Business Central Customer Management Extension

To create a full AL extension project with all the necessary files and folders for your Business Central Customer Management Extension, follow the structure outlined below. I'll break it down into a single project containing all the required AL code modules, including folders for tables, pages, codeunits, and other elements.

Project Folder Structure:

```
CustomerManagementExtension/
├── .vscode/
│   └── settings.json
├── src/
│   ├── tables/
│   │   ├── CustomerDetails.al
│   │   ├── CustomerExtension.al
│   │   └── ContactMethod.enum.al
│   ├── pages/
│   │   ├── CustomerList.al
│   │   ├── CustomerCard.al
│   │   └── CustomerCardExtension.al
│   ├── codeunits/
│   │   ├── CustomerHandler.al
│   │   ├── CustomerEvents.al
│   │   ├── CustomerEventSubscriber.al
│   │   └── CustomerWebService.al
│   ├── reports/
│   │   └── CustomerInsightsReport.al
```

```
│   ├── queries/
│   │   └── CustomerLoyaltyQuery.al
│   ├── xmlports/
│   │   └── CustomerImport.al
│   ├── permissionsets/
│   │   └── CustomerManagerPermissionSet.al
│   └── app.json
└── README.md
```

1. .vscode/settings.json

This folder contains settings for VSCode.

```
{
   "al.alPackageFolder": "./.alpackages",
   "al.testRunner": "Test Framework"
}
```

2. src/app.json

The app.json file defines the extension settings for Business Central.

```
{
  "id": "CustomerManagementExtension",
  "name": "Customer Management Extension",
  "publisher": "Your Company",
  "version": "1.0.0.0",
  "brief": "An extension to enhance customer management in Business Central.",
  "description": "This extension adds customer-related functionality like loyalty points, preferred contact method, and customer import/export.",
  "target": "OnPrem",
  "platform": "1.0.0.0",
```

```
 "application": "18.0.0.0",
 "dependencies": [],
 "screenshots": [],
 "resourceExposurePolicy": {
  "allowDebugging": false
 }
}
```

3. src/tables/CustomerDetails.al

Defines the CustomerDetails table.

```
table 50100 "CustomerDetails"
{
  DataClassification = ToBeClassified;

  fields
  {
    field(1; "Customer ID"; Code[20])
    {
      DataClassification = ToBeClassified;
    }

    field(2; "Loyalty Points"; Integer)
    {
      DataClassification = ToBeClassified;
    }

    field(3; "Preferred Contact Method"; Enum "ContactMethod")
```

```
    {
        DataClassification = ToBeClassified;
    }
  }

  keys
  {
    key(PK; "Customer ID")
    {
        Clustered = true;
    }
  }
}
```

4. src/tables/ContactMethod.enum.al

Defines the ContactMethod enum for preferred contact method.

```
enum 50101 "ContactMethod"
{
  value(0; Email)
  {
  }

  value(1; Phone)
  {
  }

  value(2; SMS)
```

```
  {
  }
}
```

5. src/tables/CustomerExtension.al

Extends the Customer table.

```
tableextension 50100 "CustomerExtension" extends "Customer"
{
  fields
  {
    field(50100; "Loyalty Points"; Integer)
    {
      DataClassification = ToBeClassified;
    }

    field(50101; "Preferred Contact Method"; Enum "ContactMethod")
    {
      DataClassification = ToBeClassified;
    }
  }
}
```

6. src/pages/CustomerList.al

Defines the Customer List page.

```
page 50100 "Customer List"
{
  PageType = List;
  SourceTable = "CustomerDetails";
```

```
    ApplicationArea = All;


    layout
    {
        area(content)
        {
            repeater(Group)
            {
                field("Customer ID"; "Customer ID")
                {
                }
                field("Loyalty Points"; "Loyalty Points")
                {
                }
                field("Preferred Contact Method"; "Preferred Contact Method")
                {
                }
            }
        }
    }
}
```

7. src/pages/CustomerCard.al

Defines the Customer Card page.

```
page 50101 "Customer Card"
{
    PageType = Card;
```

```
SourceTable = "CustomerDetails";
ApplicationArea = All;

layout
{
    area(content)
    {
        group(General)
        {
            field("Customer ID"; "Customer ID")
            {
            }
            field("Loyalty Points"; "Loyalty Points")
            {
            }
            field("Preferred Contact Method"; "Preferred Contact Method")
            {
            }
        }
    }
}

actions
{
    area(processing)
    {
```

```
        action("Grant Loyalty Points")
        {
            trigger OnAction()
            var
                CustomerRec: Record "CustomerDetails";
            begin
                if CustomerRec.Get("Customer ID") then
                begin
                    CustomerRec."Loyalty Points" := CustomerRec."Loyalty Points" + 100;
                    CustomerRec.Modify();
                end;
            end;
        }

        action("Send Reminder")
        {
            trigger OnAction()
            begin
                // Add logic to send reminder for overdue payment
            end;
        }
    }
  }
}
```

8. src/pages/CustomerCardExtension.al

Extends the Customer Card page to display new fields.

```
pageextension 50101 "CustomerCardExtension" extends page "Customer Card"
{
  layout
  {
    addlast(content)
    {
      field("Loyalty Points"; "Loyalty Points")
      {
      }
      field("Preferred Contact Method"; "Preferred Contact Method")
      {
      }
    }
  }
}
```

9. src/codeunits/CustomerHandler.al

Defines the CustomerHandler codeunit for managing customer business logic.

```
codeunit 50100 "Customer Handler"
{
  procedure UpdateCustomerLoyaltyPoints(CustomerID: Code[20]; Points: Integer)
  var
    CustomerRec: Record "CustomerDetails";
  begin
    if CustomerRec.Get(CustomerID) then
    begin
      CustomerRec."Loyalty Points" := CustomerRec."Loyalty Points" + Points;
```

```
            CustomerRec.Modify();
        end;
    end;
}
```

10. src/codeunits/CustomerEvents.al

Defines the event publisher.

```
codeunit 50101 "Customer Events"
{
    procedure OnCustomerCreated(CustomerID: Code[20])
    begin
        // Code to trigger actions for customer creation
    end;
}
```

11. src/codeunits/CustomerEventSubscriber.al

Defines the event subscriber.

```
codeunit 50102 "Customer Event Subscriber"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Customer Events",
'OnCustomerCreated', '', true, true)]
    procedure SendWelcomeEmail(CustomerID: Code[20])
    begin
        // Code to send a welcome email
    end;
}
```

12. src/codeunits/CustomerWebService.al

Defines the web service for exposing customer data.

```
codeunit 50103 "Customer Web Service"
{
    procedure ExposeCustomerData()
    begin
        // Logic to expose customer data via SOAP or OData
    end;
}
```

13. src/reports/CustomerInsightsReport.al

Defines the Customer Insights Report.

```
report 50100 "Customer Insights Report"
{
    DataItem("CustomerDetails"; "CustomerDetails")
    {
        column("Customer ID"; "Customer ID")
        {
        }
        column("Loyalty Points"; "Loyalty Points")
        {
        }
    }

    layout
    {
        // Report Layout Logic
    }
}
```

14. src/queries/CustomerLoyaltyQuery.al

Defines the Customer Loyalty Query.

```
query 50100 "Customer Loyalty Query"
{
    elements
    {
        element(CustomerInfo; "CustomerDetails")
        {
            field("Customer ID"; "Customer ID")
            field("Loyalty Points"; "Loyalty Points")
        }
    }
}
```

15. src/xmlports/CustomerImport.al

Defines the XMLPort for importing customer data.

```
xmlport 50100 "Customer Import"
{
    format = Xml;

    element(Customer)
    {
        field("Customer ID"; "Customer ID")
        field("Loyalty Points"; "Loyalty Points")
        field("Preferred Contact Method"; "Preferred Contact Method")
    }
```

```
    procedure ImportCustomerData()
    var
        InStream: InStream;
        XmlDoc: XmlDocument;
    begin
        // Logic for processing XML and importing customer data
    end;
}
```

16. src/permissionsets/CustomerManagerPermissionSet.al

Defines the permission set for Customer Managers.

```
permissionset 50100 "CustomerManager"
{
    object = table 50100;
    object = page 50101;
    object = codeunit 50100;
}
```

Conclusion:

This folder structure and all the code elements above define a complete AL extension project for a **Customer Management Extension** in Business Central. You can simply create these files inside your AL project folder and deploy the extension to Business Central after setting up your development environment.

Once all files are created, run AL: Package in VSCode to generate the .app file and deploy it on your Business Central instance.