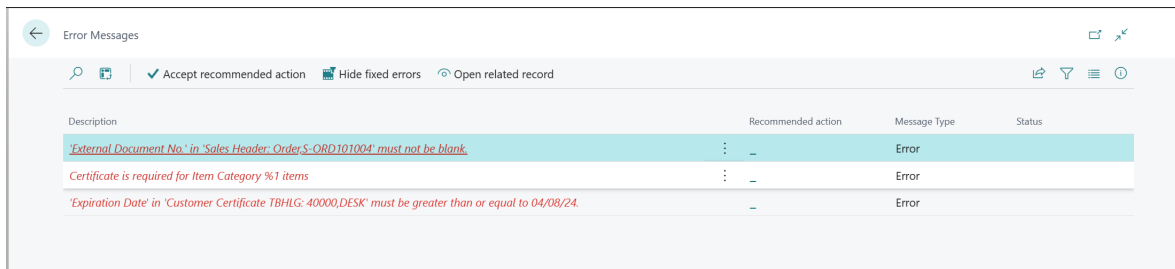
🔍 📄 Accept recommended action Hide fixed errors Open related record 🔗 🔍 ☰ ⓘ

Description	Recommended action	Message Type	Status
<i>'External Document No.' in 'Sales Header: Order,S-ORD101004' must not be blank.</i>	⋮ -	Error	
<i>Certificate is required for Item Category %1 items</i>	⋮ -	Error	
<i>'Expiration Date' in 'Customer Certificate TBHLG: 40000,DESK' must be greater than or equal to 04/08/24.</i>	-	Error	



Show multiple errors

Use the Error Message Table to collect errors, and present at the end.



Description	Recommended action	Message Type	Status
<i>'External Document No.' in 'Sales Header: Order S-ORD101004' must not be blank.</i>	—	Error	
<i>Certificate is required for Item Category %1 items</i>	—	Error	
<i>'Expiration Date' in 'Customer Certificate TBHLG: 40000,DESK' must be greater than or equal to 04/08/24.</i>	—	Error	



Show multiple errors

The Error Message table has several procedures (methods) you can call:

LogIfEmpty(RecRelatedVariant: Variant, FieldNumber: Integer, MessageType: Option): Integer

LogIfLessThan(RecRelatedVariant: Variant, FieldNumber: Integer, MessageType: Option, UpperBound: Variant): Integer

LogDetailedMessage(RecRelatedVariant: Variant, FieldNumber: Integer, MessageType: Option, NewDescription: Text, AdditionalInformation: Text[250], SupportUrl: Text[250]): Integer



Actionable Error Message

While there, sadly, isn't any example of this in the Base app, I wholeheartedly believe in the use of the Error Message Table (Error Management).

When I release a 'Document', I will run through the header and lines, and test for 'mandatory' fields (Those have been flagged as ShowMandatory).

I will also test if the affected record is 'blocked' or otherwise invalid.



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item		16,767.60	17,773.66	✖ 15			5

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

Use the Actionable Error Message

Using the Error Message Management Codeunit, you can call a codeunit to “fix it” for the user.

SalesLine.”Qty. to Ship” validation is a great example



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15		5	

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSaleShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15		5	

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSaleShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15			5

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSaleShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15			5

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSalesShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15		5	

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSalesShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

Post... Release Create Warehouse Shipment Create Inventory Put-away/Pick... Archive Document

✖ The page has an error. Refresh (F5) to undo the change, or correct the error.

Posting Date 4/2/2024 Status Open

Order Date 4/2/2024

Lines Manage Line Order

New Line Delete Line Select items... Suggest sales lines

Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice	Quantity
✖ Item	:	16,767.60	17,773.66	✖ 15			5

Qty. to Ship isn't valid
You cannot ship more than 12 units.
Set value to 12

Subtotal Excl. Tax (USD) 16,767.60 Total Excl. Tax (USD) 16,767.60

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSalesShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```



Actionable Error Message

← Sales Order ✎ 🔗 + 🗑️

S-ORD101001 · Adatum Corporation

[Home](#) [Prepare](#) [Print/Send](#) [Request Approval](#) [Order](#) [Report](#) | [Actions](#) [Related](#) [Automate](#) [Fewer opti](#)

[Post...](#) [Release](#) [Create Warehouse Shipment](#) [Create Inventory Put-away/Pick...](#) [Archive Document](#)

✖ The page has an error. [Refresh \(F5\)](#) to undo the change, or correct the error.

General

Customer Name	Adatum Corporation	Due Date	5/2/2024
Contact	Robert Townes	Requested Delivery Date	4/3/2024
Document Date	4/2/2024	External Document No.	
Posting Date	4/2/2024	Status	Open
Order Date	4/2/2024		

Lines | [Manage](#) Line Order

[New Line](#) [Delete Line](#) [Select items...](#) [Suggest sales lines](#)

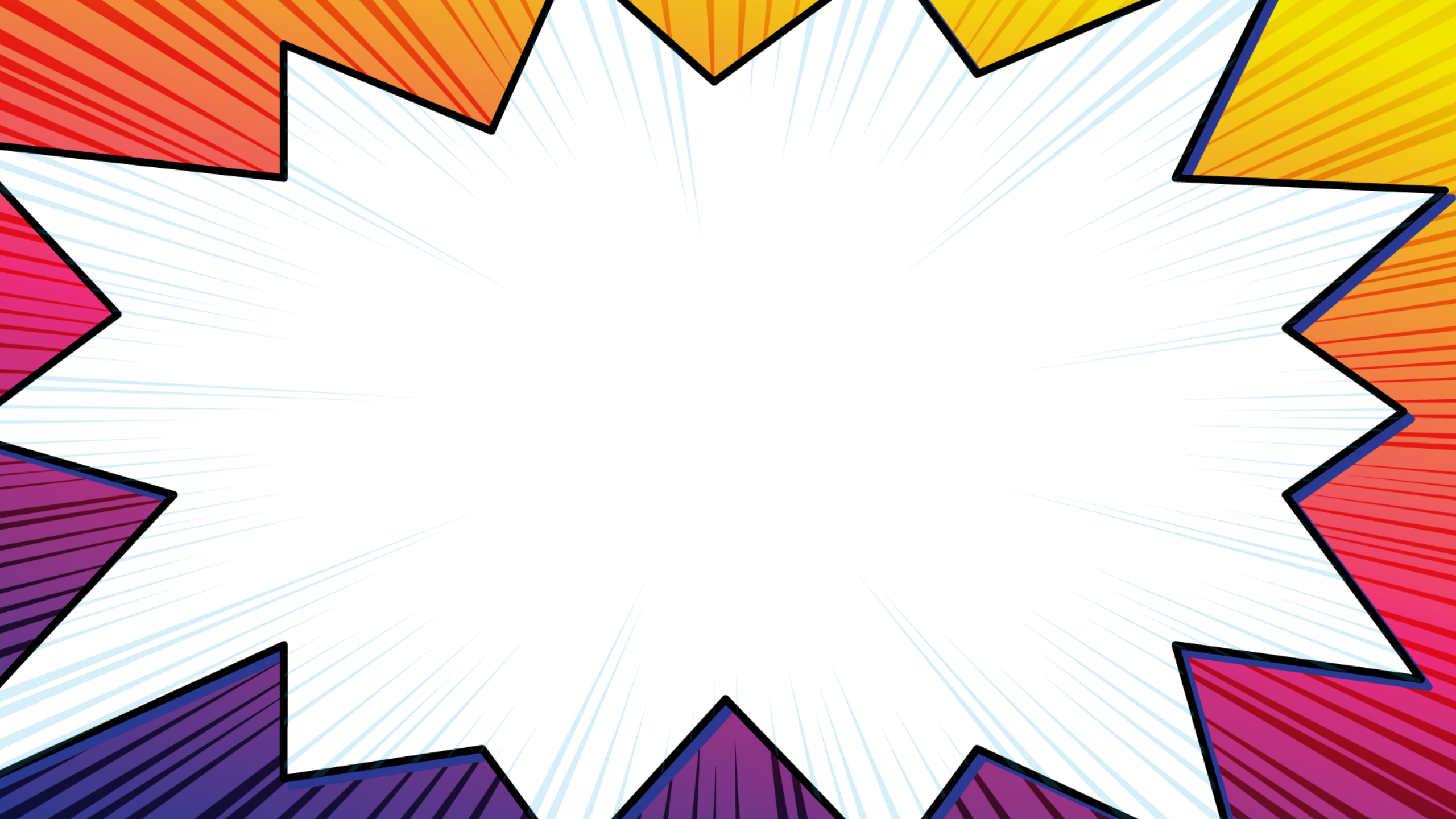
Type	Line Discount %	Line Amount Excl. Tax	Amount Including Tax	Qty. to Ship	Quantity Shipped	Qty. to Invoice
✖ Item	:	16,767.60	17,773.66	✖ 15		1

Qty. to Ship isn't valid
You cannot ship more than 12 units.
[Set value to 12](#)

Corrects Qty. to Ship to 12

```
local procedure CannotShipErrorInfo(): ErrorInfo
var
    ErrorMessageManagement: Codeunit "Error Message Management";
begin
    exit(ErrorMessageManagement.BuildActionableErrorInfo(
        QtyShipNotValidTitleLbl,
        StrSubstNo(Text007, "Outstanding Quantity"),
        Rec.RecordId,
        StrSubstNo(QtyShipActionLbl, "Outstanding Quantity"),
        Codeunit::"Sales Line-Reserve",
        'SetSaleShipQty',
        StrSubstNo(QtyShipActionDescriptionLbl,
            Rec.FieldCaption("Qty. to Ship"), Rec.Quantity));
end;
```





BuildActionableErrorInfo

```
procedure BuildActionableErrorInfo(ErrorTitle: Text; ErrorMessage: Text; RecId: RecordId; ActionMessage: Text;
ActionCodeunitId: Integer; ActionName: Text; ActionDescription: Text): ErrorInfo
var
    ReturnErrorInfo: ErrorInfo;
begin
    ReturnErrorInfo.Title := ErrorTitle;
    ReturnErrorInfo.Message := ErrorMessage;
    ReturnErrorInfo.RecordId := RecId;
    if ActionDescription <> " " then
        ReturnErrorInfo.AddAction(ActionMessage, ActionCodeunitId, ActionName, ActionDescription)
    else
        ReturnErrorInfo.AddAction(ActionMessage, ActionCodeunitId, ActionName);
    exit(ReturnErrorInfo);
end;
```





BUT WAIT.

There's more

BuildActionableErrorInfo

```
procedure BuildActionableErrorInfo(ErrorTitle: Text; ErrorMessage: Text; RecId: RecordId; ActionMessage: Text;
ActionCodeunitId: Integer; ActionName: Text; ActionDescription: Text): ErrorInfo
var
    ReturnErrorInfo: ErrorInfo;
begin
    ReturnErrorInfo.Title := ErrorTitle;
    ReturnErrorInfo.Message := ErrorMessage;
    ReturnErrorInfo.RecordId := RecId;
    if ActionDescription <> " " then
        ReturnErrorInfo.AddAction(ActionMessage, ActionCodeunitId, ActionName, ActionDescription)
    else
        ReturnErrorInfo.AddAction(ActionMessage, ActionCodeunitId, ActionName);
    exit(ReturnErrorInfo);
end;
```



Add a Second Action!

```
ErrorDialogWithTwoActions.Title := 'The line dimension value isn't valid';
```

```
    ErrorDialogWithTwoActions.Message := StrSubstNo('The dimension value must be blank for the dimension %1 for Vendor %2', DimensionCode,  
VendorCode);
```

```
    ErrorDialogWithTwoActions.DetailedMessage('Add some text to help the person troubleshooting this error.');
```

```
    ErrorDialogWithTwoActions.AddAction(  
        'Set value to blank',  
        Codeunit::FirstFixitCodeunit,  
        "FirstFixitCodeunitMethodName"  
    );
```

```
    ErrorDialogWithTwoActions.AddAction(  
        'OK',  
        Codeunit::SecondFixitCodeunit,  
        "SecondFixitCodeunitMethodName"  
    );
```

```
    EndOf(ErrorDialogWithTwoActions);
```



Actionable errors References

<https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-error-handling-guidelines>

<https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-actionable-errors>

<https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-actionable-errors#error-messages-with-multiple-actions---how-to-use-them>



10000 · Adatum Corporation

Home Request Approval New Document Prices & Discounts Customer Report Actions Related Reports Automate Fewer options

Contact Apply Template Merge With... Send Email

General

Show less

No.	<input type="text" value="10000"/>	Salesperson Code	<input type="text" value="JO"/>
Name	<input type="text" value="Adatum Corporation"/>	Responsibility Center	<input type="text"/>
Name 2	<input type="text" value="Department or extended Business Nam"/>	Document Sending Profile ..	<input type="text"/>
IC Partner Code	<input type="text"/>	Total Sales - Fiscal Year	60,672.80
Balance (\$)	0.00	Costs (\$)	40,255.70
Balance (\$) As Vendor	0.00	Profit (\$)	20,417.10
Balance Due (\$)	0.00	Profit %	33.7
Credit Limit (\$)	0.00	Last Date Modified	5/12/2024
Blocked	<input type="text"/>	Disable Search by Name	<input checked="" type="checkbox"/>
Privacy Blocked	<input type="checkbox"/>		

Address & Contact

Show more

Address		Phone No.	<input type="text"/>
Address	<input type="text" value="192 Market Square"/>	Mobile Phone No.	<input type="text"/>
Address 2	<input type="text" value="Suite / Apartment No."/>	Email	<input type="text" value="robert.townes@contoso.com"/>
Country/Region Code	<input type="text" value="US"/>	Home Page	<input type="text"/>

Details Attachments (0)

Customer Picture



Sell-to Customer Sales History

Customer No. 10000

0	0	7
Ongoing Sales Quotes	Ongoing Sales Blanket Orders	Ongoing Sales Orders
2	0	0
Ongoing Sales Invoices	Ongoing Sales Return Orders	Ongoing Sales Credit Memos
33	33	0
Posted Sales	Posted Sales	Posted Sales

es

many other
w create a
that is shown in



Give users a hint for field values

Property: **InstructionalText**

Sets the string used for instructions in the UI.

The default is an empty string, which means there are no instructions. According to the user assistance model for Business Central, apps are expected to apply instructional text, also called placeholder text, to setup guides and similar pages.

Note

The InstructionalText property can be applied to Text, BigText, Guid, and Code data type fields.



Follow Standard Nomenclature



When naming fields, make sure to use nomenclature the user will know.

Avoid:

- ID – i.e. Customer ID
- Number
- Synonyms – i.e. Use Customer, not Client. Item, not Material.



Follow Standard Nomenclature



The same rule pertains to Actions. Adding Groups for your app – especially if they contain actions that ‘fit’ in the existing groups, should be avoided.

Adding a arbitrary App Identifier to your actions, will ‘Scream’ “I am part of a separate app” – just like if You put a yellow driver side door onto your red car.

A user should NOT be able to tell ‘Standard’ and ‘App’ apart.



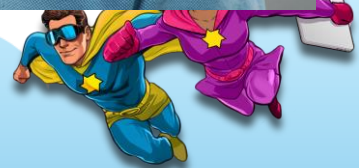
Follow Standard Nomenclature

Master Tables

- No. – Code[20]
- Has Number Series

Support tables:

- Code - Code[10]
- Description Text[50]

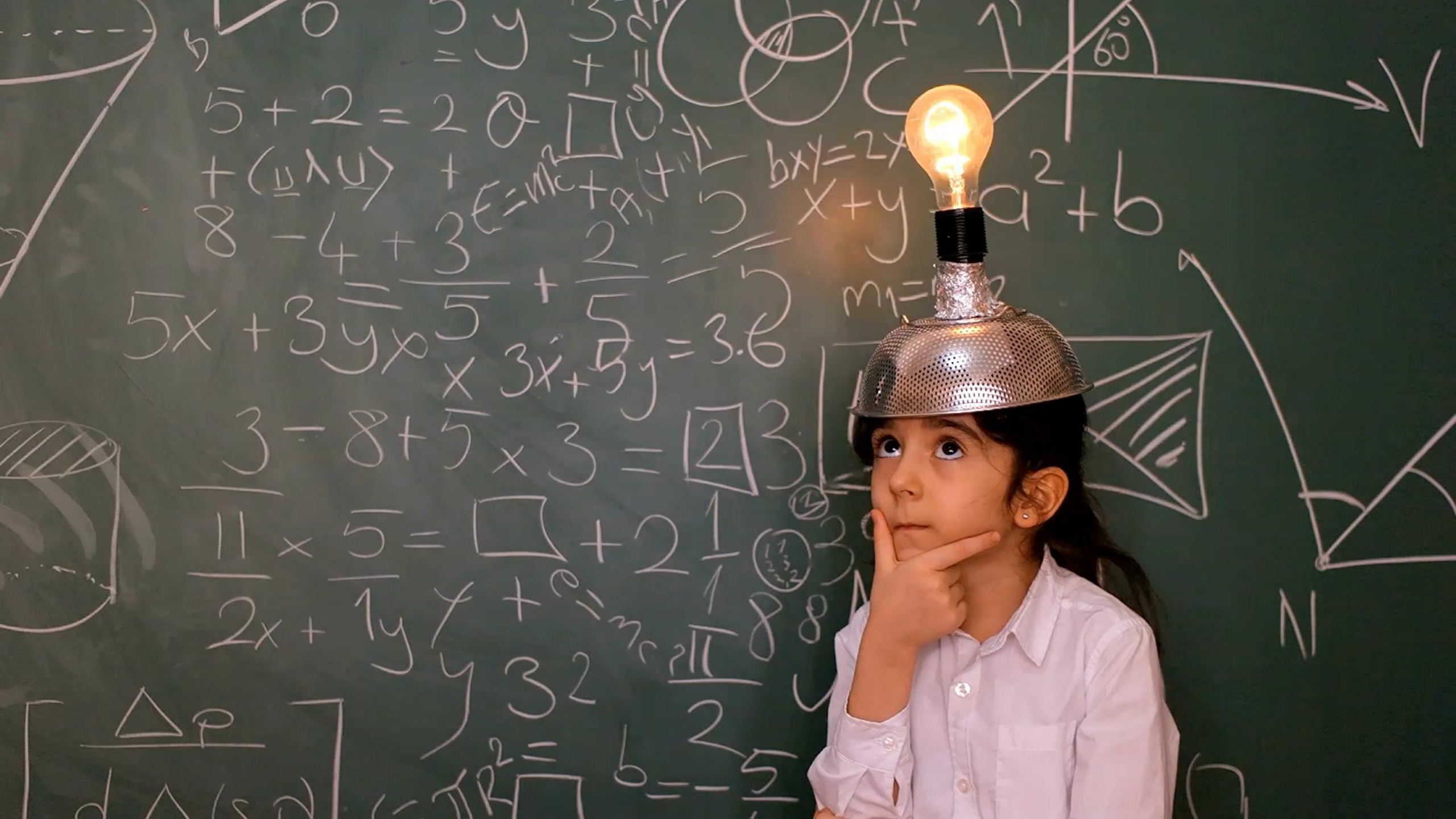


Follow Standard Nomenclature

Visit

<https://alguidelines.dev>

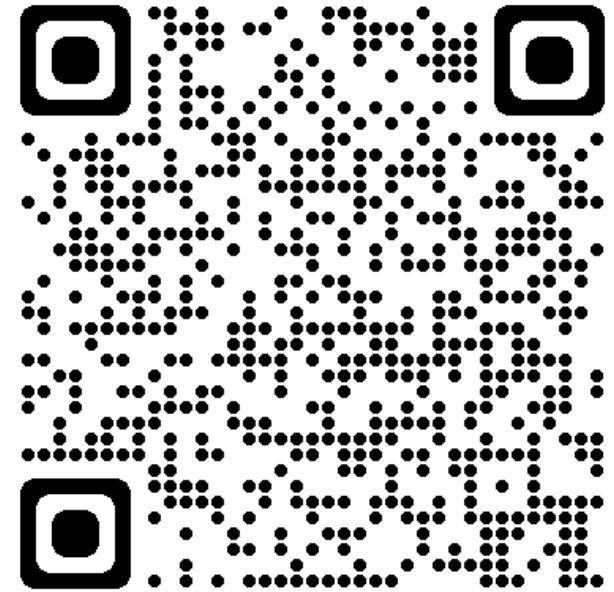




Acknowledgments

Most code examples shown here, are retrieved, thanks to Stefan Maron and his MSDyn365BC.Code.History repository on GitHub

<https://github.com/StefanMaron/MSDyn365BC.Code.History>



Questions?

- Twitter: @TheDoubleH
- Email: henrik@helgesenconsulting.com
- Blog: <https://TheDoubleH.dev>
- Github: <https://github.com/TheDoubleH>



Thank You

