POSTING

Microsoft Dynamics™ AX

Posting to the Ledger

June 2008

Microsoft Dynamics

Table of Contents

Introduction **3**

Ledger structure **3**

High-level processes **3**

Using the APIs **4**

General Facts **4**

Functionality provided 4

Object containment **5**

API definitions **6**

LedgerVoucherGroup::Construct 6
LedgerVoucher::newLedgerPost 6
LedgerVoucherObject::newVoucher 6
LedgerVoucherGroup.addLedgerVoucher 7
LedgerVoucher.addVoucher 7
LedgerVoucherTransObject::newCreateTrans 7
LedgerVoucher.addTrans 7
LedgerVoucher.end 7

Ledger Journal API to LedgerVoucher **8**

Non Ledger Journal API to LedgerVoucher **9**

Multiple entry non-ledger journal API to LedgerVoucher **10**

Introduction

Ledger is the ending point for all financial transactions. Posting to the ledger is the most critical API in any ERP system. Failure to use the APIs properly will lead to inaccurate financial statements. Every subsystem eventually needs to get their base transaction data into the ledger. This document discusses the functionality and APIs for posting to the ledger.

Ledger structure

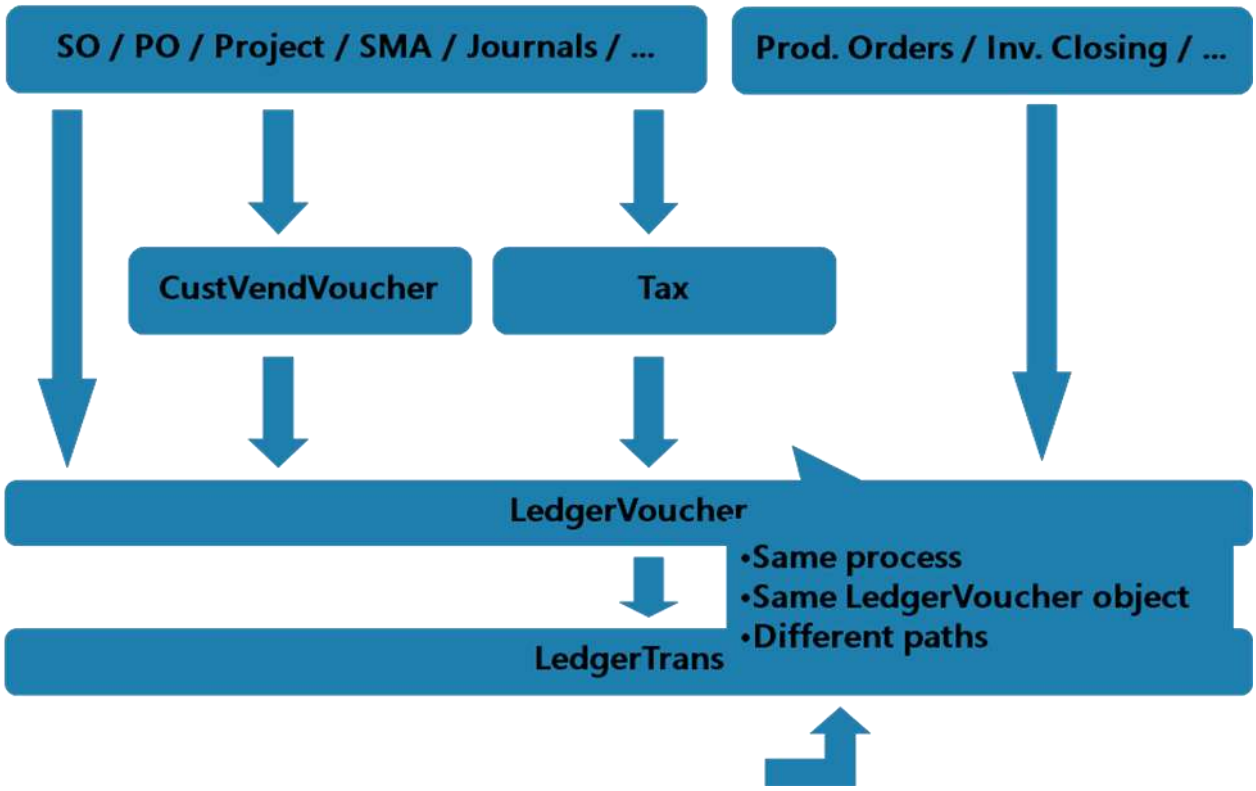
The Microsoft Dynamics AX ledger is defined in a manner very similar to manual accounting systems. The ledger contains all adjustment amounts, dates, and related information for all ledger accounts. Any transaction that originates in a subledger, such as a customer, vendor, bank, or fixed asset transaction, eventually needs to adjust ledger accounts so that the transaction amounts can be reported on financial statements.

The ledger entries are grouped by journal number and further grouped by voucher number. The journal number is a unique number that identifies a group of ledger adjustments generated through one posting. The adjustments within a journal number can be subgrouped by voucher number. The voucher number could represent a single transaction or a group of related transactions, depending on how the customer has set up their journals. While journal numbers must be unique, voucher numbers may permit duplicates if set up to do so. The adjustments must balance against each other on both a per voucher and journal number basis.

High-level processes

Ledger posting is never initiated directly by the end user. Instead, it is run by processes initiated by the user. When the user posts a purchase order invoice or performs a task, such as inventory closing, the ledger posting APIs are called to generate the required transaction in General ledger.

Using the APIs



General facts

- The primary classes are LedgerVoucherGroup, LedgerVoucher, LedgerVoucherObject and LedgerVoucherTransObject.
- The public API for ledger posting is the LedgerVoucher class.
- All submodule transaction postings into G/L (ledgertrans) are required to use the API.
- All postings that spawn allocations, accruals, or reversals are supported.
- The ability to post intercompany transactions is provided through the use ledger voucher groups.
- Handles 1..* vouchers in 1 run

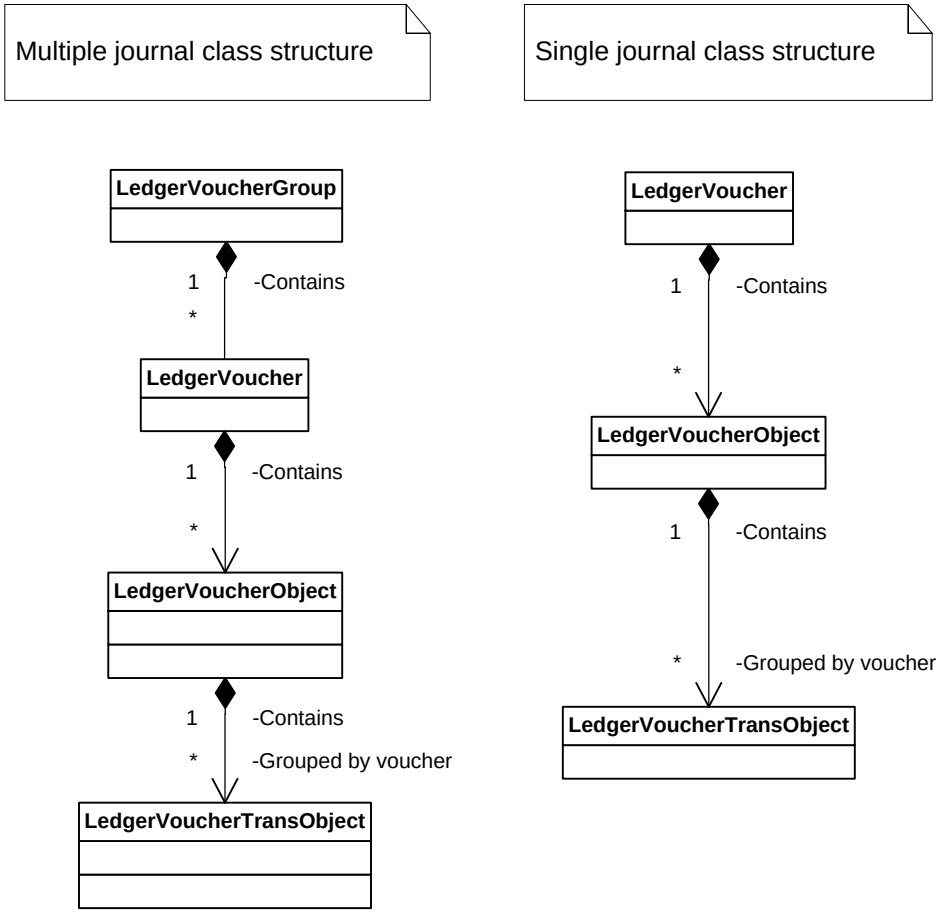
Functionality provided

- User credential validation
- Voucher number validation
- Ledger period validation
- Account/Dimensions restriction validation
- Voucher balance validation
- Secondary amounts using stored exchange rates are calculated
- Transactions when posting in summary are combined

- TransactionLog (Audit trail) records are created.

Object containment

The premise of LedgerVoucher is that an instance of a class contains instances of other classes in a hierarchical manner. This produces a single instance of LedgerVoucher that contains all the vouchers and transactions to post on a per company, journal entry basis. LedgerVoucherGroup provides the means to group LedgerVoucher objects into a single autonomous posting. This supports posting to multiple companies or journal entries in a single transaction.



- The LedgerVoucherGroup is an optional class for use when posting transactions for multiple companies or journal entries. When the need is to post transactions in multiple companies, a LedgerVoucherGroup object is instantiated.
- A LedgerVoucher object is created for each combination of company and journal entry.
- If multiple instances of LedgerVoucher are required due to multiple companies or journal entries, each instance is added to the LedgerVoucherGroup.
- An instance of LedgerVoucherobject is created for each voucher to post and is added to the LedgerVoucher instance. The transactions are grouped by voucher number for each set of transactions that balance out as a single posting<c>. Most postings only require a single voucher.
- A LedgerVoucherTransObject is instantiated for each transaction and is added to a LedgerVoucherobject instance contained in the LedgerVoucher instance.

API definitions

The APIs are listed in the order they would be invoked. The LedgerVoucherGroup methods are the only used when posting more than on journal entry.

LedgerVoucherGroup::Construct

This static method is called to instantiate an instance of the class. Only required when posting a group of journal entries in one atomic posting.

LedgerVoucher::newLedgerPost

This static method is called to instantiate an instance of the class. This is always required since it represents a journal entry and posting is done by journal entry.

Parameter	Type	Description
_detailSummary	DetailSummary Enum:	Post the ledger transaction individually (Detail) or summarized (Summary).
_sysModule	SysModule Enum	The module the transactions are originating from.
_voucherSeriesCode	String 8	The <c>NumberSequenceTable</c> id to use when creating a voucher number.
_transactionLogType	TransactionLogType Enum	The type of transaction to specify when creating transaction log entries; optional.
_transactionLogTxt	String 25	The transaction text to appear on each posted transaction; optional.
_approveJournal	boolean	Set to true if the voucher needs to support approval requirements of not checking for duplicate vouchers or removal of records from the invoice register pool; optional.
_posting	boolean	Set to false if needing to support inventory requirement of being able to post without a posting type; optional.

LedgerVoucherObject::newVoucher

The static method creates a new ledgerVoucherObject representing a new voucher. Depending on the transaction type a number of parm methods may be called to set voucher values.

Parameter	Type	Description
_voucher	Voucher	The voucher number to identify and define the object.
_transDate	Transdate	The transaction date for the voucher; optional.
_sysModule	SysModule Enum	The module the voucher is originating from; optional.
_ledgerTransType	LedgerTransType Enum	The transaction type contained in the voucher; optional.
_correction	NoYes Enum	Set to Yes if this voucher is correcting a related voucher; optional.
_operationsTax	Operationtax Enum	The ledger type the voucher is posting into if it is not the current ledger; optional.

_documentNum	String 20	The document number of the transaction if the value is to be brought into the ledger; optional.
_documentDate	DocumentDate	The document date of the transaction if the value is to be brought into the ledger; optional.
_tmpVoucherMap	Map	Used when all voucher numbers are to be continuous. The map contains all the temporary voucher numbers that will be replaced; optional.
_acknowledgementDate	AcknowledgementDate	The date to acknowledge by when the transaction requires an acknowledgement. Only a requirement in some countries; optional.

LedgerVoucherGroup.addLedgerVoucher

The method adds the new LedgerVoucher to the instance of LedgerVoucherGroup

LedgerVoucher.addVoucher

The method adds the new LedgerVoucherObject to the instance of LedgerVoucher

LedgerVoucherTransObject::newCreateTrans

The static method creates a new LedgerVoucherTransObject representing a new transaction.

LedgerVoucher.addTrans

The method adds the new LedgerVoucherTransObject to the instance of LedgerVoucherObject

LedgerVoucher.end

The method launches the posting of the vouchers.

Ledger journal API to LedgerVoucher

Any module that implements a new journal will extend journal classes and forms required for the new journal type. Extending the classes should provide the means to generate LedgerJournalTable and LedgerJournalTrans records. Posting of the new journal type should be handled by the base classes. The following is an example of the APIs used to post a journal.

Example: LedgerJournalCheckPost.postJournal

```
ledgerVoucher = LedgerVoucher::newLedgerPost();

if a ledgerJournalTable record was read
    validate the ledgerJournalTable record

for each ledgerJournalTrans

        if a new voucher is needed
            LVO = LedgerVoucherObject::newVoucher();
            ledgerVoucher.addVoucher(LVO, …);

        for each account and amount to post on the ledgerJournalTrans
            // the method creates the necessary ledgerTrans records to post
             ok = this.postTrans();

// post the vouchers after all ledgerJournalTrans records have been read
ledgerVoucher.end();
```

Non-ledger journal API to LedgerVoucher

All sub module transactions that post using transaction data from tables other than LedgerJournalTable and LedgerJournalTrans use these APIs to post.

Example: InventPostPhysicalPeriodic.postledgerTrans

```
ledgerVoucher = LedgerVoucher::newLedgerPost();

transactionTxt      = new TransactionTxt();
transactionTxt.setType();
transactionTxt.setVoucher();

// create a new voucher for posting the items together
ledgerVoucherObject = LedgerVoucherObject::newVoucher();

// assign the text to the voucher object and add to the voucher
ledgerVoucherObject.lastTransTxt(transactionTxt.txt());
ledgerVoucher.addVoucher(ledgerVoucherObject);

// if there is a header record to post loop through the lines and add a transaction
// to the voucher object for each record.

if (inventPostPhysicalTable)
{
    For each inventPostPhysicalTrans record to post
    {
        // use ledgerVoucher.findLedgerVoucherObject() to retrieve the last voucher object
        // added to the ledgerVoucher. This insures the correct voucher object is used when
        // creating the LedgerVoucherTransObject
        LVTO = LedgerVoucherTransObject::newCreateTrans(
                                    ledgerVoucher.findLedgerVoucherObject(), …);

        ledgerVoucher.addTrans(LVTO);
    }
}

// this completes the building of the ledgerVoucher.
// the call to end() will initiate the posting of the ledgerVoucher.
ledgerVoucher.end();
```

Multiple entry non-ledger journal API to LedgerVoucher

All sub modules using transaction data from tables other than LedgerJournalTable and LedgerJournalTrans use these APIs to post when multiple journal entries are required.

Example: TaxWithhold.posttaxWithhold

```
ledgerVoucherGroup = LedgerVoucherGroup::construct();

For each record (<multiple posting table>)
      {
          // if intercompany the proper company context is required to
          // access values required during voucher creation
          changecompany(<company of transaction>)
          {
              For each record (<table containing records for the transaction>
              {
                  if (<The criteria for a separate journal entry is met>)
                  {

                  // get the Number Sequence specified to generate a voucher number.
                  numberSeqRef = <where number sequence is specified>
                  numberSeq = NumberSeq::newGetVoucher(numberSeqRef);

                  // This is the journal entry for the transaction for the current company
                  LedgerVoucher = LedgerVoucher::newLedgerPost();


                  // add the voucher to the group
                  ledgerVoucherGroup.addLedgerVoucher(LedgerVoucher);
              }

              LVO = LedgerVoucherObject::newVoucher();

              // get the next voucher number and add it to the voucher
              LedgerVoucher.addVoucher(LVO, …);

              taxWithholdTransLedgerVoucher.findLedgerVoucherObject();

              // use ledgerVoucher.findLedgerVoucherObject() to retrieve the last voucher
              // object added to the ledgerVoucher. This insures the correct voucher object
              // is used when creating the LedgerVoucherTransObjec
              LVTO = LedgerVoucherTransObject::newCreateTrans(
                        ledgerVoucher.findLedgerVoucherObject(), …);

              // add the transaction to the voucher
              LedgerVoucher.addTrans(LVTO);

          }
      }
}

// this completes the building of the ledgerVoucher.
// the call to end() will initiate the posting of the ledgerVoucherGroup.
ledgerVoucherGroup.end();
```

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