PHASE 3 IMPLEMENTATION PLAN: Asset Pro - Universal Asset Management

Date: 2025-11-02 **Project:** Asset Pro - Multi-Industry Asset Management Platform **Status:** □□ AWAITING APPROVAL - DO NOT IMPLEMENT **Workflow Phase:** Phase 3 (Planning - STOP POINT)

APPROVAL REQUIRED

This document presents the complete implementation plan for Asset Pro. According to AL Development Core workflow Phase 3, **explicit approval is required** before proceeding to Phase 4 (Implementation).

Approval needed for:

1. Overall architecture approach
2. Object list and ID assignments
3. Test strategy
4. Implementation concerns and mitigation plans

Decision Points Confirmed:

- □ Universal multi-industry platform from start
- □ Unlimited hierarchy levels
- □ Sales Order integration (Phase 1), other documents (Phase 2)
- □ Hierarchical Asset table (parent/child relationships)
- □ Subscription Billing as optional add-on (defer)
- □ Production ID Range: 70182300-70182449 (150 objects)
- □ Test ID Range: 50100-50199 (100 objects)

---

1. ANALYSIS SUMMARY

Project Context

Publisher: JEMEL **App Prefix:** JMLAP (JEMEL Asset Pro) **Available Object IDs:** 70182300-70182449 (150 objects) **Base Architecture:** Adapted from Rollsberg marine vessel management system **Target:** Universal multi-industry asset management platform

Key Requirements Analysis

MUST Requirements:

1. □ Complete 3-level asset hierarchy (minimum)
2. □ Asset ownership tracking (Customer, Location, Cost Center, Vendor)
3. □ Asset transitions with history tracking
4. □ Sales Order integration for transitions
5. □ Parts can be tracked as separate assets (hierarchical)

SHOULD Requirements: 6. □ Unlimited hierarchy levels (extensible architecture) 7. □□ Transfer Order, Purchase Order integration (Phase 2) 8. □□ Subscription Billing integration (optional add-on, Phase 2+)

App.json Configuration

- **App ID:** 0e7df5a2-2732-4caf-a0e7-681a20c96f59
- **App Name:** Asset Pro
- **Publisher:** JEMEL
- **Version:** 26.0.0.0
- **Platform:** BC 26.0
- **ID Range:** 70182300-70182449 (updated from 70182300-70182349)

Symbols Verification

- □ ObjectIndex.md exists and is current
- □ Base BC objects located and verified:
  - Customer: `src\Sales\Customer\Customer.Table.al`
  - Sales Header: `src\Sales\Document\SalesHeader.Table.al`
  - Sales Line: `src\Sales\Document\SalesLine.Table.al`
  - Location: `src\Inventory\Location\Location.Table.al`
  - Vendor: `src\Purchases\Vendor\Vendor.Table.al`

Existing Files Review

Previous implementation files found (to be replaced):

- Mixed naming conventions (some proper, some with hyphens/numbers)
- 41 files total (enums, tables, pages, table extensions, page extensions, codeunits)
- Will be completely replaced with new implementation

---

2. ARCHITECTURE DESIGN

2.1 Core Design Principles

1. Universal Multi-Industry Support

- Configurable terminology per asset type
- Industry template system (Marine, Construction, Medical, IT, etc.)
- Feature toggles to enable/disable functionality per asset type
- Custom attribute framework for industry-specific fields

2. Unlimited Hierarchy

- Single Asset table with self-referential parent/child relationships
- No fixed depth limit (recursive navigation)
- Level calculation on-demand or cached
- Circular reference prevention

3. Hierarchical Parent/Child Model

- Any Asset can be parent of other Assets
- Example: Vehicle (Asset) → Engine (Asset) → Turbocharger (Asset)
- Component table separate for non-asset parts/assemblies
- BOM for maintaining parts lists

4. BC Integration Strategy

- Deep Sales document integration (Phase 1)
- Customer/Vendor/Location relationship tracking
- Dimension integration for financial reporting
- Event-driven architecture for extensibility

2.2 Data Model Architecture

```
┌──────────────────────────────────────────────────────────────┐
│                      ASSET PRO DATA MODEL                      │
└──────────────────────────────────────────────────────────────┘

┌──────────────────┐
│   Asset Type     │  Configuration & Setup
│   └─ Industry    │
│   └─ Terminology │  → Controls UI labels, features, attributes
│   └─ Features    │
│                  │
└──────────────────┘

┌──────────────────┐       ┌──────────────────┐
│      Asset       │◄─────►│  Asset (Parent)  │  Unlimited Hierarchy
│                  │1     *│                  │  (Self-referential)
│  - Asset Type    │       │  - Parent Asset  │
│  - Parent Asset  │       │  - Level         │
│  - Level         │       └──────────────────┘
│  - Status        │
│  - Location      │       ┌──────────────────┐
│                  │       │ Customer Asset   │  Many-to-Many
└──────────────────┘       │                  │  Ownership Links
         │1               *│  - Asset No.     │
         │                 │  - Customer No.  │
         │                 │  - Relationship  │
         │                 │                  │
         │                 └──────────────────┘
         │
         │1               ┌──────────────────┐
         └───────────────►│   Component      │  Optional Assembly
                         *│                  │ Management
                          │ - Asset No.      │
                          │ - Component Type │
                          │                  │
                          └──────────────────┘
                                   │1
                                   │
                                   │*
                          ┌──────────────────┐
                          │ Component BOM    │  Parts List
                          │                  │
                          │ - Component No.  │
                          │ - Item No.       │
                          │ - Quantity       │
                          │                  │
                          └──────────────────┘

┌──────────────────┐       ┌──────────────────┐
│ Asset Transition │       │ Asset Comment    │  Audit & Notes
│                  │       │                  │
│  - From Location │       │  - Asset No.     │
│  - To Location   │       │  - Comment Text  │
│  - Date/Time       │          │  - Created By      │
│  - Document Ref    │          └────────────────────┘
└────────────────────┘

BC EXTENSIONS:
┌────────────────────┐          ┌────────────────────┐          ┌────────────────────┐
│   Customer         │          │   Sales Header     │          │    Sales Line      │
│                    │          │                    │          │                    │
│ + Asset Count      │          │ + Asset No.        │          │ + Asset No.        │
└────────────────────┘          │ + Asset Type       │          │ + Component No.    │
                                │ + Primary Contact  │          └────────────────────┘
                                └────────────────────┘
```

2.3 Unlimited Hierarchy Implementation

Design Decision: Self-Referential Parent Field

```
table 70182300 "JMLAP Asset"
{
    fields
    {
        field(1; "No."; Code[20]) { }

        // Hierarchy Support
        field(10; "Parent Asset No."; Code[20])
        {
            TableRelation = "JMLAP Asset" where("No." = filter(<> field("No.")));
            // Prevents self-reference

            trigger OnValidate()
            begin
                ValidateNoCircularReference();
                RecalculateLevel();
            end;
        }

        field(11; "Level"; Integer)
        {
            // Calculated field: 0 = Root, 1 = Child, 2 = Grandchild, etc.
            Editable = false;
        }

        field(12; "Has Children"; Boolean)
        {
            FieldClass = FlowField;
            CalcFormula = exist("JMLAP Asset" where("Parent Asset No." = field("No.")));
        }
    }
}
```

Level Calculation Algorithm:

- Level 0: No parent (root assets)
- Level N: Parent's Level + 1
- Recursive calculation with circular reference detection
- Cached in Level field for performance

Navigation Methods:

- GetParent(): Returns parent asset

- GetChildren(): Returns child assets
- GetRoot(): Climbs hierarchy to root
- GetPath(): Returns full path from root
- GetAllDescendants(): Recursive descendant list

---

3. OBJECTS TO CREATE

3.1 Object ID Allocation Strategy

Range: 70182300-70182449 (150 objects)

Allocation Plan:

- **70182300-70182329:** Configuration & Setup (30 objects)
- **70182330-70182359:** Core Asset Management (30 objects)
- **70182360-70182379:** Component & BOM Management (20 objects)
- **70182380-70182399:** BC Extensions & Integration (20 objects)
- **70182400-70182419:** Supporting Enums & Utilities (20 objects)
- **70182420-70182449:** Reserved for Future Features (30 objects)

3.2 Detailed Object List

CONFIGURATION & SETUP (70182300-70182329)

ID	Type	Name	Purpose	Priority
70182300	Table	`JMLAP Asset Pro Setup`	Singleton setup table	MUST
70182301	Table	`JMLAP Asset Type`	Asset type configuration	MUST
70182302	Table	`JMLAP Industry Template`	Pre-configured industry templates	SHOULD
70182303	Table	`JMLAP Asset Attribute Def`	Custom attribute definitions	SHOULD
70182304	Table	`JMLAP Asset Attribute Value`	Custom attribute values	SHOULD
70182305	Table	`JMLAP Contact Type Config`	Configurable contact types	SHOULD
70182306	Table	`JMLAP Status Reason`	Status change reason codes	SHOULD
70182310	Page	`JMLAP Asset Pro Setup Card`	Setup page	MUST
70182311	Page	`JMLAP Asset Types`	Asset type list	MUST
70182312	Page	`JMLAP Asset Type Card`	Asset type configuration	MUST
70182313	Page	`JMLAP Industry Templates`	Industry template list	SHOULD
70182314	Page	`JMLAP Contact Type Config`	Contact type setup	SHOULD
70182315	Page	`JMLAP Status Reasons`	Status reason codes	SHOULD

CORE ASSET MANAGEMENT (70182330-70182359)

ID	Type	Name	Purpose	Priority
70182330	Table	`JMLAP Asset`	Main asset table (unlimited	MUST

70182330	Table	`JMLAP Asset`	hierarchy)	MUST
70182331	Table	`JMLAP Asset Comment Line`	Comments/notes for assets	MUST
70182332	Table	`JMLAP Customer Asset`	Many-to-many customer links	MUST
70182333	Table	`JMLAP Asset Transition`	Asset movement history	MUST
70182334	Table	`JMLAP Asset Location`	Location relationship tracking	SHOULD
70182335	Table	`JMLAP Asset Vendor`	Vendor relationship tracking	SHOULD
70182340	Page	`JMLAP Asset List`	Main asset list	MUST
70182341	Page	`JMLAP Asset Card`	Asset card with all details	MUST
70182342	Page	`JMLAP Asset Tree`	Hierarchical tree view	SHOULD
70182343	Page	`JMLAP Asset Comment Sheet`	Comment management	MUST
70182344	Page	`JMLAP Asset Comment Subform`	Inline comments	MUST
70182345	Page	`JMLAP Customer Assets`	Customer-asset links	MUST
70182346	Page	`JMLAP Asset Transitions`	Transition history	MUST
70182347	Page	`JMLAP Asset FactBox`	Asset information panel	SHOULD
70182348	Page	`JMLAP Asset Picture`	Picture management	SHOULD
70182349	Page	`JMLAP Child Assets Subform`	Child asset list	SHOULD

COMPONENT & BOM MANAGEMENT (70182360-70182379)

ID	Type	Name	Purpose	Priority
70182360	Table	`JMLAP Component Group`	Component classification level 1	SHOULD
70182361	Table	`JMLAP Component Type`	Component classification level 2	SHOULD
70182362	Table	`JMLAP Asset Component`	Components attached to assets	SHOULD
70182363	Table	`JMLAP Component BOM`	Parts list for components	SHOULD
70182364	Table	`JMLAP Operation Mode`	Component operating modes	OPTIONAL
70182365	Table	`JMLAP Development Type`	Component development types	OPTIONAL
70182370	Page	`JMLAP Component Groups`	Component group list	SHOULD
70182371	Page	`JMLAP Component Types`	Component type list	SHOULD
70182372	Page	`JMLAP Component List`	Component list	SHOULD
70182373	Page	`JMLAP Component Card`	Component details	SHOULD
70182374	Page	`JMLAP Component Subform`	Inline component list	SHOULD
70182375	Page	`JMLAP Component BOM List`	BOM management	SHOULD
70182376	Page	`JMLAP Component BOM Subform`	Inline BOM	SHOULD

BC EXTENSIONS & INTEGRATION (70182380-70182399)

ID	Type	Name	Purpose	Priority
70182380	TableExt	`JMLAP Customer Ext`	Customer → Asset counts	MUST
70182381	TableExt	`JMLAP Sales Header Ext`	Sales Header → Asset fields	MUST
70182382	TableExt	`JMLAP Sales Line Ext`	Sales Line → Asset context	MUST
70182383	TableExt	`JMLAP Location Ext`	Location → Asset tracking	SHOULD
70182384	TableExt	`JMLAP Vendor Ext`	Vendor → Asset links	SHOULD
70182385	PageExt	`JMLAP Customer Card Ext`	Customer Card → Asset FactBox	MUST

70182386	PageExt	`JMLAP Sales Order Ext`	Sales Order → Asset selection	MUST
70182387	PageExt	`JMLAP Sales Quote Ext`	Sales Quote → Asset selection	MUST
70182388	PageExt	`JMLAP Sales Invoice Ext`	Sales Invoice → Asset display	SHOULD
70182389	Codeunit	`JMLAP Asset Mgt`	Core asset management logic	MUST
70182390	Codeunit	`JMLAP Sales Integration`	Sales document integration	MUST
70182391	Codeunit	`JMLAP Hierarchy Mgt`	Hierarchy navigation & validation	MUST
70182392	Codeunit	`JMLAP Events Handler`	Event subscribers for BC	MUST

SUPPORTING ENUMS & UTILITIES (70182400-70182419)

ID	Type	Name	Purpose	Priority
70182400	Enum	`JMLAP Asset Status`	Active, Under Maintenance, etc.	MUST
70182401	Enum	`JMLAP Location Type`	Warehouse, Customer Site, etc.	MUST
70182402	Enum	`JMLAP Contact Type`	Owner, Operator, Service Provider	MUST
70182403	Enum	`JMLAP Comment Table`	Asset, Component, BOM	MUST
70182404	Enum	`JMLAP Industry Template`	Marine, Construction, Medical, etc.	MUST
70182405	Enum	`JMLAP Transition Type`	Sale, Transfer, Service, etc.	SHOULD
70182406	Enum	`JMLAP Component Status`	Operational, Faulty, Replaced	SHOULD

Total Objects Allocated: ~70 objects (47% of available range) **Reserved for Future:** 70182420-70182449 (30 objects / 20%)

---

4. IMPLEMENTATION APPROACH

4.1 Phased Development Strategy

Phase 1.1: Foundation (Core Objects)

1. Setup table & page
2. Asset Type configuration
3. Asset table with unlimited hierarchy
4. Basic Asset List & Card pages
5. Hierarchy management codeunit

Phase 1.2: Asset Management 6. Customer Asset relationships 7. Asset Comments 8. Asset Transitions 9. Customer extension 10. FactBoxes and subforms

Phase 1.3: BC Sales Integration 11. Sales Header extension 12. Sales Line extension 13. Sales Order page extension 14. Sales integration codeunit 15. Event handlers

Phase 1.4: Components & BOM (Optional for MVP) 16. Component Group/Type tables 17. Asset Component table 18. Component BOM table 19. Component pages 20. BOM management pages

Phase 1.5: Advanced Features (Optional) 21. Industry Templates 22. Custom Attributes 23. Configurable Terminology 24. Asset Tree View 25. Advanced reporting

4.2 Unlimited Hierarchy Implementation Details

Table Structure:

```
table 70182330 "JMLAP Asset"
{
    Caption = 'Asset';
    DataClassification = CustomerContent;

    fields
    {
        // PRIMARY KEY
        field(1; "No."; Code[20])
        {
            Caption = 'No.';

            trigger OnValidate()
            begin
                if "No." <> xRec."No." then begin
                    AssetProSetup.Get();
                    NoSeriesMgt.TestManual(AssetProSetup."Asset Nos.");
                    "No. Series" := '';
                end;
            end;
        }

        // HIERARCHY FIELDS
        field(10; "Parent Asset No."; Code[20])
        {
            Caption = 'Parent Asset';
            TableRelation = "JMLAP Asset" where("No." = filter(<> field("No.")));

            trigger OnValidate()
            var
                HierarchyMgt: Codeunit "JMLAP Hierarchy Mgt";
            begin
                if "Parent Asset No." <> '' then begin
                    HierarchyMgt.ValidateNoCircularReference(Rec);
                    HierarchyMgt.RecalculateLevel(Rec);
                    HierarchyMgt.InheritFromParent(Rec);
                end else begin
                    "Level" := 0;
                end;
            end;
        }

        field(11; "Level"; Integer)
        {
            Caption = 'Hierarchy Level';
            Editable = false;
        }

        field(12; "Has Children"; Boolean)
        {
            Caption = 'Has Child Assets';
            FieldClass = FlowField;
            CalcFormula = exist("JMLAP Asset" where("Parent Asset No." = field("No.")));
        }

        field(13; "Root Asset No."; Code[20])
        {
            Caption = 'Root Asset';
            TableRelation = "JMLAP Asset";
            Editable = false;
        }
        field(14; "Full Path"; Text[250])
        {
            Caption = 'Hierarchy Path';
            Editable = false;
        }

        // CORE IDENTIFICATION
        field(20; "Description"; Text[100])
        {
            Caption = 'Description';
        }

        field(21; "Description 2"; Text[100])
        {
            Caption = 'Description 2';
        }

        field(30; "Asset Type Code"; Code[20])
        {
            Caption = 'Asset Type';
            TableRelation = "JMLAP Asset Type";

            trigger OnValidate()
            begin
                if AssetType.Get("Asset Type Code") then begin
                    "Asset Type Description" := AssetType.Description;
                    // Inherit default settings from Asset Type
                end;
            end;
        }

        field(31; "Asset Type Description"; Text[50])
        {
            Caption = 'Asset Type Description';
            FieldClass = FlowField;
            CalcFormula = lookup("JMLAP Asset Type".Description where(Code = field("Asset Typ
        }

        // STATUS MANAGEMENT
        field(40; "Status"; Enum "JMLAP Asset Status")
        {
            Caption = 'Status';
        }

        field(41; "Status Date"; Date)
        {
            Caption = 'Status Date';
        }

        field(42; "Status Reason Code"; Code[10])
        {
            Caption = 'Status Reason';
            TableRelation = "JMLAP Status Reason";
        }

        // OWNERSHIP & LOCATION
        field(50; "Primary Contact Type"; Enum "JMLAP Contact Type")
        {
            Caption = 'Primary Contact Type';
        }

        field(51; "Primary Customer No."; Code[20])
        {
            Caption = 'Primary Customer';
            TableRelation = Customer;
        }
        field(52; "Primary Contact No."; Code[20])
        {
            Caption = 'Primary Contact';
            TableRelation = Contact;
        }

        field(60; "Current Location Type"; Enum "JMLAP Location Type")
        {
            Caption = 'Location Type';
        }

        field(61; "Current Location Code"; Code[20])
        {
            Caption = 'Location Code';
            TableRelation = if("Current Location Type" = const(Warehouse)) Location
                          else if("Current Location Type" = const("Customer Site")) Customer
                          else if("Current Location Type" = const(Vendor)) Vendor;
        }

        // LIFECYCLE MANAGEMENT
        field(100; "Acquisition Date"; Date)
        {
            Caption = 'Acquisition Date';
        }

        field(101; "In-Service Date"; Date)
        {
            Caption = 'In-Service Date';
        }

        field(102; "Manufacturer Code"; Code[10])
        {
            Caption = 'Manufacturer';
            TableRelation = Manufacturer;
        }

        field(103; "Model No."; Code[50])
        {
            Caption = 'Model No.';
        }

        field(104; "Serial No."; Code[50])
        {
            Caption = 'Serial No.';
        }

        field(105; "Year of Manufacture"; Integer)
        {
            Caption = 'Year of Manufacture';
        }

        // FINANCIAL TRACKING
        field(150; "Acquisition Cost"; Decimal)
        {
            Caption = 'Acquisition Cost';
        }

        field(151; "Current Book Value"; Decimal)
        {
            Caption = 'Current Book Value';
        }

        // DIMENSIONS
        field(200; "Global Dimension 1 Code"; Code[20])
        {
            Caption = 'Global Dimension 1 Code';
            TableRelation = "Dimension Value".Code where("Global Dimension No." = const(1));
        }

        field(201; "Global Dimension 2 Code"; Code[20])
        {
            Caption = 'Global Dimension 2 Code';
            TableRelation = "Dimension Value".Code where("Global Dimension No." = const(2));
        }

        // MEDIA
        field(300; Picture; MediaSet)
        {
            Caption = 'Picture';
        }

        // SYSTEM FIELDS
        field(900; "No. Series"; Code[20])
        {
            Caption = 'No. Series';
            TableRelation = "No. Series";
        }

        field(901; "Created By"; Code[50])
        {
            Caption = 'Created By';
            Editable = false;
        }

        field(902; "Created Date Time"; DateTime)
        {
            Caption = 'Created Date Time';
            Editable = false;
        }

        field(903; "Last Modified By"; Code[50])
        {
            Caption = 'Last Modified By';
            Editable = false;
        }

        field(904; "Last Modified Date Time"; DateTime)
        {
            Caption = 'Last Modified Date Time';
            Editable = false;
        }
    }

    keys
    {
        key(PK; "No.")
        {
            Clustered = true;
        }
        key(Parent; "Parent Asset No.", "No.")
        {
        }
        key(Type; "Asset Type Code", "No.")
        {
        }
        key(Level; "Level", "No.")
        {
        }
        key(Status; "Status", "No.")
        {
        }
    }

    trigger OnInsert()
    begin
        if "No." = '' then begin
            AssetProSetup.Get();
            AssetProSetup.TestField("Asset Nos.");
            NoSeriesMgt.InitSeries(AssetProSetup."Asset Nos.", xRec."No. Series", 0D, "No.",
        end;

        "Created By" := CopyStr(UserId(), 1, 50);
        "Created Date Time" := CurrentDateTime();

        UpdateHierarchyFields();
    end;

    trigger OnModify()
    begin
        "Last Modified By" := CopyStr(UserId(), 1, 50);
        "Last Modified Date Time" := CurrentDateTime();
    end;

    procedure UpdateHierarchyFields()
    var
        HierarchyMgt: Codeunit "JMLAP Hierarchy Mgt";
    begin
        HierarchyMgt.RecalculateLevel(Rec);
        HierarchyMgt.UpdateRootAsset(Rec);
        HierarchyMgt.UpdateFullPath(Rec);
    end;
}
```

Hierarchy Management Codeunit:

```
codeunit 70182391 "JMLAP Hierarchy Mgt"
{
    /// <summary>
    /// Validates that setting the parent asset won't create a circular reference.
    /// </summary>
    procedure ValidateNoCircularReference(var Asset: Record "JMLAP Asset")
    var
        TempAsset: Record "JMLAP Asset" temporary;
        CurrentAssetNo: Code[20];
        MaxIterations: Integer;
    begin
        if Asset."Parent Asset No." = '' then
            exit;

        // Build path from proposed parent to root
        CurrentAssetNo := Asset."Parent Asset No.";
        MaxIterations := 100; // Safety limit

        repeat
            if CurrentAssetNo = Asset."No." then
                Error('Circular reference detected: Asset %1 cannot be its own ancestor.', As

            if not TempAsset.Get(CurrentAssetNo) then begin
                TempAsset."No." := CurrentAssetNo;
                TempAsset.Insert();
            end else
                Error('Circular reference detected in hierarchy path.');

            if TempAsset.Get(CurrentAssetNo) then
                CurrentAssetNo := TempAsset."Parent Asset No."
            else
                CurrentAssetNo := '';

            MaxIterations -= 1;
            if MaxIterations <= 0 then
                Error('Maximum hierarchy depth exceeded (100 levels).');
        until CurrentAssetNo = '';
    end;

    /// <summary>
    /// Recalculates the level field based on parent hierarchy.
    /// </summary>
    procedure RecalculateLevel(var Asset: Record "JMLAP Asset")
    var
        ParentAsset: Record "JMLAP Asset";
    begin
        if Asset."Parent Asset No." = '' then begin
            Asset."Level" := 0;
        end else begin
            if ParentAsset.Get(Asset."Parent Asset No.") then
                Asset."Level" := ParentAsset."Level" + 1
            else
                Asset."Level" := 0;
        end;
    end;

    /// <summary>
    /// Updates the root asset reference.
    /// </summary>
    procedure UpdateRootAsset(var Asset: Record "JMLAP Asset")
    var
        CurrentAsset: Record "JMLAP Asset";
    begin
        if Asset."Parent Asset No." = '' then begin
            Asset."Root Asset No." := Asset."No.";
            exit;
        end;

        CurrentAsset := Asset;
        while CurrentAsset."Parent Asset No." <> '' do begin
            if not CurrentAsset.Get(CurrentAsset."Parent Asset No.") then
                break;
        end;

        Asset."Root Asset No." := CurrentAsset."No.";
    end;

    /// <summary>
    /// Builds full hierarchy path (e.g., "ROOT > LEVEL1 > LEVEL2").
    /// </summary>
    procedure UpdateFullPath(var Asset: Record "JMLAP Asset")
    var
        PathText: Text[250];
        CurrentAsset: Record "JMLAP Asset";
    begin
        PathText := Asset."No.";
        CurrentAsset := Asset;

        while CurrentAsset."Parent Asset No." <> '' do begin
            if CurrentAsset.Get(CurrentAsset."Parent Asset No.") then
                PathText := CurrentAsset."No." + ' > ' + PathText
            else
                break;
        end;

        Asset."Full Path" := CopyStr(PathText, 1, 250);
    end;

    /// <summary>
    /// Gets all child assets (direct children only).
    /// </summary>
```

```
    procedure GetChildren(AssetNo: Code[20]; var ChildAssets: Record "JMLAP Asset")
    begin
        ChildAssets.Reset();
        ChildAssets.SetRange("Parent Asset No.", AssetNo);
    end;

    /// <summary>
    /// Gets all descendant assets (recursive).
    /// </summary>
    procedure GetAllDescendants(AssetNo: Code[20]; var DescendantAssets: Record "JMLAP Asset"
    var
        ChildAsset: Record "JMLAP Asset";
    begin
        ChildAsset.SetRange("Parent Asset No.", AssetNo);
        if ChildAsset.FindSet() then
            repeat
                DescendantAssets := ChildAsset;
                DescendantAssets.Insert();
                GetAllDescendants(ChildAsset."No.", DescendantAssets); // Recursive call
            until ChildAsset.Next() = 0;
    end;

    /// <summary>
    /// Inherits configuration from parent asset.
    /// </summary>
    procedure InheritFromParent(var Asset: Record "JMLAP Asset")
    var
        ParentAsset: Record "JMLAP Asset";
    begin
        if Asset."Parent Asset No." = '' then
            exit;

        if not ParentAsset.Get(Asset."Parent Asset No.") then
            exit;

        // Inherit Asset Type if not set
        if Asset."Asset Type Code" = '' then
            Asset."Asset Type Code" := ParentAsset."Asset Type Code";

        // Inherit Primary Contact if not set
        if Asset."Primary Customer No." = '' then begin
            Asset."Primary Contact Type" := ParentAsset."Primary Contact Type";
            Asset."Primary Customer No." := ParentAsset."Primary Customer No.";
            Asset."Primary Contact No." := ParentAsset."Primary Contact No.";
        end;
    end;
}
```

4.3 Universal Multi-Industry Configuration

Asset Type Table (Configuration):

```
table 70182301 "JMLAP Asset Type"
{
    Caption = 'Asset Type';
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Code"; Code[20])
        {
            Caption = 'Code';
        }
```

```
field(2; "Description"; Text[100])
{
    Caption = 'Description';
}

// INDUSTRY CONFIGURATION
field(10; "Industry Template"; Enum "JMLAP Industry Template")
{
    Caption = 'Industry Template';
}

// TERMINOLOGY CONFIGURATION
field(20; "Asset Term (Singular)"; Text[50])
{
    Caption = 'Asset Term (Singular)';
    // Examples: "Vessel", "Aircraft", "Machine", "Building", "Device"
}

field(21; "Asset Term (Plural)"; Text[50])
{
    Caption = 'Asset Term (Plural)';
    // Examples: "Vessels", "Aircraft", "Machines", "Buildings", "Devices"
}

field(22; "Component Term (Singular)"; Text[50])
{
    Caption = 'Component Term (Singular)';
    // Examples: "Engine", "System", "Equipment", "Assembly"
}

field(23; "Component Term (Plural)"; Text[50])
{
    Caption = 'Component Term (Plural)';
    // Examples: "Engines", "Systems", "Equipment", "Assemblies"
}

// FEATURE TOGGLES
field(30; "Use Component Management"; Boolean)
{
    Caption = 'Use Component Management';
    InitValue = true;
}

field(31; "Use BOM Management"; Boolean)
{
    Caption = 'Use BOM Management';
    InitValue = true;
}

field(32; "Use Serial No. Tracking"; Boolean)
{
    Caption = 'Use Serial No. Tracking';
    InitValue = true;
}

field(33; "Use Location Tracking"; Boolean)
{
    Caption = 'Use Location Tracking';
    InitValue = true;
}

field(34; "Use Hierarchy"; Boolean)
{
    Caption = 'Use Hierarchy (Parent/Child)';
    InitValue = true;
}
```

```
        // NUMBER SERIES
        field(50; "Asset Nos."; Code[20])
        {
            Caption = 'Asset Number Series';
            TableRelation = "No. Series";
        }

        field(51; "Component Nos."; Code[20])
        {
            Caption = 'Component Number Series';
            TableRelation = "No. Series";
        }
    }

    keys
    {
        key(PK; "Code")
        {
            Clustered = true;
        }
    }

    trigger OnInsert()
    begin
        // Set defaults based on industry template
        InitializeFromTemplate();
    end;

    procedure InitializeFromTemplate()
    begin
        case "Industry Template" of
            "Industry Template"::Marine:
                begin
                    "Asset Term (Singular)" := 'Vessel';
                    "Asset Term (Plural)" := 'Vessels';
                    "Component Term (Singular)" := 'Engine';
                    "Component Term (Plural)" := 'Engines';
                end;
            "Industry Template"::Construction:
                begin
                    "Asset Term (Singular)" := 'Machine';
                    "Asset Term (Plural)" := 'Machines';
                    "Component Term (Singular)" := 'System';
                    "Component Term (Plural)" := 'Systems';
                end;
            "Industry Template"::Medical:
                begin
                    "Asset Term (Singular)" := 'Device';
                    "Asset Term (Plural)" := 'Devices';
                    "Component Term (Singular)" := 'Module';
                    "Component Term (Plural)" := 'Modules';
                    "Use BOM Management" := false; // Medical devices usually don't track BOM
                end;
            "Industry Template"::IT:
                begin
                    "Asset Term (Singular)" := 'Device';
                    "Asset Term (Plural)" := 'Devices';
                    "Component Term (Singular)" := 'Hardware';
                    "Component Term (Plural)" := 'Hardware';
                    "Use Component Management" := false; // Simplified for IT
                    "Use BOM Management" := false;
                end;
        end;
    end;
}
```

Industry Template Enum:

```
enum 70182404 "JMLAP Industry Template"
{
    Extensible = true;

    value(0; " ")
    {
        Caption = ' ';
    }
    value(1; Marine)
    {
        Caption = 'Marine / Maritime';
    }
    value(2; Construction)
    {
        Caption = 'Construction Equipment';
    }
    value(3; Medical)
    {
        Caption = 'Medical Equipment';
    }
    value(4; Aircraft)
    {
        Caption = 'Aircraft / Aviation';
    }
    value(5; IT)
    {
        Caption = 'IT Infrastructure';
    }
    value(6; Manufacturing)
    {
        Caption = 'Manufacturing Equipment';
    }
    value(7; Transportation)
    {
        Caption = 'Transportation / Fleet';
    }
    value(8; RealEstate)
    {
        Caption = 'Real Estate / Facilities';
    }
    value(9; Agriculture)
    {
        Caption = 'Agricultural Equipment';
    }
    value(10; Energy)
    {
        Caption = 'Energy / Utilities';
    }
    value(99; Custom)
    {
        Caption = 'Custom / Other';
    }
}
```

4.4 Sales Order Integration

Sales Header Extension:

```
tableextension 70182381 "JMLAP Sales Header Ext" extends "Sales Header"
{
    fields
    {
        field(70182300; "JMLAP Asset Type"; Code[20])
        {
            Caption = 'Asset Type';
            TableRelation = "JMLAP Asset Type";
            DataClassification = CustomerContent;
        }

        field(70182301; "JMLAP Asset No."; Code[20])
        {
            Caption = 'Asset No.';
            TableRelation = "JMLAP Asset" where("Asset Type Code" = field("JMLAP Asset Type")
            DataClassification = CustomerContent;

            trigger OnValidate()
            var
                SalesIntegration: Codeunit "JMLAP Sales Integration";
            begin
                SalesIntegration.UpdateHeaderFromAsset(Rec);
            end;
        }

        field(70182302; "JMLAP Asset Description"; Text[100])
        {
            Caption = 'Asset Description';
            FieldClass = FlowField;
            CalcFormula = lookup("JMLAP Asset".Description where("No." = field("JMLAP Asset N
        }

        field(70182303; "JMLAP Component No."; Code[20])
        {
            Caption = 'Component No.';
            TableRelation = "JMLAP Asset Component" where("Asset No." = field("JMLAP Asset No
            DataClassification = CustomerContent;

            trigger OnValidate()
            var
                SalesIntegration: Codeunit "JMLAP Sales Integration";
            begin
                SalesIntegration.UpdateHeaderFromComponent(Rec);
            end;
        }

        field(70182310; "JMLAP Primary Contact Type"; Enum "JMLAP Contact Type")
        {
            Caption = 'Primary Contact Type';
            DataClassification = CustomerContent;
        }

        field(70182311; "JMLAP Primary Customer No."; Code[20])
        {
            Caption = 'Primary Customer';
            TableRelation = Customer;
            DataClassification = CustomerContent;
        }

        field(70182312; "JMLAP Primary Contact No."; Code[20])
        {
            Caption = 'Primary Contact';
            TableRelation = Contact;
            DataClassification = CustomerContent;
        }
    }
}
```

Sales Integration Codeunit:

```
codeunit 70182390 "JMLAP Sales Integration"
{
    /// <summary>
    /// Updates sales header fields from selected asset.
    /// </summary>
    procedure UpdateHeaderFromAsset(var SalesHeader: Record "Sales Header")
    var
        Asset: Record "JMLAP Asset";
    begin
        if SalesHeader."JMLAP Asset No." = '' then
            exit;

        if not Asset.Get(SalesHeader."JMLAP Asset No.") then
            exit;

        // Copy primary contact information
        SalesHeader."JMLAP Primary Contact Type" := Asset."Primary Contact Type";
        SalesHeader."JMLAP Primary Customer No." := Asset."Primary Customer No.";
        SalesHeader."JMLAP Primary Contact No." := Asset."Primary Contact No.";

        // Update Asset Type
        SalesHeader."JMLAP Asset Type" := Asset."Asset Type Code";

        // Validate customer relationship
        ValidateCustomerAssetRelationship(SalesHeader);
    end;

    /// <summary>
    /// Updates sales header fields from selected component.
    /// </summary>
    procedure UpdateHeaderFromComponent(var SalesHeader: Record "Sales Header")
    var
        Component: Record "JMLAP Asset Component";
        Asset: Record "JMLAP Asset";
    begin
        if SalesHeader."JMLAP Component No." = '' then
            exit;

        if not Component.Get(SalesHeader."JMLAP Component No.") then
            exit;

        // Sync asset from component
        if Component."Asset No." <> SalesHeader."JMLAP Asset No." then begin
            SalesHeader."JMLAP Asset No." := Component."Asset No.";
            UpdateHeaderFromAsset(SalesHeader);
        end;
    end;

    /// <summary>
    /// Validates customer is linked to the selected asset.
    /// </summary>
    procedure ValidateCustomerAssetRelationship(var SalesHeader: Record "Sales Header")
    var
        CustomerAsset: Record "JMLAP Customer Asset";
    begin
        if (SalesHeader."JMLAP Asset No." = '') or (SalesHeader."Sell-to Customer No." = '')
            exit;

        CustomerAsset.SetRange("Asset No.", SalesHeader."JMLAP Asset No.");
        CustomerAsset.SetRange("Customer No.", SalesHeader."Sell-to Customer No.");

        if not CustomerAsset.FindFirst() then
            Message('Warning: Customer %1 is not linked to Asset %2.',
                    SalesHeader."Sell-to Customer No.",
                    SalesHeader."JMLAP Asset No.");
    end;
    /// <summary>
    /// Cascades asset context from header to all lines.
    /// </summary>
    procedure CascadeAssetToLines(var SalesHeader: Record "Sales Header")
    var
        SalesLine: Record "Sales Line";
    begin
        SalesLine.SetRange("Document Type", SalesHeader."Document Type");
        SalesLine.SetRange("Document No.", SalesHeader."No.");

        if SalesLine.FindSet(true) then
            repeat
                SalesLine."JMLAP Asset No." := SalesHeader."JMLAP Asset No.";
                SalesLine."JMLAP Component No." := SalesHeader."JMLAP Component No.";
                SalesLine.Modify();
            until SalesLine.Next() = 0;
    end;

    [EventSubscriber(ObjectType::Table, Database::"Sales Header", OnAfterModifyEvent, '', fal
    local procedure OnAfterSalesHeaderModify(var Rec: Record "Sales Header"; var xRec: Record
    begin
        // If asset changed, cascade to lines
        if Rec."JMLAP Asset No." <> xRec."JMLAP Asset No." then
            CascadeAssetToLines(Rec);
    end;
}
```

---

5. TEST STRATEGY

5.1 Test Coverage Requirements

Mandatory Test Scenarios (Minimum 3 per feature):

1. Happy path (normal, successful execution)
2. Error case (invalid input or error condition)
3. Edge case (boundary condition or unusual scenario)

5.2 Test Object Allocation

Test App: 50100-50199 (100 objects available)

Allocation:

- **50100-50129:** Unit Tests (30 objects)
- **50130-50169:** Integration Tests (40 objects)
- **50170-50189:** Test Libraries & Helpers (20 objects)
- **50190-50199:** Scenario/E2E Tests (10 objects)

5.3 Priority Test Scenarios

P1: Unlimited Hierarchy Tests (50100-50109)

Test ID	Test Name	Scenario	Expected Result
50100	CreateAsset_NoParent_LevelIsZero	Create asset without	Level = 0, Root = Self

		parent	
50101	CreateAsset_WithParent_LevelIsOne	Create asset with parent	Level = 1, Root = Parent
50102	CreateAsset_DeepHierarchy_CorrectLevel	Create 5-level hierarchy	Levels calculated correctly
50103	SetParent_CircularReference_ThrowsError	Set parent to own descendant	Error: Circular reference
50104	SetParent_SelfReference_Blocked	Set parent to self	Error: Self-reference blocked
50105	GetChildren_MultipleChildren_ReturnsAll	Get children of parent asset	All child assets returned
50106	GetDescendants_Recursive_ReturnsAll	Get all descendants recursively	All descendants returned
50107	DeleteAsset_WithChildren_BlockedOrCascade	Delete asset with children	Either blocked or cascade delete
50108	UpdateParent_RecalculatesPath_PathUpdated	Change parent asset	Full path recalculated
50109	InheritFromParent_CopiesSettings_SettingsInherited	Create child asset	Inherits Asset Type, Contacts

P2: Asset Management Tests (50110-50119)

Test ID	Test Name	Scenario	Expected Result
50110	CreateAsset_ValidData_CreatesRecord	Create asset with valid data	Asset created successfully
50111	CreateAsset_NoAssetType_ThrowsError	Create asset without type	Error: Asset Type required
50112	ModifyAsset_UpdateStatus_StatusChanged	Update asset status	Status updated, history logged
50113	DeleteAsset_NoChildren_Deleted	Delete asset without children	Asset deleted
50114	AddComment_ValidText_CommentCreated	Add comment to asset	Comment saved with timestamp
50115	LinkCustomer_ValidCustomer_LinkCreated	Link customer to asset	Customer-Asset link created
50116	LinkCustomer_Duplicate_ThrowsError	Link same customer twice	Error: Duplicate link
50117	TransitionAsset_NewLocation_TransitionLogged	Move asset to new location	Transition record created
50118	TransitionAsset_FromDocument_DocumentLinked	Transition via Sales Order	Document reference saved
50119	CalculateAssetCount_MultipleAssets_CorrectCount	Count assets per customer	Correct count returned

P3: Sales Integration Tests (50120-50129)

Test ID	Test Name	Scenario	Expected Result
50120	SelectAsset_OnSalesOrder_FieldsPopulated	Select asset on sales order	Contact fields populated
50121	SelectAsset_CustomerNotLinked_ShowsWarning	Select asset for unlinked customer	Warning message displayed
50122	ChangeAsset_OnHeader_LinesCascaded	Change asset on header	All lines updated
50123	SelectComponent_OnSalesLine_AssetInherited	Select component on line	Asset populated from component
50124	PostSalesOrder_WithAsset_TransitionCreated	Post sales order with asset	Transition record created
50125	DeleteSalesLine_WithAsset_NoOrphanData	Delete sales line	No orphan data remains

50126	ValidateCustomer_AssetSet_ChecksRelationship	Validate customer with asset	Relationship validated
50127	CopyDocument_WithAsset_AssetCopied	Copy sales document	Asset context copied
50128	ArchiveSalesOrder_WithAsset_DataArchived	Archive sales order	Asset data preserved
50129	PostAndPrint_WithAsset_AssetOnReport	Post and print	Asset appears on document

P4: Multi-Industry Configuration Tests (50130-50139)

Test ID	Test Name	Scenario	Expected Result
50130	CreateAssetType_MarineTemplate_TerminologySet	Create Marine asset type	Terminology = "Vessel", "Engine"
50131	CreateAssetType_MedicalTemplate_TerminologySet	Create Medical asset type	Terminology = "Device", "Module"
50132	CreateAssetType_ITTemplate_FeaturesDisabled	Create IT asset type	BOM disabled, simplified
50133	UpdateTerminology_UpdatesUI_LabelsChanged	Change asset type terminology	UI labels update dynamically
50134	ToggleFeature_DisableHierarchy_FieldsHidden	Disable hierarchy feature	Parent field hidden
50135	ToggleFeature_EnableBOM_PagesVisible	Enable BOM management	BOM pages accessible
50136	CreateAsset_FeatureDisabled_FieldsNotRequired	Create asset with features off	Optional fields skipped
50137	SwitchIndustry_UpdatesDefaults_SettingsChanged	Switch industry template	Defaults update
50138	CustomAttribute_AddValue_ValueStored	Add custom attribute value	Value saved and retrieved
50139	CustomAttribute_ValidateDataType_TypeEnforced	Validate attribute data type	Type validation works

P5: Test Library (50180-50189)

Object ID	Name	Purpose
50180	JMLAP Test Library	Reusable test helper procedures
50181	JMLAP Test Data Factory	Generate test assets, types, customers
50182	JMLAP Mock Data Generator	Create mock hierarchies, transitions

Test Library Key Procedures:

- `CreateTestAsset(AssetTypeCode, ParentNo, Level): AssetNo`
- `CreateTestAssetType(IndustryTemplate): AssetTypeCode`
- `CreateTestHierarchy(Levels, ChildrenPerLevel): RootAssetNo`
- `CreateTestCustomerAssetLink(AssetNo, CustomerNo)`
- `CreateTestSalesOrder(CustomerNo, AssetNo): OrderNo`
- `CleanupTestAssets(AssetNoFilter)`
- `CleanupTestSalesDocuments()`

5.4 Running Tests

Script: `Test\run-tests.ps1`

```
# Test/run-tests.ps1
param(
    [Parameter(Mandatory=$true)]
    [string]$containerName,
    [Parameter(Mandatory=$true)]
    [string]$username,

    [Parameter(Mandatory=$true)]
    [string]$password
)

Write-Host "Running Asset Pro Tests..." -ForegroundColor Cyan
Write-Host "Container: $containerName" -ForegroundColor Gray
Write-Host "Test App ID: a62ab4b7-6914-455e-b854-cb71450306c1" -ForegroundColor Gray

# Run all tests in the test app using -extensionId parameter
Run-TestsInBcContainer `
    -containerName $containerName `
    -credential (New-Object PSCredential $username, (ConvertTo-SecureString $password -AsPlai
    -extensionId "a62ab4b7-6914-455e-b854-cb71450306c1" `
    -detailed

Write-Host "Tests completed!" -ForegroundColor Green
```

Execution:

```
powershell.exe -ExecutionPolicy Bypass -File "C:\GIT\JEMEL\JML_AssetPro\Test\run-tests.ps1" -
```

5.5 Test Quality Gates

All tests must:

- □ Follow AAA pattern (Arrange-Act-Assert)
- □ Use descriptive names: `[Feature]_[Scenario]_[Expected]`
- □ Include [GIVEN], [WHEN], [THEN] comments
- □ Create unique test data (GUID-based IDs)
- □ Clean up in try-finally block
- □ Run in under 5 seconds (integration tests)
- □ Be independent (no execution order dependency)
- □ Pass 100% before feature is marked complete

---

6. CONCERNS, RISKS & ASSUMPTIONS

6.1 Technical Concerns

1. Performance with Deep Hierarchies

Concern: Recursive hierarchy navigation could be slow with 10+ levels and 1000+ assets.

Mitigation:

- Cache Level, Root Asset, and Full Path in table fields
- Use indexed keys on Parent Asset No. and Level
- Limit recursive queries to max 100 iterations
- Provide "Get Direct Children" vs "Get All Descendants" options
- Consider materialized path or nested sets if performance issues arise

Risk Level: MEDIUM

---

2. Circular Reference Prevention

Concern: User could accidentally create circular hierarchies (A → B → C → A).

Mitigation:

- Validation on Parent Asset No. field prevents self-reference
- Recursive check traverses hierarchy to root before allowing save
- MaxIterations safety limit (100 levels)
- Unit tests cover all circular reference scenarios

Risk Level: LOW (mitigated)

---

3. Unlimited Hierarchy UI Complexity

Concern: Displaying/navigating unlimited levels in UI could confuse users.

Mitigation:

- Provide both flat list and tree view pages
- Show Level field and Full Path prominently
- "Show Children" and "Go to Parent" actions
- Limit default tree view to 5 levels (expandable)
- Option to filter by Level

Risk Level: MEDIUM

---

4. Multi-Industry Configuration Complexity

Concern: Universal platform could be too generic, confusing users.

Mitigation:

- Industry Template wizard on first setup
- Pre-configured templates for 10+ industries
- Terminology automatically adapts UI labels
- Feature toggles hide unused functionality
- Simple default configuration works out-of-box

Risk Level: MEDIUM

---

5. Sales Integration Performance

Concern: Cascading asset context to many sales lines could be slow.

Mitigation:

- Only cascade when asset actually changes (not on every modify)
- Use ModifyAll for bulk line updates
- Event subscribers run after commit
- Optional: Make cascade async for large documents

Risk Level: LOW

---

6. Component vs. Asset Confusion

Concern: Users might not understand when to use Component vs. child Asset.

Mitigation:

- Clear documentation and tooltips
- Component = non-trackable assembly/part (simpler)
- Child Asset = fully tracked independent asset (more features)
- Industry templates pre-configure which to use
- Training materials with examples

Risk Level: MEDIUM

6.2 Functional Concerns

1. Asset Ownership Complexity

Concern: Multiple customers per asset (owner, operator, service provider) could be confusing.

Mitigation:

- Primary Contact prominently displayed
- Customer Asset table shows all relationships with contact type
- FactBox on Asset Card shows all linked customers
- Sales Order validation checks primary customer by default

Risk Level: LOW

---

2. Asset Transition Tracking

Concern: Manual transition logging could be forgotten by users.

Mitigation:

- Auto-create transition when sales order posted
- Validation prompt if location changed manually without transition
- Transition history visible on Asset Card
-

- Phase 2: Full Transfer Order integration (auto-create transitions)

Risk Level: MEDIUM (deferred to Phase 2)

---

3. Custom Attributes Scalability

Concern: Custom attribute framework could be slow with 100+ attributes.

Mitigation:

- Attributes stored in separate table (doesn't bloat Asset table)
- Lazy loading (only fetch when viewing attributes page)
- Index on Asset No. + Attribute Code
- Phase 1: Defer custom attributes (optional feature)

Risk Level: LOW (optional feature)

6.3 Implementation Assumptions

Assumption 1: BC Version 26.0 is stable

- No known blocking issues in BC 26.0
- AL syntax and features available as documented
- If BC version changes, re-test compatibility

Assumption 2: Symbols extracted correctly

- ObjectIndex.md is current and complete
- Base object structures haven't changed significantly
- Verify field numbers before extending

Assumption 3: 150 object IDs are sufficient

- Current plan uses ~70 objects (47%)
- 30 objects reserved for future (20%)
- If exceeded, request additional ID range

Assumption 4: Users will configure Asset Types

- Setup wizard guides initial configuration
- Industry templates provide defaults
- Documentation explains terminology options

Assumption 5: Phase 2 for advanced features

- Transfer Order integration deferred
- Purchase Order integration deferred
- Subscription Billing integration deferred
- Custom Attributes optional

- These are documented as future enhancements

6.4 Dependencies

External Dependencies:

- BC 26.0 platform
- Customer, Sales Header, Sales Line, Location, Vendor tables
- Dimension Value table (if dimension integration added)
- Contact table (if contact management added)
- No. Series setup

Internal Dependencies:

- Asset Type must be configured before creating Assets
- Parent Asset must exist before creating child
- Customer must exist before linking to Asset
- Sales Order extensions depend on core asset tables

---

7. IMPLEMENTATION PHASES (OPTIONAL SEQUENCING)

If you prefer phased delivery rather than all-at-once:

Phase 1.A: Minimal Viable Product (MVP)

Goal: Basic asset tracking with hierarchy **Timeline:** 2-3 weeks **Objects:** ~30 core objects

Includes:

- Setup table & page
- Asset Type (basic configuration only)
- Asset table with unlimited hierarchy
- Asset List & Card pages
- Customer Asset links
- Asset Comments
- Customer extension (asset counts)
- Hierarchy management codeunit
- ~15 core tests

Excludes:

- Multi-industry configuration
- Components & BOM
- Sales integration
- Advanced features

Deliverable: Users can create assets, organize hierarchies, link to customers, add comments.

Phase 1.B: Sales Integration

Goal: Asset context in sales orders **Timeline:** 1-2 weeks **Objects:** +10 objects

Adds:

- Sales Header extension
- Sales Line extension
- Sales Order page extension
- Sales integration codeunit
- Transition logging
- ~10 sales integration tests

Deliverable: Sales orders can reference assets, transitions logged on posting.

Phase 1.C: Components & BOM

Goal: Component assembly management **Timeline:** 1-2 weeks **Objects:** +15 objects

Adds:

- Component Group/Type tables
- Asset Component table
- Component BOM table
- Component pages
- BOM pages
- ~10 component tests

Deliverable: Track components and parts for assets.

Phase 1.D: Multi-Industry Configuration

Goal: Universal platform features **Timeline:** 2-3 weeks **Objects:** +15 objects

Adds:

- Industry Template enum & configuration
- Terminology configuration
- Feature toggles
- Industry Template wizard
- Custom Attributes (optional)
- ~10 configuration tests

Deliverable: Single app serves multiple industries with industry-specific terminology.

OR: Implement all phases together as originally planned (6-8 weeks total).

---

8. SUCCESS CRITERIA

8.1 Code Quality Gates

- ☐ 0 compilation errors
- ☐ 0 compilation warnings
- ☐ All object IDs within assigned range (70182300-70182449)
- ☐ All fields have Caption and ToolTip
- ☐ All fields have DataClassification
- ☐ ApplicationArea set on page fields
- ☐ No WITH statements (deprecated)
- ☐ PascalCase naming conventions
- ☐ File names follow AL best practices (no hyphens, no object numbers)
- ☐ All tables have primary keys and appropriate secondary keys

8.2 Test Quality Gates

- ☐ All tests pass (100% pass rate)
- ☐ Minimum 3 test scenarios per major feature
- ☐ All tests use AAA pattern with comments
- ☐ All tests clean up test data
- ☐ All tests run in under 5 seconds (integration tests)
- ☐ Test coverage for happy path, error case, edge case

8.3 Functional Acceptance Criteria

UC1: Create Asset Hierarchy

- ☐ Create root asset (Level 0)
- ☐ Create child asset (Level 1)
- ☐ Create grandchild asset (Level 2)
- ☐ View full hierarchy path
- ☐ Navigate between parent/child
- ☐ Prevent circular references

UC2: Link Customer to Asset

- ☐ Create customer-asset relationship
- ☐ View all assets for customer
- ☐ View all customers for asset
- ☐ Delete relationship

UC3: Sales Order with Asset

- ☐ Select asset on sales order
- ☐ Asset contact information populated
- ☐ Create sales line with asset context
- ☐ Post sales order
- ☐ Asset transition logged on posting

UC4: Multi-Industry Configuration

- ▯ Create Asset Type with Marine template
- ▯ Terminology changes to "Vessel" and "Engine"
- ▯ Create Asset Type with Medical template
- ▯ Terminology changes to "Device" and "Module"
- ▯ Feature toggles hide/show functionality

UC5: Component & BOM Management

- ▯ Create component for asset
- ▯ Add items to component BOM
- ▯ View BOM for component
- ▯ Update BOM quantities

---

9. NEXT STEPS

9.1 Approval Checklist

Please review and approve:

- Overall architecture approach (unlimited hierarchy, universal platform)
- Object list and ID assignments (70182300-70182449)
- Phasing approach (all-at-once or incremental?)
- Test strategy and coverage plan
- Concerns and mitigation plans
- Success criteria

9.2 Questions for User

Before implementation, please clarify:

1. **Phasing:** Implement all at once (6-8 weeks) or in phases (1.A → 1.B → 1.C → 1.D)?
2. **Components:** Include Component & BOM management in Phase 1, or defer to Phase 2?
3. **Custom Attributes:** Include in Phase 1, or defer as optional Phase 2 feature?
4. **Asset Tree View:** Priority for hierarchical tree UI, or list view sufficient initially?
5. **Dimension Integration:** Should Asset No. auto-create Dimension Values (Rollsberg approach)?
6. **Transition Automation:** Auto-create transitions on Sales Order posting, or manual only?

9.3 Approval Decision

▯▯ STOP - APPROVAL REQUIRED

This is Phase 3 (Planning) of the AL Development Core workflow. According to the workflow, **explicit approval is required** before proceeding to Phase 4 (Implementation).

To approve, please respond with one of:

- "Approved" or "Proceed" or "Go ahead"
- "Approved with changes: [specify changes]"
- "Not approved: [specify concerns]"

After approval:

- Phase 4: Implementation (create all objects, build, test)
- Phase 5: Documentation (user-facing changes)
- Phase 6: Summary (final report)

---

10. APPENDICES

Appendix A: File Naming Reference

Tables: `JMLAPAsset.Table.al` **Pages:** `JMLAPAssetCard.Page.al` **Codeunits:** `JMLAPHierarchyMgt.Codeunit.al` **Enums:** `JMLAPAssetStatus.Enum.al` **Table Extensions:** `JMLAPCustomerExt.TableExt.al` **Page Extensions:** `JMLAPSalesOrderExt.PageExt.al`

Rules:

- No hyphens
- No object numbers
- Proper case for object type
- JMLAP prefix on all custom objects

Appendix B: Folder Structure

```
AL/
├── src/
│   ├── setup/
│   │   ├── JMLAPAssetProSetup.Table.al
│   │   ├── JMLAPAssetProSetupCard.Page.al
│   │   ├── JMLAPAssetType.Table.al
│   │   ├── JMLAPAssetTypes.Page.al
│   │   └── JMLAPAssetTypeCard.Page.al
│   ├── asset/
│   │   ├── JMLAPAsset.Table.al
│   │   ├── JMLAPAssetList.Page.al
│   │   ├── JMLAPAssetCard.Page.al
│   │   ├── JMLAPAssetCommentLine.Table.al
│   │   ├── JMLAPAssetCommentSheet.Page.al
│   │   ├── JMLAPCustomerAsset.Table.al
│   │   ├── JMLAPCustomerAssets.Page.al
│   │   ├── JMLAPAssetTransition.Table.al
│   │   └── JMLAPAssetTransitions.Page.al
│   ├── component/
│   │   ├── JMLAPComponentGroup.Table.al
│   │   ├── JMLAPComponentType.Table.al
│   │   ├── JMLAPAssetComponent.Table.al
│   │   ├── JMLAPComponentBOM.Table.al
│   │   ├── JMLAPComponentList.Page.al
│   │   ├── JMLAPComponentCard.Page.al
│   │   └── JMLAPComponentBOMList.Page.al
│   ├── integration/
│   │   ├── JMLAPCustomerExt.TableExt.al
│   │   ├── JMLAPSalesHeaderExt.TableExt.al
│   │   ├── JMLAPSalesLineExt.TableExt.al
│   │   ├── JMLAPSalesOrderExt.PageExt.al
│   │   └── JMLAPSalesQuoteExt.PageExt.al
│   ├── codeunit/
│   │   ├── JMLAPAssetMgt.Codeunit.al
│   │   ├── JMLAPHierarchyMgt.Codeunit.al
│   │   ├── JMLAPSalesIntegration.Codeunit.al
│   │   └── JMLAPEventsHandler.Codeunit.al
│   └── enum/
│       ├── JMLAPAssetStatus.Enum.al
│       ├── JMLAPLocationType.Enum.al
│       ├── JMLAPContactType.Enum.al
│       ├── JMLAPCommentTable.Enum.al
│       └── JMLAPIndustryTemplate.Enum.al
Test/
└── src/
    ├── JMLAPHierarchyTests.Codeunit.al
    ├── JMLAPAssetMgtTests.Codeunit.al
    ├── JMLAPSalesIntegTests.Codeunit.al
    ├── JMLAPIndustryConfigTests.Codeunit.al
    └── JMLAPTestLibrary.Codeunit.al
```

Appendix C: AL Best Practices Checklist

- **Naming:** PascalCase, descriptive, no abbreviations
- **File Naming:** No hyphens, no object numbers, proper case
- **Properties:** Caption, ToolTip, DataClassification on all fields
- **ApplicationArea:** Set on all page fields
- **WITH Statements:** NEVER use (deprecated)
- **Access Modifiers:** Internal by default for helper procedures
- **Error Messages:** Include context (object no., customer, etc.)
- **FlowFields:** Use for counts and lookups instead of storing
- **Triggers:** Delegate complex logic to procedures/codeunits
- **Event Subscribers:** Use for BC integration hooks
- **Keys:** Define appropriate keys for common queries
- **Performance:** SetLoadFields() in loops, avoid database reads inside loops

Appendix D: Rollsberg Comparison

Rollsberg (Marine-Specific):

- 3 fixed levels: Item Object → Engine → Engine BOM
- Marine terminology hardcoded
- Owner + Tech Management contacts (6 fields)
- Engine BOM auto-propagation (risky)
- ~80 objects total
- Sales integration comprehensive

Asset Pro (Universal):

- Unlimited levels: Asset → Asset → Asset (recursive)
- Configurable terminology per industry
- Flexible contact types (configurable)
- Controlled propagation (optional)
- ~70 core objects + 30 reserved
- Sales integration (Phase 1), other documents (Phase 2)
- Custom attributes framework (optional)
- Industry templates (Marine, Construction, Medical, IT, etc.)

Key Improvements:

1. □ Unlimited hierarchy vs. 3 fixed levels
2. □ Universal platform vs. marine-only
3. □ Configurable terminology vs. hardcoded
4. □ Simpler parent/child model vs. separate tables
5. □ Feature toggles for flexibility
6. □ Industry templates for fast setup

END OF PHASE 3 PLAN

Status: □□ AWAITING USER APPROVAL

Prepared By: Claude Code (AL Development Core Workflow) **Date:** 2025-11-02 **Version:** 1.0