



COMMUNITY SUMMIT

DYNAMICS 365 BUSINESS CENTRAL (NAV)

Orlando, FL
October 19–23, 2025



The largest Microsoft Business Applications user conference on the planet.



AL Bootcamp: From Zero to Hero in Two Days

Day 1

A Dynamics-First, 'For User, By User' Event.

@ 2025 Dynamic Communities

| Speakers



Brad Prendergast



AJ Ansari



Brad Prendergast

- 20+ years in Microsoft Dynamics, specializing in Business Central
- Unique dual perspective as a partner & end user
- Member of Summit Programming Committee & Board of Advisors
- Co-host of the popular Dynamics Corner podcast
- Passionate about bringing people together in the Dynamics space



@ 2025 Dynamic Communities



AJ Ansari

- Community Summit Legend and BCUG All-Star
- 17 years in Microsoft Dynamics; background as certified developer, consultant and practice leader
- Member of Summit Programming Committee & Board of Advisors



Only BC Fans



Only Copilot Fans



COO & Partner

aja@dswius.com

bit.ly/AJAnsari



@ 2025 Dynamic Communities

Know Before You Go

- Restrooms are where?
- Nearest Emergency Exit is where?



| Session Objectives



- Setup a Development Environment
- Understand Business Central Architecture
- Create an extension in Business Central
- Extend Business Central



Agenda

- Day 1
- Introduction
- Getting Started
- Setting up your Development Environment
- AL Language
- Develop Your First Extension
- Creating an Extension
- Extending Business Central

- Day 2



| About You

- Name
- Company
- Role
- What are you hoping to learn?

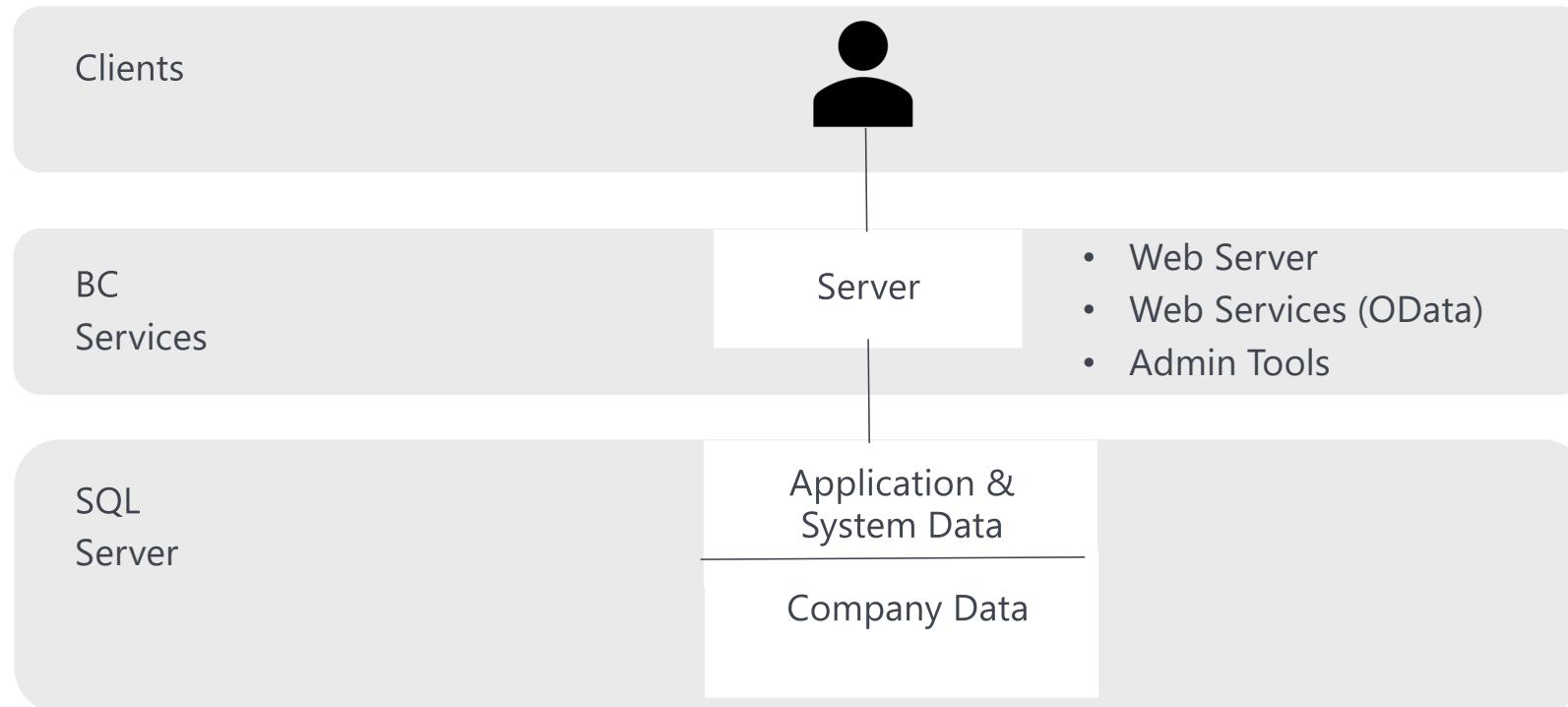


| Getting Started



Business Central Architecture

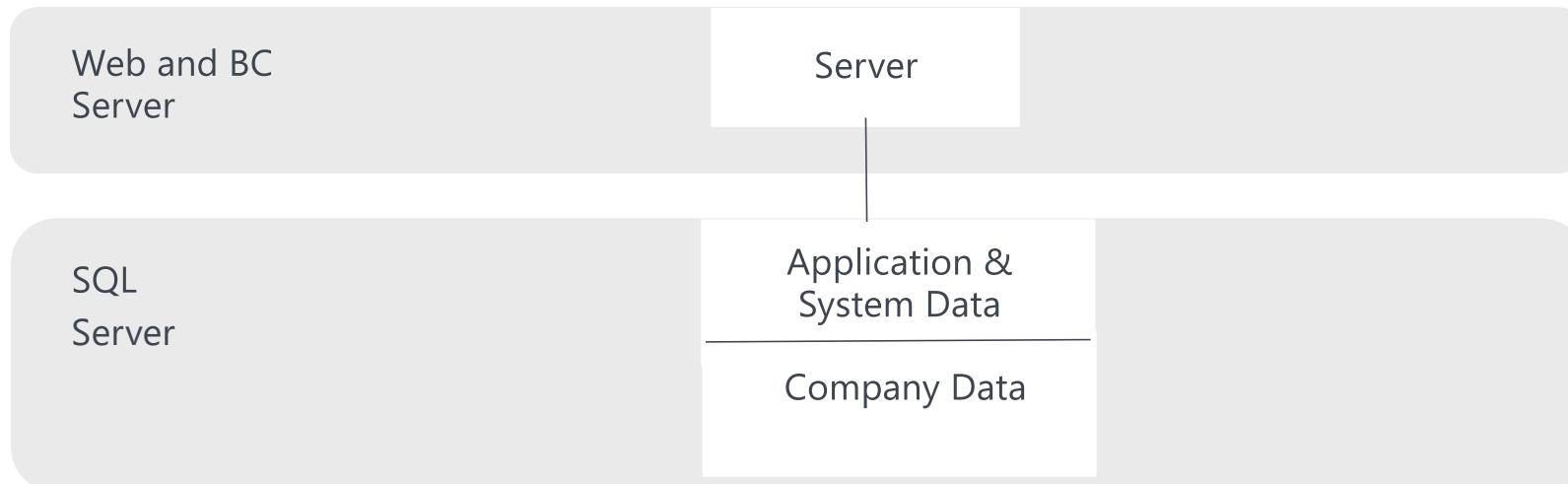
Single-Tenant or On-premises





Business Central Architecture

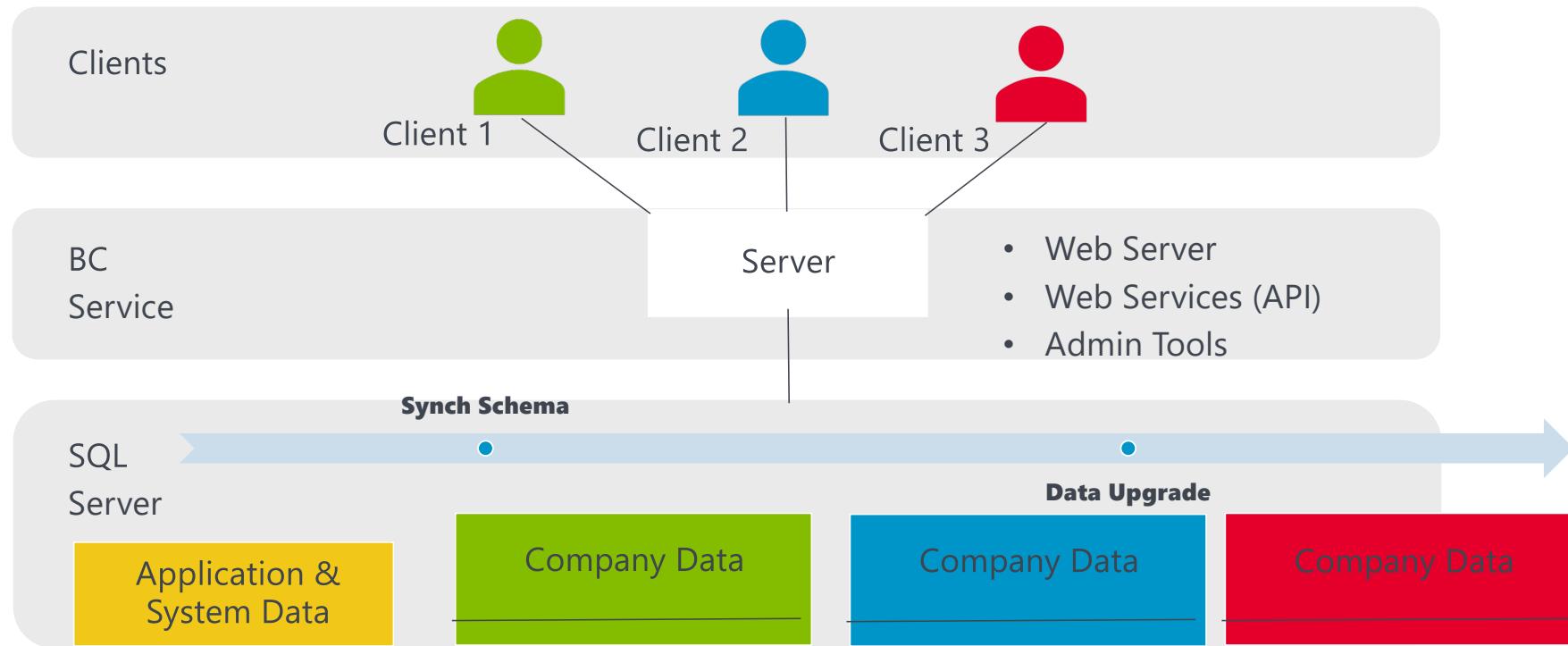
Recommended Architecture





Business Central Architecture

Multitenant Architecture





Business Central Architecture

The screenshot shows the Microsoft Dynamics 365 Business Central interface with several areas highlighted by red boxes and arrows:

- Navigation Menu:** Located at the top, showing "CRONUS USA, Inc." and categories like Finance, Cash Management, Sales, Purchasing, and Shopify.
- List Area:** A red box surrounds the navigation bar with links to Customers, Vendors, Items, Bank Accounts, and Chart of Accounts.
- Headline Area:** A red box surrounds a callout box containing the text "Want to learn more about Business Central?"
- Action Area:** A red box surrounds a list of actions including "+ Sales Quote", "+ Purchase Quote", "+ Sales Order", "+ Purchase Order", "+ Sales Invoice", "+ Purchase Invoice", "Find entries...", "Search in data...", "New", "Payments", "Reports", and "Excel Reports".
- Get started:** A red box surrounds a message "Get started: Here are a few things you can try out" with a "Show demo tours" button.
- Cues and Action Tiles:** A red box surrounds a section titled "Activities" with four cards: "Sales This Month" (\$1,906), "Overdue Sales Invoice Amount" (\$63,890), "Overdue Purch. Invoice Amount" (\$49,422), and "Sales Invoices Predicted Overdue" (0). Below this are sections for "Ongoing Sales" (with cards for Sales Quotes, Sales Orders, and Sales Invoices) and "Ongoing Purchases" (with cards for Purchase Orders, Ongoing Purch. Invoices, and Purch. Invoices Next Week).



What is the AL Language?

- AL is the program language used for Dynamics 365 Business Central; Extension Based Approach
- Origin from C/AL (Client/Server Application Language) for extending Dynamics NAV (along side C/SIDE)
- VS Code Integration
- Transpiler - source-to-source compiler - "translator" and "compiler."
- <https://www.dynamicscorner.com/2023/06/episode-220-in-the-dynamics-corner-chair-the-al-compiler-in-depth/>





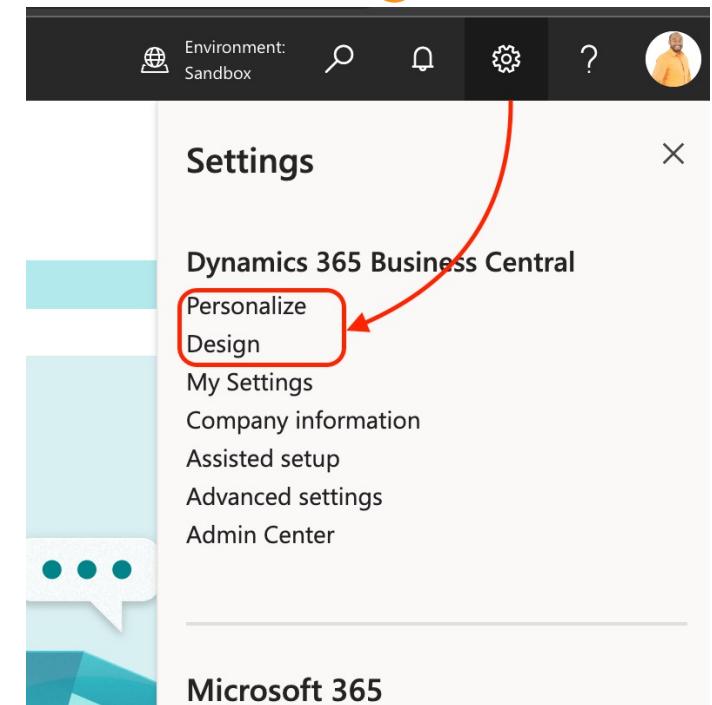
Personalize vs. Design vs. Develop

Personalize

- User Specific
- Limited
- Only Fields Available to Page

Design

- Global
- Creates new Extension as .zip
- Only from Sandbox



Personalize vs. Design vs. Develop



Existing Objects

- Tables, Pages, Reports, Enums

Create new objects of any kind

- Page Type: API

BC Online

- Cannot change base objects

BC On-Premises

- Base Objects **CAN** be modified

<https://bit.ly/d365bcuci>



Extension Types

- Global
 - Installed from the AppSource
 - Production and Sandbox
 - Preserved on upgrade for both Production and Sandbox
- Per-tenant (PTE)
 - Installed through Extension Management
 - Specific to environment
 - Production and Sandbox
 - Preserved on upgrade in Production (unless there is a problem)
 - Uninstalled from Sandbox when relocated if dependent on DEV



Extension Types

- DEV
 - Published from VS Code
 - Used for development purposes
 - Only exist in Sandbox
 - uninstalled when the sandbox environment is upgraded or relocated (data is not removed)

Version	Is Inst...	Published As
v. 20.17.34.0	<input checked="" type="checkbox"/>	Dev
v. 22.4.18.0	<input checked="" type="checkbox"/>	PTE
v. 22.5.59966.60187	<input checked="" type="checkbox"/>	Global
v. 22.5.59966.60187	<input checked="" type="checkbox"/>	Global
v. 22.5.59966.60187	<input checked="" type="checkbox"/>	Global

Extension Management



Installed Extensions

Search Manage Automate Fewer options

Upload And Deploy Extension

Manage

Install Uninstall Unpublish Set up Download Source Learn More Select More

Tigunia Royalties Management	Tigunia, LLC	v. 20.17.34.0	<input checked="" type="checkbox"/>	Dev
Data Archive	Microsoft	v. 22.5.59966.60187	<input checked="" type="checkbox"/>	Global
Data Search	Microsoft	v. 22.5.59966.60187	<input checked="" type="checkbox"/>	Global

... Royalties Management ...

Deploy Cancel



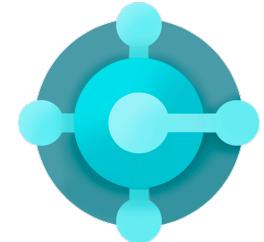


| Setup Development Environment

| Local Environment

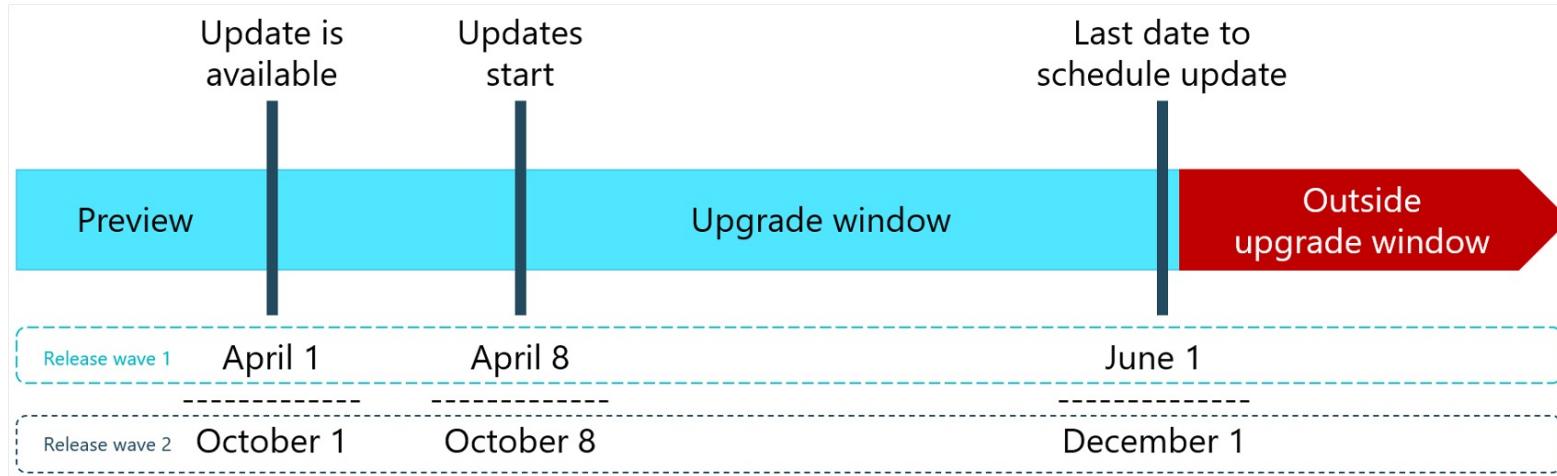


- Visual Studio Code (<https://code.visualstudio.com/Download>)
- AL Language Extension
- Other Extensions
 - AZ AL Dev Tools/AL Code Outline
 - Waldo's CRS AL Language Extension
 - Vscode-icons
- Code Spell Checker
- Create GUID
- Report Builder (<https://www.microsoft.com/en-us/download/details.aspx?id=53613>)





Business Central Tenant



**Update Window is still
currently at 60 Days**



Business Central Tenant

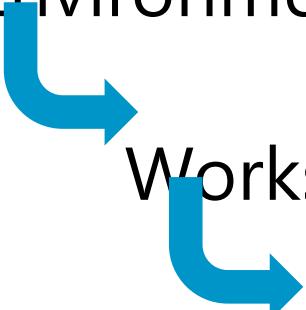
- Admin Center
- Create an Environment
 - Environment Types
 - Copy an environment
- Managing Updates
 - Set Update Window
 - Schedule an Update date
 - Get notified of Updates





Extension Settings

Environment
Workspace
Project



```
{  
  "al.incognito": true,  
  "al.browser": "Chrome",  
  "al.enableCodeAnalysis": true,  
  "al.backgroundCodeAnalysis": false,  
  "al.incrementalBuild": false,  
  "al.codeAnalyzers": [  
    "${AppSourceCop}",  
    "${CodeCop}",  
    "${PerTenantExtensionCop}",  
    "${UICop}"  
  ],  
  "al.compilationOptions": {  
    "generateReportLayout": true  
  },  
  "alOutline.completionProviders": [  
    "VariableNamesWithType",  
    "VariableDataTypes"  
  ],  
  "debug.inlineValues": "on",  
  "debug.console.fontFamily": "default",  
  "debug.console.fontSize": 14,  
}
```

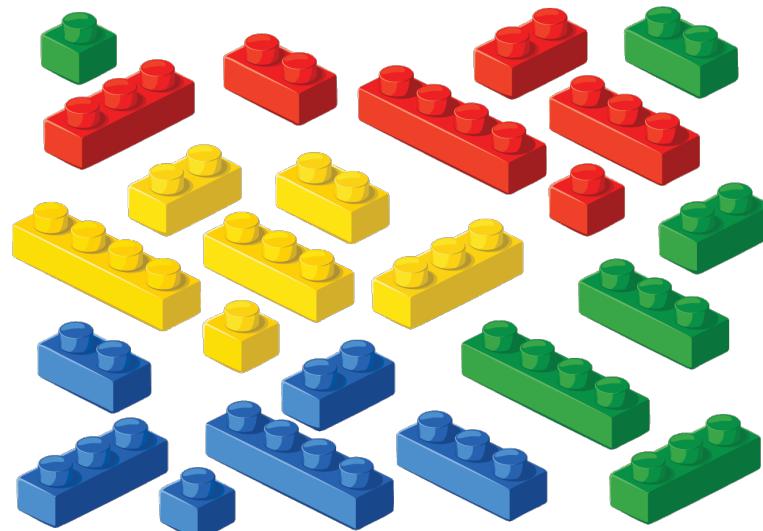




| AL Language

AL Language

- Objects
 - Table
 - Page
 - Reports
 - Codeunits
 - XMLPorts
 - Query
 - Permissionset





AL Language - Table

- Tables are the core objects used to store data in Dynamics 365 Business Central
- The structure of a table has four sections:
 - The first block contains metadata for the overall table, such as the table type.
 - The fields section describes the data elements that make up the table, such as their name and the type of data they can store.
 - The keys section contains the definitions of the keys that the table needs to support.
 - The final section details the triggers and code that can run on the table.

```
table 50104 Address
{
    Caption = 'Sample table';
    DataPerCompany = true;

    fields
    {
        field(1; Address; Text[50])
        {
            Description = 'Address retrieved by Service';
        }
        field(2; Locality; Text[30])
        {
            Description = 'Locality retrieved by Service';
        }
        field(3; "Town/City"; Text[30])
        {
            Description = 'Town/City retrieved by Service';
        }
        field(4; County; Text[30])
        {
            Description = 'County retrieved by Service';

            trigger OnValidate();
            begin
                ValidateCounty(County);
            end;
        }
    }
    keys
    {
        key(PrimaryKey; Address)
        {
            Clustered = TRUE;
        }
    }

    var
        Msg: Label 'Hello from my method';

    trigger OnInsert();
    begin
    end;

    procedure MyMethod();
    begin
        Message(Msg);
    end;
}
```



AL Language - Page

- Pages are the main way to display and organize visual data in Dynamics 365 Business Central.
- The structure of a table has four sections:
 - The first block contains metadata for the overall page; the type of the page and the source table it is showing data from.
 - The next section; the layout, describes the visual parts on the page.
 - The final section details the actions that are published on the page.

```
page 50101 SimpleCustomerCard
{
    PageType = Card;
    SourceTable = Customer;
    ContextSensitiveHelpPage = 'my-feature';
}

layout
{
    area(content)
    {
        group(General)
        {
            field("No."; "No.")
            {
                ApplicationArea = All;
                CaptionML = ENU = 'Hello';

                trigger OnValidate()
                begin
                    if "No." < '' then
                        Message('Number too small');
                end;
            }
            field(Name; Name)
            {
                ApplicationArea = All;
            }
            field(Address; Address)
            {
                ApplicationArea = All;
            }
        }
    }
}

actions
{
    area(Navigation)
    {
        action(NewAction)
        {
            ApplicationArea = All;
            RunObject = codeunit "Document Totals";
        }
    }
}
```

AL Language - Report

- Reports are used to print or display information*
- Reports consist of a data model and layout
- A report object consists of four sections
 - The first block contains metadata for the overall table
 - The second block is the dataset that defines the data model
 - The third section is the request page for accepting user input
 - The final section details the triggers and code that can run on the report

* processing reports; Excel download; Excel Layouts



```
trigger OnAfterGetRecord();
begin
    CalcFields("Balance (LCY)");
    FormatAddr.FormatAddr(
        CustAddr,"Name 2",',',Address,"Address 2",
        City,"Post Code",County,"Country/Region Code");
end;
}

requestpage
{
    SaveValues = true;
    // These properties can be set on the report
    AboutTitle = 'Awesome app';
    AboutText = 'This is a great app';
    // Use the multi-language localization
    // This property defines the language
    // Remember to also set ContextSensitiveHelpPage
    layout
    {
    }
    actions
    {
    }
    labels
    {
        LabelName = 'Label Test';
    }
    trigger OnPreReport();
    var
        CaptionManagement : Codeunit;
    begin
        CustFilter := CaptionManagement.GetCustomerFilter();
    end;
}

report 50103 "Customer List"
{
    CaptionML=ENU='Customer List';
    DefaultLayout = RDLC; // if Word use WordLayout property
    RDCLayout = 'MyRDLCReport.rdl';
}

dataset
{
    dataitem(Customer;Customer)
    {
        RequestFilterFields="No.", "Search Name", "Customer Posting Group";
        column(CompanyName;CompanyName)
        {
        }
        column(CurrReport_PageNo;Customer."No.")
        {
        }
        column(Customer_TableCaption_CustFilter;TableCaption + ': ' + CustFilter)
        {
        }
        column(CustFilter;CustFilter)
        {
        }
        column(Customer_No;"No.")
        {
        }
        column(Customer_Customer_Posting_Group;"Customer Posting Group")
        {
        }
        column(Customer_Customer_Disc_Group;"Customer Disc. Group")
        {
        }
        column(Customer_Invoice_Disc_Code;"Invoice Disc. Code")
        {
        }
        column(Customer_Customer_Price_Group;"Customer Price Group")
        {
        }
        column(Customer_Fin_Charge_Terms_Code;"Fin. Charge Terms Code")
        {
        }
        column(Customer_Payment_Terms_Code;"Payment Terms Code")
        {
        }
    }
}
```

AL Language - Codeunit



- Modularized container for AL Code that contains business logic.
- Procedures, Variables, Event Subscribers
- The Access property defines the scope of the codeunit - public or internal.

```
codeunit 50100 MyCodeunit
{
    Access = Public;
    Subtype = Normal;

    trigger OnRun()
    begin
    end;

    procedure MyFunction(Param1: Integer; Param2: Text[50]) : Boolean
    begin
    end;
}
```

| AL Language - XMLPort



- Export and Import data between an external source and Dynamics 365 Business Central
- XML Document
- Variable or Fixed Text Format
- Direction – Inbound, Outbound





AL Language - Query

- Retrieve records from one or more tables and then combine the data into rows and columns in a single dataset
- Two Types - Normal and API
- dataitems and columns
- SqlJoinType Property

```
query ID Name
{
    elements
    {
        dataitem(DataItem1; Table1)
        {
            column(Column1; Field1)
            {
            }
            column(Column2; Field2)
            {
            }
            dataitem(DataItem2; Table2)
            {
                // Sets a link between FieldY of Table2 and FieldX of Table1.
                DataItemLink = FieldY = DataItem1.FieldX;
                //The dataset contains records from Table1 and Table2 where a match is found between
                SqlJoinType = InnerJoin;

                column(Column1; Field1)
                {
                }
                dataitem(DataItem3; Table3)
                {
                    DataItemLink = FieldZ = DataItem2.FieldY;
                    SqlJoinType = InnerJoin;
                    column(Column1; Field1)
                    {
                    }
                }
            }
        }
    }
}
```

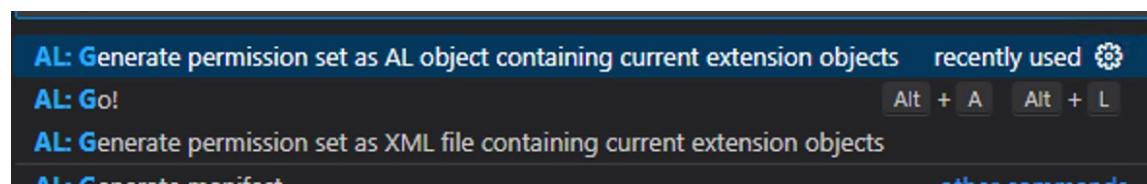
AL Language – PermissionSet



- Describes permissions on objects
- Assignable to users in Business Central
- Extended permission sets are additive

```
permissionset 50134 "Sales Person"
{
    Assignable = true;
    Caption = 'Sales Person';

    Permissions =
        tabledata Customer = RIMD,
        tabledata "Payment Terms" = RMD,
        tabledata Currency = RM,
        tabledata "Sales Header" = RIM,
        tabledata "Sales Line" = RIMD;
}
```



Page Inspector

- insight into the page design
- different page elements
- source behind the data

Ctrl + Alt + F1.



The screenshot shows the 'Page Inspection' tool window in a software application. At the top, it says 'Page Inspection' and 'Customer List (22, List)'. Below that is a link 'Explore page in Visual Studio Code'. The main area is titled 'Table' and shows 'Customer (18)'. There is a 'View table' button. At the bottom, there are tabs for 'Table Fields', 'Extensions', and 'Page Filters', along with a search icon. A detailed list of table fields is shown:

No. (1, Code[20], PK)	10000	...
Name (2, Text[100])	Adatum Corporation	...
Search Name (3, Code[100])	ADATUM CORPORATION	...
Name 2 (4, Text[50])	(Blank)	...
Address (5, Text[100])		



All Objects



All Objects with Caption

Object Type	Object ID	Object Name	Object Caption	Object Subtype	App Name
TableData	3	Payment Terms	Payment Terms	Normal	Base Application
TableData	4	Currency	Currency	Normal	Base Application
TableData	5	Finance Charge ...	Finance Charge Terms	Normal	Base Application
TableData	6	Customer Price ...	Customer Price Group	Normal	Base Application
TableData	7	Standard Text	Standard Text	Normal	Base Application
TableData	8	Language	Language	Normal	System Application
TableData	9	Country/Region	Country/Region	Normal	Base Application
TableData	10	Shipment Method	Shipment Method	Normal	Base Application
TableData	11	Country/Region ...	Country/Region Translation	Normal	Base Application
TableData	13	Salesperson/Pur...	Salesperson/Purchaser	Normal	Base Application
TableData	14	Location	Location	Normal	Base Application
TableData	15	G/L Account	G/L Account	Normal	Base Application
TableData	17	G/L Entry	G/L Entry	Normal	Base Application
TableData	18	Customer	Customer	Normal	Base Application
TableData	19	Cust. Invoice Disc.	Cust. Invoice Disc.	Normal	Base Application
TableData	21	Cust. Ledger Entry	Cust. Ledger Entry	Normal	Base Application
TableData	23	Vendor	Vendor	Normal	Base Application
TableData	24	Vendor Invoice ...	Vendor Invoice Disc.	Normal	Base Application
TableData	25	Vendor Ledger E...	Vendor Ledger Entry	Normal	Base Application
TableData	27	Item	Item	Normal	Base Application



AL Explorer



AL Explorer X

AL Explorer

Project: SummitNA 2025 Dev Sample (SummitNA...)

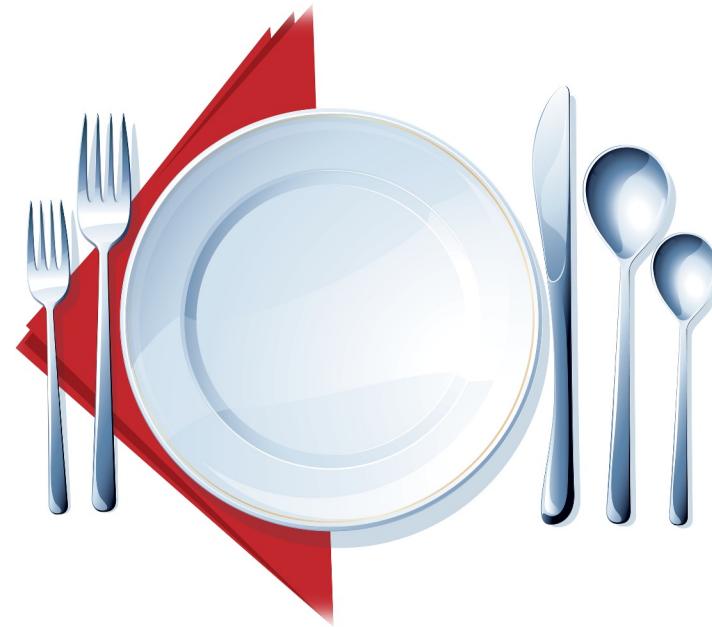
OBJECTS EVENTS APIS EXTENSIBLE ENUMS

Search: Type to filter Group by: Type Module: All

ID	Name	Subtype	Related Table	Module	Namespace
3	Payment Terms		Payment Terms	Base Application	Microsoft.Foundation.Pay...
4	Currency		Currency	Base Application	Microsoft.Finance.Currency
5	Finance Charge Terms		Finance Charge Terms	Base Application	Microsoft.Sales.FinanceCh...
6	Customer Price Group		Customer Price Group	Base Application	Microsoft.Sales.Pricing
7	Standard Text		Standard Text	Base Application	Microsoft.Utilities
8	Language		Language	System Application	System.Globalization
9	Country/Region		Country/Region	Base Application	Microsoft.Foundation.Add...
10	Shipment Method		Shipment Method	Base Application	Microsoft.Foundation.Ship...
11	Country/Region Translation		Country/Region Translation	Base Application	Microsoft.Foundation.Add...
13	Salesperson/Purchaser		Salesperson/Purchaser	Base Application	Microsoft.CRM.Team
14	Location		Location	Base Application	Microsoft.Inventory.Locati...
15	G/L Account		G/L Account	Base Application	Microsoft.Finance.General...

Source Run

| Lunch ?!?





| Develop your first App

| Develop your First App



- AL Go!
- Download Symbols
- Publish
- Project Files
- AL Home
- AL Explorer
- Source Code Analyzers



Steps to Add PTE Cop

- Step 1: In VS CODE, **press Ctrl + Shift + P**
- Step 2: In the box that appears, type WORKSPACE SETTINGS or USER SETTINGS (as needed) after the > prompt
- Select the result that reads “[Preferences: Open User Settings \(JSON\)](#)” or “[Preferences: Open Workspace Settings \(JSON\)](#)”
 - Use [User Settings](#) to set default for all projects
 - Use [Workspace Settings](#) to define project by project
- Step 3: At the top right of your VS CODE window, you will see a {} icon (two curly brackets); click that. A window will open to the side
- Step 4: If there are any lines of code / settings that appear in the window, find the last line before the } line, add a comma, and insert a new line
- Step 5: Add the following:

```
    "al.codeAnalyzers": [
        "${PerTenantExtensionCop}"
```

What mine looks like:

A screenshot of a VS Code settings.json file. The file content is as follows:

```
settings.json - NewTablesAndPage - Visual Studio Code
{
  "git.enableSmartCommit": true,
  "git.confirmSync": false,
  "team.showWelcomeMessage": false,
  "git.autofetch": true,
  "al.codeAnalyzers": [
    "${PerTenantExtensionCop}"
]
```

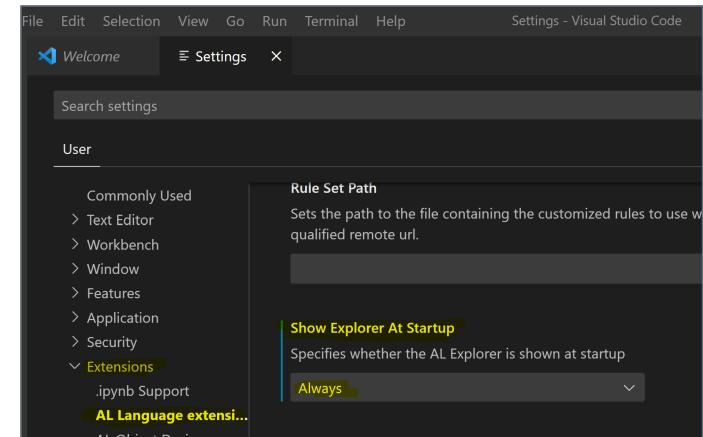
A blue oval highlights the line "al.codeAnalyzers": [and the line "\${PerTenantExtensionCop}" below it. A blue arrow points from the text "Step 5: Add the following:" in the previous slide to the highlighted area.



Steps to Enable AL Explorer

- Step 1: In VS CODE, **press Ctrl + Shift + P**
- Step 2: In the box that appears, type OPEN SETTINGS after the > prompt
- Select the result that reads “[Preferences: Open User Settings](#)” or “[Preferences: Open Settings \(UI\)](#)”
 - Use [Open User Settings](#) to set default for all projects
 - Use [Open Settings \(UI\)](#) to define by workspace
- Step 3: **Navigate to Extensions > AL Language extension**
- Step 4: In the right-hand pane, scroll down to **Show Explorer at Startup**
- Step 5: Set desired value: Always, Once (Default), Never

What mine looks like:





| Development

| Repo Structure





Development - Tables

```
0 references
table id MyTable
{
    <property> = <value>

    fields
    {
        0 references
        field(1; MyField; Integer)
        {
            <property> = <value>
        }
    }

    keys
    {
        key(Key1; MyField)
        {
            Clustered = true;
        }
    }

    fieldgroups
    {
        fieldgroup(Name; Fields)
        {
        }
    }

    var
    myInt: Integer;

    trigger <OnInsert()> <OnModify()> <OnDelete()> <OnRename()>
    begin
    end;

    <local> procedure <method_name>(parameter list) <return_value_name> : <data_type>[<length>]
    begin
    end;
}
```

Development

- Assignment Statements
- Control Statements
- Repetitive Statements

```
procedure assignments()
var
  item: Record Item;
  itemNo: Code[20];
  price: Decimal;
  counter: Integer;
  i: Integer;
  color: Text;

begin
  itemNo := '1000';
  item.Get(itemNo);
  Counter := 1;

  if counter > 1 then begin
    price := item."Unit Price" * Counter;
  end else begin
    price := item."Unit Price";
  end;

  for i := 1 to 10 do begin
    counter += 1;
  end;

  while counter > 0 do begin
    counter -= 1;
  end;

  repeat
    counter += 1;
  until counter >= 10;

  case color of
    'blue':
      Message('blue');
    'black':
      Message('black');
    'green':
      Message('green');
    else
      Message('no color');
  end;
end;
```



Development - Pages



```
references
page Id MyPage
{
    PageType = Card;
    ApplicationArea = All;
    UsageCategory = Administration;
    SourceTable = TableName;

    layout
    {
        0 references
        area(Content)
        {
            0 references
            group(GroupName)
            {
                0 references
                field(Name; NameSource)
                {
                    ApplicationArea = All;
                }
            }
        }
    }

    actions
    {
        0 references
        area(Processing)
        {
            0 references
            action(ActionName)
            {
                ApplicationArea = All;
                0 references
                trigger OnAction()
                begin
                end;
            }
        }
    }

    var
        0 references
        myInt: Integer;
    }
}
```

A screenshot of the Microsoft Dynamics 365 Customer Card interface for Adatum Corporation. The card displays general information such as No. (10000), Name (Adatum Corporation), Balance (\$), and Total Sales - Fiscal Year (60,672.80). It also shows address and contact details like Address (192 Market Square), City (Atlanta), and Contact (Robert Townes). The interface includes tabs for Home, Request Approval, New Document, Prices & Discounts, Customer, Report, and More options.

Development - Pages



Page type	Examples of use	Main data display
RoleCenter	Overview of business performance and the start page for a specific user profile.	Defined by the embedded parts.
Card	Master, reference, and set up data management. Card page example	Single entity
Document	Transaction and other document management.	Single entity
ListPlus	Statistics, details, and related data management.	Single entity
List	Entity overviews and navigation, and inline editing of simple entities. List page example	Collection of entities/entries
Worksheet	Line-based data entry tasks (such as journals) and inquiries.	Collection of entities
CardPart	A page that is embedded in another page, such as in a FactBox.	Single entity
ListPart	A page that is embedded in another page, such as in a FactBox.	Collection of entities/entries





Exercise 1

The Contoso Training and Events (CTE) company would like the ability to capture Instructor profile. They need to be able to capture detail information regarding the instructor. CTE would like to capture an instructor ID, Name, Type, and contact details. They would also like to be able to list Training and Events Sessions that contains the session ID, Name or description, capacity, Type, and if possible, start and end date.

Then create a location for where those sessions will be held.





Development



- Table Relations - Sets up a lookup into another table

```
TableRelation = <TableName>[.<FieldName>] [WHERE(<TableFilters>)] |  
[IF(<Conditions>) <TableName>[.<FieldName>] [WHERE(<TableFilters>)] ELSE <TableRelation>]  
<Conditions> ::= <TableFilters>  
<TableFilters> ::= <TableFilter> {,<TableFilter>}  
<TableFilter> ::= <DestinationFieldName>=CONST(<FieldConst>) | FIELD(<SourceFieldName>)
```

```
... field(5; "Instructor No."; Code[20])  
{  
... Caption = 'Instructor No.';  
... TableRelation = "DEV Instructor"."No.";  
... }
```

Development



- Enumerations – set of named constants

```
enum 50101 "DEV_Program_Level"
{
    Extensible = true;

    0 references
    value(0; "Beginner")
    {
        Caption = 'Beginner';
    }
    0 references
    value(1; Intermediate)
    {
        Caption = 'Intermediate';
    }
    0 references
    value(2; Advanced)
    {
        Caption = 'Advanced';
    }
}
```



Development

- PageParts

```
part(SalesLines; "DEV Registration Subform")
{
    ApplicationArea = Basic, Suite;
    SubPageLink = "Registration No." = field("No.");
    UpdatePropagation = Both;
}
```

- PageActions

```
action(Instructors)
{
    ApplicationArea = Basic, Suite;
    Caption = 'Instructors';
    Image = Resource;
    RunObject = page "DEV Session Instructors";
    RunPageLink = "Session No." = field("No.");
    ToolTip = 'View or edit the instructors for this session.';

    0 references
    trigger OnAction()
    begin
        InstructorList.Run();
    end;
}
```





Exercise 2



The Contoso Training and Events (CTE) company would like to extend the functionality beyond just managing the instructor profiles into a page that shows what sessions the instructors are assigned to. Contoso would like to be able to capture registrations that contains the session and Contact/Attendees. Create a page and subpage.



| Extending Business Central

Extending Business Central



- Table Extension Object
 - allows you to add additional fields or to change some properties on a table
 - define keys for fields added in the table extension
 - define keys for fields in base table
- Page Extension Object
 - adds or overrides the functionality
 - Use keywords to place actions and controls (addfirst, addlast, add after)



Extending Business Central

- FlowFields - display the result of the calculation described in the CalcFormula Property.
 - Sum The sum of a specified set in a column in a table.
 - Average The average value of a specified set in a column in a table.
 - Exist Indicates whether any records exist in a specified set in a table.
 - Count The number of records in a specified set in a table.
 - Min The minimum value in a column in a specified set in a table.
 - Max The maximum value in a column in a specified set in a table.
 - Lookup Looks up a value in a column in another table.

```
    . . . Reference
    . . . field(80; "No. Sessions"; Integer)
    . . .
    . . . {
    . . .     Caption = 'No. Sessions';
    . . .     Editable = false;
    . . .     FieldClass = FlowField;
    . . .     CalcFormula = count("DEV Session Instructor" where("Instructor No." = field("No.")));
    . . .
    }
```



Extending Business Central

- FactBoxes - right-most side of a page for displaying content – other pages, charts, system parts

```
page 50100 "Simple Customercard Page"
{
    PageType = Card;

    layout
    {
        area(FactBoxes)
        {
            part(MyPart; "Acc. Sched. KPI Web Srv. Lines")
            {
                ApplicationArea = All;
                SubPageView = SORTING ("Acc. Schedule Name");
            }
        }
        systempart(Links; Links)
        {
            ApplicationArea = All;
        }
        systempart(Notes; Notes)
        {
            ApplicationArea = All;
        }
    }
}
```



The screenshot shows the Sales Order page for Sales Order S-ORD101001. The main area displays the order details for Adatum Corporation. On the right side, there are three FactBoxes:

- Sell-to Customer Sales History**: A grid showing sales history with columns for Ongoing Sales Quotes, Ongoing Sales Orders, Ongoing Sales Segments, and Ongoing Sales Returns. The data is as follows:

Ongoing Sales Quotes	Ongoing Sales Orders	Ongoing Sales Segments	Ongoing Sales Returns
0	0	6	2
0	0	33	33
0	0	0	0
- Customer Details**: Displays customer information for Adatum Corporation, including name, phone number, email, fax number, credit limit, available credit, payment terms code, and contact person.
- Sales Line Details**: Displays details for the sales line item 1996-S, including item number, required quantity, and unit of measure.



Exercise 3

The Contoso Training and Events (CTE) company would like the ability to see the instructor information quickly from the registration list without having to drill into the record itself. Create a factbox that shows additional detail of the instructor. It must list the Instructor ID, Name, Type, and Job Title.





@ 2025 Dynamic Communities

Day 1 Recap

- Set up Development Environment
- AL Language
- Creating an App
- Extending Business Central



| Thank You





COMMUNITY SUMMIT

DYNAMICS 365 BUSINESS CENTRAL (NAV)

© 2025 Dynamic Communities. All rights reserved.



COMMUNITY SUMMIT

DYNAMICS 365 BUSINESS CENTRAL (NAV)

Orlando, FL
October 19–23, 2025



The largest Microsoft Business Applications user conference on the planet.



AL Bootcamp: From Zero to Hero in Two Days

Day 2

A Dynamics-First, 'For User, By User' Event.

@ 2025 Dynamic Communities

Day 1 Recap

- Set up Development Environment
- AL Language
- Creating an App
- Extending Business Central





Agenda

- Day 1
- Day 2
- Working with multiple Projects
- Reports and Report Extensions
- Codeunits
- Functions
- Debugging in AL



Working with Multiple Projects

Working with Multiple Projects



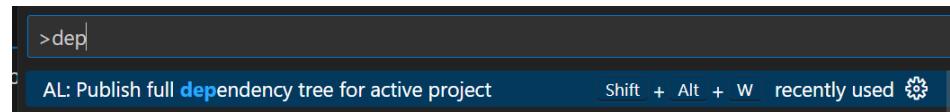
- Workspaces
 - A Visual Studio Code "workspace" is the collection of one or more folders that are opened in a VS Code window (instance)
 - <name>.code-workspace

Working with Multiple Projects



- Dependencies
 - Packages that the extension is dependent upon

```
  "dependencies": [
    {
      "id": "eeb5ab52-dbde-4bac-b267-c44446a1b7ef",
      "name": "SummitNA 2023 Dev Sample",
      "publisher": "SummitNA 2023",
      "version": "22.5.0.0"
    }
  ],
```

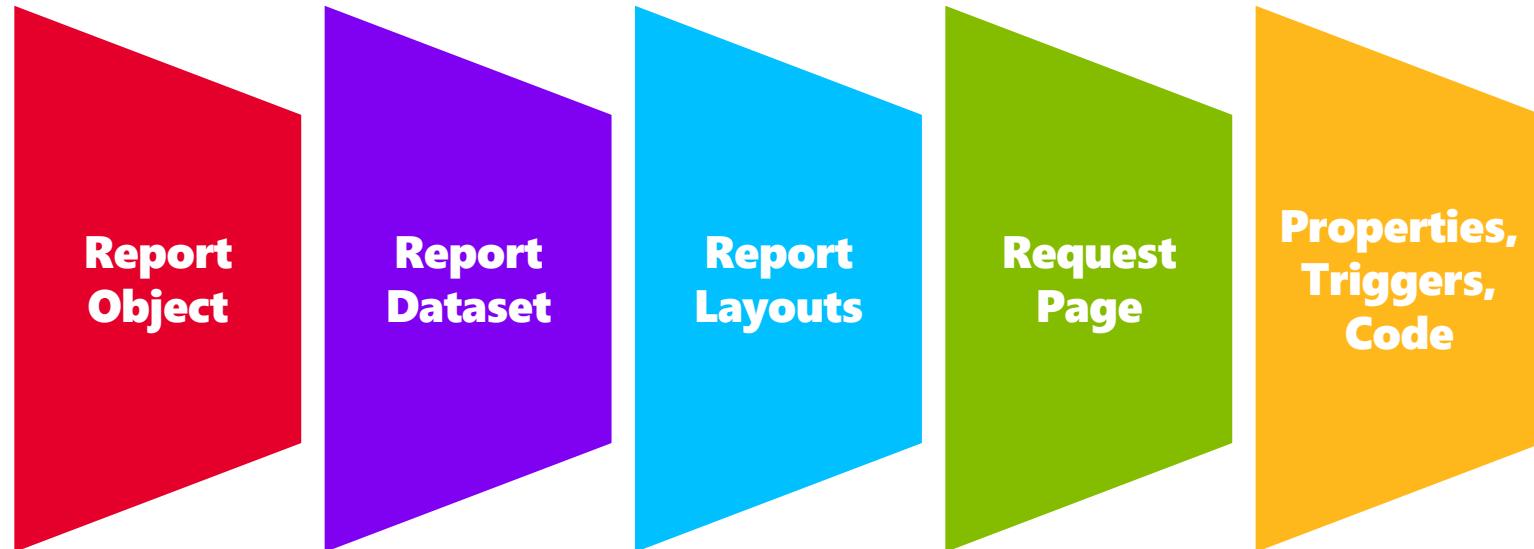






| Reports

| Report Architecture



| Report Layouts



RDL

Word

Excel

Improvements to Word Layouts



- <https://aka.ms/bcReporting>
- <https://aka.ms/BCYouTube>
 - <https://www.youtube.com/watch?v=V8CSor5qBRE>
 - https://www.youtube.com/watch?v=hn92Al_x-s8

Extending Base Report



You can:

- Add to Request Pages
- Add new Data Items
- Implement Triggers
- Add Columns to existing data items in the report dataset
- Add new Report layout to reflect the new fields



Multiple Report Layouts

- treporttext
- Rendering
- Layout in section
- Caption
- Summary
- LayoutFile



```
nds "Employee - List"
```

```
own in Pivot table in Excel';  
ot.xlsx';
```

```
rted by last name in Excel';  
stName.xlsx';
```

```
}  
  
layout(LayoutWord)  
{  
    Type = Word;  
    Caption = 'WordList';  
    Summary = 'Employee list sorted by last name in Word';  
    LayoutFile = 'EmpSortedByLastName.docx';  
}  
}
```



Processing Reports

- Designed primarily to perform data processing tasks
- Handle backend operations efficiently
- Request Page for Filters



Exercise 4



The Contoso Training and Events (CTE) company would now like to have a few reports built.

- Create a report that list the sessions.
- Extend an existing Report
- Use Report Layout of your choice on one
- Utilize the Multiple Report Layouts with Rendering





| Codeunits



Codeunits

- Modularized container for AL Code that contains business logic.

```
0 references
codeunit 50101 "DEV·Rental·Management"
{
    Access = Public;
    Subtype = Normal;
    TableNo = "DEV·Book·Rental·Header";

    0 references
    trigger OnRun()
    begin
        CheckRental(Rec);
    end;
}
```

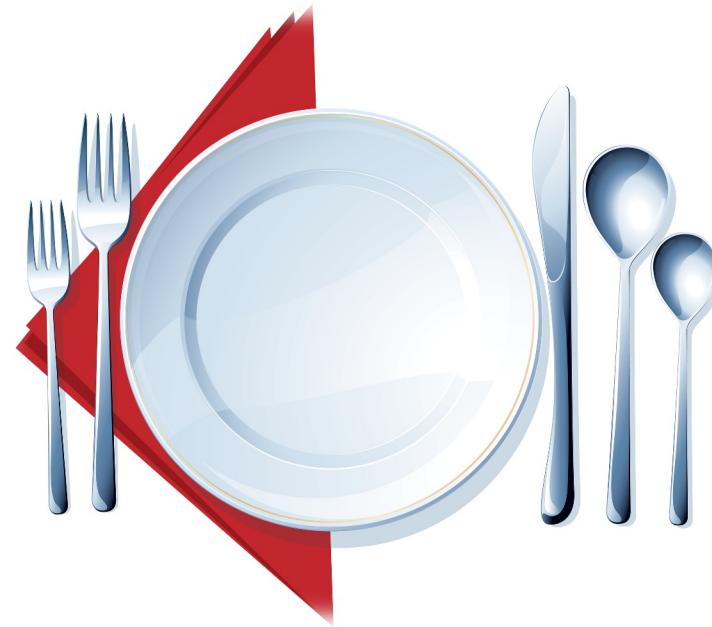
| Codeunits - Procedures



- group of statements that perform an operation or task
- Scope determines if the procedures can be called, from the same object in which they are declared or from other parts of the application

```
...//Attributes(arguments·list)
...//local·procedure·<method_name>(parameter·list)·<return_value_name>·:><data_type>[<length>]
```

| Lunch ?!?





Exercise 5

The Contoso Training and Events (CTE) company would now like to

Create a new extension that adds two new fields to the instructor table and displays those fields on the instructor card

Create a codeunit that can accept the Registration Table as a source that will check for specific conditions of your choice on the registration

Create a library codeunit with a set of procedures for performing mathematical functions on sets of numbers.

Create a page that accepts user input and processes the data by using the library procedures





@ 2025 Dynamic Communities



@ 2023 Dynamic Communities

@ 2025 Dynamic Communities

| GitHub Copilot





Debugging in AL

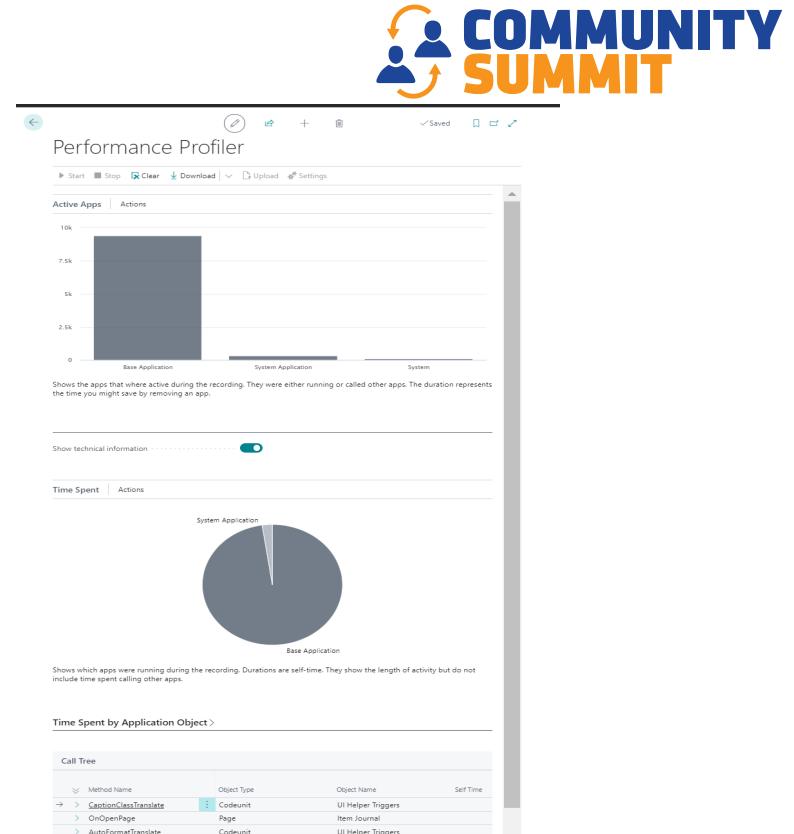
- Breakpoints
- Break on Errors
- Break on Record Change
- Attach and Debug Next
- Watches

```
{  
    "name": "Microsoft.cloud.sandbox",  
    "request": "attach",  
    "type": "al",  
    "environmentType": "Sandbox",  
    "environmentName": "sandbox",  
    "tenant": "117d2cc0-dbb2-4e85-b82d-96d851551dff",  
    "breakOnError": "All",  
    "breakOnRecordWrite": "ExcludeTemporary",  
    "userId": "brad",  
    "enableLongRunningSqlStatements": true,  
    "enableSqlInformationDebugger": true  
},
```



Performance Profiler

- In-client Performance Profiler, to record a snapshot of the process
- Profiler monitors all of the apps that are involved in the process
- Identify where there may be a holdup



| Vibe Coding



Day 2 Recap

- Working with multiple Projects
- Codeunits
- Functions
- Debugging in AL
- It doesn't end here!



GitHub Repos



SCAN ME

Full AL Bootcamp Solution:
<https://bit.ly/albootcampsummit2024>



SCAN ME

Additional Resources (Processing Reports,
Report Extensions, Page Actions):
<https://bit.ly/ajsalgithubrepo>

| Suggested Resources





Thank you for Attending



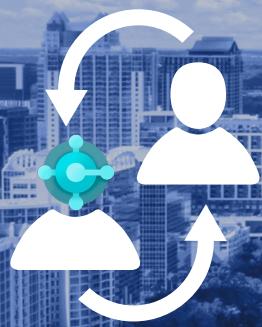
Brad Prendergast



AJ Ansari



Only BC Fans



COMMUNITY SUMMIT

DYNAMICS 365 BUSINESS CENTRAL (NAV)

© 2025 Dynamic Communities. All rights reserved.



COMMUNITY SUMMIT

DYNAMICS 365 BUSINESS CENTRAL (NAV)

Orlando, FL
October 19–23, 2025





AL Bootcamp: Extending Extension Development for Business Central

@ 2024 Dynamic Communities

@ 2025 Dynamic Communities

| Speakers



Brad Prendergast



AJ Ansari



Brad Prendergast

- 20+ years in Microsoft Dynamics, specializing in Business Central
- Unique dual perspective as a partner & end user
- Member of Summit Programming Committee & Board of Advisors
- Co-host of the popular Dynamics Corner podcast
- Passionate about bringing people together in the Dynamics space



@ 2025 Dynamic Communities



AJ Ansari

- Community Summit Legend and BCUG All-Star
- 17 years in Microsoft Dynamics; background as certified developer, consultant and practice leader
- Member of Summit Programming Committee & Board of Advisors



Only BC Fans



Only Copilot Fans



COO & Partner

aja@dswius.com

bit.ly/AJAnsari



@ 2025 Dynamic Communities

Know Before You Go

- Restrooms are where?
- Nearest Emergency Exit is where?



Session Objectives



- Utilize events to implement custom business logic and extend Business Central functionality
- Implement robust error-handling strategies to improve extension reliability and user experience.
- Develop and expose APIs to enable integration with external systems and enhance interoperability.
- AI Toolkit – overview touchpoint – mention sessions (Summit and Outside)

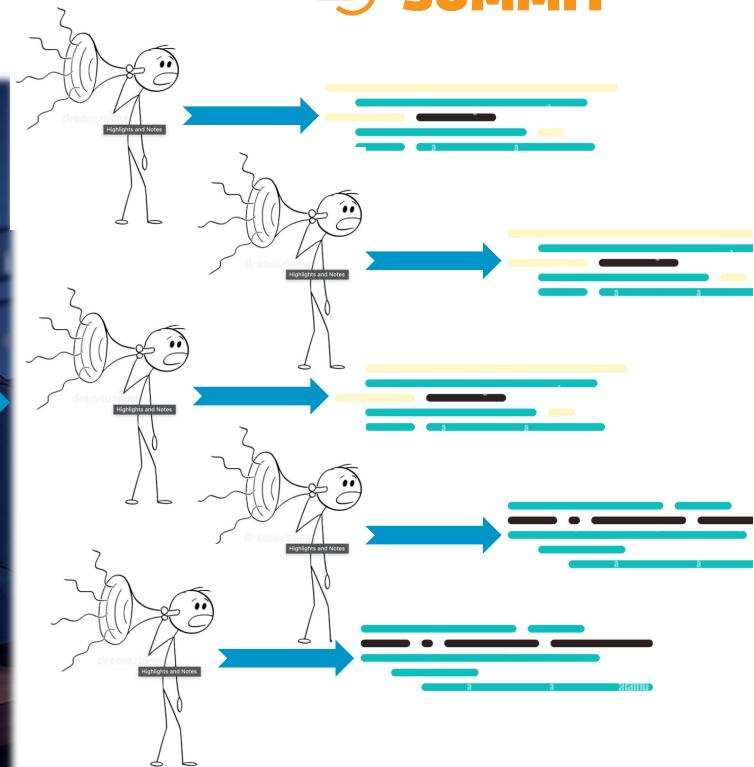
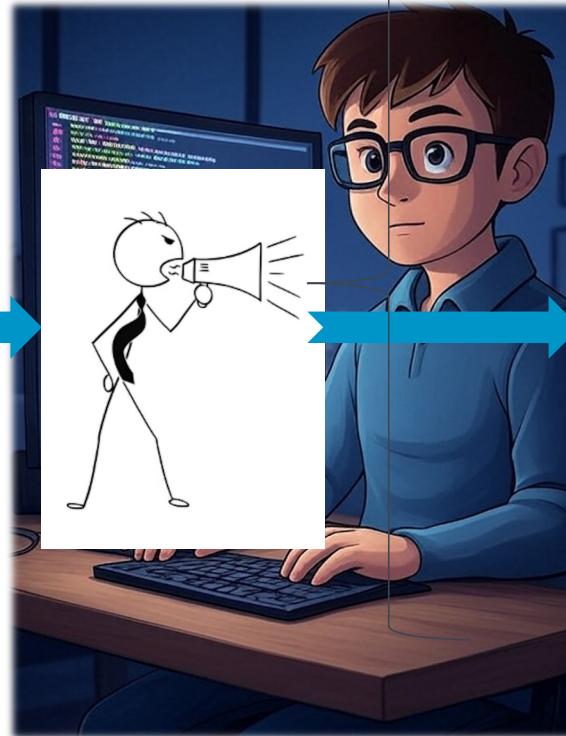
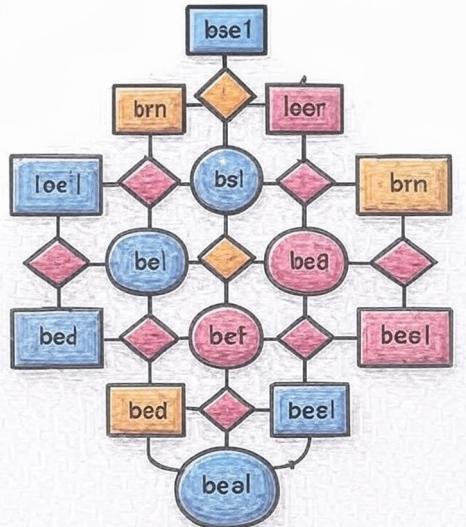


What is an Event

- Significant occurrence or change in state within an application that can trigger a response or action



How Do Events Work





Types of Events

- Business Event
 - Defined that structure is not going to change
 - Facilitate communication between BC and external systems
 - Do not have access to global variables

```
[BusinessEvent(IncludeSender: Boolean [, Isolated: Boolean])]
```

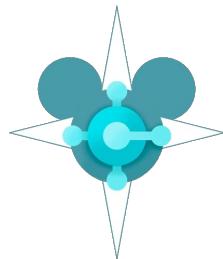
```
[ExternalBusinessEvent(Name: Text, DisplayName: Text, Description: Text, Category: enum [, Version: Text])]
```



Types of Events

- Business Event

```
[ExternalBusinessEvent('PurchaseReceiptPosted', 'Purchase receipt posted',
    | | | 'This business event is triggered when a purchase order is received.', EventCategory:::"My Purchase Events")]
1 reference
local procedure MyBusinessEventPurchaseReceivedPosted(PurchaseInvoiceId: Guid; Url: Text[250]; Base64Text: Text[250])
begin
end;
```





Types of Events

- Integration Events
 - Extend Business Central at specific points in application logic
 - Base Application
 - ISV Solutions
 - Your own solutions

```
[IntegrationEvent(IncludeSender: Boolean, GlobalVarAccess: Boolean [, Isolated: Boolean])]
```



Types of Events

- In
... SalesHeader := SalesHeader2;
... OnCodeOnBeforeFillTempLines(SalesHeader, CalledBy);
... FillTempLines(SalesHeader, TempSalesLineGlobal);
•
... // Check that the invoice amount is zero or greater
... OnRunOnBeforeCheckTotalInvoiceAmount(SalesHeader);

```
[IntegrationEvent(false, false)]
local procedure OnRunOnBeforeCheckTotalInvoiceAmount(var SalesHeader: Record "Sales Header")
begin
```

```
[IntegrationEvent(false, false)]
local procedure OnRunOnBeforeCheckAndUpdate(var SalesHeader: Record "Sales Header")
begin
end;
```

```
... OnRunOnBeforeFinalizePosting()
```

```
[IntegrationEvent(false, false)]
local procedure OnRunOnBeforeFinalizePosting(var SalesHeader: Record "Sales Header"; var SalesShipmentHeader: Record "Sales Shipment"
begin
end;
FinalizePosting(SalesHeader, EverythingInvoiced, TempOpshipCostBuffer);
```



Types of Events

- Internal Event
 - Most restrictive in scope
 - Can only be subscribed within same module



Types of Events

• Global Events

- Global events are predefined system events that are automatically raised by various base application codeunits.

Codeunit ID	Codeunit Name	Event
9170	Conf./Personalization Mgt.	OnRoleCenterOpen OnAfterLogInEnd OnBeforeLogInStart OnBeforeCompanyOpen OnAfterCompanyOpen OnBeforeCompanyClose OnAfterCompanyClose
42	TextManagement	OnBeforeMakeTextFilter OnAfterMakeDateTimeFilter OnAfterMakeDateFilter OnAfterMakeTextFilter OnAfterMakeTimeFilter
49	GlobalTriggerManagement	OnAfterGetGlobalTableTriggerMask OnAfterOnGlobalInsert OnAfterOnGlobalModify OnAfterOnGlobalDelete OnAfterOnGlobalRename OnAfterGetDatabaseTableTriggerSetup OnAfterOnDatabaseInsert OnAfterOnDatabaseModify OnAfterOnDatabaseDelete OnAfterOnDatabaseRename OnBeforeOnDatabaseInsert OnBeforeOnDatabaseModify OnBeforeOnDatabaseDelete OnBeforeOnDatabaseRename



Event Recorder

- Built in tool designed to help discover events raised during a process
- Includes name, type, and where it is published
- AL Snippet Generation



Event Recorder

Record Events

Call Order ↑	Object Type ↑	Object Name	Event Name ↑	Element Name ↑	Calling Object Type ↑	Calling Object Name	Calling Method ↑	Get AL Snippet
342	Table	Customer	OnAfterSetLastModifiedDateTime		Table	Customer	SetLastModifiedDateTime	Get AL Snippet
399	Table	Customer	OnAfterSetLastModifiedDateTime		Customer	SetLastModifiedDateTime	Get AL Snippet	
624	Table	Customer	OnAfterSetLastModifiedDateTime		Customer	SetLastModifiedDateTime	Get AL Snippet	
867	Table	Customer	OnAfterSetLastModifiedDateTime		Customer	SetLastModifiedDateTime	Get AL Snippet	
1106	Table	Customer	OnAfterSetLastModifiedDateTime		Customer	SetLastModifiedDateTime	Get AL Snippet	
1349	Table	Customer	OnAfterSetLastModifiedDateTime		Customer	SetLastModifiedDateTime	Get AL Snippet	
→ 1547	Table	Customer	OnAfterSetLastModifiedDateTime	[EventSubscriber(ObjectType:Table, Database::"Customer", 'OnAfterSetLastModifiedDateTime', ", true, true)] local procedure MyProcedure() begin end;	Customer	SetLastModifiedDateTime	Get AL Snippet	
1098	Table	Customer	OnAfterValidateEvent	CRM Customer Name	Customer	City - OnValidate	Get AL Snippet	
620	Table	Customer	OnAfterValidateEvent		Customer		Get AL Snippet	
1099	Table	Customer	OnAfterValidateEvent	City	Customer		Get AL Snippet	
617	Table	Customer	OnAfterValidateEvent	Name	Customer		Get AL Snippet	
338	Table	Customer	OnAfterValidateEvent	Payment Method Code	Customer		Get AL Snippet	
1540	Table	Customer	OnAfterValidateEvent	Post Code	Customer		Get AL Snippet	
1539	Table	Customer	OnAfterValidatePostCode		Table	Customer	Post Code - OnValidate	Get AL Snippet
232	Table	Customer	OnBeforeInsert		Table	Customer	OnInsert	Get AL Snippet

OK

Shipments: 22
Posted Sales Credit Memos

Invoices: 0
Customer Subscription Contracts

Return Receipts: 0
Subscriptions



Event Subscriptions

AL Explorer

Project: ALProject1 (Default Publisher)

OBJECTS EVENTS APIS EXTENSIBLE ENUMS

Search: Type to filter Group by: None Module: All

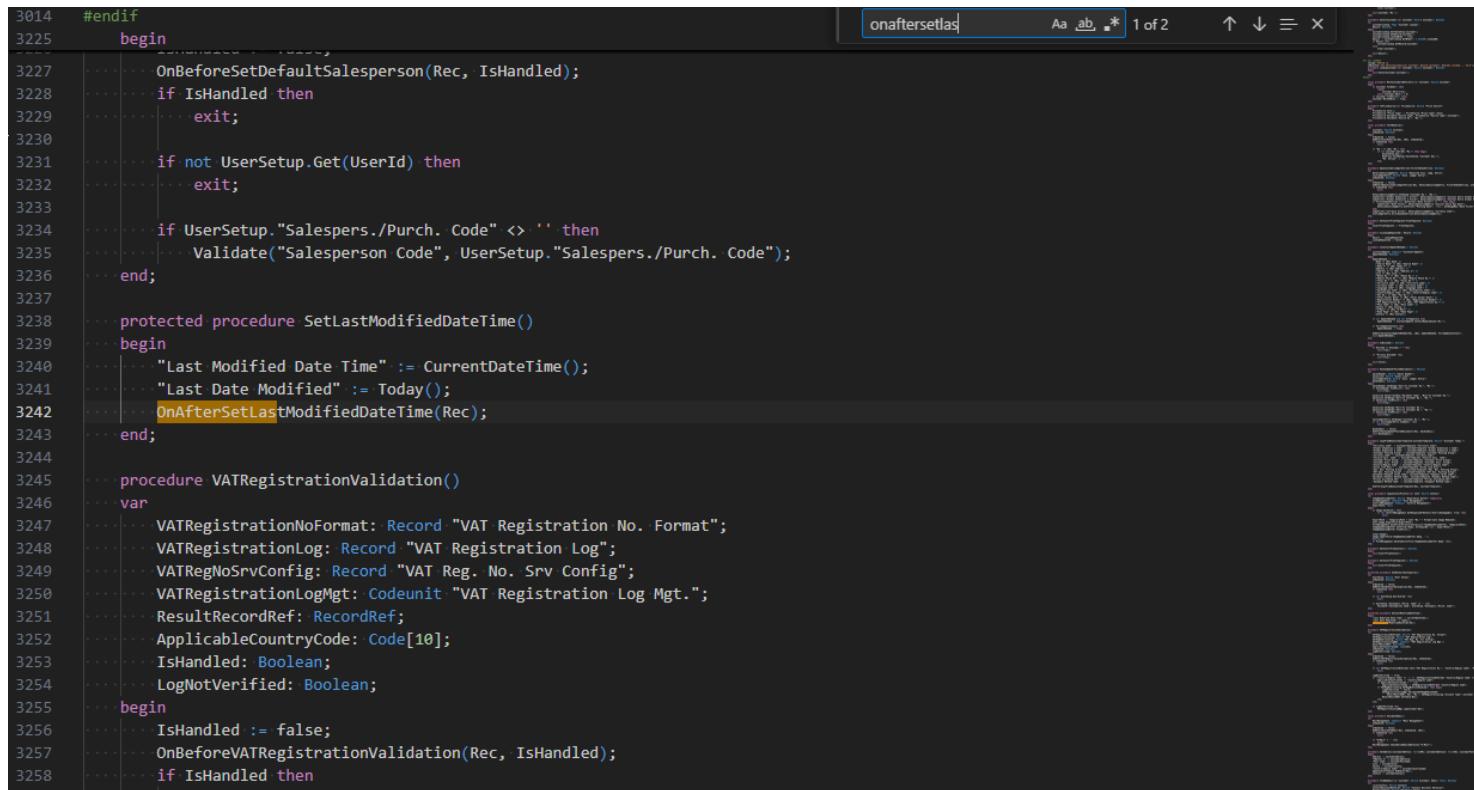
Signature	Type	Object Name ↑	Module	Namespace
OnBeforeValidateRegistrationNumber(var Customer: Rec...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnBeforeValidateShortcutDimCode(var Customer: Reco...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnBeforeValidateVATRegistrationNo(var Customer: Reco...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnBeforeVATRegistrationValidation(var Customer: Reco...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetBalanceAsVendorOnBeforeCalcBalance(var Vendor...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetCustNoOpenCardOnAfterMarkCustomersWithSimi...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetCustNoOpenCardOnAfterOnAfterCustomerFilterFr...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetCustNoOpenCardOnAfterSetCustomerFilters(var C...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetCustNoOpenCardOnBeforeCustomerFindSet(var Cu...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetCustNoOpenCardOnBeforeFilterCustomer(var Cus...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetTotalAmountLCYOnAfterCalcFields(var Customer: ...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnGetTotalAmountLCYUIOnAfterSetAutoCalcFields(var C...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnMarkCustomersWithSimilarNameOnBeforeCustomerF...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnShowContactOnBeforeOpenContactCard(var Contact:...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnShowContactOnBeforeOpenContactList(var Contact: ...	Table	Customer	Base Application	Microsoft.Sales.Customer
OnAfterCalcRemainingAmount(var TempCustLedgerEntr...	Report	Customer - Balance to Date	Base Application	Microsoft.Sales.Reports

Source Subscribe

A large blue arrow points from the "Subscribe" button at the bottom left of the table to the "Subscribe" button at the bottom right of the slide.



Event Subscriptions



A screenshot of a code editor displaying a segment of Delphi-like code. The code is part of a class definition, likely for a database or application component. It includes several event handlers and variable declarations. The code is numbered from 3014 to 3258. The editor interface shows tabs for 'onaftersetlas' and '1 of 2', and various toolbars at the top.

```
3014  #endif
3225  begin
3226    OnBeforeSetDefaultSalesperson(Rec, IsHandled);
3227    if IsHandled then
3228      exit;
3229
3230    if not UserSetup.Get(UserId) then
3231      exit;
3232
3233    if UserSetup."Salespers./Purch. Code" <> '' then
3234      Validate("Salesperson Code", UserSetup."Salespers./Purch. Code");
3235    end;
3236
3237  protected procedure SetLastModifiedDateTime()
3238  begin
3239    "Last Modified Date Time" := CurrentDateTime();
3240    "Last Date Modified" := Today();
3241    OnAfterSetLastModifiedDateTime(Rec);
3242  end;
3243
3244
3245  procedure VATRegistrationValidation()
3246  var
3247    VATRegistrationNoFormat: Record "VAT Registration No. Format";
3248    VATRegistrationLog: Record "VAT Registration Log";
3249    VATRegNoSrvConfig: Record "VAT Reg. No. Srv Config";
3250    VATRegistrationLogMgt: Codeunit "VAT Registration Log Mgt.";
3251    ResultRecordRef: RecordRef;
3252    ApplicableCountryCode: Code[10];
3253    IsHandled: Boolean;
3254    LogNotVerified: Boolean;
3255  begin
3256    IsHandled := false;
3257    OnBeforeVATRegistrationValidation(Rec, IsHandled);
3258    if IsHandled then
3259      exit;
```



Event Exercise

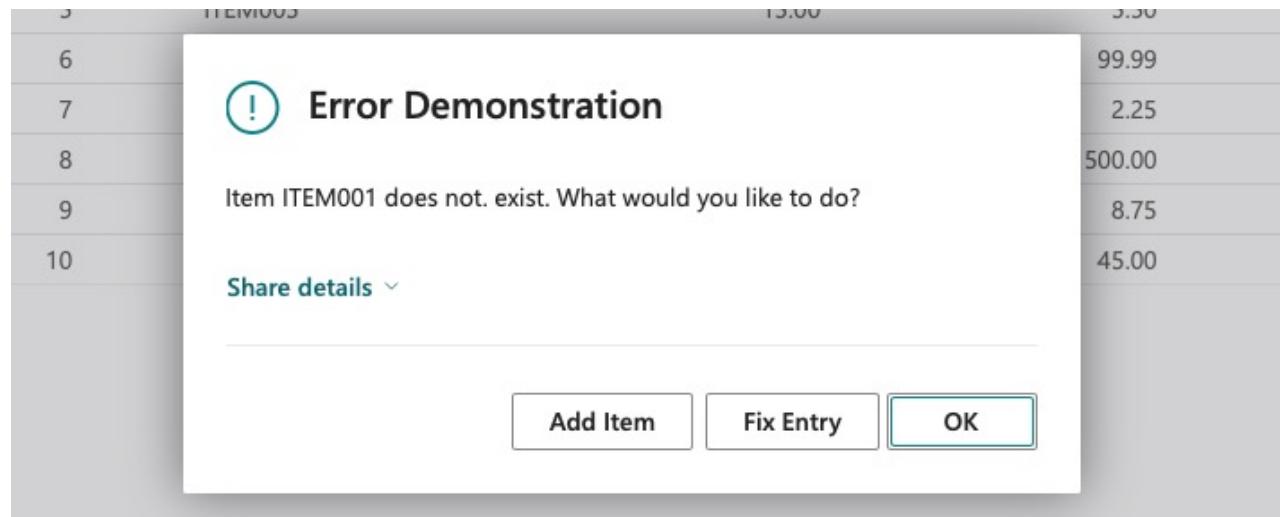


You are developing an extension for a fictional company, "AdventureWorks Cycles," that wants to extend its sales order process in Business Central. Without using the Change Log or the Monitor Fields features of Business Central, the company would like to monitor changes made to a Sales Order or Sales Invoice. The company requires a custom log to track the date, time, user, and field information for modifications made to the shipping and pricing information of the order.

The company also requires custom logging to track the duration when posting a sales order. They would like to track the start and end times of the posting process, as well as the time it takes to process the lines.



Error Handling



Error Handling

- `ErrorInfo` Data Type
 - Provides a structure for grouping information about an error.





Error Handling Exercise

Scenario:



You are developing an extension for a fictional company, "Cronus Retail," that manages a customer loyalty program. The company wants a custom page called "Loyalty Card Setup" where users can enter details for a loyalty card, including a customer number, loyalty points, and a card activation date. The system must validate the input and provide detailed error messages using the ErrorInfo data type if the input is invalid.

If any validation fails, the system should display a detailed error message with a custom title, description, and a link to the relevant record or page for correction.



Error Exercise



Create a Table Extension:

Extend the Customer table to include a new field: Loyalty Card No. (Code[20]).

Ensure the field is linked to a new table called Loyalty Card Setup.

Create a New Table:

Create a table called Loyalty Card Setup with the following fields:

Card No. (Code[20], Primary Key)

Customer No. (Code[20], linked to the Customer table)

Loyalty Points (Integer)

Activation Date (Date)

Create a Page for Data Entry:

Create a new card page called Loyalty Card Setup Card for the Loyalty Card Setup table.

Add fields for Card No., Customer No., Loyalty Points, and Activation Date.

Ensure the page allows users to input and save data.

Error Exercise



Implement Error Handling with ErrorInfo:

The Customer No. exists in the Customer table.

The Loyalty Points is positive and does not exceed 10,000.

The Activation Date is not in the past (relative to the system date, e.g., WorkDate()).

Use the ErrorInfo data type to create detailed error messages for each validation failure. For example:

- If the customer number is invalid, provide a message like "Invalid Customer No." with a link to the Customer List page.
- If loyalty points exceed 10,000, include a custom title like "Loyalty Points Error" and a detailed message explaining the limit.
- If the activation date is in the past, provide a message with a call to action to correct the date.
- Use ErrorInfo properties such as Title, Message, RecordId, TableId, and AddAction to enhance the error details.

| Working with APIs





Working with APIs

Bound unbound actions – AJ has sample
Sample API page - screen

Working with APIs



- What are APIs?
 - Application Programming Interface
 - allows two applications to talk to each other
 - Standardized

Working with APIs



- API Pages
 - API Page types create versioned, webhook-supported, OData v4 enabled REST web services
 - page cannot be displayed in the user interface
- Naming
 - camelCase for naming attributes, tables, as well as APIPublisher, APIGroup, EntityName, and EntitySetName
 - Alphanumeric characters
 - APIVersion follows the pattern vX.Y

Working with APIs



```
page 50200 "DEV Widget Entity"
{
    APIGroup = 'Sample';
    APIPublisher = 'SummitNA';
    APIVersion = 'v2.0';
    Caption = 'widgetEntity', Locked = true;
    ChangeTrackingAllowed = true;
    DelayedInsert = true;
    EntityName = 'widget';
    EntitySetName = 'widgets';
    OdataKeyFields = SystemId;
    PageType = API;
    SourceTable = "DEV Widget";

    layout
    {
        0 references
        area(content)
        {
            0 references
            repeater(General)
            {
                0 references
                field(id; Rec.SystemId)
                {
                    ApplicationArea = All;
                    Caption = 'Id', Locked = true;
                    Editable = false;
                }
                0 references
                field(no; Rec."No.")
                {
                    ApplicationArea = All;
                    caption = 'no', Locked = true;
                }
                0 references
                field(description; Rec.Description)
                {
                    ApplicationArea = All;
                    caption = 'description', Locked = true;
                }
            }
        }
    }
}
```

| PowerPlatform



- Consume API Pages



API Exercise



The Contoso Training and Events (CTE) company would like to provide an endpoint for a 3rd-Party, Vendor, or Website to consume registrations, sessions available, or Instructor information.

- You should create an API Page for information to be consumed externally.

| Vibe Coding

