Microsoft Dynamics® AX 2012

Implementing Budgeting for Microsoft Dynamics AX 2012 Applications

White Paper

This document describes new development patterns in budgeting and their implementation.

http://microsoft.com/dynamics/ax

Date: July 2011

Author: Kim Kroetsch, Senior Development Lead

Send suggestions and comments about this document to adocs@microsoft.com. Please include the title with your feedback.

Microsoft Dynamics

Table of Contents

Overview

In Microsoft Dynamics® AX 2012, budgeting capabilities have been enhanced to provide substantially more functionality than in earlier versions. To support this new functionality, budgeting has been integrated with the ledger, the new accounting distributions framework, the chart of accounts, and the new account and financial dimensions framework. Budgeting has also been integrated with currency and exchange rate types, and integration data patterns have been completely revised.

In Microsoft Dynamics AX 2009 and earlier versions, all budgeting data was stored in the LedgerBudget table. Budget allocations were stored in the same table, and revisions were supported through a relationship with the BudgetRevision table. A record was considered to be always available for processing and reporting unless specifically marked otherwise. The user was required to reference a main account number. There was no way to distinguish budget types or to determine whether budget amounts were for revenue projections or expenditures. There was no concept of workflow for budgets or budget changes.

Microsoft Dynamics AX 2012 introduces the concept of a "budget transaction." The LedgerBudget table has been normalized into a BudgetTransactionHeader table and a BudgetTransactionLine table. The BudgetModel table has been maintained without changes from Microsoft Dynamics AX 2009, but the tables that supported the LedgerBudget table (LedgerBudgetSettlement and BudgetRevision) have been deprecated. Budgeting is supported by the new budget control framework.

This document does not discuss all of the new budgeting functionality. Instead, it focuses on new development patterns and their implementation. Note that these budgeting pattern enhancements do not affect the budget patterns in other modules in any way.

Audience

This document targets developers who are building new applications for Microsoft Dynamics AX 2012 or who are updating their existing application code and data.

Terminology

Microsoft Dynamics AX 2012 terms:

Term	Definition
Budget	A financial plan that controls expenditures for planned activities.
Ledger	The part of an accounting system that is used for classifying the monetary value of economic transactions by using a chart of accounts, a fiscal calendar, and one or more currencies.
Ledger dimension	A classifier created from the combination of financial dimension values listed in a chart of accounts and used to classify the financial consequences of economic activity.
Budget model	A planning structure used to schedule budget fund allocations and expenditures.

Implementing budgeting

This white paper describes the new patterns used in budgeting in Microsoft Dynamics AX 2012 and how to implement them.

The information provided in this paper is intended for use by developers who need to perform the following tasks for budgeting:

- Code upgrade
- Data upgrade

- Microsoft Dynamics AX 2012 service integration

Changes to the data model

Budgeting is supported in Microsoft Dynamics AX 2012 by budget transactions, which are referred to in the application as *budget register entries* or *budget register account entries*. For the physical data model for the new budgeting tables, see the Appendix.

Previous version

In Microsoft Dynamics AX 2009, a record in the LedgerBudget table contained the details of the budget, including references to the budget model, ledger account, financial dimensions, currency, dates, and amounts. A record also held allocation information and a reference to a revision that indicated whether the record had been changed as a result of that revision. A record was considered to be always available for processing and reporting unless specifically marked otherwise.

Microsoft Dynamics AX 2012

In Microsoft Dynamics AX 2012, the LedgerBudget table has been normalized into a header table (the BudgetTransactionHeader table) and a lines table (the BudgetTransactionLine table), with the header table having a one-to-many relationship with its lines table. Budget transaction tables have their **SaveDataPerCompany** table properties set to **No**.

BudgetTransactionHeader table

A BudgetTransactionHeader table contains a reference to the Ledger table (in its PrimaryLedger field) that identifies the legal entity associated with the budget transaction data.

A BudgetTransactionHeader table contains the following values that help categorize the transaction:

- A reference to the BudgetModel table, which holds user-defined reference data.
- A reference to the BudgetTransactionCode table, which provides the transaction code.
- A type value (in the BudgetTransactionType field).

The BudgetTransactionHeader table also contains a status value (in the BudgetTransactionStatus field) that supports "Draft" as a status for the transaction entry in the form, and a process for converting the status to "Completed." A "Completed" status means that the budget transaction can no longer be edited and can be included in reporting.

BudgetTransactionLine table

Microsoft Dynamics AX 2012 continues to support budget allocations by using period allocation keys (as was done in previous versions), but it uses new budget allocation terminology. Budget transaction lines can now also be generated by using recurrence patterns for recurring budget amounts.

In addition, when a budget is transferred from other modules, such as **Project**, **Fixed Asset**, and **Inventory and Warehouse Management**, transaction amounts that have already been transferred can be reversed instead of having to be deleted, as was the case in Microsoft Dynamics AX 2009. This feature allows for an enhanced audit trail of changes to the budget. A reversal is tracked by the BudgetTransactionLineReverse table, which contains references to both the BudgetTransactionLine that is being reversed and the line that holds the reversal information.

A BudgetTransactionLine table contains a ledger dimension reference, which is held in its LedgerDimension field. This reference must always be set to the enumeration value of "Budget" from the **LedgerDimensionType** enumeration. The **LedgerDimensionType** enumeration differentiates the types of ledger dimensions that can be stored in the DimensionAttributeValueCombination table. The **LedgerDimensionBudget** extended data type (EDT) is used to support the differentiation of a budget ledger dimension.

Only financial dimensions that are designated as enabled for budgeting can be included in a budget ledger dimension. This designation is stored in the BudgetPrimaryLedgerDimensionAttribute table. This table holds references to the financial dimensions that are found in the chart of accounts for a ledger and that are enabled for budgeting. This table associates a ledger record with a DimensionAttribute record.

Budget transaction lines store the amount of the adjustment to the budget. This amount is entered in the transaction currency specified on the line. If the budget transaction line has a **BudgetType** enumeration value of "Revenue" in its BudgetType field, the sign (positive or negative) of the amount is adjusted when it is stored in the database. If the budget transaction line has a **BudgetType** enumeration value of "Expense," the sign of the amount is stored as entered.

The line amount is stored in both the transaction currency (in the TransactionCurrencyAmount field) and in the accounting currency of the ledger (in the AccountingCurrencyAmount field). The amount is converted to the accounting currency by using the budget exchange rate type that is specified for the ledger. The **calculateTransAmountToAccountingAmount** method of the **BudgetTransactionManager** class performs this calculation.

Transfers of budgets between modules

Transfers of budgets and forecasts from the **Project**, **Inventory and Warehouse Management,** and **Fixed Asset** modules to the **General Ledger** module are preserved in Microsoft Dynamics AX 2012, as is the transfer of a budget from the **General Ledger** module to the **Cost Accounting** module. However, the opposite action—the transfer of a budget from the **Cost Accounting** module to the **General Ledger** module—has been removed in Microsoft Dynamics AX 2012.

To support the transfers of budgets and forecasts between modules, different patterns of transfer of financial dimension data are needed. This is because only a subset of the dimension attributes specified by the transferred budget might be enabled for budgeting in the **General Ledger** module. These patterns are described in the X++ code patterns section of this document.

Workflow

In Microsoft Dynamics AX 2012, workflow has been implemented for budget transactions. Both header-level and line-level workflows are supported.

Services

In Microsoft Dynamics AX 2012, budget transactions have a service interface implemented in the Application Integration Framework (AIF). The Office Business Application add-in for Microsoft Excel® makes use of services for read/write access to data that the services support. This allows you to use Excel for developing and adjusting budgets.

The following service operations are supported for budget transactions:

- Read
  - Input parameter: AifEntityKeyList
  - Return value: BudgetTransaction
- Find
  - Input parameter: AifQueryCriteria
  - Return value: BudgetTransaction
- FindKeys
  - Input parameter: AifQueryCriteria
  - Return value: AifEntityKeyList
- Delete

    - Input parameter: AifEntityKeyList
    - Return value: void
- Create
    - Input parameter: BudgetTransaction
    - Return value: AifEntityKeyList
- Update
    - Input parameter: AifEntityKeyList, BudgetTransaction
    - Return value: void

Implementing the new data pattern

The budget ledger dimension is a new pattern in Microsoft Dynamics AX 2012.

Budget ledger dimension

A budget ledger dimension contains the account structure and financial dimension values. A foreign key representing a budget ledger dimension is a 64-bit integer field that contains the data from the corresponding RecId field of the DimensionAttributeValueCombination (or LedgerDimension) table. Foreign key fields for ledger accounts are named LedgerDimension because that is the alias used for the DimensionAttributeValueCombination table.

Data model and extended data type

New EDT	LedgerDimensionBudget
New field	LedgerDimension The LedgerDimension field contains a foreign key to the DimensionAttributeValueCombination table.

Microsoft Dynamics AX form control

The Budget Ledger Dimension control represents a combination of the Segmented Entry control and the **BudgetLedgerDimensionController** class. The Segmented Entry control is a general-purpose control that has been introduced in Microsoft Dynamics AX 2012. The **BudgetLedgerDimensionController** class handles events raised by the Segmented Entry control.

This combination allows the control to handle the entry and display of budget dimension values in Microsoft Dynamics AX forms. The following sections show you how to implement this control on a Microsoft Dynamics AX form.

Changes needed on the form

In most scenarios, the changes needed on a Microsoft Dynamics AX form are as follows:

1. Verify that the table holding the foreign key to the DimensionAttributeValueCombination table is a data source on the form.
2. Drag the **LedgerDimension** field from the data source to the desired location on the form design. This creates a Segmented Entry control with the appropriate **DataSource** and **ReferenceField** property values. Alternatively, you can add a Segmented Entry control to the design and manually set the **DataSource** and **ReferenceField** properties.

3. Override the class declaration and the **init** method on the form. If these methods already exist, just add the code to the methods.

```
public class FormRun extends ObjectRun
{
    BudgetLedgerDimensionController budgetLedgerDimensionController;
}

public void init()
{
    super();
    budgetLedgerDimensionController =
BudgetLedgerDimensionController::construct({BackingDataSource_ds},
fieldstr({BackingTable}, LedgerDimension));
}
```

4. Override the following methods on the Segmented Entry control instance in the form design:

```
public void jumpRef()
{
    budgetLedgerDimensionController.jumpRef();
}

public boolean validate()
{
    boolean isValid;

    isValid = super();
    isValid = budgetLedgerDimensionController.validate() && isValid;

    return isValid;
}

public void segmentValueChanged(SegmentValueChangedEventArgs _e)
{
    super(_e);
    budgetLedgerDimensionController.segmentValueChanged(_e);
}

public void loadSegments()
{
    super();
    budgetLedgerDimensionController.parmControl(this);
    budgetLedgerDimensionController.parmDimensionAccountStorageUsage(DimensionAccountStora
geUsage::Transactional);
    budgetLedgerDimensionController.parmDate({BackingTable}.Date);

    budgetLedgerDimensionController.loadSegments();
}

public void loadAutoCompleteData(LoadAutoCompleteDataEventArgs _e)
{
    super(_e);
    budgetLedgerDimensionController.loadAutoCompleteData(_e);
}
```

5. Override the **resolveReference** method on the data source field that backs the Segmented Entry control:

```
public Common resolveReference(FormReferenceControl _formReferenceControl)
{
    return budgetLedgerDimensionController.resolveReference();
}
```

6. Set the *parmAccountStructure* parameter to the RecId of an account structure in the system. In the most common scenario, the user can choose from multiple account structures. In this case, create an edit method that sets the *parmAccountStructure* parameter to the account structure that is selected on the form.

   If the Segmented Entry control is bound to a specific account structure, the RecId of that account structure can be set in the **loadSegments** method of the Segmented Entry control.

```
budgetLedgerDimensionController.parmAccountStructure({Account Structure RecId});
```

Control options

Several parameters affect the validation, lookup, and storage of a budget ledger dimension as performed by the Segmented Entry control.

- *DataAreaId*: Specifies the legal entity associated with the control being managed. This is used for validation and lookup to restrict valid values. The default value is the current legal entity (this is the most common scenario). If this parameter is provided by a field that the user can manipulate on the form, we recommend that you call the corresponding **parm** method from the **modified** method of the control for the DataAreaId field.
- *Date*: Specifies the date of the transaction associated with the control being managed. This parameter is used for validation. The default value is empty and no validation is done against the date.
- *DimensionAccountStorageUsage*: Specifies how the control being managed is used. This parameter is used for validation and for saving combinations. Use the **DimensionAccountStorageUsage** enumeration and pass in the "Transactional" value.

These parameters should be specified in the **loadSegments** method of the Segmented Entry control. They are called every time that the control receives focus. Always declaring the parameters in one method ensures that a developer can easily verify whether all parameters are being set properly.

Considerations for specific scenarios

When working with the budget ledger dimension control, there are additional considerations for the following scenarios:

- When multiple fields have an account number control, each field should have its own **BudgetLedgerDimensionController** instance.
- Each instance must have a unique name that should include a description of how the backing field is used. You need to add code that is similar to the example shown in the Changes needed on the form section for each controller instance.
- When one field needs to be edited in multiple places on the same form, create a single **BudgetLedgerDimensionController** instance for the field.
- When two controls share the same controller, the **loadSegments** method for each control should always contain a call to the **parmControl(this)** method before the call to the **loadSegments** method for the controller.

X++ code patterns

The **BudgetTransactionManager** class is a helper class that supports budget transactions. Some of the key methods in this class are described below.

Create budget ledger dimensions

The following methods are used to create budget ledger dimensions:

- **saveLedgerDimensionBudget**

  Creates a ledger dimension of type budget by using the dimension attribute values from another ledger dimension reference. Uses only dimension attributes that are enabled for budget to create the budget ledger dimension.

- **getLedgerDimensionBudget**

  Combines a default account reference for a main account and a default dimension reference into one budget ledger dimension. Uses only dimension attributes that are enabled for a budget to create the budget ledger dimension.

- **mergeDefaultDimWithLedgerDim**

  Combines a ledger dimension with a default dimension into one budget ledger dimension. Uses only dimension attributes that are enabled for the budget to create the budget ledger dimension.

Convert a transaction amount to the accounting currency

The following method is used to convert currency amounts:

- **calculateTransAmountToAccountingAmount**

  Converts an amount to the accounting currency of the ledger using the budget exchange rate type specified in the ledger.

Validate budget transactions

All validation logic for budget transactions is performed by methods in the **BudgetTransactionManager** class to allow transactions to be shared between the rich client and the service entry points. The following methods are used for validation:

- **validateAccountStructure**
- **validateBudgetModel**
- **validateCurrency**
- **validateDimensionFocus**
- **validateLedgerDimension**
- **validateTransactionDate**

Data upgrade

This section describes the conversion of data that occurs during the upgrade for budgeting from the source environment (Microsoft Dynamics AX 4.0 or Microsoft Dynamics AX 2009) to the target environment (Microsoft Dynamics AX 2012).

LedgerBudget table normalization

The LedgerBudget table in Microsoft Dynamics AX 4.0 or Microsoft Dynamics AX 2009 must be normalized to the BudgetTransactionHeader and BudgetTransactionLine tables in the new data model.

During the upgrade process, a BudgetTransactionHeader record is generated for each LedgerBudget record from the previous version.

If a LedgerBudget record has been "allocated"—that is, broken down into finer amounts—no BudgetTransactionLine record will be created for it. A BudgetTransactionLine record will only be created for each expanded LedgerBudget record. The expanded LedgerBudget records will reference the allocated LedgerBudget record.

In other words, the creation of a BudgetTransactionLine record depends on the values in the ExpandId and AllocateMethod fields on the LedgerBudget record. If a LedgerBudget record was allocated, the ExpandId field is set to **zero** and the AllocateMethod field is set to a value other than "None". In this case, no transaction line is created. Otherwise, a budget transaction line is created.

LedgerBudget table conversions

The fields from the LedgerBudget table in Microsoft Dynamics AX 4.0 and Microsoft Dynamics AX 2009 are mapped to BudgetTransactionHeader and BudgetTransactionLine table fields in Microsoft Dynamics AX 2012 as shown in the following table.

LedgerBudget table field	Microsoft Dynamics AX 2012 table	Microsoft Dynamics AX 2012 field	Value mapping
RecId	BudgetTransactionHeader	TransactionNumber	
Active	BudgetTransactionHeader	TransactionStatus	Completed where Active = Yes Draft where Active = No
BudgetModel	BudgetTransactionHeader	BudgetModel	
BudgetModel	BudgetTransactionHeader	BudgetModelSubId	
BudgetModel	BudgetTransactionHeader	BudgetModelType	BudgetModel.Type based on join to BudgetModel record
BudgetModel	BudgetTransactionHeader	BudgetModelDataAreaId	BudgetModel.DataAreaId based on join to BudgetModel record
	BudgetTransactionHeader	BudgetTransactionType	Original budget by default
RevisionDate	BudgetTransactionHeader	BudgetTransactionType	Amendment where RevisionDate has a value
AssetId	BudgetTransactionHeader	BudgetTransactionType	Fixed asset where AssetId has a value
ProjTransId	BudgetTransactionHeader	BudgetTransactionType	Project where ProjTransId has a value
InventTableId	BudgetTransactionHeader	BudgetTransactionType	Demand forecast where InventTableId = ForecastSales Supply forecast where InventTableId = ForecastPurch
StartDate	BudgetTransactionHeader	Date	
	BudgetTransactionHeader	BudgetTransactionCode	Not set during upgrade
AssetId	BudgetTransactionLine	AssetBudget	AssetBudget.RecId based on join to AssetBudget record

ProjTransId	BudgetTransactionLine	ProjTransBudgetTransId	ProjTransBudget.TransId based on join to ProjTransBudget record
AccountNum	BudgetTransactionLine	BudgetType	Revenue or Expense based on join to LedgerTable referencing AccountPlType and DebCredProposal values
StartDate	BudgetTransactionLine	Date	
AccountNum + Dimension	BudgetTransactionLine	LedgerDimension	DimensionAttributeValueCombination.RecId based on AccountNum and Dimension field values
CurrencyCode	BudgetTransactionLine	TransactionCurrency	
Qty	BudgetTransactionLine	Quantity	
Price	BudgetTransactionLine	Price	
Amount	BudgetTransactionLine	TransactionCurrencyAmount	
AmountMST (only in AX 2009)	BudgetTransactionLine	AccountingCurrencyAmount	
Cov	BudgetTransactionLine	IncludeInCashFlowForecast	
TaxGroup	BudgetTransactionLine	TaxGroup	
Comment	BudgetTransactionLine	Comment	

Additional data revisions

The following data revisions also occur during the upgrade to Microsoft Dynamics AX 2012.

Normalized table

The following table has been normalized.

Microsoft Dynamics AX 4.0/2009 table	Microsoft Dynamics AX 4.0/2009 field	Microsoft Dynamics AX 2012 table	Microsoft Dynamics AX 2012 field
LedgerParameters	BudgetSettle	BudgetParameters	CashFlowForecastPeriodAllocationKey

Deleted tables

The following tables have been deleted:

- LedgerBudSettlement
- BudgetRevision

Deleted fields

The following fields have been deleted:

- BudgetModel table: TrackRevisions field
- LedgerAllocation table: Freq field; FreqCode field; StartDate field; Stop field
- LedgerParameters table: BudgetCheck field

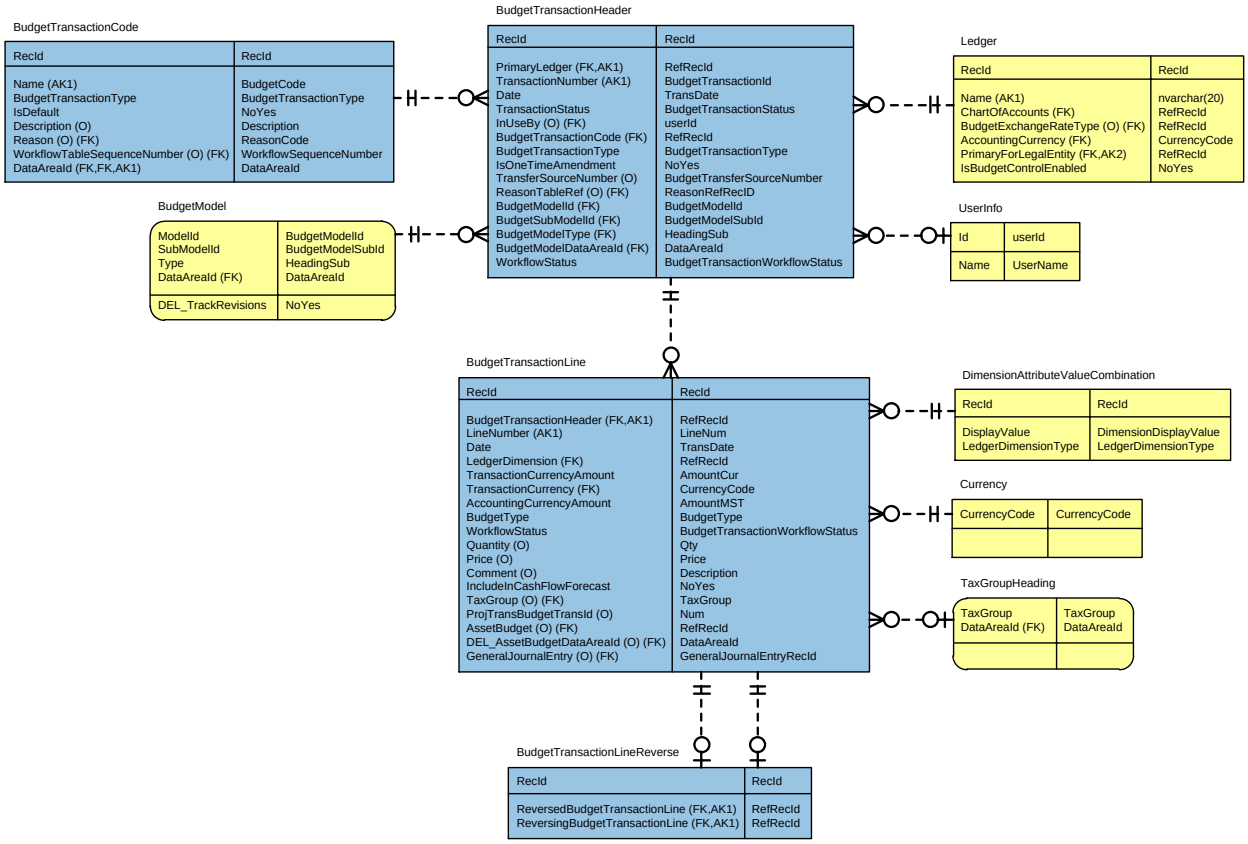
Data cleanup

All records that contain references to the ledger budget are deleted from the following tables as a result of the upgrade process. After the upgrade has been completed, the user can run the relevant processes to regenerate data in these tables.

- LedgerAllocation
- LedgerConsolidateHistRef
- LedgerAccountCov

Appendix

Budget data tables



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