Microsoft Dynamics® AX 2012

Implementing the Item-Product data management framework for Microsoft Dynamics AX 2012 Applications

White Paper

This document highlights the new patterns used to represent item-product master data. When detailing the new patterns, the document also describes the pattern that is being replaced and how developers should approach updating their code.

http://microsoft.com/dynamics/ax

Date: January 2011

Author: Ievgenii Korovin, Inventory Management, Dynamics AX

Microsoft Dynamics

Table of Contents

Overview **3**
- Terminology 3
- Document purpose 4

Patterns **5**
- Products patterns 5
- Product translations pattern 6
- Product and item identification pattern 6
- Variant configuration technology pattern 7
- Product, Storage, and Tracking dimension groups pattern 7
- Product dimension values pattern 8
- Released product and released product variant pattern 8
- Item group pattern 9
- Item model group pattern 9
- Default order type pattern 10

Products **10**

Product translations **12**

Product and item identification **14**

Variant configuration technology **15**

Product, storage, and tracking dimension groups **17**

Product dimension values **20**

Released product and released product variant **23**

Item group and Item Model group **25**

Default order type (item type BOM removal) **28**

Services **30**
- Common usage scenarios 30
- EcoResProductService 30
- EcoResProductMasterDimValueService 31
- InventItemService 32
- InventInventDimCombinationService 33

Data upgrade **34**
- Product Upgrade 34
- Inventory dimension groups upgrade 36

Appendix **38**
- Data model diagrams 38

Overview

In Microsoft Dynamics AX 2012, Microsoft has enhanced the item-product data management framework to provide flexibility and sustainability across the organization. To accomplish the new functionality, the data model was completely reworked, making many of the core item master tables shared. A side effect of this rework is that every reference to entities that are now shared must be updated to reference the new tables. This document targets developers building new applications for Microsoft Dynamics AX 2012 as well as developers updating their existing application code and data.

The core product data model is completely new. The item data model underwent extensive normalization.

In Microsoft Dynamics AX 2009, all items were stored in a single table (InventTable). There were three item types available: service, BOM (bill of materials), and item. Each item was associated with a mandatory dimension group, item group, and item model group. All possible item combinations were stored in a single table (InventDimCombination). All item creations and setup processes were stored within a company account context, and the virtual table collection approach was used to share the item master data within different company accounts.

In Microsoft Dynamics AX 2012, however, all product master data is shared across all companies, and the virtual table collection concept is no longer available for product master data management. The old item representation (InventTable) still exists. However, it now has a foreign key to the shared product instance (EcoResProduct hierarchy), and it represents the released product concept or a given enterprise product that has become authorized for use within a legal entity. In addition, products can now only be a service type or an item type.

BOM has been removed as an item type, though an item can still have a BOM associated with it; see Default order type (item type BOM removal) later in this document for details. The new default order-type policy on the legal-entity level defines whether the BOM structure will be used and whether a production order should be created to fulfill the demand. You can set the order type to "purchase" and associate a BOM with it if the product type is set to "item."

The inventory dimension group has been split and refactored into three groups: product dimension, storage dimension, and tracking dimension. These groups can be associated with the shared product entity or overridden at the legal-entity level. The old item combinations representation (InventDimCombination) still exists, but it now has a foreign key to the shared product variant entity (EcoResDistinctProductVariant table) and represents a released product variant or a given shared product variant that is authorized for use within a legal entity. All products have the enterprise data definition set up like translation data. First, the product entity must be created on the enterprise or system level. Then the product must be released to a given legal entity, and all company-specific master data must be specified before the released product can be authorized for use for certain processes. The item group and item model group are still defined on legal-entity level; however, the relationship between entities and the old item entity has been refactored completely.

Terminology

Microsoft Dynamics AX 2012 terms:

Term	Definition
Product	An item, service, or right that results from an economic activity.
Distinct product	A uniquely identifiable product. It serves as a core product that does not vary and therefore no product dimensions can be associated with the definition.

Product master	A standard or functional product representation that is the basis for configuring product variants. The variations are configured by selecting a configuration technology, which can be either a set of predefined product dimensions or configurators in sales scenarios.
Product variant	A product variant is the configuration of a product master. Based on the choice of the configuration technology, the variant can be either predefined by using the product dimensions of its master or configured by using a configurator tool.

Document purpose

This document highlights the new patterns used to represent item-product master data. When detailing the new patterns, the document also describes an existing pattern that is being replaced and how a developer should approach updating their legacy code. This document does not describe all new functionality within the item-product data management feature. Instead, this document focuses on the development patterns and how they are implemented. A comprehensive list of new tables and their replacements can be found in the Appendix.

Sections to read if you are developing new code

Developers developing new code for Microsoft Dynamics AX 2012 that references item-product master data should read the following sections of this document:

- Patterns
- Data model diagrams

Sections to read if you are performing code and data upgrade

Developers performing a code upgrade for existing applications should first attempt to identify all references to the defined code patterns and then following the instructions in the related sections to upgrade their code. The code upgrade can be done in any sequence. The following steps are required.

- Identify the pattern your code uses today. Add new fields in the product data model if you want to use enterprise product definition references. You can still create a new reference to the released product, which is stored in the InventTable.
- Add new fields in the data model to hold foreign keys to the new entities in case the old way of referencing the entity is obsolete.
- Prefix the old foreign key fields with "DEL_" to mark them for future deletion.
- Create a data upgrade script to populate the new fields with data from the old fields (see Data upgrade). Reuse the out-of-the-box item-to-product upgrade features to manage and consolidate your product master data.
- Update the user interface to use new controls appropriate for the pattern defined. The new controls will leverage the new foreign keys that you added to your data model. New lookup forms are present to lookup product and product dimension values.
- Update the references and business logic in your X++ classes and table methods to use the new code patterns defined in the Patterns section of this document.
- Update existing reports to leverage the new data model.

Patterns

This section describes the patterns you should be familiar with when working with the item-product data management framework.

To review the physical models for the tables, see the Data model diagrams section of this document.

Products patterns

This pattern applies to the following tables:

- EcoResProduct
- EcoResProductMaster
- EcoResDistinctProductVariant
- EcoResDistinctProduct

Previous versions

In previous versions, all inventory items and services were company-specific and were stored in the InventTable. Items could vary in system-predefined item dimensions (size, color, and configuration). The system used the concept of item combinations to associate an ItemId with item dimensions. All item and item-combinations master data were stored in a company context.

The virtual table collection approach was the recommended way to share item master data across companies, if needed.

Microsoft Dynamics AX 2012

All products are stored as system master data, which allows organizations to create and maintain shared product definition data.

The "product master" concept encompasses the product definition, which can have variants in product dimensions (color, size, and configuration).

The "product variant" concept encompasses a product that has a number of associated product dimension values (color, size, and configuration). It replaces the previous "item combination" concept. It is important to note that a product variant has all attributes and behaviors that any other type of product has.

The "distinct product" concept encompasses a product that does not vary in product dimensions and therefore cannot have a product dimension group associated to it.

The products in a system can be released to a legal entity to become available for various processes (sales, purchase, and production). The same shared product variants can be available for production in one company, but not available for the production in another legal entity.

Product dimension values like colors, sizes, and configuration are stored in the system tables (EcoResColor, EcoResConfiguration, and EcoResSize) and are immutable. The InventDim product dimensions values point to the EcoResColor.Name, EcoResConfiguration.Name, and EcoResSize.Name fields.

Product translations pattern

This pattern applies to the following table:

- EcoResProductTranslation

Previous versions

In previous versions, the system allowed the user to specify descriptions for items and item combinations in different languages. These descriptions were stored in the Item translation table (InventTxt).

At the same time, there could be only one Name specified for an item or an item combination (inventTable.itemName and inventDimCombination.Name fields) in the company-specific language.

The item entity utilized the kernel "alias" functionality. The InventTable.nameAlias field could be used as the default substitution of the ItemId field, for instance, on the order line to find the item. The Name "alias" value was set in the company-specific language.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the system allows users to define multi-language translations for shared product definitions. Out of the box, the product localization attributes are the product name and the product description. Product search name has been introduced in order to control shared and legal-entity–specific values to the product "alias" functionality.

By default, the product has its name and translation in the system language (enterprise organizational language).

The pattern that is applied to model product localization is consistent across the application to model localization support for shared entities. The general system translation forms are used to manage product translation, which enables a consistent user experience in the way that the user interacts with the various entities in the system regarding multi-language translation (product, attributes, catalog descriptions, and so on).

Product and item identification pattern

This pattern applies to the following:

- ProductNumber
- ItemId
- EcoResProductIdentification
- EcoResProduct.DisplayProductName
- InventTable.ItemId

Previous versions

In Microsoft Dynamics AX 2009, ItemId was the main item identification value. A common pattern was to add a relation to InventTable by the ItemId field in order to model a relationship to the item master; and then to add the InventDimID field to the table to represent an item master (in case InventDimID was a blank value) or a given item combination; and finally to specify additional storage dimensions, which were controlled by the item dimension group setup.

There was no single way to ensure that the same items in the different companies were equal. The best practice was to reuse the same ItemId values or to use virtual table collections to share the item master data across companies.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, product is identified by a unique shared product number value, which is a natural key of the product entity. Not all types of products can have a unique product number. The product variants do not have their own shared product number since they are always identified by the relation to their product master.

The **DisplayProductNumber** attribute helps to identify product variants quickly and uniquely in the application user interface.

Internally, all products have their own system-unique surrogate key, which should always be used to model the relationship to the product.

The InventTable still has an ItemId value, which by default will be equal to the corresponding product shared number value. However, in certain cases, the system allows having an itemId value different from its product number value. This option allows having different identification strategies for the same product instance in different legal entities.

Variant configuration technology pattern

This pattern applies to the following table:

- EcoResVariantConfigurationTechnology

Previous versions

In Microsoft Dynamics AX 2009, the item master contained a number of fields that were used for various features to create new item combinations. This included features like Product Builder (PBAItemConfigurable, PBAItem*) and Standard Configurator (Configurable, ConfigSimilar). In addition, the item master contained fields that directly affect the creation of new item combinations (ItemDimCombinationAutoCreate).

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the product master contains a mandatory policy that identifies the process of its variant creation. This shared policy is called "variant configuration technology." This setup identifies the only possible feature that can be used to create related product variants. The Microsoft Dynamics AX 2009 code base is adjusted and modified to follow the product master setup, so any modification in this area should be adjusted as well.

Product, Storage, and Tracking dimension groups pattern

This pattern applies to the following tables:

- EcoResProductDimensionGroup
- EcoResStorageDimensionGroup
- EcoResTrackingDimensionsGroup

Previous versions

In Microsoft Dynamics AX 2009, the item dimension group was a mandatory item master setting. You could not create an item without specifying a dimension group, and each item always had to have one—and only one—dimension group associated to it. The dimension group was company-specific and was logically grouped around item dimensions and storage dimensions. The virtual table collection approach was the recommended way to share dimensions groups across companies, if necessary.

Microsoft Dynamics AX 2012

The item dimension group is split into three groups: the product dimension group, storage dimension group, and tracking dimension group. These are the shared entities. They are assigned to the system product definition and can be assigned to the legal-entity level as well. In other words, you can define

how your product should behave in a centralized way (always controlled by serial number) or you can define legal-entity–specific behavior. Note that the dimensions group setup has been changed in order to ensure better consistency.

Product dimension values pattern

This pattern applies to the following:

- EcoResProductMasterDimValue and EcoResProductVariantDimValue hierarchies
- InventDim

Previous versions

In Microsoft Dynamics AX 2009, item master dimensions (color, size, and configuration) were stored in company-specific tables and could not be shared across multiple companies. When the item or item combination was selected on the order line, the InventDim structure held relationships to the company-specific item dimensions.

The system allowed lookup of company-specific item dimensions on the order line and always validated them against existing item combinations.

The item master contained default dimensions, which could be set and used as the default user choice during journal line creation.

Microsoft Dynamics AX 2012

Product dimensions are shared and are attribute-based. Three predefined attributes exist, representing color, size, and configuration dimensions. These attributes are related to the product dimensions and have nothing to do with the EcoRes* attributes framework in Microsoft Dynamics AX 2012.

The product dimension values are stored in a shared, immutable table (EcoResColor, EcoResSize, and EcoResConfiguration). The database schema around product dimensions allows adding new attributes, which can be used to represent new product dimensions. For example, you could have T-shirt sizes (X, L, M) and shoe sizes (8, 10, 12).

The product master holds relations to all possible product dimensions values within one product attribute. The product variant holds one—and only one—relation to the product dimensions value within one product attribute.

The InventDim structure holds the relationships to the shared product dimension values.

The system allows lookup only of legal-entity–specific product dimensions on the order line and always validates them against release product variants for the current legal entity. On the shared level (product management forms, tables, and so on), the system allows lookup of all product dimensions and validates them again all product variants.

The “view details” action on the form and list pages (like the “go to main table” action in Microsoft Dynamics AX 2009) follows the logic described later in this document.

Released product and released product variant pattern

This pattern applies to the following tables:

- InventTable
- InventDimCombination

Previous versions

All inventory items and services were stored in the InventTable and were company-specific. Items could have various predefined item dimensions (size, color, and configuration). The system used the

concept of item combinations (ItemId plus item dimensions). All item and item-combinations master data were stored in a company context.

The virtual table collection approach was the recommended way to share item master data across companies, if necessary.

Microsoft Dynamics AX 2012

Products and product variants must be released to a specific legal entity before they can be used with that legal entity. The InventTable and its associated tables use the concept of the released product and can also be treated as the instance of the particular product in the current company. The mandatory product foreign key is added to the InventTable. The InventTable represents the instance of either a product master or a distinct product.

The InventDimCombination table and its associated tables use the concept of the released product variant and can also be treated as the instance of a product variant in the current company. The mandatory DistinctProductVariant foreign key is added to the InventDimCombination table. The InventDimCombination always represents the instance of a product variant.

Item group pattern

This pattern applies to the following table:

- InventItemGroup

Previous versions

In Microsoft Dynamics AX 2009, the item group was a mandatory item master setting. You could not create an item without specifying the item group, and every item always had to have one—and only one—item group associated with it. The item group was company-specific and contained important account setup for purchase, inventory, and sales processes. The virtual table collection approach was a recommended way to share these settings across companies, if necessary.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the item group is not mandatory for the product, but it is still company-specific. In other words, you can create the instance of a released product without specifying the item group at the creation step or at the product release step. The item group can be assigned later to the product when needed (or by applying a product template).

The product can be used for certain processes without having these values set (for example, trade agreement setup). If a process expects an item group to be set up (for example, sales-order line creation), a run-time error will occur, and the user will be prompted to set the value on the product before continuing his or her action.

Item model group pattern

This pattern applies to the following table:

- InventModelGroup

Previous versions

In Microsoft Dynamics AX 2009, the item model group was a mandatory item master setting. You could not create an item without specifying the item model group, and each item always had to have—one and only one—item model group associated with it. The item model group was company-specific and contained important setup information about the inventory model and warehouse management settings. The virtual table collection approach was a recommended way to share these settings across companies, if necessary.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the item model group is not mandatory for the product, but it is still company-specific. In other words, you can create the instance of a released product without specifying the item model group at the creation step or at the product-release step. The item model group can be assigned later to the product when needed (or by applying a product template).

The product can be used for certain processes without having these values set (for example, trade agreement setup). If a process expects an item model group to be set up (for example, sales order line creation), a run-time error will occur and the user will be prompted to set the value on the product before continuing his or her action.

Default order type pattern

This pattern applies to the following table:

- InventItemSetupSupplyType

Previous versions

In Microsoft Dynamics AX 2009, a special value of ItemType::BOM could be associated to an item. This itemType meant that the item could have a BOM and route associated with it. In other words, the item master had to be of "BOM" type in order to have a Bill of Materials and to be produced. An item master whose type was set to "Item" normally would represent only items designated for purchase.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the ItemType::BOM value has been removed. The value that represents items that can be manufactured or purchased is now itemType::Item. A new mandatory default order setting has been introduced to indicate when an item must be purchased or manufactured by default. This policy is stored in the InventItemSetupSupplyType table.

Products

As in Microsoft Dynamics AX 2009, the product type classifies whether a product is tangible or intangible (an item or a service). In Microsoft Dynamics AX 2012, this distinction is made as a subclassification of the product.

There are three subtypes of products: product master, distinct product, and product variant. Product variants will be variants of product masters.

This concept is designed as a super-type/subtype table hierarchy to decouple specific properties between different entities that are of the same nature.

A product master is associated to the EcoResProductDimensionGroup and can have a range of predefined product dimensions available. For example, a T-shirt could be a product master with various sizes (S, M, L) and colors (Blue, Green). The product variant represents variations of the product master.

By contrast, a distinct product cannot have variations (that is, it cannot have any product dimensions specified).

EcoResProduct	
FieldID	**Description**
RecId	Surrogate product key. Use RecID as the primary reference to the product entity.
DisplayProductNumber	DisplayProductNumber is concatenated display value based on: • Product number for Product masters and distinct products • Product Master product number + ":" + listed set of the product dimensions for the distinct product variant (for example, "MyProductMasterId : Red : 10 : MyConfiguration"). The Product number is mandatory for product masters and distinct products, while it can never be set for product variants.
Search name	Product search name
ProductType	Can be Item or Service

EcoResProductMaster	
FieldID	**Description**
RecId	Surrogate product key. Should be used as the primary reference to the product entity
VariantConfigurationTechnology	Defines product variant configuration technology. Can be Predefined variant, Dimension-based configuration, or Constraint-based configuration

EcoResDistinctProduct	
FieldID	**Description**
RecId	Surrogate product key. Should be used as the primary reference to the product entity

EcoResDistinctProductVariant	
FieldID	**Description**
RecId	Surrogate product key. Should be used as the primary reference to the product entity
Product master	Surrogate key of the related product master

```
// Example – perform specific logic for product master of type services
// if the product can be used for non-inventory trade

if (productMaster.ProductType == EcoResProductType::Service
&& productMaster.isSupportedForNonInventoriedItems())
)
{
 // do something
}
```

```
// Example – find reference for product variant for the current product master
// which has following product dimensions:
// configuration – 001
// size – XL
// color - Red

EcoResDistinctProductVariantRecId productVariantRecId;

container  productDimensions = EcoResProductVariantDimValue::getDimensionValuesContainer('001',
'XL', 'Red');

// find product variant surrogate key for product master and specified product dimensions
productVariantRecId  =
EcoResProductVariantExistMgr::findVariantByMasterAndDimensions(productMaster.RecId,
productDimensions));
```

```
// Example – create new product variant for the current product master
// which has following product dimensions:
// configuration – 001
// size – XL
// color - Red
// Set search name to "gold variant"

EcoResDistinctProductVariantRecId productVariantRecId;

container  productDimensions = EcoResProductVariantDimValue::getDimensionValuesContainer('001',
'XL', 'Red');

// create new product variant for product master and specified product dimensions
    productVariantRecId = EcoResProductVariantManager::createProductVariant(
         productMaster.RecId,
        'gold variant',
        productDimensions);
```

Product translations

All product translations are stored in the EcoResProductTranslation table.

EcoResProductTranslation	
FieldID	**Description**
Product	Reference to the product surrogate key.
LanguageID	Reference to language
Description	Product description
Name	Product name

Here is the table mapping:

Previous versions	Microsoft Dynamics AX 2012
InventTxt	EcoResProductTranslation
InventTable.itemName	EcoResProductTranslation
InventDimCombination.Name	EcoResProductTranslation

Existing code that uses an InventTxt table in the following manner:

```
        itemFreeTxt = InventTxt::find(this.ItemId,languageId).Txt;
```

… should be replaced with the following pattern:

```
EcoResProductDescription   productDescription;

    productDescription = EcoResProductTranslation::findByProductLanguage(this.Product,
languageId).Description;
```

Existing code that uses inventTable.itemName in the following manner:

```
//BP Deviation documented
display Name displayItemName()
{
    ;
    return InventTable::find(this.ItemId).ItemName;
}
```

… should be replaced with the following pattern:

```
//BP Deviation documented
display Name displayItemName()
{
    ;
    return InventTable::find(this.ItemId).defaultProductName();
}
```

Existing code that uses InventDimCombination.Name in the following manner:

```
salesBasketLine.ItemConfiguration = InventDimCombination::find(ItemId, inventDim).Name;
```

… should be replaced with the following pattern:

```
salesBasketLine.ItemConfiguration = InventDimCombination::find(ItemId,
inventDim).defaultProductName();
```

The system form should display product data in system language. Consider adding an outer join EcoResProductTranslation data source with the following **init** method:

```
public void init()
{
    super();

    EcoResProductTranslation::queryAddDefaultLanguage(this.query());
}
```

The legal entity form should display product data in company language. Consider adding an outer join EcoResProductTranslation data source with the following **init** method:

```
public void init()
{
    super();

    EcoResProductTranslation::queryAddCompanyLanguage(this.query());
}
```

The InventTable.nameAlias field will be set during the product release process and is set from the product search name. Code upgrade for this field is not required. However, if the product search name will be changed on the shared level, the inventTable.nameAlias field will be synchronized to the latest version, unless it already has been modified locally.

Product and item identification

The EcoResProductIdentifier table contains product IDs or unique product numbers. The product master and distinct product can have product numbers.

The product variants are always identified by the related product master number and the set of product dimensions for the particular variants. Consider the following examples:

Example 1

```
// Example - find product master record by product number

EcoResProductMaster  productMaster;

// find product master buffer by providing product number
productMaster = EcoResProduct::findByProductNumber("MyProductMAster");
```

Example 2

```
// Example - populate DisplayProductNumber for product variant

EcoResDisplayProductNumber    displayProductNumber;
EcoResProductNumber       productNumber;

container  productDimensions =
EcoResProductVariantDimValue::getDimensionValuesContainer('Config', 'Size', 'Color');

// assume you have valid reference to productMaster
productNumber = EcoResProductIdentifier::findByProduct(productMasterRecId).ProductNumber;
```

```
// populate DisplayProductNumber for product variants
displayProductNumber =
EcoResProductNumberBuilderVariant::buildFromProductNumberAndDimensions(productNumber,
productDimensions);

// use displayProductNumber to present user with the product variant identification
```

When the product is released to the legal entity, the ItemId will default to the product number in the **EcoResProductReleaseManager.setInventTableFields** method.

Variant configuration technology

The various fields on the InventTable in Microsoft Dynamics AX 2009 were used by various features to create new item combinations. In Microsoft Dynamics AX 2012, this setup is reflected in product master modeling policy.

Here is the table mapping:

Previous versions	Microsoft Dynamics AX 2012
InventTable.Configurable	EcoResProductMaster.VariantConfigurationTechnology == DimensionBased
InventTable.ConfigSimilar	EcoResProductMasterModelingPolicy.IsReuseConfigurationEnabled
InventTable, ItemDimCombinationAutoCreate	EcoResProductMaster.VariantConfigurationTechnology == Predefined variants
	EcoResProductMasterModelingPolicy.IsAutomaticVariantGenerationEnabled

The following code patterns should be used for code upgrades for the Microsoft Dynamics AX 2009 Standard Configuration feature. The examples are taken from the InventTable.IsConfigurable table instance method.

Microsoft Dynamics AX 2009

```
boolean isConfigurable()
{
    return this.Configurable && this.inventItemType().canHaveBOM();
}
```

Microsoft Dynamics AX 2012

```
public boolean isConfigurable()
{
    return EcoResProductMaster::find(this.Product).isDimensionBased() &&
this.inventItemType().canHaveBOM();
}
```

Microsoft Dynamics AX 2009

```
if (!inventTable.ConfigSimilar)
{
// do something
}
```

Microsoft Dynamics AX 2012

```
isReuseExistingConfigurationEnabled =
EcoResProductMaster::find(inventTable.Product).modelingPolicy().isReuseExistingConfigurationEnabl
ed();


if (!isReuseExistingConfigurationEnabled)
{
    // do something
}
```

The following code patterns should be used for code upgrades for Microsoft Dynamics AX 2009 automatic item combination creations.

Microsoft Dynamics AX 2009

```
result = this.inventTable().ItemDimCombinationAutoCreate
if (result)
{
// do something
}
```

Microsoft Dynamics AX 2012

```
if (this.isPredefinedVariants())
{
    result =
EcoResProductMasterModelingPolicy::findByProductMaster(this.RecId).isAutomaticVariantGenerationEn
abled();
}


If (result)
{
// do something
}
```

The new Product Configuration feature available in Microsoft Dynamics AX 2012 can operate with the product master only if the variant configuration technology is set to ConstraintBased products. Consider the following code pattern for potential code upgrade purposes:

```
// Example – verifies if the current released product can be used by Product Configuration
features and
// has Default Order Type set to Production to allow the product be used for master planning with
in
// current company


// assumption that inventTable is a valid record
isPCandProducedByDefault =   EcoResProductMaster::find(inventTable.Product).isConstraintBased()
            && this.isProducedByDefault();


If (isPCandProducedByDefault)
{
    // do something
}
```

Product, storage, and tracking dimension groups

In Microsoft Dynamics AX 2012, the setup of inventory dimension groups has been changed. This section describes the primary changes and illustrates how existing Microsoft Dynamics AX 2009 code patterns can be rewritten to use the new classes.

In Microsoft Dynamics AX 2009, the inventory dimensions were divided into two categories:

- Item dimensions
- Storage dimensions

In Microsoft Dynamics AX 2012, inventory dimensions have been divided into the following three categories:

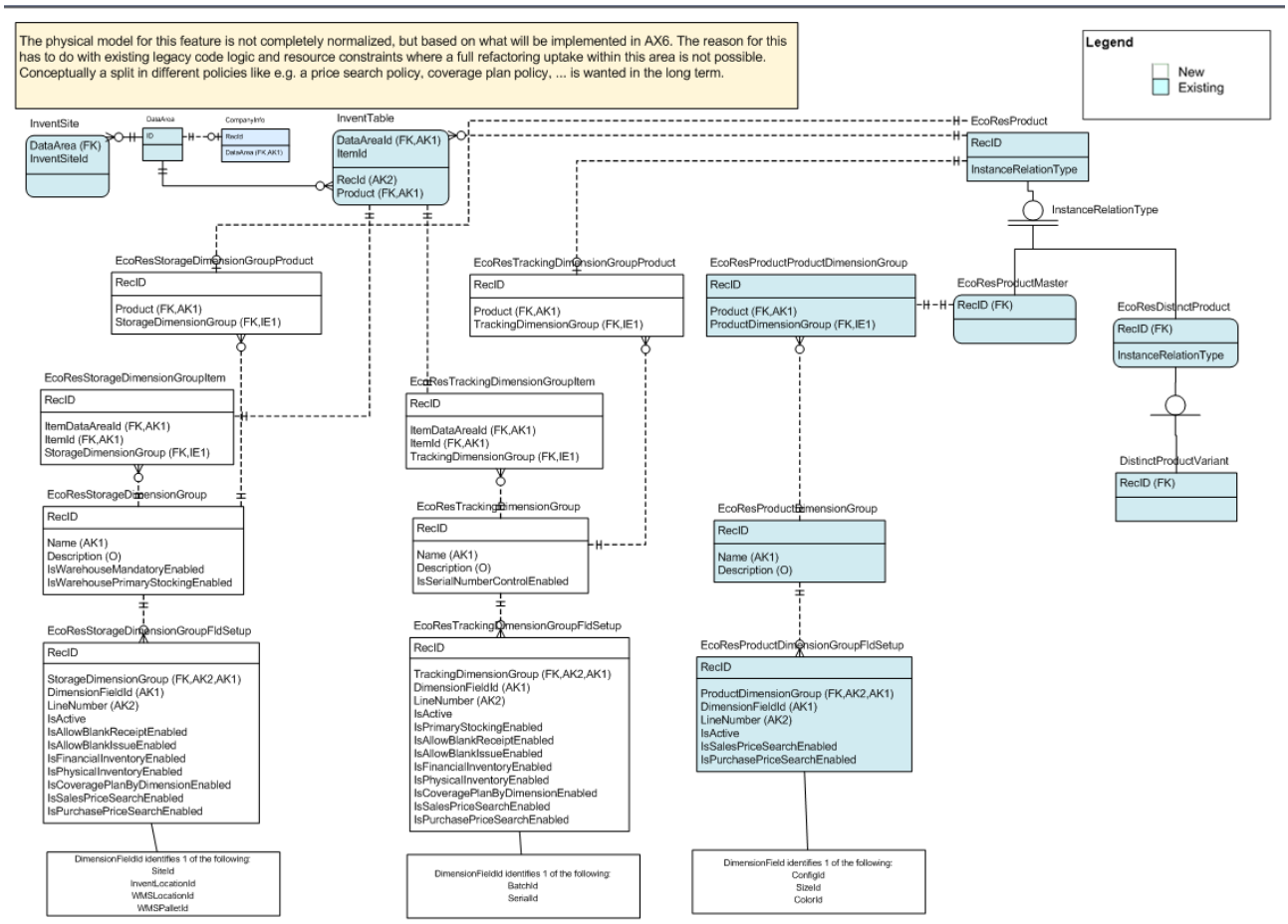
- Product dimensions. These correspond to the old item dimensions.
- Storage dimensions. These are Site, Warehouse, Location, and Pallet.
- Tracking dimensions. These are Batch number and Serial number.

The product dimension group can be assigned only when creating the product.

The storage and tracking dimensions groups can be assigned to the product on two levels:

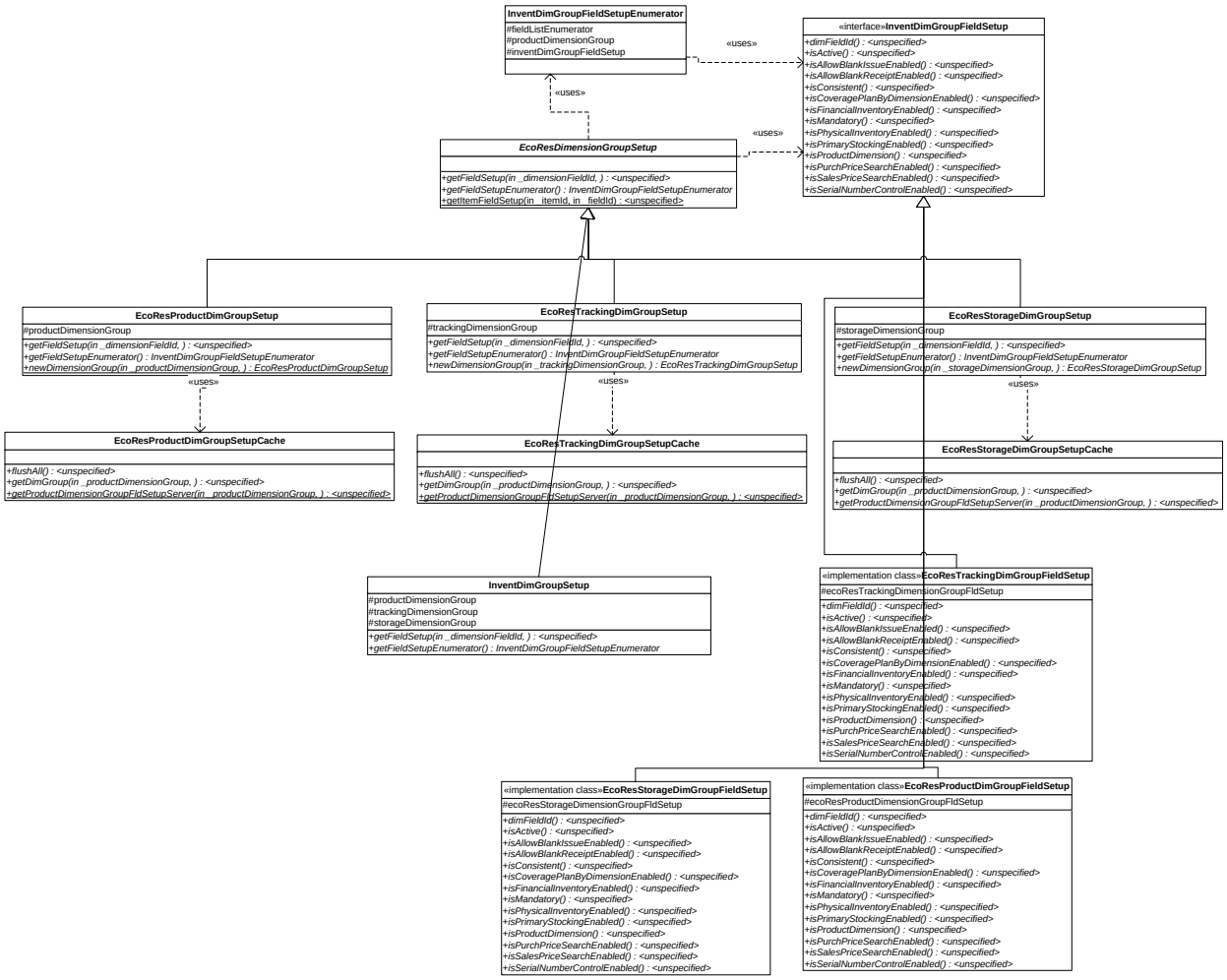
- System level, in which case they will be used in all legal entities where the product is released.
- Legal-entity level, in which case they will be used for the product only in that specific legal entity.

The following diagram shows the physical data model:



In Microsoft Dynamics AX 2009, the **InventDimSearch** class was used to get information about the inventory dimension setup. This class has been deleted and several new classes have been implemented to make it possible to get information about the dimension setup.

The following diagram shows the classes:



In order to determine whether a dimension for a specific field is active in Microsoft Dynamics AX 2009, the following code could be used:

```
InventDimSearch     inventDimSearch = new InventDimSearch();
InventDimGroupId    dimGroupId = InventTable::find(ItemId). DimGroupId;
    ;
    if (inventDimSearch.findActive(dimGroupId,fieldNum(InventDim,wmsPalletId)))
    {
        info(strFmt("The palletId dimension is active for dimension group %1",dimGroupId));
    }
```

In Microsoft Dynamics AX 2012, this can be achieved by the following code:

```
    InventTable                 inventTable;
    InventDimGroupSetup         inventDimGroupSetup;
    InventDimGroupFieldSetup    inventDimGroupFieldSetup;

    inventDimGroupSetup = InventDimGroupSetup::newInventTable(inventTable);
    inventDimGroupFieldSetup =
inventDimGroupSetup.getFieldSetup(fieldNum(InventDim,WMSPalletId));

    if (inventDimGroupFieldSetup.isActive())
    {
info(strFmt("The palletId dimension is active for dimension group
%1",inventDimGroupSetup.getStorageDimensionGroup()));));
    }
```

Product dimension values

Color, size, and configuration are used to define the "item dimensions" concept in Microsoft Dynamics AX 2009. A similar approach is used in Microsoft Dynamics AX 2012 to define product dimensions, but these entities are placed on the system level.

Here is the table mapping:

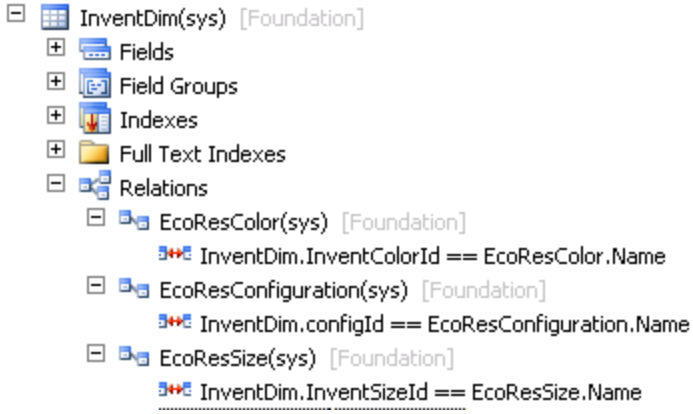
Previous versions	Microsoft Dynamics AX 2012
InventSize	EcoResSize, EcoResProductMasterSize
InventColor	EcoResColor, EcoResProductMasterColor
ConfigTable	EcoResConfiguration, EcoResProductMasterConfiguration,
InventDim (InventColorId, InventSizeId, ConfigId fields)	EcoResProductDimensionAttribute
InventDimCombination	EcoResProductVariantSize , EcoResProductVariantColor, EcoResProductVariantConfiguration

This EcoResProductMaster* tables describe all possible dimension values that can be associated with the products, such as size, color, and so on. Each such relation is categorized by a product dimension attribute, which is stored in EcoResProductDimensionAttribute table.

The three out-of-the-box product dimension attributes are shared across product models and product variants. The attributes are used to establish relationships for InventDim color, size, and configuration fields and to establish main mapping between InventDim values and product dimensions. The attributes are:

- Color
- Size
- Configuration

At the same time, the InventDim product dimension fields also hold relationships to the EcoRes* tables.



Consider the following example tables for size attributes; color and configuration attributes are treated in the same way:

EcoResProductDimensionAttribute		
ID	**Name**	**InstanceRelationType**
1	Size	1000 (EcoResSize)
2	Color	1001 (EcoResColor)
3	Configuration	1002 (EcoResConfiguration)

EcoResProduct	
RecId	**DisplayProductNumber**
1	GenericTshirt-A
2	GenericTshirt-B

EcoResProducMaster	
RecId	**VariantConfigurationTechnology**
1	DimensionBased
2	ConstraintBased

EcoResSize	
RecId	**Name**
1	S
2	M
3	L

EcoResProducMasterSize				
RecId	**ProductMaster**	**ProductDimensionAttribute**	**Size**	**Description**
1	1 (GenericTshirt-A)	1 (Size)	2 (M)	EU medium
2	1 (GenericTshirt-A)	1 (Size)	3 (L)	EU large
3	2 (GenericTshirt-B)	1 (Size)	1 (S)	EU small
4	2 (GenericTshirt-B)	1 (Size)	3 (L)	US large

For product variants, product dimensions are stored in EcoResProductVariant* tables, and size attributes are stored in EcoResProductVariantSize tables. Only one size can be assigned to the same product variant, so multiple sizes would require setting up multiple variants, as shown in the following examples:

EcoResProduct	
RecId	**ProductId**
3	SpecificTshirt-A
4	SpecificTshirt-B

EcoResProductVariant	
RecId	**ProductMaster**
3	1
4	2

EcoResProductVariantDimensionValue			
RecId	**InstanceRelationType**	**DistinctProductVariant**	**ProductDimensionAttribute**
1	EcoResProductVariantSize	3 (SpecificTshirt-A)	1 (Size)
2	EcoResProductVariantSize	4 (SpecificTshirt-B)	1 (Size)

EcoResProductVariantSize	
RecId	**Size**
1	2 (M)
2	3 (L)

Example 1

```
//Example - find all active released configuration for current release product
// call local checkRoute operation for each found configuration

    inventTable = this.inventTable();

    if (inventTable)
    {
        if (inventTable.configActive())
        {
            configurations              =
EcoResProductVariantDimValue::newProductVariantDim_ConfigId().getDimValues(inventTable.Product);
            configurationsEnumerator    = configurations.getEnumerator();

            while (configurationsEnumerator.moveNext())
            {
                configId = configurationsEnumerator.current();
                checkRoute(configId);
            }
        }
    }
```

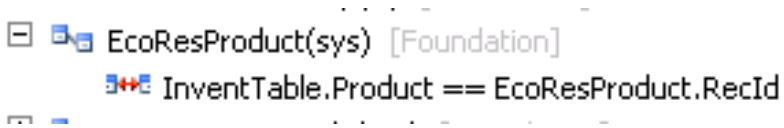
Example 2

```
//Example - add new configuration product dimension value to the product master
// assume we have valid product master reference value - productMasterRecId
// please note that we use
EcoResProductDimensionAttribute::inventDimFieldId2DimensionAttributeRecId()
// method to find default product dimension attribute for the fieldnum(InventDim,ConfigId) field.

EcoResProductMasterManager::addProductDimensionValue(
            productMasterRecId,

EcoResProductDimensionAttribute::inventDimFieldId2DimensionAttributeRecId(fieldnum(InventDim,
ConfigId)),
            configurationName,
            '',
            '');
```

Released product and released product variant

In Microsoft Dynamics AX 2012, **InventTable** represents the concept of a product released to the given legal entity.

A mandatory product foreign key (FK) is added to the **InventTable**:



Here is the mapping for the InventTable fields that are deleted (DEL_):

Previous versions	Microsoft Dynamics AX 2012
ItemName	See product translation pattern.
Configurable	See variant configuration technology pattern.
ConfigSimilar	See variant configuration technology pattern.
DimGroupId	See product dimension group pattern.
ItemDimCombinationAutoCreate	See variant configuration technology pattern.
ItemIdCompany	The company item concept is obsolete. The intercompany chain will use the same product reference (EcoResProduct.RecId FK reference).
ItemType	EcoResProduct.ProductType.

```
//Example 1- find released instance of the product in current company
// if found than print product name in Russian language

// assumption that you have valid product reference (ProductRecId)
InventTable inventTable = InventTable::findByProduct(ProductRecId)

If (inventTable)
{
    Info(inventTable.productName('ru'));
}
```

In Microsoft Dynamics AX 2012, InventDimCombination represents the concept of released product variant to a given legal entity.

There is a mandatory DistinctProductVariant foreign key (FK) added to the InventDimCombination table.

Here is the mapping for the InventTable fields that are marked with "DEL_" for future deletion:

Previous versions	Microsoft Dynamics AX 2012
ConfigId	See product dimension values pattern.
InventSizeId	See product dimension values pattern.
InventColorId	See product dimension values pattern.
Name	See product translation pattern.
AutoCreate	See variant configuration technology pattern.
ItemIdCompany	The company item concept is obsolete. The intercompany chain will use the same product variant reference (EcoRes DistinctProductVariant.RecId FK reference).

```
//Example 1- find released instance of the product variant in current company
// if found than print product variant namein Russian language

// assumption that you have valid product variant reference (ProductVariantRecId)
InventDimCombination inventDimCombination =
InventDimCombination::findByDistinctProductVariant(ProductVariantRecId);

If (inventDimCombination)
{
    Info(inventDimCombination.productName('ru'));
}
```

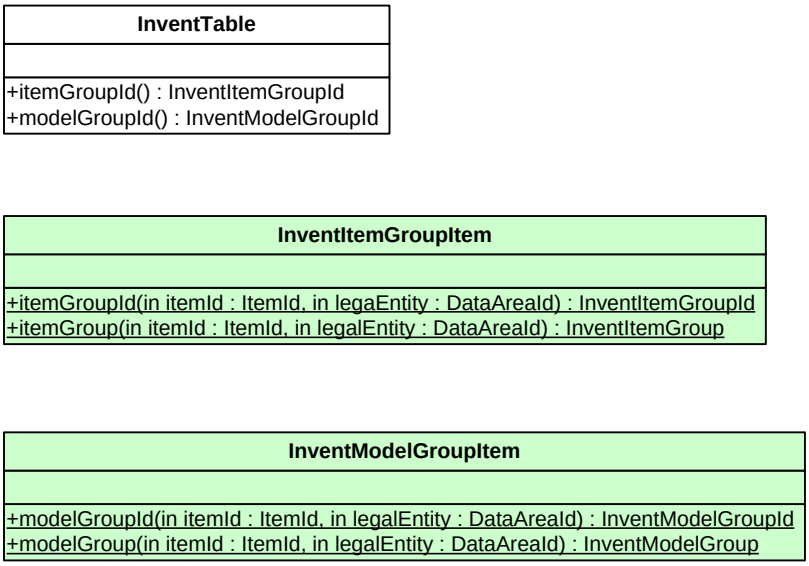
Item group and Item Model group

In Microsoft Dynamics AX 2012, the item group and item model group are optional policies for the released product. These polices are not set by default during product release process unless they are not a part of the product template definition. The product templates can be automatically applied when the product is created from a legal-entity context (such as a Release product list page).

Here is the mapping for InventTable fields that are marked with "DEL_" for future deletion:

Previous versions	Microsoft Dynamics AX 2012
ItemGroupId	InventItemGroupItem, InventItemGroup
ModelGroupId	InventModelGroupItem, ItemModelGroup

The InventItemGroupItem and InventModelGroupItem tables hold the relationships between released product and item group/time model groups. These two methods should be considered for code upgrade purposes.



Here are descriptions of the methods used in the preceding tables:

- **InventTable.itemGroupId**: An instance method that retrieves item group ID for the InventTable record. This method will call InventItemGroupItem::itemGroupId and therefore error will be thrown if item group ID is not specified for the item.

- **InventTable.modelGroupId:** An instance method that retrieves model group ID for the InventTable record. This method will call InventModelGroupItem::modelGroupId and therefore an error will be thrown if model group ID is not specified for the item.

- **InventItemGroupItem.itemGroupId(ItemId _itemId, DataAreaId _dataAreaId):** A static method that retrieves item group ID for the given item ID and legal entity. An error will be thrown if item group ID is not specified for the item.

- **InventItemGroupItem.itemGroup(ItemId _itemId, DataAreaId _dataAreaId):** A static method that retrieves item group for the given item. This method will call InventItemGroupItem::itemGroupId and therefore error will be thrown if item group ID is not specified for the item.

- **inventModelGroupItem::modelGroupId(ItemId _itemId, DataAreaId _dataAreaId):** A static method that retrieves model group ID for the given item ID and legal entity. An error will be thrown if model group ID is not specified for the item.

- **inventModelGroupItem::modelGroup(ItemId _itemId, DataAreaId _dataAreaId):** A static method that retrieves model group for the given item. This method will call InventModelGroupItem::modelGroupId and therefore error will be thrown if model group ID is not specified for the item.

Plan to replace code that uses itemGroupId or modelGroupId fields from the InventTable table, such as in the following two examples:

Example 1

```
if (! itemGroup)
{
itemGroup  = InventTable::find(itemId).ItemGroupId;
}
```

Example 2

```
if (!modelGroup)
{
modelGroup= InventTable::find(itemId).ModelGroupId;
}
```

with the following patterns:

Example 1A

```
    if (!ItemGroup)
    {
        ItemGroup = InventTable::find(_itemId). ItemGroupId();
    }
```

Example 2A

```
    if (!modelGroup)
    {
        modelGroup = InventTable::find(_itemId). ModelGroupId();
    }
```

Note: By default, these APIs will throw exceptions if the groups are not specified (which should be a rare case as these policies are essentially part of the release product setup within a legal entity). The API should ensure that the processes that require the item group and model group to be set up will fail if those groups are missing or not set. In some cases—for example, if you create a trade agreement and the model might not be necessary to the current process—the following API is the recommended way to search for item group and model group.

```
inventTable = InventTable::find('myItem');

If (InventModelGroupItem::findByItemIdLegalEntity(_inventTable.ItemId, _inventTable.dataAreaId))
{
// do something
}

If (InventitemGroupItem::findByItemIdLegalEntity(_inventTable.ItemId, _inventTable.dataAreaId))
{
// do something
}
```

Default order type (item type BOM removal)

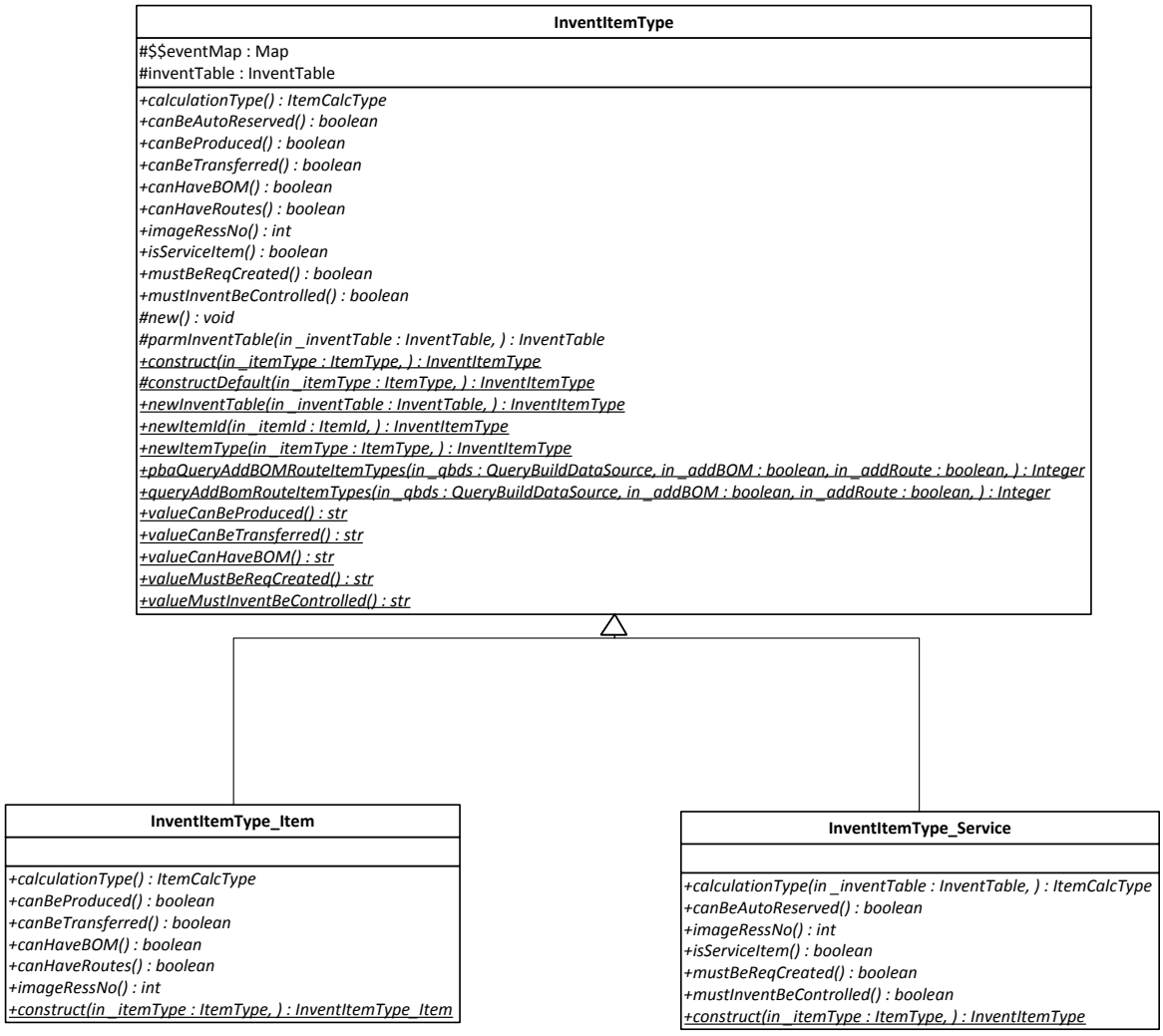
Microsoft Dynamics AX 2009 has items of the following types (**enum ItemType**):

- Item (items that are purchased)
- BOM (items that are produced)
- Service (services)

The issue with this approach was that there is an artificial split between Item and BOM. In reality, both types are items. Furthermore, some items that might normally be purchased might also be manufactured, and vice versa. To remove this limitation, all items in Microsoft Dynamics AX 2012 are combined under the type ItemType::Item, and the ItemType::BOM is removed. At the same time, the system needs to know the default way to acquire the item (purchase or produce). The new InventItemSetupSupplyType table indicates which type of order—a purchase order or a production order—should be used by default (for example, by Material Requirements Planning).

In Microsoft Dynamics AX 2009, the behavior of an item was solely determined by its type. In the **InventItemType** class hierarchy, concrete classes (**InventItemType_Item**, **InventItemType_BOM**, **InventItemType_Service**) were created based on ItemType. These classes were used to determine, for example, whether the item can have BOM, or be produced, or have a route, etc.

In Microsoft Dynamics AX 2012, the **InventItemType_BOM** has been deleted and **InventItemType_Item** is used for all the items.



The use of these classes is slightly different in Microsoft Dynamics AX 2012 than in Microsoft Dynamics AX 2009. In the earlier version, an item's type statically determined whether the item could be produced, with the code relying on the value returned by **InventItemType.canHaveBOM** and **InventItemType.canHaveRoute** methods. Those methods returned true for ItemType::BOM and false for ItemType::Item respectively.

In Microsoft Dynamics AX 2012, ItemType::Item also can have a BOM and route associated. However, the actual order that should be created by default is determined by invoking method **isProducedByDefault** on the InventTable record (or by fetching the value of **InventItemSetupSupplyType.DefaultOrderType**, which is the same).

For example, code that is written like this in Microsoft Dynamics AX 2009:

```
if (bomChanged.inventTable().inventItemType().canHaveBOM())
{ … }
```

… has to be written like this in Microsoft Dynamics AX 2012:

```
if (bomChanged.inventTable().isProducedByDefault())
{ … }
```

Services

Some of the services in previous versions of Microsoft Dynamics AX have been modified in Microsoft Dynamics AX 2012. In addition, new services have been added to fully support read and write operations for distinct products, product masters, product variants, etc.

An assumption has been made that the reader is familiar with the new product model and Application Integration Framework (AIF) services in general.

Common usage scenarios

In previous versions of Microsoft Dynamics AX, an item was created by adding a record to the InventTable table and related tables. The new product model requires that each item (product per company) be related to a product in the EcoResProduct table.

Create a distinct product and release it to a company

Through forms you would normally create a distinct product using the **Product** form and then release it to one or more companies. Services work in a similar fashion.

Creating and releasing a product to a company will require calls to two different services. First, use the product service (EcoResProductService) to define and create the actual product. Then use the item service (InventItemService) to create an instance of the product in a specific company. The item should be associated with the product by specifying the product-identifier in the product tag. The item service will release the product to the company associated to the called endpoint.

Note: Even though creating and releasing a product will require two service calls, you can still add multiple entities to each document. Therefore, creating and releasing products would still only require two calls.

Create product master and product variant

Just as with distinct products, the product service is used to create both product masters and product variants.

First, a product master is defined and created using the product service (EcoResProductService). Then product master dimension attributes can be created—and associated to the product master—using the product master dimension value service (EcoResProductDimValueService). Finally, one or more product variants can be assigned to the product master by using the product service (EcoResProductService).

Release product variant to a company and assign combinations

After a product master has been created by using the product service (EcoResProductService) and master dimension values have been assigned by using the product master dimension value service (EcoResProductDimValueService), you can release the variant to companies. This is done by using the inventory dimension combination service (InventInventDimCombinationService).

The service takes the item ID and inventory dimensions as parameters.

EcoResProductService

The EcoResProductService is a hierarchical service covering three different schemas. By defining different instance relations for the entities, the service can be used to both read and create distinct products, product masters, or product variants.

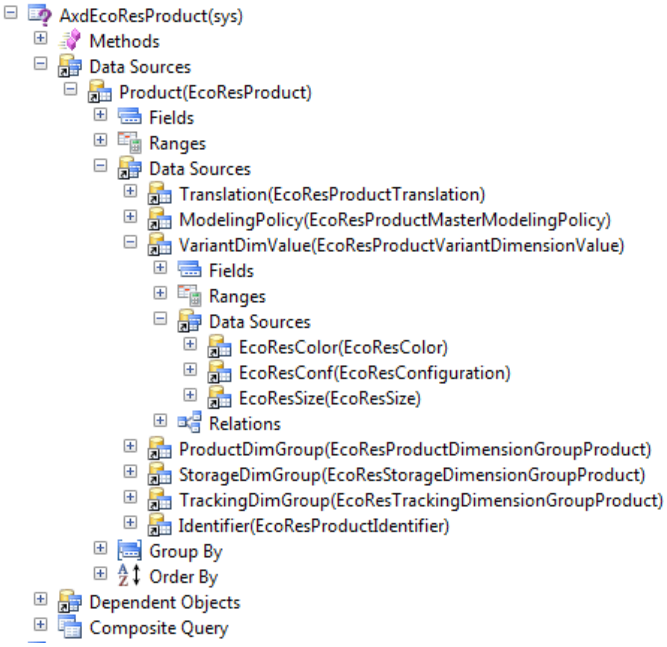
This service uses surrogate key replacement for its references.

Supported operations

The service supports the following operations:

- Create
- Read
- Find
- Findkeys

Query



EcoResProductMasterDimValueService

The EcoResProductMasterDimValueService is used to read or create product dimension values and associate them with an existing product master.

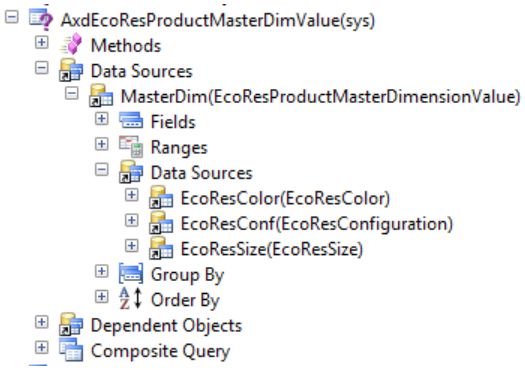
This service uses surrogate key replacement for its references.

Supported operations

The service supports the following operations:

- Create
- Read
- Find
- Findkeys

Query



InventItemService

The InventItemService looks similar to previous versions of Microsoft Dynamics AX. However, there are a few important changes. Creating an item now requires that you pass a reference to a distinct product or a product variant by using the Product tag in the Item entity. The service will automatically create a default (blank) warehouse item, but it no longer supports creating specific warehouse items during creation of the item. Instead, the new InventItemLocationService should be used to assign warehouse items to items.

If the related product has storage or tracking dimensions assigned, the created item must either have the same storage and tracking dimensions, or the fields should be left blank. For data-consistency reasons, having conflicting storage or tracking dimensions is not allowed. If the fields are left blank for the item, the service will automatically copy the storage and tracking dimensions from the assigned product.

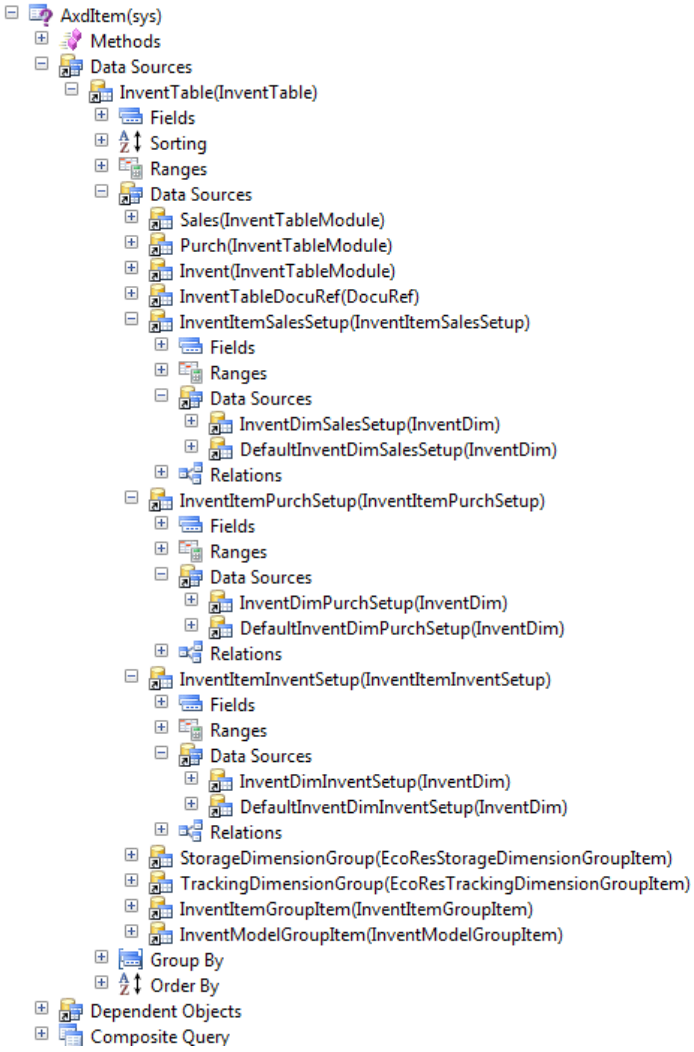
This service uses surrogate key replacement for its references.

Supported operations

The service supports the following operations:

- Create
- Read
- Find
- Findkeys

Query



InventInventDimCombinationService

The InventInventDimCombinationService should be used to read and write combinations of released product variants. The structure of the service is quite simple. It requires a product variant and an item ID is passed in along with an InventDim. If the InventDim does not already exist, it will be implicitly created.

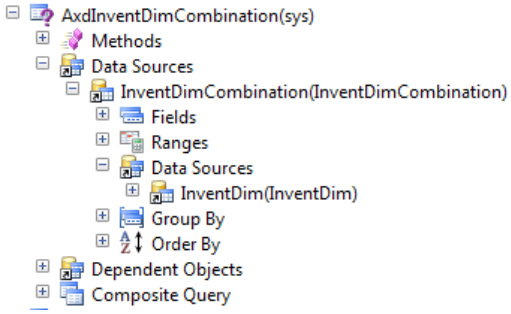
This service uses surrogate key replacement for its references.

Supported operations

The service supports the following operations:

- Create
- Read
- Find
- Findkeys

Query



Data upgrade

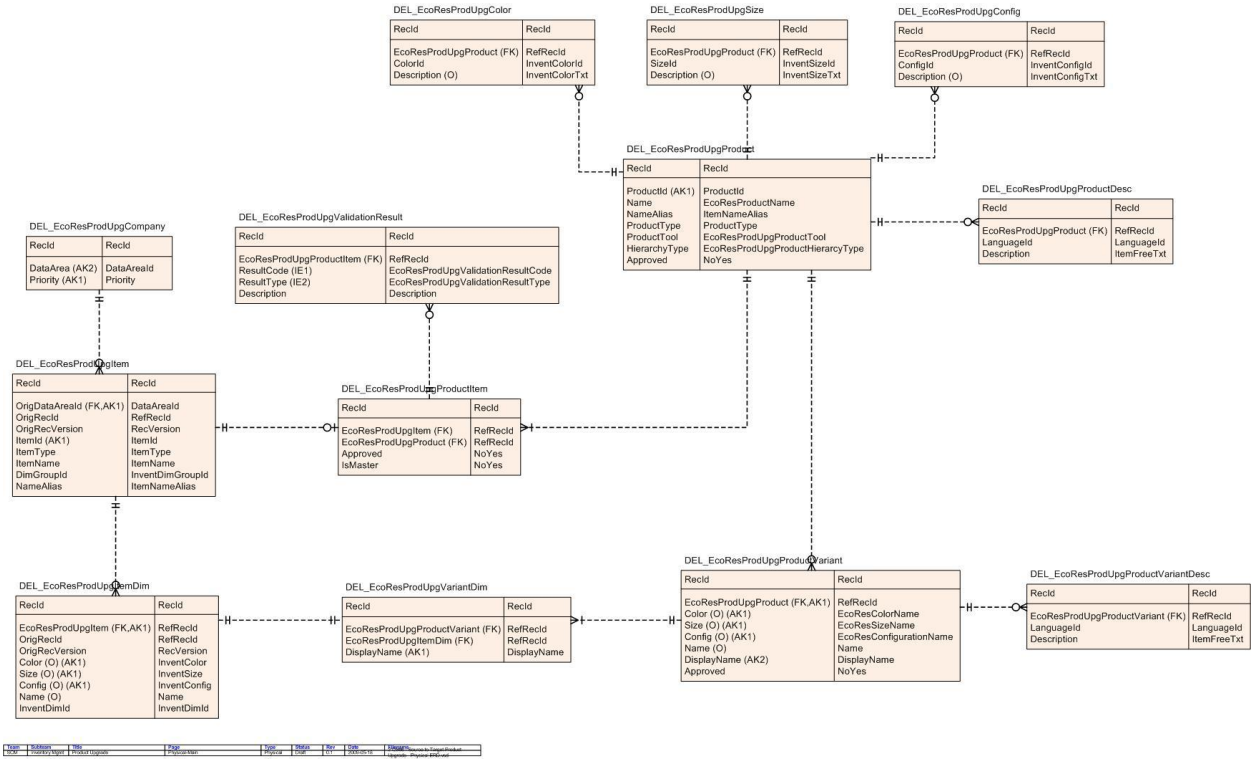
This section describes how to upgrade product and inventory data.

Product Upgrade

In order to upgrade to the new product model, data from several existing, non-shared tables (tables with the property **saveDataPerCompany=Yes**) must be copied to new shared tables. This represents a challenge because it potentially involves merging data from multiple legal entities. In order to achieve this, user interaction is required. The upgrade is performed by using a source-to-target upgrade.

Through several forms on the source system, developers can provide input and validate the resulting product models.

The data model for used for the upgrade of products is illustrated below:



This data model serves two main purposes:

- Providing access to data from multiple companies that is accessible from X++ in order to perform mapping and validation.
- Storing the product data that will be copied to the source environment and be used to create the product on the target environment.

The upgrade sequence consists of three primary steps, each of which will be described in detail.

1. Synchronization of data from the different companies to shared tables.
2. Mapping of items to products.
3. Validation of the mappings and products.

Synchronization

For technical reasons, data from several tables is copied into shared tables. This is done so that work on data across companies can be performed efficiently.

The following schema lists where the data is copied from and to, and which class performs the synchronization.

From table	To table	Class
InventTable	DEL_EcoResProdUpgItem	**EcoResProdUpgSyncItem**
InventDimCombination	DEL_EcoResProdUpdItemDim	**EcoResProdUpgSyncItemDim**
DataArea	Del_EcoResProdUpgCompany	**EcoResProdUpgSyncCompany**

The data in the DEL_ tables is only used on the source system and is not copied over to the target system.

Mapping

Once data is synchronized, product mapping can be performed.

Product mapping is the step where items are mapped to new products. There are three ways to map items to products provided out of the box:

- **Mapping based on itemId**: With this mapping, all items with the same itemId are mapped to one product. The product gets its values from the item that comes from the company with the highest priority (which is determined by the values from the DEL_EcoResProdUpgCompany table).
- **One-to-one mapping**: With this mapping, each item is mapped to one product.
- **Manual mapping**: This mapping is accomplished by manually entering product numbers.

The mapping is handled by the following classes:

- **DEL_EcoResProdUpgProductMapper**: The abstract base class.
- **DEL_EcoResProdUpgItemProductMapper**: Used for item-to-product mapping.
- **DEL_EcoResProdUpgUniqueProductMapper**: Used for one-to-one mapping.
- **DEL_EcoResProdUpgProductMapperSingleItemMapper**: Used for manual mapping.

Mapping results in data being created in several tables, the most important of which are:

- **DEL_EcoResProdUpgradeProduct**: This table holds the data used to create the products on the source system.
- **DEL_EcoResProdUpgradeProductItem**: This table holds the relationship between product and item.

Validation

After the product mapping, a validation of the products must be performed in order to ensure that the data can be upgraded to the Microsoft Dynamics AX 2012 data model. Examples of errors that would prevent upgrade are two items with different dimension setups being mapped to the same product, or two items within the same company being mapped to the same product.

Validation is handled by the following classes:

- **DEL_EcoResProdUpgValidate**: This class produces detailed information about the validation errors and warnings that were found.
- **DEL_EcoResProdUpgValidateFinal**: This class is a lightweight version of the preceding class that, for performance reasons, will return on the first error encountered.

Inventory dimension groups upgrade

The data model of inventory dimension groups has been split into the three separate entities: product, storage, and tracking dimension groups. Mapping existing inventory dimension groups into these new entities requires user interaction because existing inventory dimension groups from different companies can be merged.

The upgrade is performed by using the source-to-target framework. The user can use a form on the source system to merge and prepare the data. This data will be used when performing the actual upgrade on the target system.

The data model for the inventory dimension group upgrade consists of four tables:

Table name	Description
DEL_EcoResDimGroupUpgrade	Contains all inventory dimension groups from all companies and stores references to all three tables below.
DEL_EcoResProductDimGroupUpgrade	Contains IDs and descriptions of product dimension groups.
DEL_EcoResStorageDimGroupUpgrade	Contains IDs and descriptions of storage dimension groups.
DEL_EcoResTrackingDimGroupUpgrade	Contains IDs and descriptions of tracking dimension groups.

The upgrade sequence consists of three main steps, each of which will be described in detail.

1. Synchronization of data from the different companies to DEL_EcoResDimGroupUpgrade.
2. Mapping of inventory dimension groups to product, storage, and tracking dimension groups.
3. Validation of the mappings.

Synchronization

To improve performance, data is synchronized from the InventDim table when the inventory dimension group upgrade form (**DEL_EcoResDimGroupUpgrade**) is initialized. For the implementation of the synchronization, please see the **fillDEL_EcoResProductDimGroupUpgrade** method in the **ReleaseUpdateTransformDB40_EcoResDimG** class.

Mapping

After the synchronization has been performed, the data can be mapped. There are three default mappings. If needed, additional mappings can easily be created. The default mappings are:

- **Mapping dimension groups 1:1**: This mapping uses the **DEL_EcoResDimGroupUniqueMapper** class. It creates one product, storage, and tracking dimension group for each existing inventory dimension group. No data is merged.
- **Mapping dimension group IDs**: This mapping uses the **DEL_EcoResDimGroupIdMapper** class. It creates product, storage, and tracking dimension groups having the same names as the existing inventory dimension groups. If multiple inventory dimension groups (in different companies) have the same name but different setups, validation will fail. If validation fails, the user must manually fix the validation errors.
- **Mapping dimension groups by setup**: This mapping uses the **DEL_EcoResDimGroupSetupMapper** class. It analyzes the inventDimSetups and creates one product, storage, and tracking dimension group per unique setup. This option is useful if you want to heavily merge your dimension groups.

The abstract **DEL_EcoResDimGroupMapper** class is the base class for the default mappers. This class can be extended if there is a need for a custom mapper.

Validation

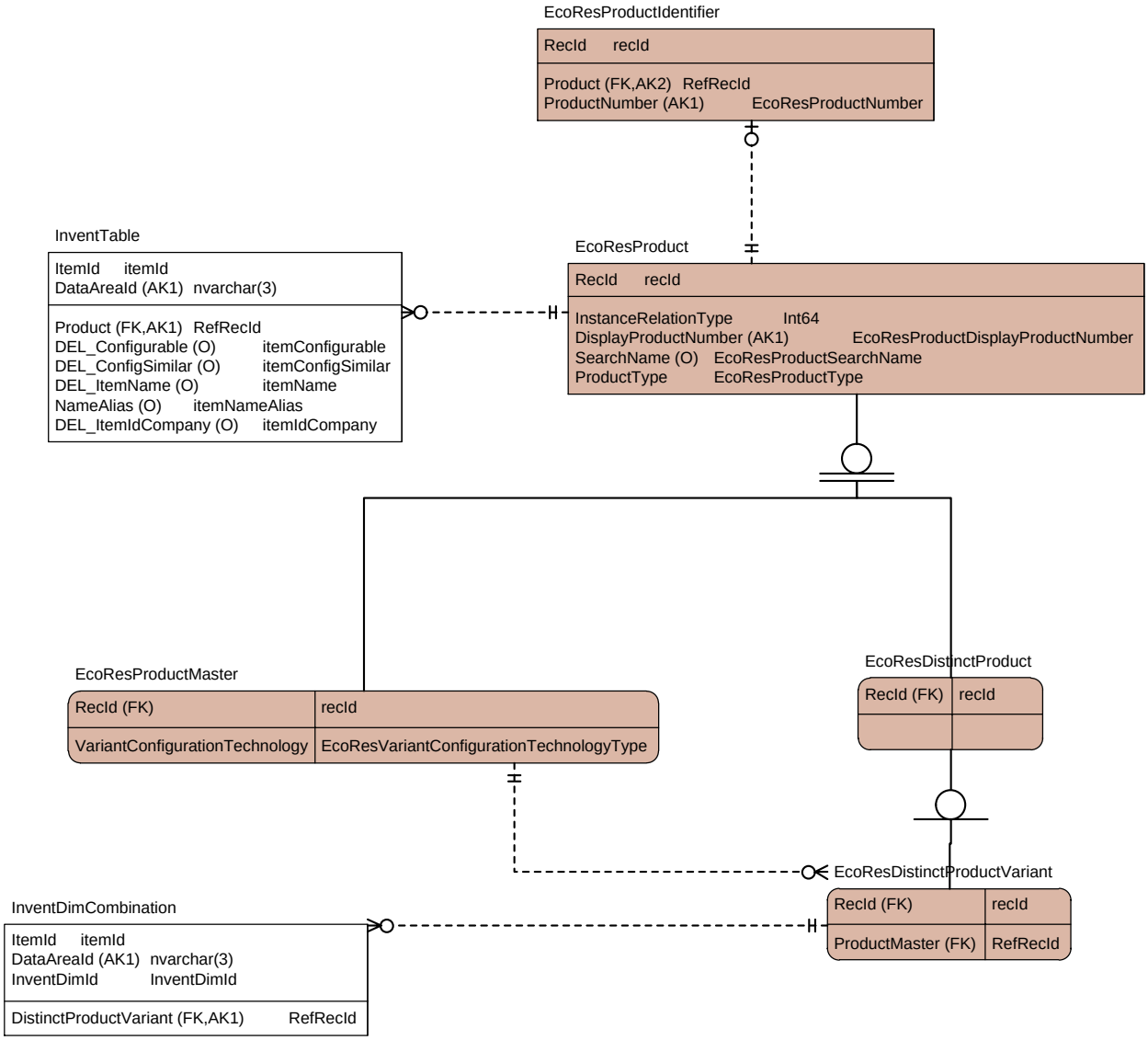
After mapping the data, validation can be run manually by clicking the **Validate** button. Validation also will occur before the data is set to ready for upgrade.

Validation will ensure that product, storage, and tracking dimension groups are mapped to inventory dimension groups that have the same setup.

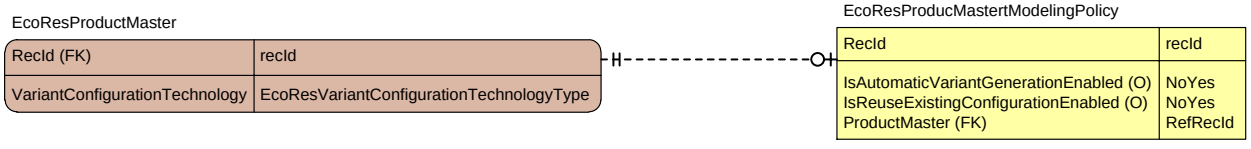
Appendix

Data model diagrams

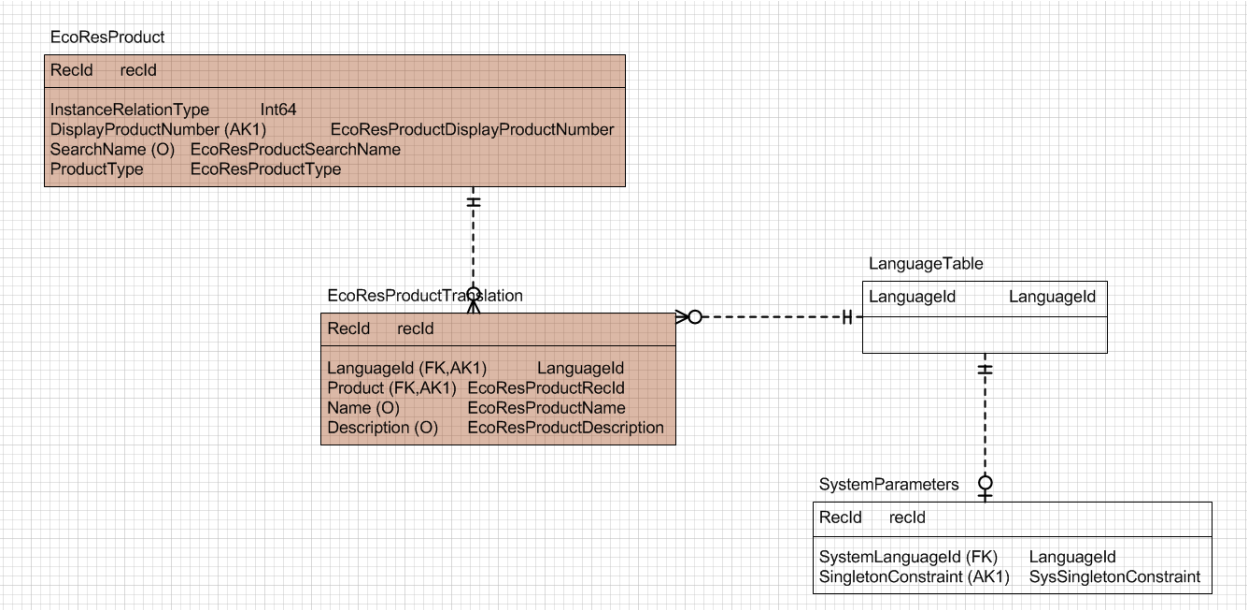
Product shared definition and relations to legal entity tables



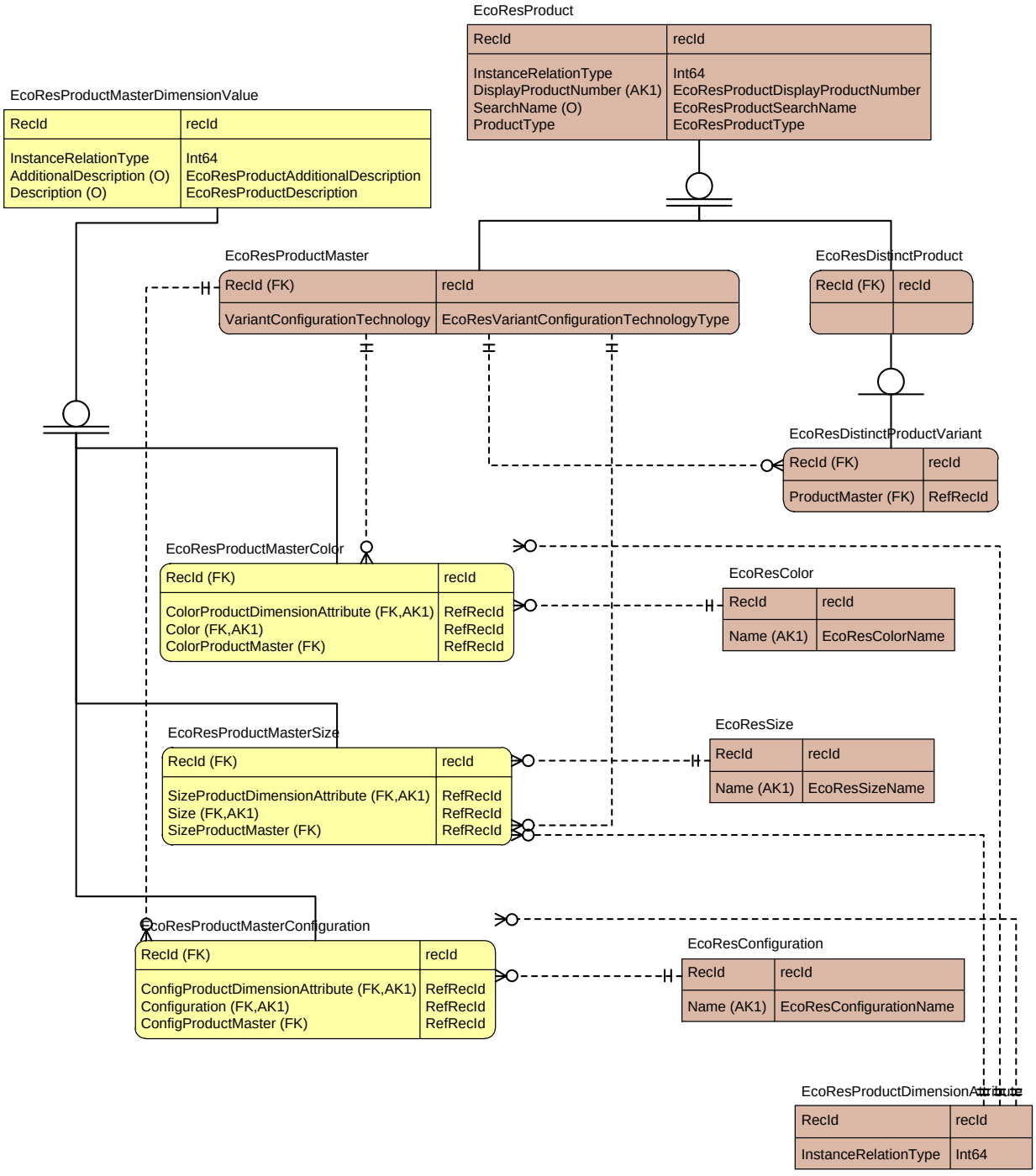
Product master modeling policy



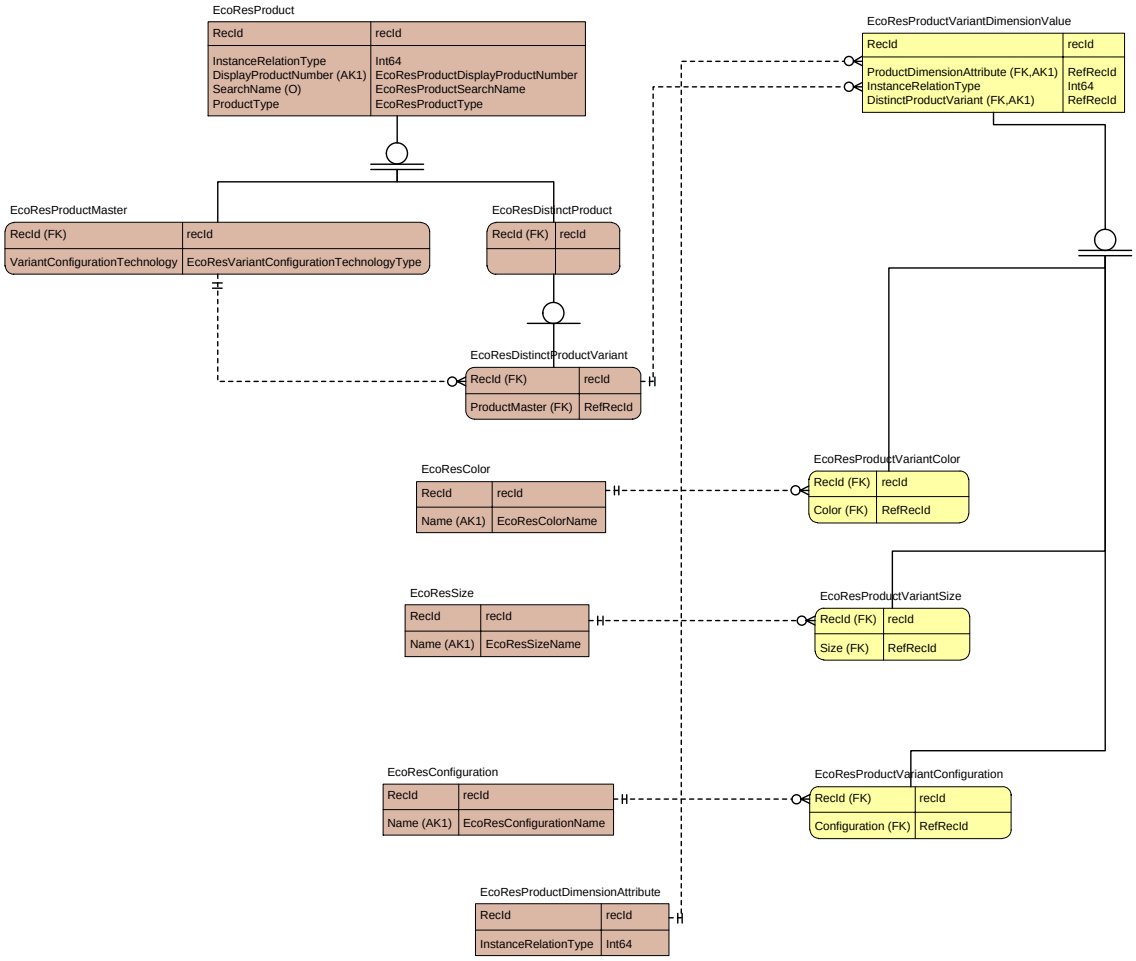
Product localization



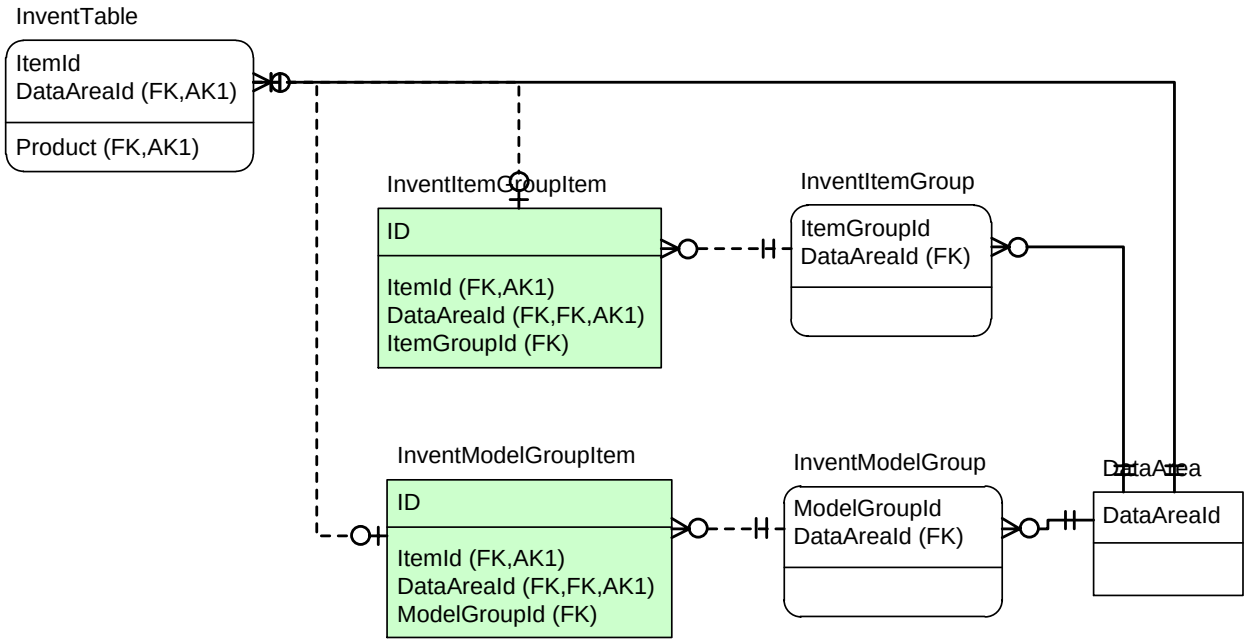
Product master dimensions



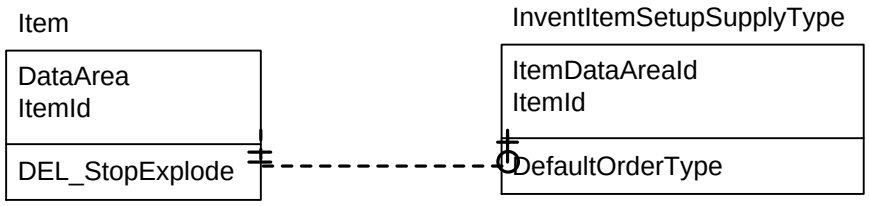
Product variant dimensions



Item group and item model group



Default order type



Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989

Worldwide +1-701-281-6500

www.microsoft.com/dynamics