Asset Pro - Comprehensive Analysis and Planning Document v2

Project: Asset Pro - Multi-Industry Asset Management for Business Central **Publisher:** JEMEL **App Prefix:** JML **Date:** 2025-11-05 **Status:** Architecture Analysis - Awaiting Approval **Workflow Mode:** Analysis (Relaxed) **Document Version:** 2.0

Executive Summary

Vision Statement

> "Track any asset, any industry, your way - with unlimited flexibility and complete terminology adaptation"

The Challenge

Traditional asset management solutions force businesses into rigid structures:

- Fixed 2-3 level hierarchies
- Generic terminology ("Asset", "Component", "Part")
- Either classification OR physical relationships, not both
- One-size-fits-all approach that fits nobody perfectly

The Solution: Asset Pro's Two-Structure Architecture

Asset Pro introduces a revolutionary **dual-structure architecture** that separates concerns:

STRUCTURE 1: Classification Hierarchy (Organizational)

- Unlimited configurable levels per industry
- Dynamic terminology that adapts completely
- Example: "Fleet" → "Vessel Type" → "Vessel Model" → "Vessel Unit"
- Used for: Organization, filtering, reporting, access control

STRUCTURE 2: Physical Composition (Parent-Child Assets)

- Self-referential asset relationships
- Represents actual physical assembly/containment
- Example: Vessel (Asset) → Engine (Asset) → Turbocharger (Asset)
- Used for: Component tracking, maintenance, BOM, service history

STRUCTURE 3: Component BOM (Items, not Assets)

- Standard BC Items for non-tracked parts
- Example: Filters, taps, cables, consumables
- Used for: Parts inventory, ordering, replacement

Key Innovation

Separation of Classification from Composition:

```
Traditional (Confused):
  Level 1: Vehicle Type
    Level 2: Vehicle Model
      Level 3: Specific Vehicle  <-- Where do components go?
        Level 4: Engine??? (Breaks the classification logic)

Asset Pro (Clear):
  CLASSIFICATION:
    Industry: Fleet
      Level 1: Vessel Type
        Level 2: Vessel Model
          Asset Classification: Cargo Ship, Model XYZ

  PHYSICAL COMPOSITION:
    Asset: Vessel HMS-001
      Parent: (none)
      Children:
        → Engine (Asset)
        → Generator (Asset)
      Components (Items):
        → Propeller Blade (Item, Qty 4)
        → Navigation Light (Item, Qty 12)
```

Market Position

Target Market:

- Multi-industry asset-intensive businesses using Business Central
- Companies with 100-100,000+ assets to manage
- Industries: Marine, Medical, Construction, IT, Manufacturing, Rental

Competitive Advantage:

1. **Universal Adaptability** - One codebase serves all industries
2. **Complete Terminology Transformation** - System speaks your language
3. **Dual-Structure Clarity** - No confusion between classification and composition
4. **Unlimited Depth** - Each industry defines its own hierarchy complexity
5. **Native BC Integration** - Deep integration with Sales, Purchasing, Transfer documents

Pricing Strategy:

- Professional: $49/user/month (vs. competitors at $99/user/month)
- Target: 50-100 customers, 2,500+ users by Year 3
- ARR Target: $2-4M by Year 3

Architecture Overview

The Two-Structure Model Explained

Structure 1: Classification Hierarchy (What IS it?)

Purpose: Organizational taxonomy for filtering, reporting, and access control

Characteristics:

- Tree-like structure defined per industry
- Each industry has 1-10 levels (configurable)
- Each level has configurable name and terminology
- Assets are classified at ONE point in this tree
- Think: "What category/type does this asset belong to?"

Example - Fleet Management:

```
Industry: Fleet Management
  Level 1: "Fleet" (Commercial, Fishing, Passenger)
    Level 2: "Vessel Type" (Cargo Ship, Trawler, Ferry)
      Level 3: "Vessel Model" (Custom designation, e.g., "Panamax Bulk Carrier")

Asset HMS-001 Classification:
  - Industry: Fleet Management
  - Level 1 Value: Commercial
  - Level 2 Value: Cargo Ship
  - Level 3 Value: Panamax Bulk Carrier
```

Example - Water Dispensers:

```
Industry: Dispenser Management
  Level 1: "Product Line" (Office, Industrial, Residential)
    Level 2: "Model Series" (WD-100 Series, WD-200 Series)

Asset D-12345 Classification:
  - Industry: Dispenser Management
  - Level 1 Value: Office
  - Level 2 Value: WD-200 Series
```

Structure 2: Physical Composition (What's INSIDE it?)

Purpose: Actual physical assembly and component relationships

Characteristics:

- Self-referential Asset table (Parent Asset No. field)
- Represents physical containment/assembly
- Unlimited nesting depth (Asset → Asset → Asset...)
- Independent of classification hierarchy
- Think: "What physical components make up this asset?"

Example - Fleet Management:

```
Vessel HMS-001 (Classification: Commercial/Cargo Ship/Panamax)
  ├─ Main Engine ME-001 (Classification: Marine Equipment/Main Engine/Diesel)
  │    ├─ Turbocharger TC-001 (Classification: Marine Equipment/Turbocharger/Model-X)
  │    └─ Fuel Injection System FIS-001 (Classification: Marine Equipment/Fuel System/Electron
  └─ Auxiliary Generator AG-001 (Classification: Marine Equipment/Generator/750kW)
```

Example - Water Dispensers:

```
Dispenser D-12345 (Classification: Office/WD-200 Series)
  ├─ Electronic Control Unit ECU-789 (Classification: Dispenser Parts/Electronics/Control)
  └─ Cooling Compressor CC-456 (Classification: Dispenser Parts/Cooling/Compressor)
```

Structure 3: Component BOM (What standard PARTS does it use?)

Purpose: Non-tracked consumable items and standard parts

Characteristics:

- Links Asset to BC Items (existing Item table)
- Quantity-based (not serial number tracked)
- Used for consumables, common parts, replacements
- Think: "What do I need to order/stock for this asset?"

Example - Fleet Management:

```
Vessel HMS-001 Components:
  - Item 10001: Propeller Blade (Qty: 4)
  - Item 10002: Navigation Light, Red (Qty: 6)
  - Item 10003: Navigation Light, Green (Qty: 6)
  - Item 10004: Life Vest, Adult (Qty: 50)
```

Example - Water Dispensers:

```
Dispenser D-12345 Components:
  - Item 20001: Water Filter, 5-micron (Qty: 2)
  - Item 20002: Water Tap, Chrome (Qty: 1)
  - Item 20003: Drip Tray, Plastic (Qty: 1)
  - Item 20004: UV Lamp, Replacement (Qty: 1)
```

How the Three Structures Work Together

Asset Creation Flow:

1. **Choose Classification** - Where does it fit organizationally?
    - Select Industry: "Fleet Management"
    - Select Level 1: "Commercial"
    - Select Level 2: "Cargo Ship"
    - Select Level 3: "Panamax Bulk Carrier"
2. **Create Asset Record**
    - Asset No.: HMS-001
    - Description: "MV Pacific Star"
    - Classification: As above
    -

- Parent Asset: (none, this is a top-level asset)

3. **Add Physical Child Assets** (if applicable)
   - Create Engine asset ME-001
   - Set Parent = HMS-001
   - Engine has its own classification (Marine Equipment/Main Engine/Diesel)
4. **Add Component BOM** (standard parts)
   - Add Item 10001 (Propeller Blade), Qty 4
   - Add Item 10002 (Navigation Light), Qty 12
   - Etc.

Validation Logic:

- Classification validation: Must select valid industry and level values
- Physical validation: Parent asset must exist, cannot be self, no circular references
- Component validation: Item must exist in BC

Search/Filter Scenarios:

- "Show all Cargo Ships" → Filter by Classification Level 2
- "Show all assets containing Turbochargers" → Filter by Child Assets
- "Show all assets needing Item 20001" → Filter by Component BOM

Detailed Requirements Analysis

Requirement 1: Unlimited Classification Hierarchy (MUST)

Original Requirement:

> "Unlimited levels of asset hierarchy. We start from defining structure per industry (root) then setting up child levels. Compare with how Item Categories organized in BC."

Refined Requirement:

- **Unlimited** = Configurable depth per industry (recommended max: 10 levels, system supports up to 50)
- **Structure per industry** = Each industry defines its own levels independently
- **Configurable terminology** = Each level has custom name (singular and plural)
- **NOT like Item Categories** = More flexible, fully dynamic terminology

Architecture Solution:

Table: JML Asset Industry

- Defines each industry (Fleet Management, Dispenser Management, Medical Equipment, etc.)
- One record per industry

Table: JML Asset Classification Level

- Defines levels within an industry
- Fields: Industry Code, Level Number (1-50), Level Name, Level Name Plural
- Example: Industry="FLEET", Level=1, Name="Fleet", NamePlural="Fleets"

Table: JML Asset Classification Value

- Actual values at each level
- Fields: Industry Code, Level Number, Value Code, Description, Parent Value Code
- Example: Industry="FLEET", Level=1, Code="COMM", Description="Commercial", Parent=""

Validation Rules:

1. Level numbers must be sequential (1, 2, 3...), no gaps
2. Cannot delete level if values exist
3. Cannot delete value if assets are classified under it
4. Parent value must exist at level N-1 for values at level N
5. Value code must be unique within Industry + Level

Use Case Examples:

Simple Industry (2 levels):

```
Industry: IT Equipment
  Level 1: "Equipment Type" (Server, Storage, Network)
    Level 2: "Device" (individual assets)
```

Complex Industry (5 levels):

```
Industry: Healthcare Equipment
  Level 1: "Facility Type" (Hospital, Clinic, Lab)
    Level 2: "Department" (Radiology, Surgery, ICU)
      Level 3: "Equipment Category" (Imaging, Monitoring, Surgical)
        Level 4: "Equipment Type" (MRI, CT, X-Ray)
          Level 5: "Device" (individual assets)
```

Requirement 2: Self-Referential Asset Relationships (MUST)

Original Requirement:

> "Single Asset table with self-referential parent/child relationships. Any Asset can be parent of other Assets. Example: Vehicle (Asset) → Engine (Asset) → Turbocharger (Asset)."

Refined Requirement:

- One Asset table with Parent Asset No. field
- Circular reference prevention (cannot create loops)
- Hierarchy compatibility validation (discussed below)
- Support unlimited nesting depth (practical limit: 20 levels)

Architecture Solution:

Table: JML Asset

- Field: Parent Asset No. (Code[20], TableRelation to JML Asset)
- Field: Has Children (Boolean, calculated FlowField)
- Field: Child Count (Integer, calculated FlowField)
- Field: Level in Hierarchy (Integer, calculated from root)

Circular Reference Prevention Algorithm:

```
procedure ValidateParentAsset(NewParentNo: Code[20])
var
    CheckAsset: Record "JML Asset";
    CurrentNo: Code[20];
    Depth: Integer;
begin
    if NewParentNo = '' then
        exit; // No parent is OK

    if NewParentNo = "No." then
        Error('Asset cannot be its own parent.');

    CurrentNo := NewParentNo;
    Depth := 0;

    while (CurrentNo <> '') and (Depth < 100) do begin
        if not CheckAsset.Get(CurrentNo) then
            Error('Parent asset %1 does not exist.', CurrentNo);

        if CheckAsset."Parent Asset No." = "No." then
            Error('Circular reference detected: Asset %1 is already a child of current asset.

        CurrentNo := CheckAsset."Parent Asset No.";
        Depth += 1;
    end;

    if Depth >= 100 then
        Error('Maximum parent-child depth (100) exceeded.');
end;
```

Hierarchy Compatibility Validation:

Question: What does "check if hierarchy levels are compatible, not allow to overlap levels" mean?

Answer: Prevent illogical classification assignments. Two rules:

Rule 1: Classification Consistency

- If Parent and Child are in the same Industry, Child's classification level must be equal or deeper
- Example (VALID): Parent classified at Level 2 (Vessel Type), Child at Level 3 (Vessel Model)
- Example (INVALID): Parent at Level 3, Child at Level 1 (doesn't make sense)

Rule 2: Cross-Industry Relationships

- Allow parent-child across industries (e.g., Vehicle contains Generic Equipment)
- No validation on levels if industries differ
- Example: Fleet vessel (Industry: Fleet) can contain Electronics (Industry: Generic Equipment)

```
procedure ValidateClassificationCompatibility(ParentAsset: Record "JML Asset")
```

```
begin
    if "Industry Code" = ParentAsset."Industry Code" then begin
        // Same industry: child must be at equal or deeper level
        if "Classification Level No." < ParentAsset."Classification Level No." then
            Error('Child asset cannot be at a higher classification level than its parent wit
    end;
    // Different industries: no level validation
end;
```

Requirement 3: Configurable Terminology with CaptionClass (MUST)

Original Requirement:

> "Every level has name and code, so it makes configurable terminology. May be use of CaptionClass for dynamic renaming of levels in UI? Like Dimensions do?"

Refined Requirement:

- Complete terminology transformation (fields, pages, reports)
- User never sees generic "Level 1", "Level 2" - only their terms
- Implementation via CaptionClass mechanism (like Dimensions)
- Dynamic page title updates

Architecture Solution:

CaptionClass Implementation:

Business Central's CaptionClass system allows runtime caption resolution. Asset Pro will use area 'JML-ASSET' with expressions for each level.

Setup Table Extension:

```
table 70182300 "JML Asset Setup"
{
    fields
    {
        // Store current industry context (set by user navigation)
        field(100; "Current Industry Code"; Code[20]) { }

        // No need to store captions - they come from Classification Level table
    }
}
```

CaptionClass Codeunit:

```
codeunit 70182380 "JML Asset Caption Mgmt"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Caption Class", 'OnResolveCaptionClass'
    local procedure OnResolveCaptionClass(CaptionArea: Text; CaptionExpr: Text; Language: Int
    var
        AssetSetup: Record "JML Asset Setup";
        ClassLevel: Record "JML Asset Classification Level";
        IndustryCode: Code[20];
        LevelNo: Integer;
        Plural: Boolean;
    begin
        if CaptionArea <> 'JML-ASSET' then
```

```
            exit;

        // CaptionExpr format: "INDUSTRY:LEVEL:PLURAL"
        // Example: "FLEET:1:N" = Fleet industry, Level 1, Singular
        // Example: "FLEET:2:Y" = Fleet industry, Level 2, Plural

        ParseCaptionExpr(CaptionExpr, IndustryCode, LevelNo, Plural);

        if ClassLevel.Get(IndustryCode, LevelNo) then begin
            if Plural then
                Caption := ClassLevel."Level Name Plural"
            else
                Caption := ClassLevel."Level Name";
            Resolved := true;
        end else begin
            Caption := StrSubstNo('Level %1', LevelNo); // Fallback
            Resolved := true;
        end;
    end;

    local procedure ParseCaptionExpr(CaptionExpr: Text; var IndustryCode: Code[20]; var Level
    var
        Parts: List of [Text];
    begin
        Parts := CaptionExpr.Split(':');
        if Parts.Count >= 2 then begin
            IndustryCode := CopyStr(Parts.Get(1), 1, 20);
            Evaluate(LevelNo, Parts.Get(2));
        end;
        if Parts.Count >= 3 then
            Plural := (Parts.Get(3) = 'Y');
    end;
}
```

Page Field with Dynamic Caption:

```
page 70182333 "JML Asset Card"
{
    field("Classification Level 1"; Rec."Classification Level 1 Code")
    {
        CaptionClass = GetLevelCaption(1, false);
        // Resolves to: "Fleet", "Product Line", "Facility Type", etc.
        // depending on asset's industry
    }

    field("Classification Level 2"; Rec."Classification Level 2 Code")
    {
        CaptionClass = GetLevelCaption(2, false);
        // Resolves to: "Vessel Type", "Model Series", "Department", etc.
    }

    local procedure GetLevelCaption(LevelNo: Integer; Plural: Boolean): Text
    begin
        exit(StrSubstNo('JML-ASSET:%1:%2:%3',
            Rec."Industry Code",
            LevelNo,
            IIF(Plural, 'Y', 'N')));
    end;
}
```

Dynamic Page Titles:

For list pages showing classification values, page title should also adapt:

```
page 70182335 "JML Classification Values"
{
    Caption = 'Classification Values'; // Default
    SourceTable = "JML Asset Classification Value";

    trigger OnOpenPage()
    var
        ClassLevel: Record "JML Asset Classification Level";
    begin
        // Update page caption based on filtered level
        if ClassLevel.Get(GetFilter("Industry Code"), GetFilter("Level Number")) then
            CurrPage.Caption := ClassLevel."Level Name Plural";
    end;
}
```

Result:

- Vehicles industry: User sees "Fleet", "Vessel Type", "Vessel Model"
- Dispensers industry: User sees "Product Line", "Model Series"
- Medical industry: User sees "Facility Type", "Department", "Equipment Category"

Requirement 4: Asset Attributes per Industry/Level (SHOULD)

Original Requirement:

> "Asset attributes defined per industry and per level in hierarchy. Attributes can be of different types: text, number, date, option, boolean, lookup"

Refined Requirement:

- Define custom attributes per Industry + Level combination
- Support data types: Text, Integer, Decimal, Date, Boolean, Option
- Mandatory/optional configuration per attribute
- Default values supported
- Attribute values stored separately (not bloat Asset table)

Architecture Solution:

Table: JML Asset Attribute Definition

```
table 70182304 "JML Asset Attribute Defn"
{
    fields
    {
        field(1; "Industry Code"; Code[20]) { TableRelation = "JML Asset Industry"; }
        field(2; "Level Number"; Integer) { } // 0 = All levels
        field(3; "Attribute Code"; Code[20]) { }
        field(10; "Attribute Name"; Text[50]) { }
        field(20; "Data Type"; Enum "JML Attribute Data Type") { } // Text, Integer, Decimal,
        field(21; "Option String"; Text[250]) { } // For Option type: "Red,Blue,Green"
        field(30; "Mandatory"; Boolean) { }
        field(31; "Default Value"; Text[250]) { }
        field(40; "Display Order"; Integer) { }
    }

    keys
    {
```

```
        key(PK; "Industry Code", "Level Number", "Attribute Code") { Clustered = true; }
    }
}
```

Table: JML Asset Attribute Value

```
table 70182305 "JML Asset Attribute Value"
{
    fields
    {
        field(1; "Asset No."; Code[20]) { TableRelation = "JML Asset"; }
        field(2; "Attribute Code"; Code[20]) { }
        field(10; "Value Text"; Text[250]) { }
        field(11; "Value Integer"; Integer) { }
        field(12; "Value Decimal"; Decimal) { }
        field(13; "Value Date"; Date) { }
        field(14; "Value Boolean"; Boolean) { }
        // FlowFields for display
        field(100; "Attribute Name"; Text[50]) { FieldClass = FlowField; CalcFormula = Lookup
        field(101; "Data Type"; Enum "JML Attribute Data Type") { FieldClass = FlowField; Cal
    }

    keys
    {
        key(PK; "Asset No.", "Attribute Code") { Clustered = true; }
    }
}
```

Enum: JML Attribute Data Type

```
enum 70182403 "JML Attribute Data Type"
{
    Extensible = true;

    value(0; Text) { Caption = 'Text'; }
    value(1; Integer) { Caption = 'Integer'; }
    value(2; Decimal) { Caption = 'Decimal'; }
    value(3; Date) { Caption = 'Date'; }
    value(4; Boolean) { Caption = 'Boolean'; }
    value(5; Option) { Caption = 'Option'; }
}
```

Attribute Display: FactBox

Attributes displayed in a FactBox on the Asset Card (not inline fields, to keep card clean):

```
page 70182338 "JML Asset Attributes FactBox"
{
    PageType = CardPart;
    SourceTable = "JML Asset";

    layout
    {
        area(content)
        {
            // Dynamic repeater that shows all attributes for this asset
            repeater(Attributes)
            {
                field(AttributeName; TempAttributeValue."Attribute Name") { }
                field(Value; GetDisplayValue(TempAttributeValue)) { }
```

```
            }
        }
    }

    trigger OnAfterGetRecord()
    begin
        LoadAttributeValues();
    end;

    local procedure LoadAttributeValues()
    var
        AttributeValue: Record "JML Asset Attribute Value";
    begin
        TempAttributeValue.Reset();
        TempAttributeValue.DeleteAll();

        AttributeValue.SetRange("Asset No.", Rec."No.");
        if AttributeValue.FindSet() then
            repeat
                TempAttributeValue := AttributeValue;
                TempAttributeValue.Insert();
            until AttributeValue.Next() = 0;
    end;

    local procedure GetDisplayValue(var AttrValue: Record "JML Asset Attribute Value" tempora
    begin
        AttrValue.CalcFields("Data Type");
        case AttrValue."Data Type" of
            AttrValue."Data Type"::Text: exit(AttrValue."Value Text");
            AttrValue."Data Type"::Integer: exit(Format(AttrValue."Value Integer"));
            AttrValue."Data Type"::Decimal: exit(Format(AttrValue."Value Decimal"));
            AttrValue."Data Type"::Date: exit(Format(AttrValue."Value Date"));
            AttrValue."Data Type"::Boolean: exit(Format(AttrValue."Value Boolean"));
            AttrValue."Data Type"::Option: exit(AttrValue."Value Text");
        end;
    end;

    var
        TempAttributeValue: Record "JML Asset Attribute Value" temporary;
}
```

Attribute Management: Separate Page

```
page 70182337 "JML Asset Attributes"
{
    PageType = List;
    SourceTable = "JML Asset Attribute Value";
    Caption = 'Asset Attributes';

    layout
    {
        area(content)
        {
            repeater(Group)
            {
                field("Attribute Name"; Rec."Attribute Name") { Editable = false; }
                field(ValueText; Rec."Value Text") { Visible = IsTextAttribute; }
                field(ValueInteger; Rec."Value Integer") { Visible = IsIntegerAttribute; }
                field(ValueDecimal; Rec."Value Decimal") { Visible = IsDecimalAttribute; }
                field(ValueDate; Rec."Value Date") { Visible = IsDateAttribute; }
                field(ValueBoolean; Rec."Value Boolean") { Visible = IsBooleanAttribute; }
            }
        }
    }
```

```
    trigger OnAfterGetRecord()
    begin
        UpdateVisibility();
    end;

    local procedure UpdateVisibility()
    begin
        Rec.CalcFields("Data Type");
        IsTextAttribute := (Rec."Data Type" = Rec."Data Type"::Text) or (Rec."Data Type" = Re
        IsIntegerAttribute := (Rec."Data Type" = Rec."Data Type"::Integer);
        IsDecimalAttribute := (Rec."Data Type" = Rec."Data Type"::Decimal);
        IsDateAttribute := (Rec."Data Type" = Rec."Data Type"::Date);
        IsBooleanAttribute := (Rec."Data Type" = Rec."Data Type"::Boolean);
    end;

    var
        IsTextAttribute, IsIntegerAttribute, IsDecimalAttribute, IsDateAttribute, IsBooleanAt
}
```

Use Case Example: Vehicles Industry

```
Industry: Fleet Management
Level 3: Vessel Unit (specific assets)

Attributes defined for Level 3:
  - VIN (Text, Mandatory, Default: "")
  - Year of Manufacture (Integer, Mandatory, Default: CurrentYear)
  - Hull Material (Option: "Steel,Aluminum,Fiberglass", Mandatory)
  - Gross Tonnage (Decimal, Optional)
  - Last Survey Date (Date, Optional)
  - Ice Class Certified (Boolean, Optional, Default: false)

Asset HMS-001 Attribute Values:
  - VIN: "IMO9876543"
  - Year: 2018
  - Hull Material: "Steel"
  - Gross Tonnage: 75000.00
  - Last Survey Date: 2024-06-15
  - Ice Class: true
```

Requirement 5: Current Owner/Holder Tracking (MUST)

Original Requirement:

> "Current owner/holder of asset can be: Customer, Vendor, Location (item location in BC), Cost Center. Asset can be transitioned between locations, customers, cost centers, vendors. Track history (from-to)."

Refined Requirement:

- Track current holder: Customer, Vendor, Location, Cost Center
- Track holder type and code
- Maintain complete transition history with audit trail
- Date-stamped transitions with reason/document reference
- **Phase 2:** Automatic transitions via BC documents (Sales Order, Transfer Order, etc.)

Architecture Solution:

Table: JML Asset (Current Holder fields)

```
table 70182301 "JML Asset"
{
    fields
    {
        // ... other fields ...

        field(200; "Current Holder Type"; Enum "JML Asset Holder Type") { }
        field(201; "Current Holder Code"; Code[20])
        {
            TableRelation = if("Current Holder Type"=const(Customer)) Customer."No."
                            else if("Current Holder Type"=const(Vendor)) Vendor."No."
                            else if("Current Holder Type"=const(Location)) Location.Code
                            else if("Current Holder Type"=const("Cost Center")) "Dimension Va
        }
        field(202; "Current Holder Name"; Text[100])
        {
            FieldClass = FlowField;
            CalcFormula = Lookup(Customer.Name where("No."=field("Current Holder Code")));
            // Note: Real implementation would need multiple FlowFields or calculation
        }
        field(203; "Current Holder Since"; Date) { }

        // Ownership roles (multiple customers can be linked)
        field(210; "Owner Customer No."; Code[20]) { TableRelation = Customer; }
        field(211; "Operator Customer No."; Code[20]) { TableRelation = Customer; }
        field(212; "Lessee Customer No."; Code[20]) { TableRelation = Customer; }
    }
}
```

Enum: JML Asset Holder Type

```
enum 70182400 "JML Asset Holder Type"
{
    Extensible = true;

    value(0; " ") { Caption = ' '; }
    value(1; Customer) { Caption = 'Customer'; }
    value(2; Vendor) { Caption = 'Vendor'; }
    value(3; Location) { Caption = 'Location'; }
    value(4; "Cost Center") { Caption = 'Cost Center'; }
}
```

Table: JML Asset Holder History

```
table 70182306 "JML Asset Holder History"
{
    fields
    {
        field(1; "Asset No."; Code[20]) { TableRelation = "JML Asset"; }
        field(2; "Sequence No."; Integer) { AutoIncrement = true; }
        field(10; "Transition Date"; Date) { }
        field(11; "Transition Time"; Time) { }
        field(20; "From Holder Type"; Enum "JML Asset Holder Type") { }
        field(21; "From Holder Code"; Code[20]) { }
        field(22; "From Holder Name"; Text[100]) { }
        field(30; "To Holder Type"; Enum "JML Asset Holder Type") { }
        field(31; "To Holder Code"; Code[20]) { }
        field(32; "To Holder Name"; Text[100]) { }
        field(40; "Transition Type"; Enum "JML Asset Transition Type") { }
        field(41; "Document Type"; Enum "JML Asset Document Type") { }
        field(42; "Document No."; Code[20]) { }
```

```
        field(50; "Reason Code"; Code[10]) { TableRelation = "Reason Code"; }
        field(51; "Reason Description"; Text[100]) { }
        field(60; "User ID"; Code[50]) { }
    }

    keys
    {
        key(PK; "Asset No.", "Sequence No.") { Clustered = true; }
        key(TransDate; "Transition Date", "Asset No.") { }
    }
}
```

Enum: JML Asset Transition Type

```
enum 70182401 "JML Asset Transition Type"
{
    Extensible = true;

    value(0; " ") { Caption = ' '; }
    value(1; "Manual Transfer") { Caption = 'Manual Transfer'; }
    value(2; Sale) { Caption = 'Sale'; }
    value(3; Purchase) { Caption = 'Purchase'; }
    value(4; "Internal Transfer") { Caption = 'Internal Transfer'; }
    value(5; "Lease Out") { Caption = 'Lease Out'; }
    value(6; "Lease Return") { Caption = 'Lease Return'; }
    value(7; Service) { Caption = 'Service'; }
    value(8; "Service Return") { Caption = 'Service Return'; }
}
```

Enum: JML Asset Document Type

```
enum 70182406 "JML Asset Document Type"
{
    Extensible = true;

    value(0; " ") { Caption = ' '; }
    value(1; "Sales Order") { Caption = 'Sales Order'; }
    value(2; "Purchase Order") { Caption = 'Purchase Order'; }
    value(3; "Transfer Order") { Caption = 'Transfer Order'; }
    value(4; "Service Order") { Caption = 'Service Order'; }
    value(10; Manual) { Caption = 'Manual'; }
}
```

Codeunit: JML Asset Transfer Management

```
codeunit 70182385 "JML Asset Transfer Mgt"
{
    procedure TransferAsset(var Asset: Record "JML Asset"; NewHolderType: Enum "JML Asset Hol
    var
        HolderHistory: Record "JML Asset Holder History";
    begin
        // Validate
        ValidateTransfer(Asset, NewHolderType, NewHolderCode);

        // Create history record
        CreateHistoryEntry(Asset, NewHolderType, NewHolderCode, TransitionType, DocumentType,

        // Update asset
        Asset.Validate("Current Holder Type", NewHolderType);
        Asset.Validate("Current Holder Code", NewHolderCode);
        Asset.Validate("Current Holder Since", Today);
```

```
        Asset.Modify(true);
    end;

    local procedure ValidateTransfer(var Asset: Record "JML Asset"; NewHolderType: Enum "JML A
    var
        Customer: Record Customer;
        Vendor: Record Vendor;
        Location: Record Location;
    begin
        if NewHolderCode = '' then
            Error('Holder code must be specified.');

        // Validate holder exists
        case NewHolderType of
            NewHolderType::Customer:
                if not Customer.Get(NewHolderCode) then
                    Error('Customer %1 does not exist.', NewHolderCode);
            NewHolderType::Vendor:
                if not Vendor.Get(NewHolderCode) then
                    Error('Vendor %1 does not exist.', NewHolderCode);
            NewHolderType::Location:
                if not Location.Get(NewHolderCode) then
                    Error('Location %1 does not exist.', NewHolderCode);
            // Cost Center validation would go here
        end;

        // Cannot transfer to same holder
        if (Asset."Current Holder Type" = NewHolderType) and (Asset."Current Holder Code" = Ne
            Error('Asset is already at this holder.');
    end;

    local procedure CreateHistoryEntry(var Asset: Record "JML Asset"; NewHolderType: Enum "JML
    var
        HolderHistory: Record "JML Asset Holder History";
    begin
        HolderHistory.Init();
        HolderHistory."Asset No." := Asset."No.";
        HolderHistory."Transition Date" := Today;
        HolderHistory."Transition Time" := Time;
        HolderHistory."From Holder Type" := Asset."Current Holder Type";
        HolderHistory."From Holder Code" := Asset."Current Holder Code";
        HolderHistory."To Holder Type" := NewHolderType;
        HolderHistory."To Holder Code" := NewHolderCode;
        HolderHistory."Transition Type" := TransitionType;
        HolderHistory."Document Type" := DocumentType;
        HolderHistory."Document No." := DocumentNo;
        HolderHistory."Reason Code" := ReasonCode;
        HolderHistory."User ID" := UserId;
        HolderHistory.Insert(true);
    end;
}
```

Use Case Example: Venden Dispensers

```
Dispenser D-12345 Lifecycle:

1. Initial Purchase (2025-01-10):
   Transition: Vendor → Location (Warehouse)
   - From: Vendor "V001" (Manufacturer)
   - To: Location "WH01" (Our Warehouse)
   - Type: Purchase
   - Document: Purchase Order PO-001

2. Installation at Customer (2025-01-15):
   Transition: Location → Customer
```

```
   - From: Location "WH01"
   - To: Customer "C001" (Acme Corp)
   - Type: Lease Out
   - Document: Sales Order SO-001
   - Holder Since: 2025-01-15

3. Service (2025-06-01):
   Transition: Customer → Vendor
   - From: Customer "C001"
   - To: Vendor "V002" (Service Provider)
   - Type: Service
   - Document: Service Order SV-001

4. Return from Service (2025-06-03):
   Transition: Vendor → Customer
   - From: Vendor "V002"
   - To: Customer "C001"
   - Type: Service Return
   - Document: Service Order SV-001

Current Status:
   - Current Holder: Customer "C001" (Acme Corp)
   - Current Holder Since: 2025-06-03
   - Owner: JML Company (we own it)
   - Operator: Customer "C001" (they lease/use it)
```

Requirement 6: BC Document Integration (MUST - Phase 2)

Original Requirement:

> "Transition between locations, customers, cost centers, vendors must be made by standard BC documents (Transfer Order, Sales Order, Purchase Order). This is phase 2."

Refined Requirement:

- **Phase 1:** Manual asset transfers (via procedure)
- **Phase 2:** Automatic transfers triggered by BC document posting
- Documents: Sales Order, Purchase Order, Transfer Order, (Service Order future)
- Asset field added to document headers/lines
- Event subscribers update asset on posting

Architecture Solution (Phase 2):

Table Extension: Sales Header

```
tableextension 70182423 "JML Sales Header Ext" extends "Sales Header"
{
    fields
    {
        field(70182400; "JML Asset No."; Code[20])
        {
            Caption = 'Asset No.';
            TableRelation = "JML Asset";
            DataClassification = CustomerContent;

            trigger OnValidate()
            var
                Asset: Record "JML Asset";
            begin
                if "JML Asset No." <> '' then begin
```

```
                    Asset.Get("JML Asset No.");
                    // Could populate fields from asset (description, etc.)
                end;
            end;
        }

        field(70182401; "JML Asset Description"; Text[100])
        {
            Caption = 'Asset Description';
            FieldClass = FlowField;
            CalcFormula = Lookup("JML Asset".Description where("No."=field("JML Asset No.")))
            Editable = false;
        }
    }
}
```

Table Extension: Sales Line

```
tableextension 70182424 "JML Sales Line Ext" extends "Sales Line"
{
    fields
    {
        field(70182400; "JML Asset No."; Code[20])
        {
            Caption = 'Asset No.';
            TableRelation = "JML Asset";
            DataClassification = CustomerContent;
        }
    }
}
```

Codeunit: JML Document Integration (Event Subscribers)

```
codeunit 70182388 "JML Document Integration"
{
    // Event: After posting Sales Order
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"Sales-Post", 'OnAfterSalesInvHeaderInse
    local procedure OnAfterSalesInvHeaderInsert(var SalesInvHeader: Record "Sales Invoice Hea
    var
        Asset: Record "JML Asset";
        AssetTransferMgt: Codeunit "JML Asset Transfer Mgt";
    begin
        if SalesHeader."JML Asset No." = '' then
            exit;

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

        // Transfer asset to customer
        AssetTransferMgt.TransferAsset(
            Asset,
            Asset."Current Holder Type"::Customer,
            SalesHeader."Sell-to Customer No.",
            Asset."Transition Type"::Sale,
            Asset."Document Type"::"Sales Order",
            SalesInvHeader."No.",
            '');
    end;

    // Event: After posting Transfer Order
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"TransferOrder-Post Shipment", 'OnAfterT
    local procedure OnAfterTransferOrderPostShipment(var TransferHeader: Record "Transfer Hea
    var
```

```
        Asset: Record "JML Asset";
        AssetTransferMgt: Codeunit "JML Asset Transfer Mgt";
        TransferLine: Record "Transfer Line";
    begin
        TransferLine.SetRange("Document No.", TransferHeader."No.");
        TransferLine.SetFilter("JML Asset No.", '<>%1', '');
        if TransferLine.FindSet() then
            repeat
                if Asset.Get(TransferLine."JML Asset No.") then begin
                    // Transfer asset to new location
                    AssetTransferMgt.TransferAsset(
                        Asset,
                        Asset."Current Holder Type"::Location,
                        TransferHeader."Transfer-to Code",
                        Asset."Transition Type"::"Internal Transfer",
                        Asset."Document Type"::"Transfer Order",
                        TransferHeader."No.",
                        '');
                end;
            until TransferLine.Next() = 0;
    end;
}
```

Important Notes:

- Phase 2 implementation only (not in initial release)
- Requires thorough testing of all document posting scenarios
- Must handle errors gracefully (what if transfer fails?)
- Consider: Should posting be blocked if asset transfer fails?

Complete Data Model

Entity Relationship Diagram

erDiagram ASSET-INDUSTRY ||--o{ CLASSIFICATION-LEVEL : "defines levels" CLASSIFICATION-LEVEL ||--o{ CLASSIFICATION-VALUE : "has values" CLASSIFICATION-VALUE ||--o{ ASSET : "classifies at Level 1" CLASSIFICATION-VALUE ||--o{ ASSET : "classifies at Level 2" CLASSIFICATION-VALUE ||--o{ ASSET : "classifies at Level 3" ASSET ||--o{ ASSET : "parent-child" ASSET ||--o{ ASSET-COMPONENT : "has parts" ASSET ||--o{ ASSET-ATTRIBUTE-VALUE : "has attribute values" ASSET ||--o{ ASSET-HOLDER-HISTORY : "tracks transitions" CLASSIFICATION-LEVEL ||--o{ ATTRIBUTE-DEFINITION : "defines attributes" ATTRIBUTE-DEFINITION ||--o{ ASSET-ATTRIBUTE-VALUE : "defines structure" CUSTOMER ||--o{ ASSET : "owns/operates" VENDOR ||--o{ ASSET : "maintains" LOCATION ||--o{ ASSET : "stores" ITEM ||--o{ ASSET-COMPONENT : "is used in" ASSET-INDUSTRY { Code industry_code PK Text name Text description Boolean blocked } CLASSIFICATION-LEVEL { Code industry_code PK Integer level_number PK Text level_name Text level_name_plural Integer parent_level_number Boolean use_in_lists } CLASSIFICATION-VALUE { Code industry_code PK Integer level_number PK Code value_code PK Text description Code parent_value_code Boolean blocked } ASSET { Code no PK Text description Code industry_code FK Code classification_l1_code FK Code classification_l2_code FK Code classification_l3_code FK Code parent_asset_no FK Enum current_holder_type Code current_holder_code Date current_holder_since Code owner_customer_no Code operator_customer_no Enum status Date acquisition_date Decimal acquisition_cost } ASSET-COMPONENT { Code asset_no PK_FK Integer line_no PK Code item_no FK Decimal quantity Code unit_of_measure Text position Code serial_no Boolean blocked } ATTRIBUTE-DEFINITION { Code industry_code PK_FK Integer level_number PK_FK Code attribute_code PK Text attribute_name Enum data_type Text option_string Boolean mandatory Text default_value Integer display_order } ASSET-ATTRIBUTE-VALUE { Code asset_no PK_FK Code attribute_code PK_FK Text value_text Integer value_integer Decimal value_decimal Date value_date Boolean value_boolean } ASSET-HOLDER-HISTORY { Code asset_no PK_FK Integer sequence_no PK

Date transition_date Time transition_time Enum from_holder_type Code from_holder_code Enum to_holder_type Code to_holder_code Enum transition_type Enum document_type Code document_no Code reason_code Code user_id }

Core Table Definitions

Table 70182300: JML Asset Setup

```
table 70182300 "JML Asset Setup"
{
    Caption = 'Asset Setup';
    DataClassification = CustomerContent;

    fields
    {
        field(1; "Primary Key"; Code[10])
        {
            Caption = 'Primary Key';
            DataClassification = SystemMetadata;
        }

        field(10; "Asset Nos."; Code[20])
        {
            Caption = 'Asset Nos.';
            TableRelation = "No. Series";
        }

        field(20; "Default Industry Code"; Code[20])
        {
            Caption = 'Default Industry Code';
            TableRelation = "JML Asset Industry";
        }

        field(30; "Enable Classification"; Boolean)
        {
            Caption = 'Enable Classification';
            InitValue = true;
        }

        field(31; "Enable Attributes"; Boolean)
        {
            Caption = 'Enable Attributes';
            InitValue = true;
        }

        field(32; "Enable Holder History"; Boolean)
        {
            Caption = 'Enable Holder History';
            InitValue = true;
        }
    }

    keys
    {
        key(PK; "Primary Key")
        {
            Clustered = true;
        }
    }
}
```

Table 70182301: JML Asset

```
table 70182301 "JML Asset"
{
    Caption = 'Asset';
    DataClassification = CustomerContent;
    LookupPageId = "JML Asset List";
    DrillDownPageId = "JML Asset List";

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

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

        field(2; "No. Series"; Code[20])
        {
            Caption = 'No. Series';
            TableRelation = "No. Series";
            Editable = false;
        }

        field(10; Description; Text[100])
        {
            Caption = 'Description';
        }

        field(11; "Description 2"; Text[50])
        {
            Caption = 'Description 2';
        }

        field(12; "Search Description"; Code[100])
        {
            Caption = 'Search Description';
        }

        // Classification (Structure 1)
        field(100; "Industry Code"; Code[20])
        {
            Caption = 'Industry';
            TableRelation = "JML Asset Industry";

            trigger OnValidate()
            begin
                if "Industry Code" <> xRec."Industry Code" then begin
                    // Clear classification values if industry changed
                    "Classification Level 1" := '';
                    "Classification Level 2" := '';
                    "Classification Level 3" := '';
                end;
            end;
        }

        field(101; "Classification Level 1"; Code[20])
        {
            Caption = 'Level 1';
            TableRelation = "JML Asset Classification Value".Code where("Industry Code"=field
```

```
        CaptionClass = GetLevelCaption(1);
    }

    field(102; "Classification Level 2"; Code[20])
    {
        Caption = 'Level 2';
        TableRelation = "JML Asset Classification Value".Code where("Industry Code"=field
        CaptionClass = GetLevelCaption(2);
    }

    field(103; "Classification Level 3"; Code[20])
    {
        Caption = 'Level 3';
        TableRelation = "JML Asset Classification Value".Code where("Industry Code"=field
        CaptionClass = GetLevelCaption(3);
    }

    // Additional levels 4-10 would follow same pattern...

    // Physical Composition (Structure 2)
    field(200; "Parent Asset No."; Code[20])
    {
        Caption = 'Parent Asset No.';
        TableRelation = "JML Asset" where("Industry Code"=field("Industry Code"));

        trigger OnValidate()
        begin
            ValidateParentAsset();
        end;
    }

    field(201; "Has Children"; Boolean)
    {
        Caption = 'Has Children';
        FieldClass = FlowField;
        CalcFormula = Exist("JML Asset" where("Parent Asset No."=field("No.")));
        Editable = false;
    }

    field(202; "Child Asset Count"; Integer)
    {
        Caption = 'Child Asset Count';
        FieldClass = FlowField;
        CalcFormula = Count("JML Asset" where("Parent Asset No."=field("No.")));
        Editable = false;
    }

    field(203; "Hierarchy Level"; Integer)
    {
        Caption = 'Hierarchy Level';
        Editable = false;
    }

    // Current Holder (Structure 3 / Ownership)
    field(300; "Current Holder Type"; Enum "JML Asset Holder Type")
    {
        Caption = 'Current Holder Type';
    }

    field(301; "Current Holder Code"; Code[20])
    {
        Caption = 'Current Holder Code';
        TableRelation = if("Current Holder Type"=const(Customer)) Customer."No."
                        else if("Current Holder Type"=const(Vendor)) Vendor."No."
                        else if("Current Holder Type"=const(Location)) Location.Code;
    }
```

```
field(302; "Current Holder Since"; Date)
{
    Caption = 'Current Holder Since';
}

field(310; "Owner Customer No."; Code[20])
{
    Caption = 'Owner Customer No.';
    TableRelation = Customer;
}

field(311; "Operator Customer No."; Code[20])
{
    Caption = 'Operator Customer No.';
    TableRelation = Customer;
}

field(312; "Lessee Customer No."; Code[20])
{
    Caption = 'Lessee Customer No.';
    TableRelation = Customer;
}

// Status and Dates
field(400; Status; Enum "JML Asset Status")
{
    Caption = 'Status';
}

field(410; "Acquisition Date"; Date)
{
    Caption = 'Acquisition Date';
}

field(411; "In-Service Date"; Date)
{
    Caption = 'In-Service Date';
}

field(412; "Last Service Date"; Date)
{
    Caption = 'Last Service Date';
}

field(413; "Next Service Date"; Date)
{
    Caption = 'Next Service Date';
}

// Financial
field(500; "Acquisition Cost"; Decimal)
{
    Caption = 'Acquisition Cost';
    AutoFormatType = 1;
}

field(501; "Current Book Value"; Decimal)
{
    Caption = 'Current Book Value';
    AutoFormatType = 1;
}

// Additional Info
field(600; "Serial No."; Code[50])
{
    Caption = 'Serial No.';
}
```

```
    field(601; "Manufacturer Code"; Code[10])
    {
        Caption = 'Manufacturer Code';
        TableRelation = Manufacturer;
    }

    field(602; "Model No."; Code[50])
    {
        Caption = 'Model No.';
    }

    field(603; "Year of Manufacture"; Integer)
    {
        Caption = 'Year of Manufacture';
    }

    // System Fields
    field(900; Blocked; Boolean)
    {
        Caption = 'Blocked';
    }

    field(910; "Last Date Modified"; Date)
    {
        Caption = 'Last Date Modified';
        Editable = false;
    }
}

keys
{
    key(PK; "No.")
    {
        Clustered = true;
    }
    key(Industry; "Industry Code", "Classification Level 1", "Classification Level 2")
    {
    }
    key(Holder; "Current Holder Type", "Current Holder Code")
    {
    }
    key(Parent; "Parent Asset No.")
    {
    }
}

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

    "Last Date Modified" := Today;
    CalculateHierarchyLevel();
end;

trigger OnModify()
begin
    "Last Date Modified" := Today;
end;

var
    AssetSetup: Record "JML Asset Setup";
    NoSeriesMgt: Codeunit NoSeriesManagement;
```

```
    local procedure ValidateParentAsset()
    var
        ParentAsset: Record "JML Asset";
        CheckAsset: Record "JML Asset";
        CurrentNo: Code[20];
        Depth: Integer;
    begin
        if "Parent Asset No." = '' then begin
            "Hierarchy Level" := 1;
            exit;
        end;

        // Cannot be own parent
        if "Parent Asset No." = "No." then
            Error('Asset cannot be its own parent.');

        // Parent must exist
        if not ParentAsset.Get("Parent Asset No.") then
            Error('Parent asset %1 does not exist.', "Parent Asset No.");

        // Check circular reference
        CurrentNo := "Parent Asset No.";
        Depth := 0;

        while (CurrentNo <> '') and (Depth < 100) do begin
            if CheckAsset.Get(CurrentNo) then begin
                if CheckAsset."Parent Asset No." = "No." then
                    Error('Circular reference detected: Asset %1 is already a child of curren
                CurrentNo := CheckAsset."Parent Asset No.";
            end else
                CurrentNo := '';
            Depth += 1;
        end;

        if Depth >= 100 then
            Error('Maximum parent-child depth (100) exceeded.');

        CalculateHierarchyLevel();
    end;

    local procedure CalculateHierarchyLevel()
    var
        ParentAsset: Record "JML Asset";
    begin
        if "Parent Asset No." = '' then
            "Hierarchy Level" := 1
        else if ParentAsset.Get("Parent Asset No.") then
            "Hierarchy Level" := ParentAsset."Hierarchy Level" + 1
        else
            "Hierarchy Level" := 1;
    end;

    local procedure GetLevelCaption(LevelNo: Integer): Text
    begin
        exit(StrSubstNo('JML-ASSET:%1:%2:N', "Industry Code", LevelNo));
    end;
}
```

Supporting Tables (Summary)

Due to length, providing summary structures:

Table 70182302: JML Asset Industry

- Primary Key: Code
- Fields: Code, Name, Description, Blocked

Table 70182303: JML Asset Classification Level

- Primary Key: Industry Code, Level Number
- Fields: Industry Code, Level Number, Level Name, Level Name Plural, Parent Level Number
- Purpose: Define hierarchy levels per industry

Table 70182304: JML Asset Classification Value

- Primary Key: Industry Code, Level Number, Code
- Fields: Industry Code, Level Number, Code, Description, Parent Value Code, Blocked
- Purpose: Actual classification values (e.g., "Commercial", "Cargo Ship")

Table 70182305: JML Asset Attribute Defn

- Primary Key: Industry Code, Level Number, Attribute Code
- Fields: Attribute Name, Data Type, Option String, Mandatory, Default Value
- Purpose: Define custom attributes per industry/level

Table 70182306: JML Asset Attribute Value

- Primary Key: Asset No., Attribute Code
- Fields: Value Text, Value Integer, Value Decimal, Value Date, Value Boolean
- Purpose: Store attribute values for assets

Table 70182307: JML Asset Component

- Primary Key: Asset No., Line No.
- Fields: Asset No., Item No., Quantity, Unit of Measure, Position, Serial No., Blocked
- Purpose: Link assets to BC Items (BOM)

Table 70182308: JML Asset Holder History

- Primary Key: Asset No., Sequence No.
- Fields: Transition Date/Time, From/To Holder Type/Code, Transition Type, Document Type/No., Reason Code, User ID
- Purpose: Audit trail of ownership/location transitions

Table 70182309: JML Asset Comment Line

- Primary Key: Asset No., Line No.
- Fields: Comment, Date, User ID
- Purpose: Comments/notes for assets

Object Inventory

Production Objects (70182300-70182449)

Tables (70182300-70182329)

ID	Object Name	Purpose	Priority
70182300	JML Asset Setup	Company-wide configuration	Phase 1
70182301	JML Asset	Main asset master record	Phase 1
70182302	JML Asset Industry	Industry definitions	Phase 1
70182303	JML Asset Classification Level	Hierarchy level definitions	Phase 1
70182304	JML Asset Classification Value	Classification values	Phase 1
70182305	JML Asset Attribute Defn	Custom attribute definitions	Phase 1
70182306	JML Asset Attribute Value	Asset attribute values	Phase 1
70182307	JML Asset Component	Non-asset parts/items (BOM)	Phase 1
70182308	JML Asset Comment Line	Comments for assets	Phase 1
70182309	JML Asset Holder History	Ownership transition audit trail	Phase 1
70182310	JML Industry Template	Pre-built industry templates	Phase 2

Pages (70182330-70182379)

ID	Object Name	Type	Purpose	Priority
70182330	JML Asset Setup	Card	Configuration page	Phase 1
70182331	JML Asset Setup Wizard	Navigate	Initial setup wizard	Phase 1
70182332	JML Asset List	List	Main asset list	Phase 1
70182333	JML Asset Card	Card	Asset details	Phase 1
70182334	JML Asset Industries	List	Industry setup	Phase 1
70182335	JML Classification Levels	List	Level definitions	Phase 1
70182336	JML Classification Values	List	Classification values	Phase 1
70182337	JML Attribute Definitions	List	Attribute setup	Phase 1
70182338	JML Asset Attributes FactBox	CardPart	Attribute display/edit	Phase 1
70182339	JML Asset Holder History	List	Transition history	Phase 1
70182340	JML Asset Components	ListPart	BOM subpage	Phase 1
70182341	JML Asset Tree	List	Hierarchical view	Phase 2
70182342	JML Industry Templates	List	Template selection	Phase 2
70182343	JML Asset FactBox	FactBox	Asset summary	Phase 2
70182344	JML Holder History FactBox	FactBox	Recent transitions	Phase 2

Codeunits (70182380-70182399)

ID	Object Name	Purpose	Priority
70182380	JML Asset Management	Core asset operations	Phase 1
70182381	JML Asset Setup Wizard	Setup wizard logic	Phase 1
70182382	JML Asset Caption Mgmt	Dynamic caption resolution	Phase 1
70182383	JML Classification Mgmt	Classification operations	Phase 1
70182384	JML Asset Attribute Mgmt	Attribute operations	Phase 1
70182385	JML Asset Transfer Mgt	Ownership transfer logic	Phase 1
70182386	JML Asset Copy	Copy asset with children	Phase 2
70182387	JML Asset Validation	Business rule validation	Phase 2
70182388	JML Document Integration	BC document event handlers	Phase 2

Enums (70182400-70182409)

ID	Object Name	Values	Priority
70182400	JML Asset Holder Type	Customer, Vendor, Location, Cost Center	Phase 1
70182401	JML Asset Transition Type	Sale, Transfer, Lease, Service, Return, etc.	Phase 1
70182402	JML Asset Status	Active, Inactive, Maintenance, Decommissioned	Phase 1
70182403	JML Attribute Data Type	Text, Integer, Decimal, Date, Boolean, Option	Phase 1
70182404	JML Industry Template Type	Vehicles, Medical, Construction, IT, Custom	Phase 1
70182406	JML Asset Document Type	Sales Order, Purchase Order, Transfer Order, etc.	Phase 2

Reports (70182410-70182419)

ID	Object Name	Purpose	Priority
70182410	JML Asset List	Asset inventory report	Phase 2
70182411	JML Asset Classification	Classification structure report	Phase 2
70182412	JML Asset Transitions	Holder history report	Phase 2
70182413	JML Assets by Holder	Assets grouped by current holder	Phase 2
70182414	JML Asset Hierarchy Tree	Parent-child structure report	Phase 2

Table Extensions (70182420-70182429)

ID	Object Name	Extends	Purpose	Priority
70182420	JML Customer Ext	Customer	Asset count FlowFields	Phase 2
70182421	JML Vendor Ext	Vendor	Maintained asset count	Phase 2
70182422	JML Location Ext	Location	Asset count at location	Phase 2
70182423	JML Sales Header Ext	Sales Header	Asset context fields	Phase 2
70182424	JML Sales Line Ext	Sales Line	Asset reference	Phase 2
70182425	JML Transfer Header Ext	Transfer Header	Asset list	Phase 2
70182426	JML Transfer Line Ext	Transfer Line	Asset reference	Phase 2

Page Extensions (70182430-70182439)

ID	Object Name	Extends	Purpose	Priority
70182430	JML Customer Card Ext	Customer Card	Asset FactBox	Phase 3
70182431	JML Vendor Card Ext	Vendor Card	Asset FactBox	Phase 3
70182432	JML Location Card Ext	Location Card	Asset FactBox	Phase 3
70182433	JML Sales Order Ext	Sales Order	Asset fields	Phase 3
70182434	JML Transfer Order Ext	Transfer Order	Asset fields	Phase 3

Test Objects (50100-50199)

Unit Tests (50100-50149)

ID	Object Name	Purpose
50100	JML Asset Setup Tests	Test setup wizard
50101	JML Classification Tests	Test classification hierarchy
50102	JML Asset Creation Tests	Test asset creation
50103	JML Circular Reference Tests	Test circular ref prevention
50104	JML Attribute Tests	Test attribute management
50105	JML Asset Transfer Tests	Test ownership transfers
50106	JML Parent-Child Tests	Test asset parent-child relationships

Integration Tests (50150-50179)

ID	Object Name	Purpose
50150	JML Asset Workflow Tests	End-to-end workflows
50151	JML Document Integration Tests	BC document integration
50152	JML Multi-Industry Tests	Multiple industries
50153	JML Venden Scenario Tests	Venden use case testing
50154	JML Rollsberg Scenario Tests	Rollsberg use case testing

Test Library (50180-50189)

ID	Object Name	Purpose
50180	JML Asset Test Library	Test data creation helpers

Implementation Phases

Phase 1: Foundation (Weeks 1-4) - Core Data Model

Objective: Implement two-structure architecture and basic asset management

Week 1: Setup & Classification Structure

Tables:

- 70182300: JML Asset Setup
- 70182302: JML Asset Industry
- 70182303: JML Asset Classification Level
- 70182304: JML Asset Classification Value

Pages:

- 70182330: JML Asset Setup
- 70182334: JML Asset Industries
- 70182335: JML Classification Levels
- 70182336: JML Classification Values

Codeunits:

- 70182381: JML Asset Setup Wizard

- 70182383: JML Classification Mgmt

Enums:

- 70182404: JML Industry Template Type

Deliverables:

- User can define industries with up to 10 classification levels
- Each level has configurable name (singular/plural)
- Setup wizard creates sample industry (Vehicles or Dispensers)
- Validation prevents orphaned values and level deletion

Tests:

- 50100: Test setup wizard creates valid classification structure
- 50101: Test classification level validation (sequential levels, no gaps)
- 50101: Test cannot delete level with existing values
- 50101: Test cannot delete industry with assets

Exit Criteria:

- All tables compile with 0 errors, 0 warnings
- Setup wizard completes in < 2 minutes
- Can create industry with 3 levels and 5 values per level
- All Phase 1 Week 1 tests pass

Week 2: Asset Master Table & Parent-Child

Tables:

- 70182301: JML Asset

Pages:

- 70182332: JML Asset List
- 70182333: JML Asset Card

Codeunits:

- 70182380: JML Asset Management
- 70182382: JML Asset Caption Mgmt

Enums:

- 70182402: JML Asset Status

Deliverables:

- Asset table with classification fields (Structure 1)
- Self-referential parent-child relationship (Structure 2)
-

Circular reference prevention algorithm
- Hierarchy level calculation
- Dynamic field captions via CaptionClass

Tests:

- 50102: Test asset creation with classification
- 50103: Test circular reference prevention (A→B→C→A blocked)
- 50106: Test parent-child validation (parent must exist)
- 50106: Test hierarchy level calculation
- 50102: Test classification filtering

Exit Criteria:

- Can create asset with classification
- Can create parent-child relationships (3+ levels deep)
- Circular reference prevention works
- Field captions change based on industry terminology
- All Phase 1 Week 2 tests pass

Week 3: Attributes Framework

Tables:

- 70182305: JML Asset Attribute Defn
- 70182306: JML Asset Attribute Value

Pages:

- 70182337: JML Attribute Definitions
- 70182338: JML Asset Attributes FactBox

Codeunits:

- 70182384: JML Asset Attribute Mgmt

Enums:

- 70182403: JML Attribute Data Type

Deliverables:

- Define custom attributes per industry/level
- Support all data types (Text, Integer, Decimal, Date, Boolean, Option)
- Mandatory attribute validation
- Default value application
- FactBox shows/edits attributes

Tests:

- 50104: Test attribute definition (all data types)
- 50104: Test mandatory attribute validation

- 50104: Test default value application
- 50104: Test option type with option string
- 50104: Test attribute display in FactBox

Exit Criteria:

- Can define 10+ attributes for an industry/level
- Mandatory attributes enforced on asset creation
- FactBox displays attributes correctly
- All data types work (Text, Integer, Decimal, Date, Boolean, Option)
- All Phase 1 Week 3 tests pass

Week 4: Ownership & Components

Tables:

- 70182307: JML Asset Component
- 70182308: JML Asset Comment Line
- 70182309: JML Asset Holder History

Pages:

- 70182339: JML Asset Holder History
- 70182340: JML Asset Components

Codeunits:

- 70182385: JML Asset Transfer Mgt

Enums:

- 70182400: JML Asset Holder Type
- 70182401: JML Asset Transition Type

Deliverables:

- Track current holder (Customer, Vendor, Location, Cost Center)
- Manual transfer procedure
- Complete transition history with audit trail
- Component BOM (Items linked to assets)
- Comments for assets

Tests:

- 50105: Test manual asset transfer (Customer → Location)
- 50105: Test holder history creation
- 50105: Test holder validation (holder must exist)
- 50105: Test cannot transfer to same holder
- 50102: Test component BOM creation

Exit Criteria:

- Can manually transfer asset between holders
- Holder history records created automatically
- Component BOM works (link Items to Assets)
- Comments work
- All Phase 1 Week 4 tests pass

Phase 1 Exit Criteria:

- All Phase 1 tables, pages, codeunits compile (0 errors, 0 warnings)
- Setup wizard creates working industry structure
- Can create assets with classification, parent-child, attributes
- Can manually transfer asset ownership
- Can add Items as components
- All Phase 1 tests pass (50100-50106)
- Code review completed
- Documentation updated

Phase 2: Advanced Features & Integration (Weeks 5-8)

Objective: Industry templates, reporting, BC document integration

Week 5: Industry Templates

Tables:

- 70182310: JML Industry Template

Pages:

- 70182331: JML Asset Setup Wizard (enhanced)
- 70182342: JML Industry Templates

Codeunits:

- 70182381: JML Asset Setup Wizard (enhanced)

Deliverables:

- Pre-built templates: Vehicles, Dispensers, Medical, IT
- Template includes: classification levels, sample values, attributes
- Setup wizard offers template selection
- One-click template application
- Custom template export/import

Template Examples:

Vehicles Template:

```
Classification:
  Level 1: "Fleet" (Commercial, Fishing, Passenger)
  Level 2: "Vessel Type" (Cargo Ship, Trawler, Ferry)
```

```
  Level 3: "Vessel Model" (Custom)

Attributes:
  - VIN (Text, Mandatory)
  - Year (Integer, Mandatory)
  - Hull Material (Option: Steel,Aluminum,Fiberglass)
  - Gross Tonnage (Decimal)
```

Dispensers Template:

```
Classification:
  Level 1: "Product Line" (Office, Industrial, Residential)
  Level 2: "Model Series" (WD-100, WD-200, WD-300)

Attributes:
  - Serial Number (Text, Mandatory)
  - Cooling Type (Option: Compressor,Thermoelectric)
  - Tank Capacity Liters (Decimal)
  - Installation Date (Date)
```

Tests:

- 50152: Test Vehicles template application
- 50152: Test Dispensers template application
- 50152: Test template validation
- 50152: Test custom template creation

Exit Criteria:

- 4 templates available (Vehicles, Dispensers, Medical, IT)
- Templates apply in < 10 seconds
- Can export/import custom templates
- All Phase 2 Week 5 tests pass

Week 6: Reporting

Reports:

- 70182410: JML Asset List
- 70182411: JML Asset Classification
- 70182412: JML Asset Transitions
- 70182413: JML Assets by Holder
- 70182414: JML Asset Hierarchy Tree

Deliverables:

- Asset inventory report (filtering, grouping)
- Classification structure report (tree view)
- Transition history report (audit)
- Assets by holder report
- Parent-child hierarchy report
- Excel export for all reports

Report Features:

- Filter by: Industry, Classification, Holder, Status, Date Range
- Group by: Industry, Classification Level 1/2, Holder, Status
- Include: Attributes, Components, Children (configurable)
- Output: PDF, Excel, Print

Exit Criteria:

- All 5 reports generate correct data
- Reports support filtering and grouping
- Excel export works
- Performance: < 5 seconds for 1,000 assets

Week 7: BC Document Integration (Design & Implement)

Table Extensions:

- 70182423: JML Sales Header Ext
- 70182424: JML Sales Line Ext
- 70182425: JML Transfer Header Ext
- 70182426: JML Transfer Line Ext

Codeunits:

- 70182388: JML Document Integration

Deliverables:

- Sales Order: Asset No. field, auto-transfer on posting
- Transfer Order: Asset list, location update on posting
- Event subscribers for document posting
- Error handling if transfer fails
- Integration tests

Posting Behavior:

Sales Order Posting:

```
Before Posting:
  - Asset Current Holder: Location "WH01"

Sales Order:
  - Customer: "C001"
  - Asset No.: "D-12345"

After Posting (Success):
  - Asset Current Holder: Customer "C001"
  - Holder History: WH01 → C001 (Sale, SO-001)
```

Transfer Order Posting:

```
Before Posting:
  - Asset Current Holder: Location "WH01"
```

```
Transfer Order:
  - From Location: "WH01"
  - To Location: "WH02"
  - Asset No.: "D-12345"

After Posting (Success):
  - Asset Current Holder: Location "WH02"
  - Holder History: WH01 → WH02 (Transfer, TO-001)
```

Tests:

- 50151: Test sales order asset transfer
- 50151: Test transfer order location update
- 50151: Test holder history on posting
- 50151: Test error handling (asset not found)
- 50151: Test rollback if transfer fails

Exit Criteria:

- Sales Order posting updates asset holder
- Transfer Order posting updates asset location
- Holder history records created
- Error handling works correctly
- All Phase 2 Week 7 tests pass

Week 8: UI Enhancements

Pages:

- 70182341: JML Asset Tree (hierarchical view)
- 70182343: JML Asset FactBox
- 70182344: JML Holder History FactBox

Deliverables:

- Tree view page showing parent-child structure
- Asset summary FactBox (for other pages)
- Recent transitions FactBox
- Improved page layouts
- Field grouping and visibility

Tree View Features:

- Expand/collapse parent-child relationships
- Visual indentation for hierarchy levels
- Show: Asset No., Description, Classification, Holder, Status
- Drill-down to asset card
- Refresh on changes

Exit Criteria:

- Tree view displays assets correctly
- FactBoxes work on relevant pages

- Page layouts improved based on user feedback
- All Phase 2 tests pass (50150-50154)

Phase 2 Exit Criteria:

- Industry templates functional
- All reports generate correct data
- BC document integration working
- Tree view and FactBoxes implemented
- All Phase 2 tests pass (50150-50154)
- User acceptance testing completed
- Documentation updated

Phase 3: Extensions & Polish (Weeks 9-12)

Objective: BC extensions, performance optimization, production readiness

Week 9: Customer/Vendor/Location Extensions

Table Extensions:

- 70182420: JML Customer Ext
- 70182421: JML Vendor Ext
- 70182422: JML Location Ext

Page Extensions:

- 70182430: JML Customer Card Ext
- 70182431: JML Vendor Card Ext
- 70182432: JML Location Card Ext
- 70182433: JML Sales Order Ext
- 70182434: JML Transfer Order Ext

Deliverables:

- Customer Card: Owned/operated assets count, FactBox
- Vendor Card: Maintained assets count, FactBox
- Location Card: Assets at location count, FactBox
- Sales Order: Asset fields visible
- Transfer Order: Asset fields visible

FlowFields:

```
// Customer Extension
field(70182400; "JML Assets Owned Count"; Integer)
{
    FieldClass = FlowField;
    CalcFormula = Count("JML Asset" where("Owner Customer No."=field("No.")));
}

field(70182401; "JML Assets Operated Count"; Integer)
{
    FieldClass = FlowField;
```

```
    CalcFormula = Count("JML Asset" where("Operator Customer No."=field("No.")));
}

field(70182402; "JML Assets at Customer Count"; Integer)
{
    FieldClass = FlowField;
    CalcFormula = Count("JML Asset" where("Current Holder Type"=const(Customer), "Current Hol
}
```

Exit Criteria:

- FlowFields calculate correctly
- FactBoxes display on extended cards
- Asset fields visible on documents
- Performance acceptable (< 100ms for FlowField calculation)

Week 10: Performance Optimization

Activities:

- Index optimization (add keys for common filters)
- Query performance tuning
- Circular reference check optimization (cache results)
- Attribute loading optimization (batch load)
- FlowField calculation optimization

Performance Targets:

- Asset Card load: < 500ms
- Asset List (1,000 assets): < 1 second
- Classification navigation: < 100ms
- Attribute FactBox load: < 50ms
- Circular reference check: < 200ms
- Search (10,000 assets): < 2 seconds

Optimization Techniques:

- Add SIFT keys for common FlowFields
- Cache classification level names
- Lazy load attributes (only when FactBox visible)
- Batch attribute value loading
- Optimize circular reference check (memoization)

Exit Criteria:

- All performance targets met
- Load testing completed (10,000+ assets)
- No performance regressions
- Optimization documented

Week 11: Testing & Quality Assurance

Activities:

- Complete unit test coverage (Phases 1-3)
- Integration testing (all scenarios)
- Performance testing (load, stress, endurance)
- Security testing (permissions, data classification)
- Accessibility testing (keyboard navigation, screen readers)
- Browser compatibility (Web Client)

Test Coverage Targets:

- Unit tests: 100% for core logic
- Integration tests: All critical workflows
- Performance tests: All pages and reports
- Security tests: All permission sets

Deliverables:

- Test report (all tests passing)
- Performance benchmark report
- Security audit report
- Accessibility compliance report

Exit Criteria:

- 100% of tests passing
- No critical or high-severity bugs
- Performance targets met
- Security audit passed
- Accessibility standards met

Week 12: Documentation & Release Preparation

Activities:

- User documentation (setup, daily use, reporting)
- Technical documentation (architecture, extensibility, API)
- Administrator guide (setup wizard, configuration)
- Developer guide (extending Asset Pro, custom attributes)
- Video tutorials (setup, asset creation, transfers)
- Release notes
- Deployment guide

Deliverables:

- User Manual (50+ pages)
- Technical Documentation (30+ pages)
- Administrator Guide (20+ pages)
- Developer Guide (15+ pages)
- 5+ video tutorials (5-10 minutes each)
- Release Notes v1.0
- Deployment Guide

Exit Criteria:

- All documentation complete
- Videos recorded and published
- Release notes finalized
- Deployment guide tested
- Ready for production release

Phase 3 Exit Criteria:

- All extensions implemented
- Performance optimized and verified
- All tests passing (unit, integration, performance, security)
- Documentation complete
- Ready for production deployment
- Sign-off from stakeholders

Validation Rules Summary

Classification Hierarchy Validation

1. **Industry Level:**
    - Industry code must be unique
    - Cannot delete industry if assets exist
    - Cannot modify industry code if in use
2. **Classification Level:**
    - Level numbers must be sequential (1, 2, 3...), no gaps
    - Level 1 must exist before Level 2
    - Cannot delete level if values exist at that level
    - Cannot change level number if values exist
    - Level name must be unique within industry
3. **Classification Value:**
    - Value code must be unique within Industry + Level
    - Parent value must exist at Level N-1 (for Level N values)
    - Cannot delete value if assets classified under it
    - Cannot create orphaned values (parent must exist first)

Asset Validation

1. **Classification:**
    - Industry code is mandatory
    - At least Level 1 classification required
    - Level 2 requires Level 1 to be set

- Level 3 requires Level 2 to be set
- Cannot mix classification values from different industries
- Cannot select Level 2 value whose parent doesn't match Level 1

2. **Parent-Child Relationships:**
   - Parent asset must exist
   - Cannot be own parent (Asset A → Asset A)
   - Cannot create circular references (A → B → C → A)
   - Maximum depth: 100 levels (practical: 20)
   - Parent and child can be in different industries (allowed)
   - If same industry: child classification level ≥ parent classification level
3. **Holder/Ownership:**
   - Current holder code must exist in respective table
   - Holder type must match holder code table
   - Cannot transfer to same holder (same type + code)
   - Transition date cannot be in future

Attribute Validation

1. **Attribute Definition:**
   - Attribute code must be unique within Industry + Level
   - Data type must be valid enum value
   - Option string required for Option data type
   - Default value must match data type
   - Option string format: "Value1,Value2,Value3" (comma-separated)
2. **Attribute Value:**
   - Attribute must be defined for asset's industry/level
   - Mandatory attributes must have values
   - Value must match defined data type
   - Integer: Must be whole number
   - Decimal: Must be numeric
   - Date: Must be valid date
   - Boolean: Must be true/false
   - Option: Must be from option string

Component (BOM) Validation

1. **Component:**
   - Item must exist in BC Item table
   - Quantity must be > 0
   - Unit of measure must be valid for item
   - Cannot add duplicate Item No. on same asset
   - Serial no. optional (for non-serialized items)

Use Case Examples

Use Case 1: Rollsberg Marine Fleet Management

Setup: Industry Configuration

```
Industry: Fleet Management

Classification Levels:
  Level 1: "Fleet" (singular), "Fleets" (plural)
  Level 2: "Vessel Type" (singular), "Vessel Types" (plural)
  Level 3: "Vessel Model" (singular), "Vessel Models" (plural)

Classification Values:
  Level 1:
    - COMM: Commercial
    - FISH: Fishing
    - PASS: Passenger
  Level 2 (under COMM):
    - CARGO: Cargo Ship
    - TANKER: Tanker
    - BULK: Bulk Carrier
  Level 3 (under CARGO):
    - PANAMAX: Panamax Bulk Carrier
    - HANDYMAX: Handymax Bulk Carrier

Attributes for Level 3 (Vessel Model):
  - IMO Number (Text, Mandatory)
  - Year Built (Integer, Mandatory)
  - Gross Tonnage (Decimal, Mandatory)
  - Deadweight Tonnage (Decimal)
  - Hull Material (Option: Steel,Aluminum)
  - Ice Class (Boolean)
  - Last Survey Date (Date)
```

Asset Creation: MV Pacific Star

```
Asset No.: HMS-001
Description: MV Pacific Star
Classification:
  - Industry: Fleet Management
  - Fleet: Commercial
  - Vessel Type: Cargo Ship
  - Vessel Model: Panamax Bulk Carrier

Attributes:
  - IMO Number: IMO9876543
  - Year Built: 2018
  - Gross Tonnage: 75000.00
  - Deadweight Tonnage: 82000.00
  - Hull Material: Steel
  - Ice Class: true
  - Last Survey Date: 2024-06-15

Ownership:
  - Owner Customer: "C-ROLLSBERG" (Rollsberg Shipping AS)
  - Operator Customer: "C-ROLLSBERG"
  - Current Holder: Customer "C-ROLLSBERG"
  - Current Holder Since: 2018-03-15

Status: Active
Acquisition Date: 2018-03-15
```

Acquisition Cost: 45,000,000.00

Physical Composition: Add Main Engine

Asset No.: ME-001
Description: Wärtsilä 6RT-flex50-D
Parent Asset: HMS-001
Classification:
  - Industry: Marine Equipment (different industry!)
  - Equipment Type: Main Engine
  - Engine Model: 6RT-flex50-D

Attributes:
  - Serial Number: WRT-2018-001
  - Power Output kW: 14,400
  - Fuel Type: Heavy Fuel Oil
  - Year Built: 2018

Ownership:
  - Owner Customer: "C-ROLLSBERG"
  - Current Holder: (inherited from parent HMS-001)

Physical Composition: Add Turbocharger (child of engine)

Asset No.: TC-001
Description: ABB A100-L Turbocharger
Parent Asset: ME-001 (child of ME-001, grandchild of HMS-001)
Classification:
  - Industry: Marine Equipment
  - Equipment Type: Turbocharger
  - Model: A100-L

Attributes:
  - Serial Number: ABB-TC-2018-050
  - Max RPM: 18,000

Component BOM (Standard Parts for HMS-001):

Item No.	Description	Quantity	Position
10001	Propeller Blade	4	Propeller
10002	Navigation Light, Red	2	Port Side
10003	Navigation Light, Green	2	Starboard Side
10004	Life Vest, Adult	50	Safety Equipment
10005	Life Raft, 25-person	4	Deck

Holder Transitions:

Sequence 1:
  Date: 2018-03-15
  From: Vendor "V-SHIPYARD" (Shipyard)
  To: Customer "C-ROLLSBERG"
  Type: Purchase
  Document: Purchase Order PO-2018-001

Sequence 2:
  Date: 2024-06-01
  From: Customer "C-ROLLSBERG"
  To: Vendor "V-SERVICE" (Maintenance facility)

```
  Type: Service
  Document: Service Order SV-2024-050
  Reason: Annual survey and dry-docking

Sequence 3:
  Date: 2024-07-15
  From: Vendor "V-SERVICE"
  To: Customer "C-ROLLSBERG"
  Type: Service Return
  Document: Service Order SV-2024-050
```

Search/Filter Scenarios:

1. "Show all Cargo Ships" → Filter: Fleet Type = "CARGO"
2. "Show all vessels owned by Rollsberg" → Filter: Owner Customer = "C-ROLLSBERG"
3. "Show all assets at service facility" → Filter: Current Holder = Vendor "V-SERVICE"
4. "Show all vessels with Main Engines" → Filter: Has Child Assets where Child Type = "Main Engine"
5. "Show all vessels needing Life Vest replenishment" → Filter: Component Item = "10004", Quantity < 40

Use Case 2: Venden Water Dispensers

Setup: Industry Configuration

```
Industry: Dispenser Management

Classification Levels:
  Level 1: "Product Line" (singular), "Product Lines" (plural)
  Level 2: "Model Series" (singular), "Model Series" (plural)

Classification Values:
  Level 1:
    - OFFICE: Office
    - INDUST: Industrial
    - RESIDE: Residential
  Level 2 (under OFFICE):
    - WD100: WD-100 Series (Basic)
    - WD200: WD-200 Series (Premium)
    - WD300: WD-300 Series (Deluxe)

Attributes for Level 2 (Model Series):
  - Cooling Type (Option: Compressor,Thermoelectric, Mandatory)
  - Tank Capacity Liters (Decimal, Mandatory)
  - Hot Water (Boolean)
  - UV Sterilization (Boolean)
  - Filter Type (Option: Standard,Advanced,Ultra)
  - Dimensions H×W×D cm (Text)
```

Asset Creation: Dispenser for Acme Corp

```
Asset No.: D-12345
Description: Water Dispenser WD-200 Premium
Classification:
  - Industry: Dispenser Management
  - Product Line: Office
  - Model Series: WD-200 Series

Attributes:
  - Cooling Type: Compressor
  - Tank Capacity Liters: 5.0
  - Hot Water: true
```

```
  - UV Sterilization: true
  - Filter Type: Advanced
  - Dimensions: 110×32×38

Serial No.: VENDEN-WD200-2025-0012
Year of Manufacture: 2025
Acquisition Date: 2025-01-10
Acquisition Cost: 450.00 EUR

Ownership:
  - Owner Customer: (empty - we own it)
  - Operator Customer: (empty initially)
  - Lessee Customer: "C-ACME" (Acme Corp)
  - Current Holder: Location "WH01" (Our Warehouse)
  - Current Holder Since: 2025-01-10

Status: Active
```

Physical Composition: Add Cooling Compressor

```
Asset No.: CC-456
Description: Danfoss BD35F Compressor
Parent Asset: D-12345
Classification:
  - Industry: Dispenser Parts
  - Component Type: Cooling System
  - Model: Compressor

Serial No.: DANFOSS-BD35F-2025-789
Year of Manufacture: 2025

Ownership:
  - Owner: (inherited from parent)
  - Current Holder: (same as parent)
```

Physical Composition: Add Electronic Control

```
Asset No.: ECU-789
Description: Control Board WD-200
Parent Asset: D-12345
Classification:
  - Industry: Dispenser Parts
  - Component Type: Electronics
  - Model: Control Board

Serial No.: VENDEN-ECU-2025-345
```

Component BOM (Consumable Parts for D-12345):

Item No.	Description	Quantity	Position
20001	Water Filter, 5-micron	2	Filter Housing
20002	Water Tap, Chrome	1	Front Panel
20003	Drip Tray, Plastic	1	Bottom
20004	UV Lamp, Replacement	1	UV Chamber
20005	Power Cable, EU	1	Rear Panel

Lifecycle: Installation at Customer

Step 1: Purchase from Manufacturer

```
Date: 2025-01-10
Document: Purchase Order PO-2025-001
Action: Receive at warehouse

Holder Transition:
  - From: Vendor "V-VENDEN" (Manufacturer)
  - To: Location "WH01" (Our Warehouse)
  - Type: Purchase
  - Document: PO-2025-001
```

Step 2: Lease to Customer

```
Date: 2025-01-15
Document: Sales Order SO-2025-050 (Rental Agreement)
Action: Deliver to customer site

Holder Transition:
  - From: Location "WH01"
  - To: Customer "C-ACME" (Acme Corp)
  - Type: Lease Out
  - Document: SO-2025-050

Update Asset:
  - Operator Customer: "C-ACME"
  - Lessee Customer: "C-ACME"
  - Current Holder: Customer "C-ACME"
  - Current Holder Since: 2025-01-15
  - In-Service Date: 2025-01-15
  - Next Service Date: 2025-07-15 (6 months)

Asset Location (custom field):
  - Customer Site: "Acme HQ"
  - Floor: "Floor 2"
  - Room: "Break Room"
  - GPS: 56.9496° N, 24.1052° E
```

Step 3: Maintenance Service

```
Date: 2025-07-15
Action: Filter replacement

Service Record (could be via Service Order in Phase 2):
  - Service Type: Preventive Maintenance
  - Work Done: Replace water filters, clean UV lamp, test cooling
  - Technician: "TECH-05" (John Doe)
  - Duration: 30 minutes

Components Replaced:
  - Item 20001 (Water Filter): Quantity 2 replaced

Update Asset:
  - Last Service Date: 2025-07-15
  - Next Service Date: 2026-01-15 (6 months)

(No holder transition - serviced on-site)
```

Step 4: End of Lease

```
Date: 2027-01-20
Action: Customer returns dispenser

Holder Transition:
  - From: Customer "C-ACME"
  - To: Location "WH01"
  - Type: Lease Return
  - Document: Credit Memo CM-2027-010

Update Asset:
  - Operator Customer: (clear)
  - Lessee Customer: (clear)
  - Current Holder: Location "WH01"
  - Current Holder Since: 2027-01-20
  - Status: Inactive (pending inspection)
```

Search/Filter Scenarios:

1. "Show all dispensers at Acme Corp" → Filter: Current Holder = Customer "C-ACME"
2. "Show all WD-200 series dispensers" → Filter: Model Series = "WD200"
3. "Show all dispensers needing service this month" → Filter: Next Service Date between 2025-07-01 and 2025-07-31
4. "Show all dispensers with UV sterilization" → Filter: Attribute "UV Sterilization" = true
5. "Show all dispensers at warehouse" → Filter: Current Holder Type = Location, Current Holder Code = "WH01"
6. "Which customer has dispenser D-12345?" → Lookup by Asset No., see Current Holder
7. "Show service history for D-12345" → View Holder History + Service Records

Billing Integration (Future):

```
Monthly Rental Invoice for Acme Corp:
  - Dispenser D-12345: 50.00 EUR/month (from lease agreement)
  - Last service: 2025-07-15 (filter replacement)
  - Next service due: 2026-01-15
  - Location: Acme HQ, Floor 2, Break Room
```

Risks and Mitigation

1. Technical Risks

Risk: CaptionClass Limitations

Impact: High **Probability:** Medium **Description:** CaptionClass might not work in all UI contexts (reports, Excel export, some page types)

Mitigation:

- Prototype caption management in Week 2
- Test on all page types: List, Card, FactBox, Report
- Create fallback strategy: Static captions + industry code display
- Document known limitations early

Contingency:

- If CaptionClass insufficient for some areas, use Industry Code prefix
- Example: "Fleet (COMM)" instead of just "Fleet"
- Accept some technical terminology in reports/exports
- Provide documentation mapping technical names to business terms

---

Risk: Circular Reference Performance

Impact: Medium **Probability:** Low **Description:** Checking 100-level circular references could timeout with large datasets

Mitigation:

- Set practical max depth to 20 levels (covers 99% of cases)
- Use iterative algorithm (not recursive) to avoid stack overflow
- Cache parent chain during session
- Add timeout protection (1 second max check time)

Contingency:

- Reduce max depth to 10 levels
- Implement async validation for batch operations
- Add database trigger for circular reference check (if AL performance insufficient)

Risk: Attribute Scalability

Impact: High **Probability:** Medium **Description:** 50+ attributes per asset could slow page load, especially FactBox

Mitigation:

- Lazy load attributes (only when FactBox visible)
- Index Attribute Value table on Asset No.
- Cache attribute definitions (don't reload each time)
- Batch load attribute values (one query, not 50)
- Show only top 10 attributes in FactBox, "Show All" button for more

Contingency:

- Reduce max attributes to 20 per industry/level
- Implement attribute groups/tabs (Basic, Advanced, Technical)
- Use separate page for attribute management (not FactBox)
- Consider: Store frequently-used attributes directly on Asset table

---

Risk: Dynamic Page Title Complexity

Impact: Low **Probability:** Medium **Description:** Changing page titles dynamically might not work in all navigation scenarios

Mitigation:

- Test page title changes in all contexts (Role Center, navigation, search)
- Use OnOpenPage trigger to set CurrPage.Caption
- Provide fallback generic caption if terminology not loaded

Contingency:

- Accept generic page titles ("Asset List", "Classification Values")
- Show industry-specific terminology in page header/description instead
- Use breadcrumb FactBox to show navigation path with correct terms

---

2. Business Risks

Risk: User Adoption Resistance (Complexity)

Impact: High **Probability:** Medium **Description:** Users might find two-structure model (classification + physical) too complex

Mitigation:

- Setup wizard with clear explanation and examples
- Video tutorials showing both Rollsberg and Venden use cases
- Start simple: Template creates 2-3 levels only, user expands as needed
- Provide "Quick Start" guide (5 steps to first asset)
- Hands-on training sessions for pilot users

Contingency:

- Offer "Simple Mode" preset: 2 classification levels, no attributes, no parent-child
- Provide consulting services for setup (billable)
- Create industry-specific onboarding packages
- Consider: Phased rollout (classification first, parent-child second)

Risk: Terminology Confusion

Impact: Medium **Probability:** Medium **Description:** Users might confuse classification (organizational) with physical (parent-child)

Mitigation:

- Clear visual distinction in UI: Separate field groups
- Classification section labeled "Category/Type"
- Physical section labeled "Parent Asset / Components"
- Help text explaining difference
- Examples in documentation showing both structures

Contingency:

- Add tooltip help on Asset Card explaining each section
- Provide industry-specific diagrams showing structure
- Consider: Rename "Classification" to "Category" or "Type" (clearer)
- Add wizard step explaining difference with examples

Risk: Competitive Response

Impact: Medium **Probability:** High **Description:** Dynaway or competitors might add multi-industry + configurable terminology

Mitigation:

- Move fast: 12-week development, 6-month launch
- Patent/trademark key innovations (two-structure model, terminology engine)
- Build strong customer relationships (lock-in via customization)
- Focus on superior UX and setup simplicity
- Emphasize BC native integration (not bolt-on)

Contingency:

- Compete on price: $49/user vs. $99/user (50% discount)
- Compete on depth: Unlimited hierarchy vs. competitor's 3 levels
- Bundle with implementation services (value-add)
- Partner with industry associations for credibility

---

3. Integration Risks

Risk: BC Version Compatibility

Impact: High **Probability:** Low **Description:** Future BC versions might break event subscribers or table extensions

Mitigation:

- Use only stable, documented BC APIs
- Avoid deprecated features (already doing: no WITH statements)
- Comprehensive integration test suite
- Monitor BC release notes for breaking changes
- Test on BC preview versions (Insider program)

Contingency:

- Maintain version-specific branches (BC v23, v24, v25)
- Offer rapid hotfix releases (< 48 hours)
- Provide migration tools for breaking changes
- Consider: Version matrix support (test on 3 BC versions)

Risk: Document Integration Failures

Impact: Medium **Probability:** Medium **Description:** Sales/Transfer posting might not trigger events reliably in all scenarios

Mitigation:

- Test all document types thoroughly (Quote, Order, Invoice, Return, Transfer)
- Subscribe to multiple event points (Before, After, OnRun)
- Implement validation before posting (prevent bad data)
- Provide manual correction tools (if auto-update fails)
- Log all integration attempts (success/failure)

Contingency:

- Offer manual holder update procedure (always available)
- Implement batch reconciliation job (find discrepancies)
- Provide audit report for mismatches (Asset vs. Posted Documents)
- Consider: User confirmation prompt before auto-update

---

4. Project Delivery Risks

Risk: Scope Creep

Impact: High **Probability:** High **Description:** Customers/stakeholders might request features beyond core plan

Mitigation:

- Lock scope for Phases 1-3 (12 weeks)
- Maintain Phase 4 backlog for post-launch features
- Clear communication: "That's a great idea for v1.1"
- Offer customization as paid service (separate from base product)
- Steering committee approval required for scope changes

Contingency:

- Extend timeline by 2 weeks if critical feature discovered
- Negotiate reduced scope in other areas (trade-offs)
- Release as v1.1 instead of v1.0 (additional 4 weeks)
- Defer Phase 3 features to v1.1 if needed

Risk: Resource Availability

Impact: High **Probability:** Medium **Description:** Developers might be pulled to other urgent projects

Mitigation:

- Dedicate 2-3 developers full-time (100% allocation)
- Cross-train team members (redundancy)
- Pair programming on critical components (knowledge sharing)
- Document architecture continuously (not just at end)

- Weekly status reviews with management

Contingency:

- Extend timeline by 1 week per developer-week lost
- Reduce Phase 3 scope (defer UI polish, some extensions)
- Hire contractor for specific components (if budget available)
- Prioritize core functionality over polish

Risk: Inadequate Testing Time

Impact: High **Probability:** Medium **Description:** Discovering bugs late in Week 11-12 might delay launch

Mitigation:

- Test continuously (not just Week 11)
- Automated tests run on every commit (CI/CD)
- Phase 1 tests must pass before Phase 2 starts
- User acceptance testing in Week 10 (not Week 12)
- Buffer 2 weeks for bug fixes (Weeks 11-12)

Contingency:

- Extend Week 11-12 by 2 weeks if needed (Week 13-14)
- Release v1.0 with known minor bugs (documented workarounds)
- Rapid v1.0.1 hotfix release (1 week after launch)
- Beta release to pilot customers first (get feedback early)

---

5. Data Migration Risks (Rollsberg)

Risk: Rollsberg Data Migration Complexity

Impact: High **Probability:** Medium **Description:** Existing Rollsberg vessel data might not map cleanly to new model

Mitigation:

- Build migration codeunit in Phase 2 Week 6
- Test migration with real Rollsberg data (anonymized)
- Provide data mapping wizard (user confirms mappings)
- Offer parallel run period (30 days: old + new system)
- Rollback procedure documented and tested

Contingency:

- Maintain compatibility layer: Keep Rollsberg objects, sync to Asset Pro
- Offer manual data migration service (billable consulting)
- Extend transition period to 60-90 days
- Gradual migration: Migrate 10% of assets, test, then 100%

Next Steps

Immediate Actions (This Week - Week 0)

1. Stakeholder Review & Approval

- Present this document to project sponsor
- Review architecture with technical lead
- Confirm budget and resource allocation
- Get sign-off on timeline (12 weeks)

2. Team Assignment

- Assign 2-3 developers full-time (100% dedicated)
- Assign QA resource (50% starting Week 8)
- Assign technical writer (25% starting Week 10)
- Assign UX designer (25% for page layouts)

3. Environment Setup

- Provision BC development container (BC version 25 or 26)
- Set up version control (Git repository)
- Configure CI/CD pipeline (build + test automation)
- Set up project management tool (task tracking)

4. Symbol Extraction

- Run `../.claude/Extract-All-Dependencies.ps1`
- Verify ObjectIndex.md created successfully
- Test symbol lookup (search for "Table 18" Customer)

5. Kickoff Meeting

- Review architecture with development team
- Discuss design decisions (two-structure model)
- Review object ID ranges (70182300-70182449)
- Clarify coding standards (al-best-practices skill)
- Set up daily standup schedule

Week 1 Actions (Phase 1 Start)

Monday:

- Create AL/src folder structure (Tables, Pages, Codeunits, Enums)
- Create Test/src folder structure
- Update app.json with object ID ranges
- Create git branch: feature/phase1-foundation

Tuesday-Wednesday:

- Implement tables: Asset Setup, Industry, Classification Level, Classification Value
- Implement enums: Industry Template Type
- Create unit tests for table validation

Thursday:

- Implement pages: Asset Setup, Industries, Classification Levels, Classification Values
- Test page navigation and CRUD operations

Friday:

- Implement setup wizard (basic version)
- Create Vehicles industry template (hardcoded for now)
- Code review and merge to main branch
- Sprint retrospective

Decision Points

Before Starting Implementation:

1. Approve Architecture □ / □

- Two-structure model (Classification + Physical) approved?
- CaptionClass strategy viable for terminology?
- Object ID ranges sufficient (70182300-70182449)?
- Test ID ranges acceptable (50100-50199)?

2. Approve Timeline □ / □

- 12-week timeline realistic?
- Phases correctly prioritized (Foundation → Advanced → Polish)?
- Phase 4 scope deferred appropriately (post-launch)?
- Buffer time adequate (Week 11-12 for testing)?

3. Approve Resource Plan □ / □

- 2-3 developers sufficient (skill mix: senior + mid)?
- QA resource availability confirmed (Week 8+)?
- Technical writer availability confirmed (Week 10+)?
- Backup developers identified (if resource unavailable)?

4. Approve Budget □ / □

- Development costs approved (personnel: 12 weeks × 2.5 FTE)?
- Testing environment costs approved (BC containers)?
- Marketing/launch costs approved (post-Week 12)?
- Support infrastructure costs approved?

Sign-off Required:

```
Technical Lead:    _____  Date: _____
```

Project Sponsor: _____ Date: _____

Product Owner: _____ Date: _____

Appendices

Appendix A: Complete Object ID Allocation

Production Range: 70182300-70182449 (150 objects)

Object Type	ID Range	Count	Assigned	Available
Tables	70182300-70182329	30	11	19
Pages	70182330-70182379	50	15	35
Codeunits	70182380-70182399	20	9	11
Enums	70182400-70182409	10	6	4
Reports	70182410-70182419	10	5	5
Table Extensions	70182420-70182429	10	7	3
Page Extensions	70182430-70182439	10	5	5
Reserved	70182440-70182449	10	0	10

Test Range: 50100-50199 (100 objects)

Object Type	ID Range	Count	Assigned	Available
Unit Tests	50100-50149	50	7	43
Integration Tests	50150-50179	30	5	25
Test Library	50180-50189	10	1	9
Reserved	50190-50199	10	0	10

Appendix B: Comparison with Previous Planning

Previous Document (Analysis_and_Planning_Asset_Pro.md):

- Single hierarchy structure (like Item Categories)
- Fixed levels (not truly unlimited)
- Asset-level assignment unclear
- Attribute system similar (good foundation)
- Ownership tracking similar
- Missing: Two-structure separation

Current Document (v2):

- **Two-structure model** (Classification + Physical) - Key innovation
- Truly unlimited classification levels (configurable per industry)
- Clear separation: Classification (organizational) vs. Physical (composition)
- Component tracking: Hybrid (Assets + Items) - Based on user feedback
- Complete CaptionClass implementation (terminology transformation)
- BC document integration deferred to Phase 2 (user feedback)
- More detailed implementation phases (week-by-week)

Why the Change: Based on user feedback during requirement gathering, it became clear that:

1. Classification (organizational taxonomy) and Physical relationships (composition) should be separate
2. Users selected "Completely different from Item Categories" - needed custom approach
3. Users confirmed hybrid component tracking (some as Assets, some as Items)
4. Phase 2 for BC integration preferred (build solid foundation first)

Appendix C: Glossary

Classification Hierarchy: Organizational taxonomy for filtering and reporting. Example: Industry → Type → Category → Asset.

Physical Composition: Actual parent-child relationships representing physical assembly. Example: Vehicle → Engine → Turbocharger.

Component BOM: Bill of Materials - list of BC Items (non-asset parts) used in an asset.

Holder: Current location/owner of an asset. Can be: Customer, Vendor, Location, Cost Center.

Attribute: Custom field defined per industry/level. Example: "VIN", "Serial Number", "Cooling Type".

Industry: Top-level classification root. Example: "Fleet Management", "Dispenser Management", "Medical Equipment".

Classification Level: A level in the organizational hierarchy. Example: Level 1 = "Fleet", Level 2 = "Vessel Type".

Classification Value: Actual value at a classification level. Example: Level 1 Value = "Commercial", Level 2 Value = "Cargo Ship".

CaptionClass: BC mechanism for dynamic field/page captions. Used to transform "Level 1" → "Fleet" based on setup.

Circular Reference: Invalid parent-child relationship where A → B → C → A (creates a loop). Prevented by validation.

Transition: Change of asset holder/owner. Example: Asset moves from Warehouse to Customer.

Template: Pre-configured industry setup (levels, values, attributes). Example: "Vehicles Template", "Dispensers Template".

Appendix D: References

Business Central Documentation:

- AL Development
- Table Relations
- Caption Class
- Event Subscribers

Competitor Analysis:

- Dynaway EAM: https://www.dynaway.com/eam-for-business-central/
- Microsoft AppSource: https://appsource.microsoft.com/

Previous Research:

- `.claude/Objects_System_Research_Report.md` - Rollsberg analysis (2800+ lines)
- `.claude/Analysis_and_Planning_Asset_Pro.md` - Previous planning (1669 lines)

Document Control

Version History:

Version	Date	Author	Changes
1.0	2025-11-04	Claude	Initial planning document (previous)
2.0	2025-11-05	Claude	Complete rewrite with two-structure architecture, based on user requirements gathering

Approval Status: DRAFT - Awaiting Stakeholder Review

Next Review Date: 2025-11-12 (1 week)

Questions for User:

1. Is the two-structure model (Classification + Physical) clear and acceptable?
2. Is the CaptionClass approach for terminology transformation acceptable?
3. Are you comfortable with Phase 2 timing for BC document integration?
4. Do you have any concerns about the implementation timeline (12 weeks)?
5. Should we prioritize Rollsberg or Venden use case for initial testing?

END OF DOCUMENT

Total Pages: ~80 Total Words: ~28,000 Diagrams: 1 (Mermaid ERD) Tables: 20+ Code Examples: 30+ Use Cases: 2 (detailed)

Status: Ready for review and approval. Once approved, development can begin Week 1 (Phase 1).