

### 3.0 File Overview

The Simply Accounting application stores data in fifty-two different files, plus four additional files for each general ledger that uses bank reconciliation. Forty of those files will be empty in a typical file set.

Six files are of primary significance: the ASC, ASJ, IT0, IT2, IT3, and IT4.

#### 3.1 ASC File

The ASC file is the most complex and the most important file that the Simply Accounting application uses. It can be internally divided into eight parts:

- *SN\_REC* Header (application defaults)
- General Ledger Accounts (*GYN* records)
- Payable Ledger Accounts (*VYD* and *VYN* records)
- Receivable Ledger Accounts (*VYD* and *VYN* records)
- Payroll Ledger Accounts (*MYN* or *USMYN* records)
- Inventory Ledger Accounts (*INV* records)
- Job Cost Ledger Accounts (*JYN* records)
- Journal Entry Indices (*JIDXREC* records)

The *SN\_REC* describes the file set. It contains the format version, integration account numbers, taxation rate defaults, user preferences, and entity information.

Integration accounts are general ledger accounts that have special meaning. For example, the general ledger that accumulates provincial sales tax will be an integration account because it has special accounting significance. The *SN\_REC.INTEG\_REC* struct is required because integration accounts do not have consistent account numbers across different file sets.

Integration accounts of importance when filing an invoice are *SN\_REC.INTEG\_REC.tAcNadvRec* (“Accounts Payable”), *SN\_REC.INTEG\_REC.tAcNpstPay* (“PST Payable”), and *SN\_REC.INTEG\_REC.tAcNgstRec1* (“GST Receivable, Rate 1”).

General ledger accounts are “The Books”. They can represent bank accounts, discounts, project allocations, budgets, or many other things that require accumulation tracking.

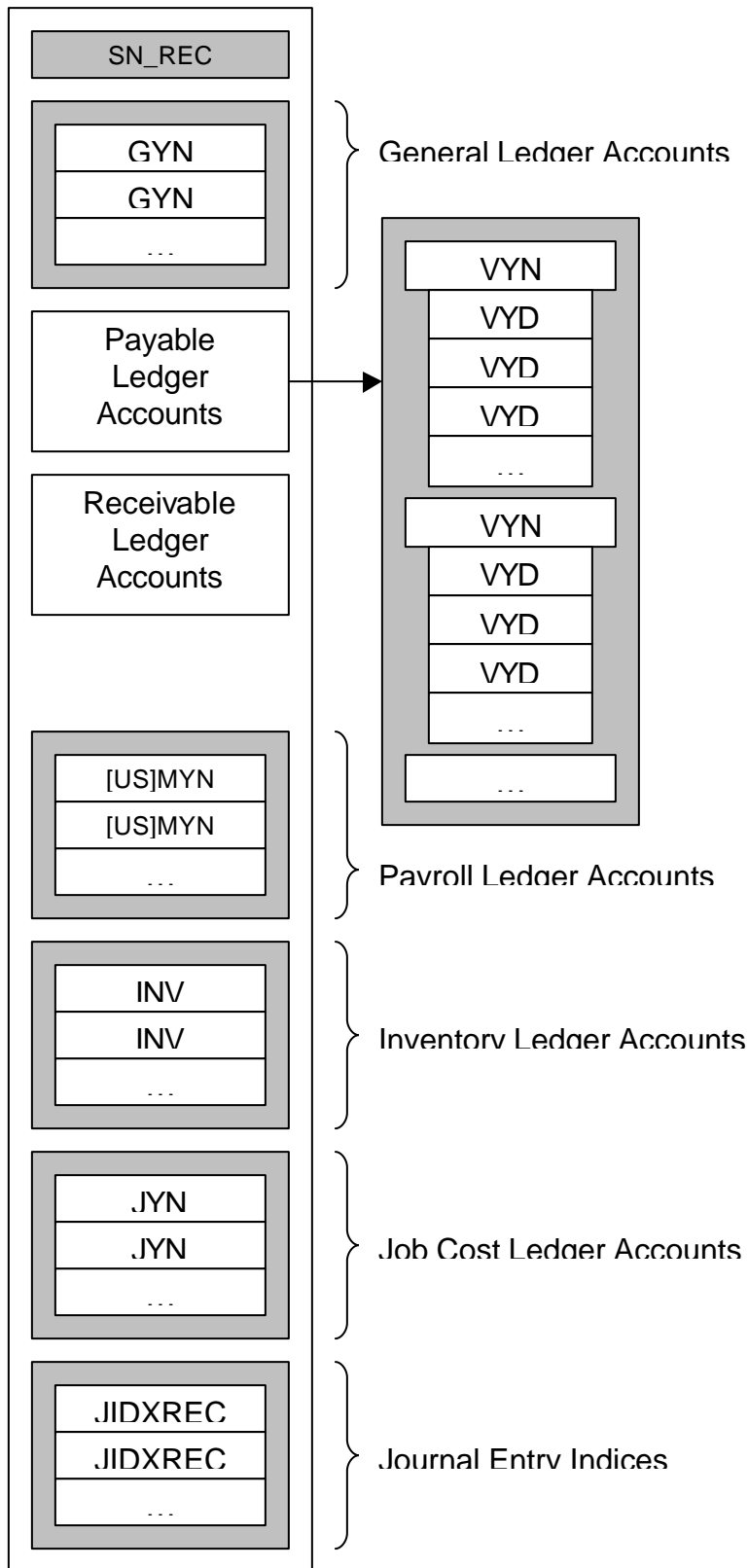
Payable ledger accounts contain information about customers. Every tracked customer is represented by one *VYN* record, which contains information like a shipping address, contact specifics, and sales summaries. The customer *VYN* is followed by a series of *VYD* records that contain summary information about sales invoices filed against that customer.

Receivable ledger accounts are identical to payable ledger accounts, except that they represent vendors and purchase invoices. Payable records can be found at offset “sizeof(SN\_REC) + *SN\_REC.size[GEN]*”. Receivable records can be found at offset “sizeof(SN\_REC) + *SN\_REC.size[GEN]* + *SN\_REC.size[CEN]*”.

Inventory ledger accounts contain the summary statistics of tracked goods or services. There is one inventory ledger account for each tracked part number. Untracked inventory items have no inventory ledger account, so the *SN\_REC* record maintains the required pointers instead of an *INV* record. (These pointers are described later in this document.) Inventory ledger accounts are updated during the invoice insertion process.

Payroll ledger accounts and job cost ledger accounts were not examined because they are not involved with invoice filing. Nevertheless, they must be correctly read from and written to the ASC file during updates.

The journal entry indices are an array of pointers into the ASJ file that do not have an analog in the Simply Accounting application interface.



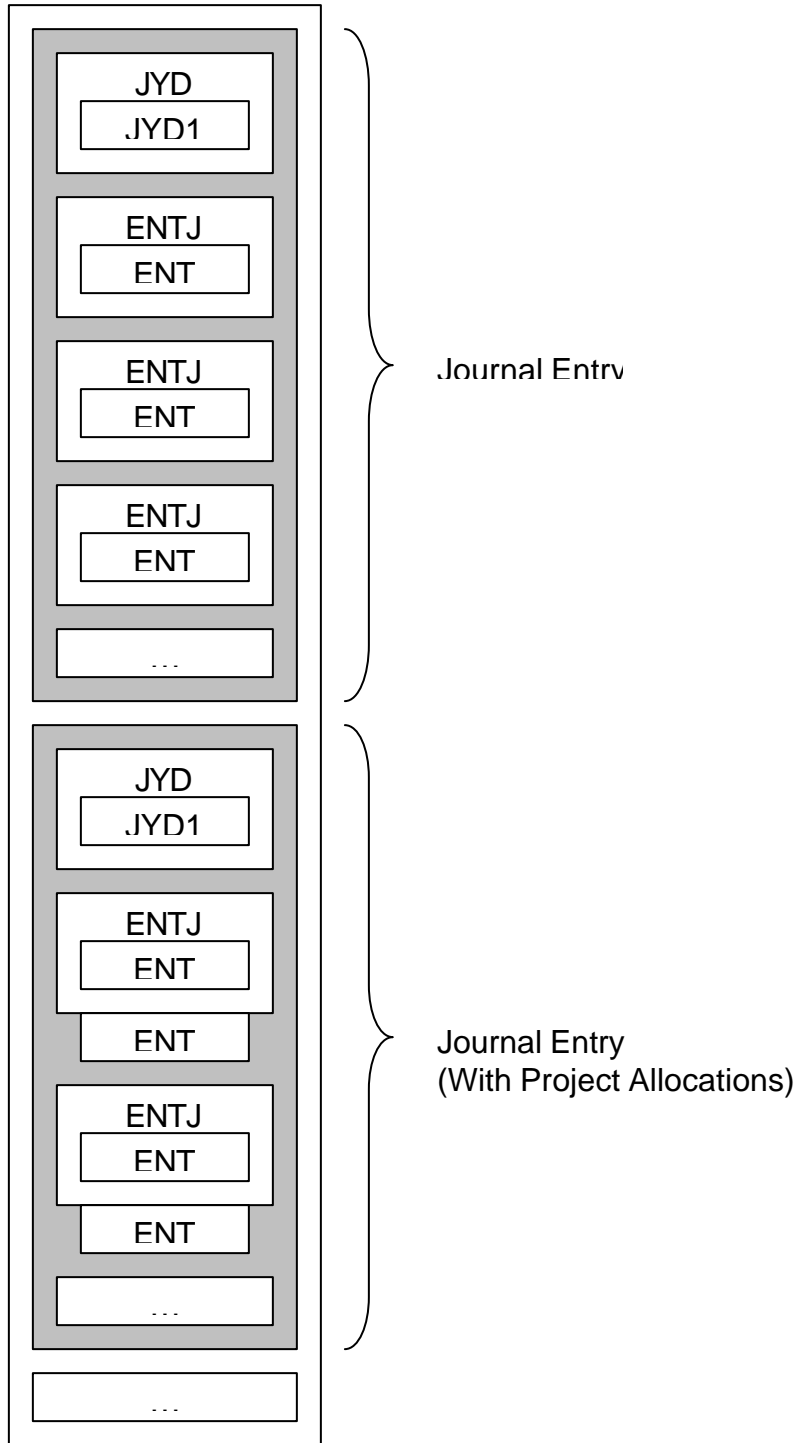
**Figure 1: ASC File Layout**

### 3.2 ASJ File

The ASJ file contains journal entries. The journal is a history of changes made to the general ledger accounts.

Every journal record has a *JYD* header that indicates its date, type, and source. The *JYD* header is followed by a series of *ENTJ* records that store changes to individual accounts. Invoices are transactions that modify accounts. Therefore, each *VYD* record in the ASC file will have a corresponding journal record in the ASJ file.

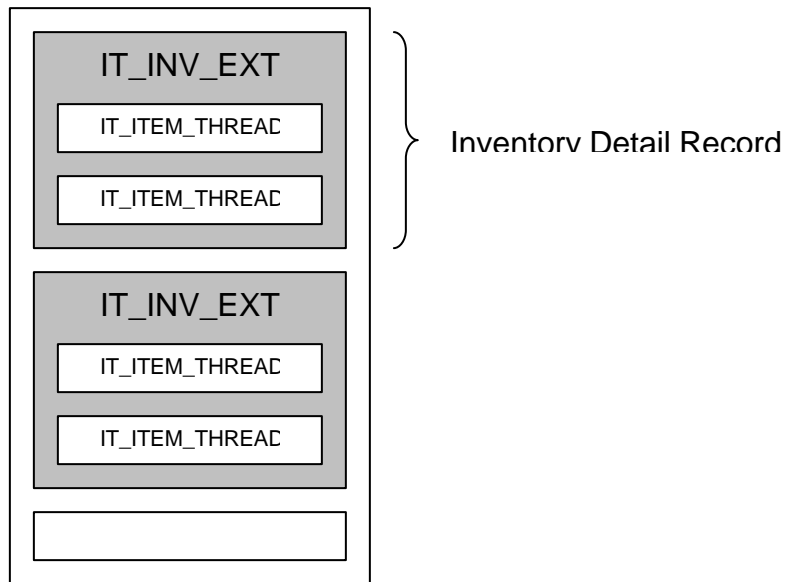
Each *ENTJ* indicates the account number and amount of a general ledger debit or credit. The sign of the *ENTJ.accRec.amt* field does not correspond to debit or credit. If the transaction causes an increase in the amount of the account that it is applied against, then the *ENTJ.accRec.amt* field will have a positive value. The converse is also true. Furthermore, atomic *ENT* records that indicate changes to project allocation accounts will follow the *ENTJ* records where appropriate.



**Figure 2: ASJ File Layout**

### 3.3 IT0 File

The IT0 file contains a series of *IT\_INV\_EXT* records that individually complement the inventory ledger accounts in the ASC file. A new *IT\_INV\_EXT* record is created for every tracked part number. If any record in the IT0 file disagrees with an *INV* record in the ASC file, then the Simply Accounting application will discard the entire IT0 file.



**Figure 3: IT0 File Layout**

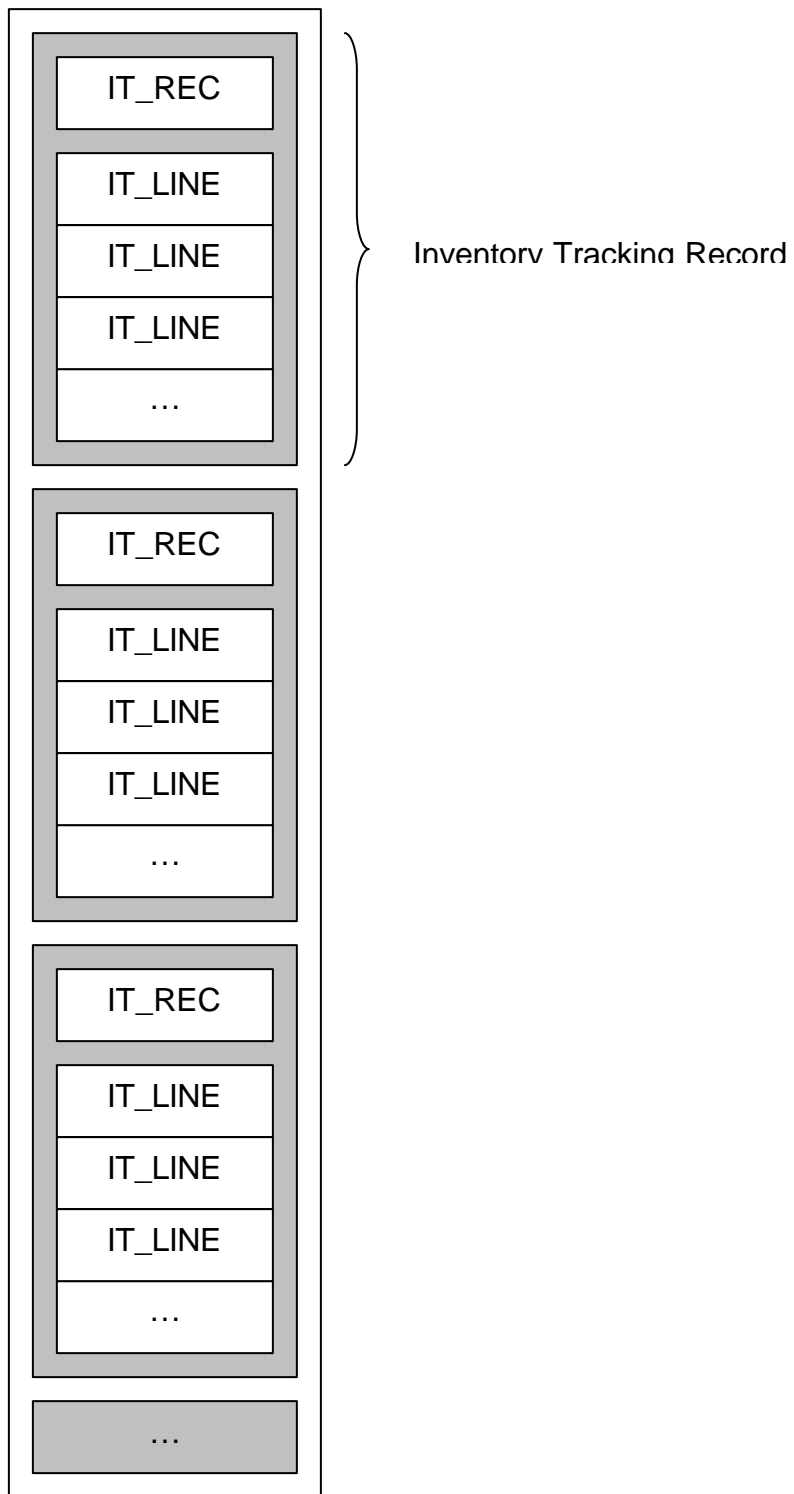
### 3.4 IT2 File

Invoices are broken into two parts that are stored in the IT2 file and the IT3 file. The IT2 file stores inventory tracking records. The IT3 file stores inventory lookup records. The IT2 file design is almost identical to the IT3 file design. Invoices are represented by one record in the IT2 file, and one record in the IT3 file, together.

Inventory tracking records consist of an *IT\_REC* header followed by one *IT\_LINE* record for each line item on the invoice. The *IT\_REC* stores information about the type of invoice, the invoice number, the discount terms, the taxation totals, and the total invoice amount.

Each *IT\_LINE* stores information about the type of line item that it represents, the internal part number, the external part name, the part unit information, and the general ledger account against which the line item is posted. The unit information consists of the total line item amount, quantity, cost, and price.

The Simply Accounting application probably generates inventory reports and histories from the IT2 file. If the inventory record pairs or their threads become inconsistent, then the Simply Accounting application will delete both files.



**Figure 4: IT2 File Layout**

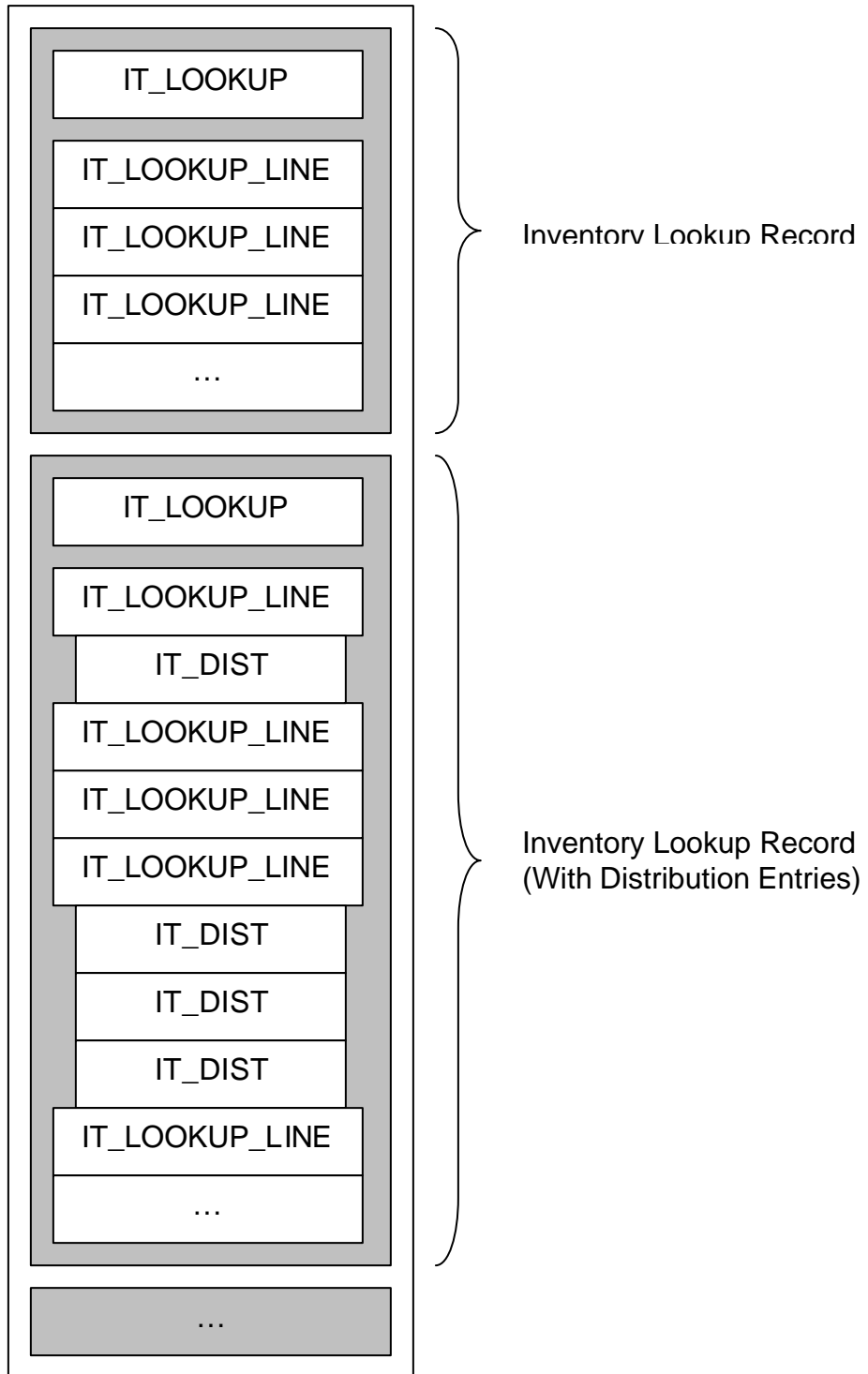
### 3.5 IT3 File

The IT3 file stores inventory lookup records. Each inventory lookup record is an *IT\_LOOKUP* header followed by a series of *IT\_LOOKUP\_LINE* records, which represent invoice line items, and *IT\_DIST* records, which represent distributed line items. Distributed line items are special instances of line items that contribute to more than one general ledger account. Note that distributed line items are not reflected in the IT2 file.

The record header contains the default tax rates from the ASC file at posting time, the receivable account to which the invoice total was posted, a variety of miscellaneous subtotals, and customer information. The *IT\_LOOKUP\_LINE* records store subtotal information about their respective line items.

The Simply Accounting application probably generates customer reports and histories from the IT3 file. It is known that the Simply Accounting application uses the IT3 file to generate invoice browse lists.

It is possible that the Simply Accounting application does not implement distributed line item handling. No examples of distributed line items could be created with application testing. Further testing is required to determine whether distributed line items are an artifact of a previous Simply Accounting version.

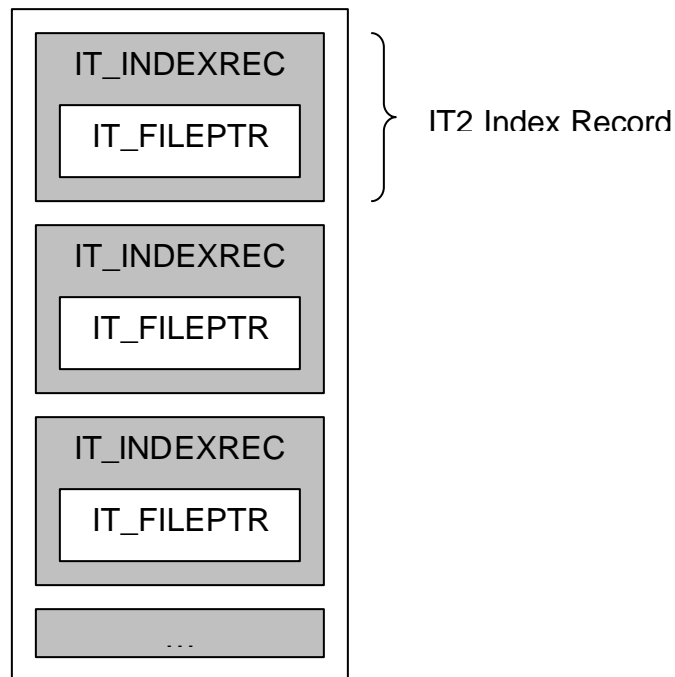


**Figure 5: IT3 File Layout**

### 3.6 IT4 File

The IT4 file is an array of type *IT\_INDEXREC* that points into the IT2 file. Every IT4 record points to one corresponding IT2 record. If the IT2 file is ever deleted by the Simply Accounting application, then the IT4 file is also deleted.

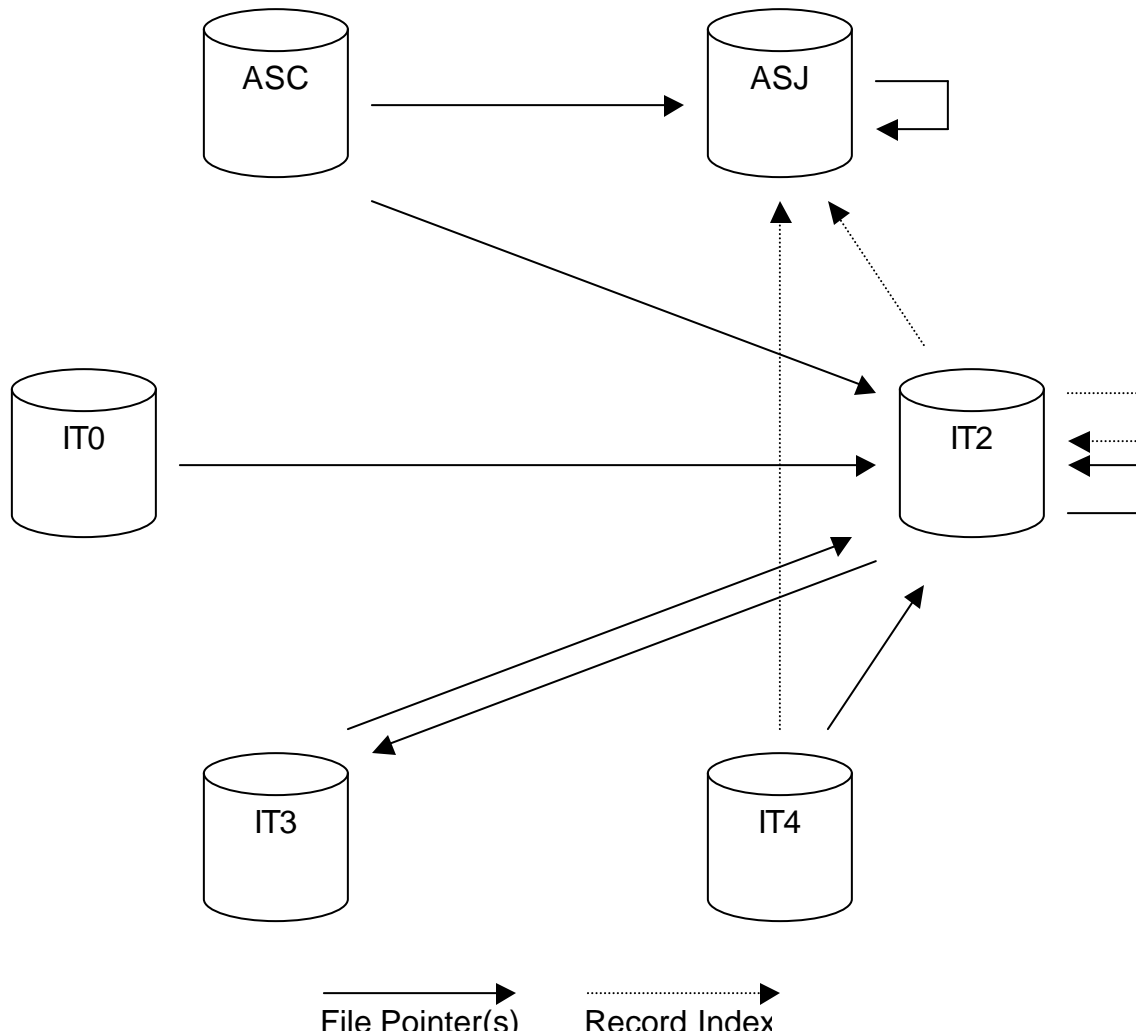
The Simply Accounting application might use the IT4 file to begin the generation of invoice reports because IT4 records contain the invoice source, the invoice total, the journal entry number, and the journal posting date of a transaction.



**Figure 6: IT4 File Layout**

## 4.0 Record Threading

The Simply Accounting application maintains a convoluted web of file pointers and record indices that 'thread' records together into groups. Threads are used to quickly retrieve information from files and maintain associations between related records.



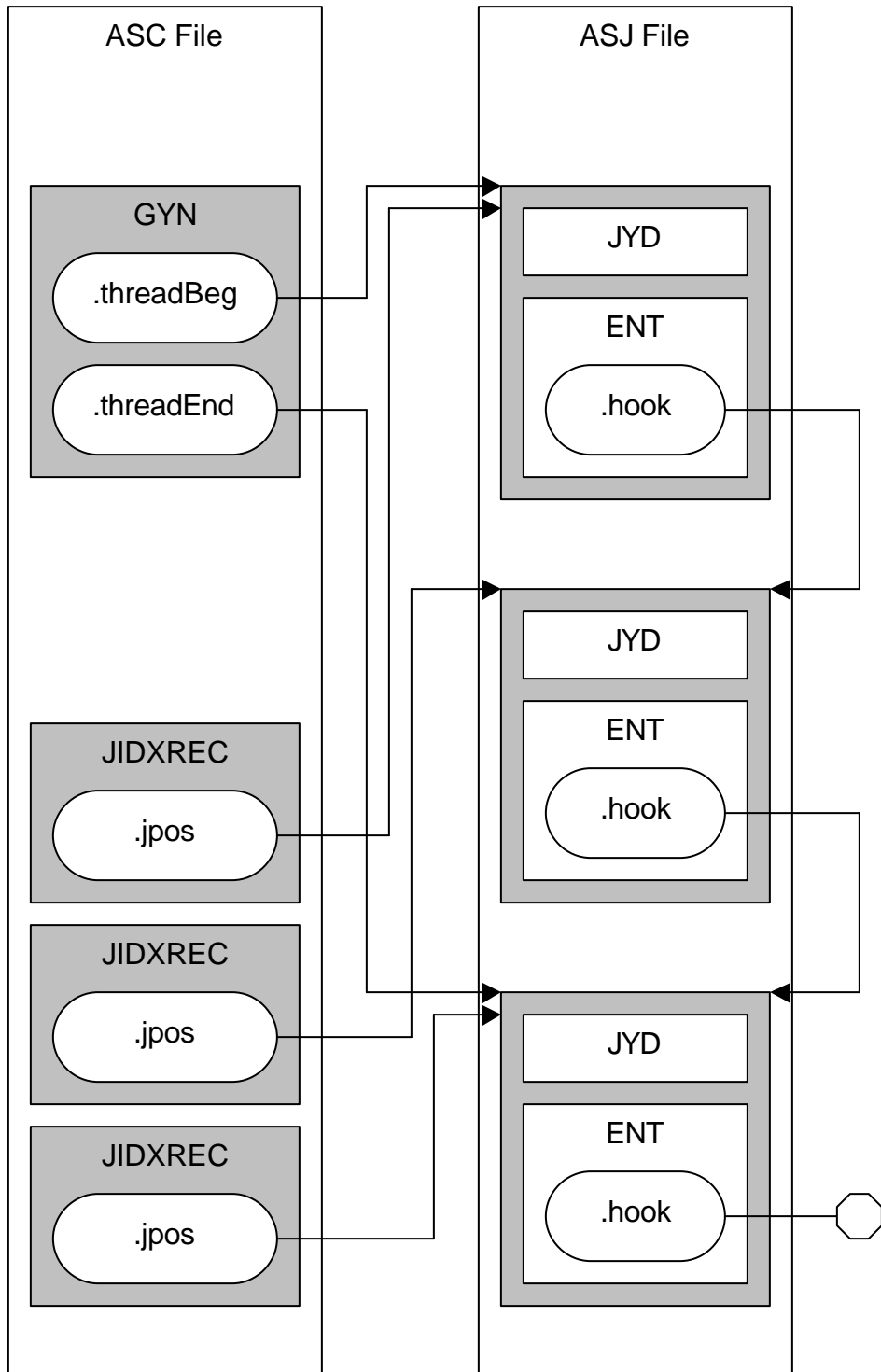
**Figure 7: File Threading Overview**

## 4.1 Journal Threading

Journal records are threaded together by general ledger account number with the *ENT.hook* field. The *GYN.threadBeg* field and the *GYN.threadEnd* field of the appropriate general ledger account respectively mark the head *JYD* and tail *JYD* of this thread.

Journal records must never contain journal entries with duplicate *ENT.acnt* values because account debits and credits for a transaction are recorded in the aggregate. It follows that a journal record will never be self-threaded, and that it will be a member of a different thread for each *ENT* record that it contains. Figure 8 presents an example where a general ledger account has three transactions applied against it.

Every journal record is indexed by a *JIDXREC* in the ASC file. The *JIDXREC.jpos* field points to the *JYD* header of the record. If the journal record is the result of an invoice transaction, then the *IT\_REC.nJorNum* field of the IT2 record and the *IT\_INDEXREC.nJorNum* field of the IT4 record must equal the *JYD1.jorn* field. (These record indices are not depicted by figures in this document.)



**Figure 8: Journal Threading Diagram**

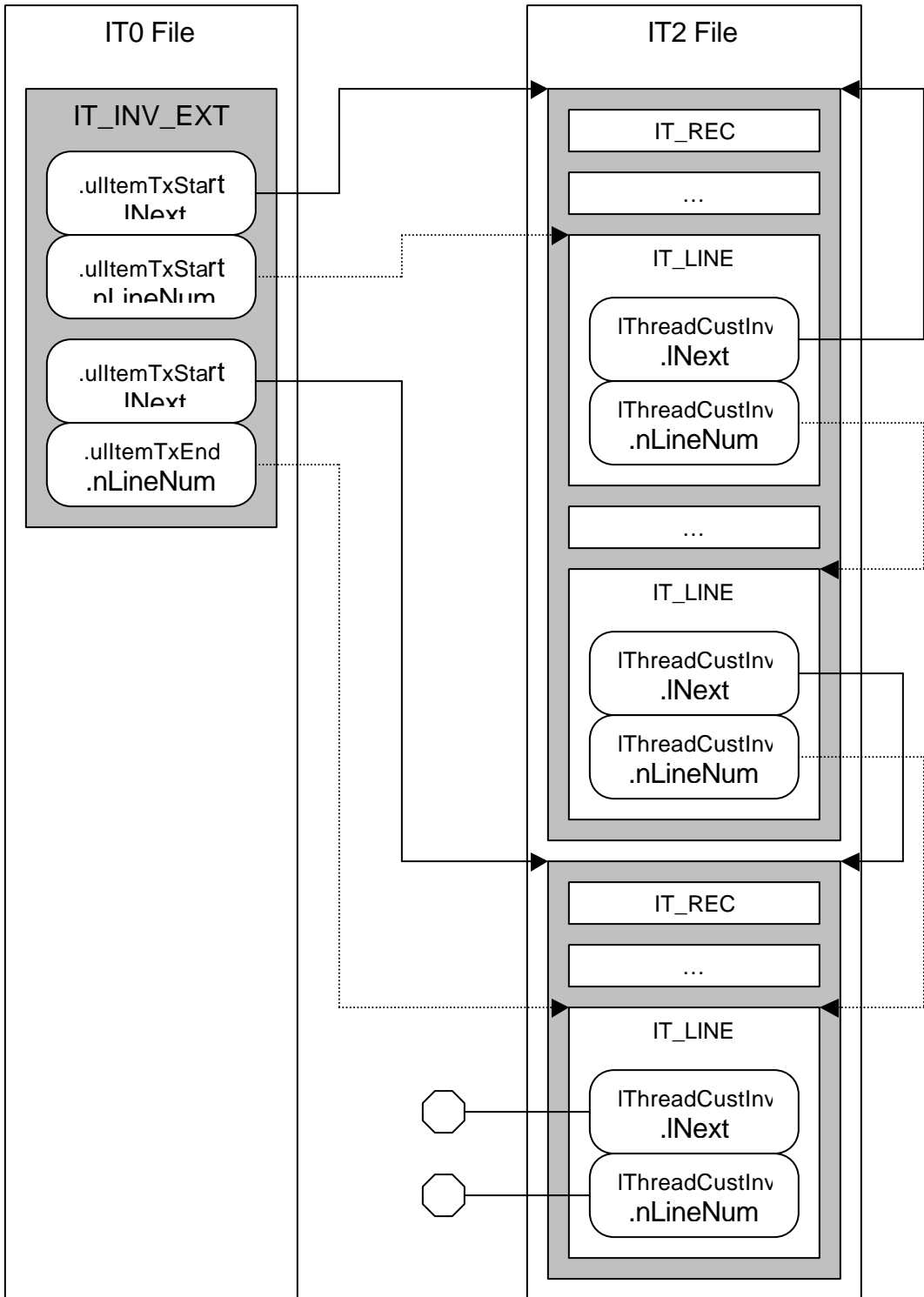
## 4.2 Inventory Threading

In the IT2 file, inventory tracking records are threaded together by internal part number with their child *IT\_LINE.sNxtItemLine* fields. All threading elements are of type *IT\_ITEM\_THREAD*, which contains an *IT\_REC* pointer called *INext* and an *IT\_LINE* index called *nLineNum*.

From the IT0 record for an internal part number, the *IT\_INV\_EXT.ulItemTxStart* field gives the head of the thread, and the *IT\_INV\_EXT.ulItemTxEnd* field gives the tail of the thread.

Inventory tracking records will be self-threaded if they have more than one line item with a particular part number. In self-threaded records, for all but the last instance of a part number within the IT2 record, every *IT\_LINE.{IT\_ITEM\_THREAD}.INext* field will be a self-reference to its parent *IT\_REC* header.

Figure 9 illustrates a thread with three contributing line items in two different inventory tracking records. The first inventory tracking record is self-threaded.



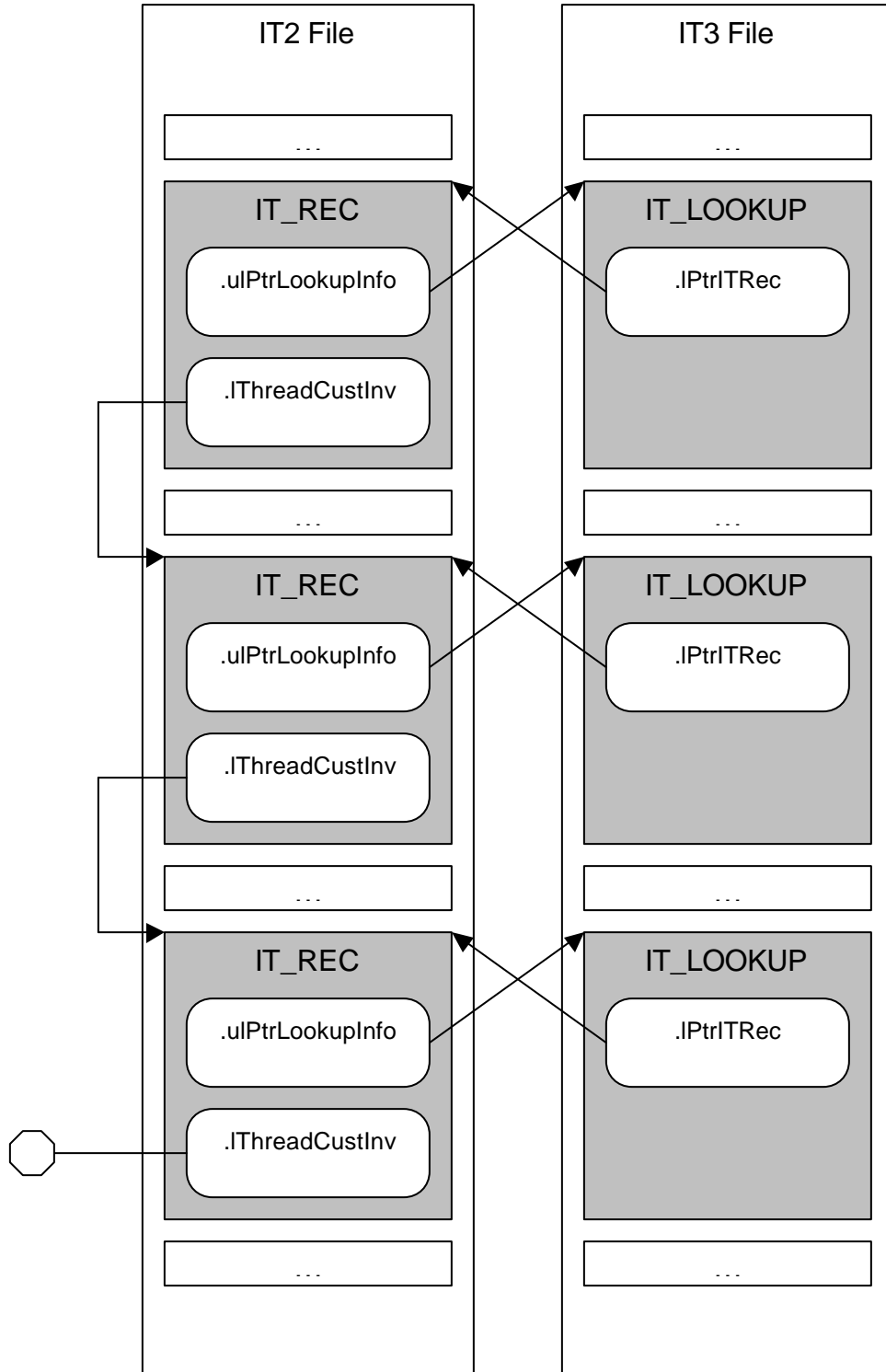
**Figure 9: Inventory Threading Diagram**

### 4.3 Customer Threading

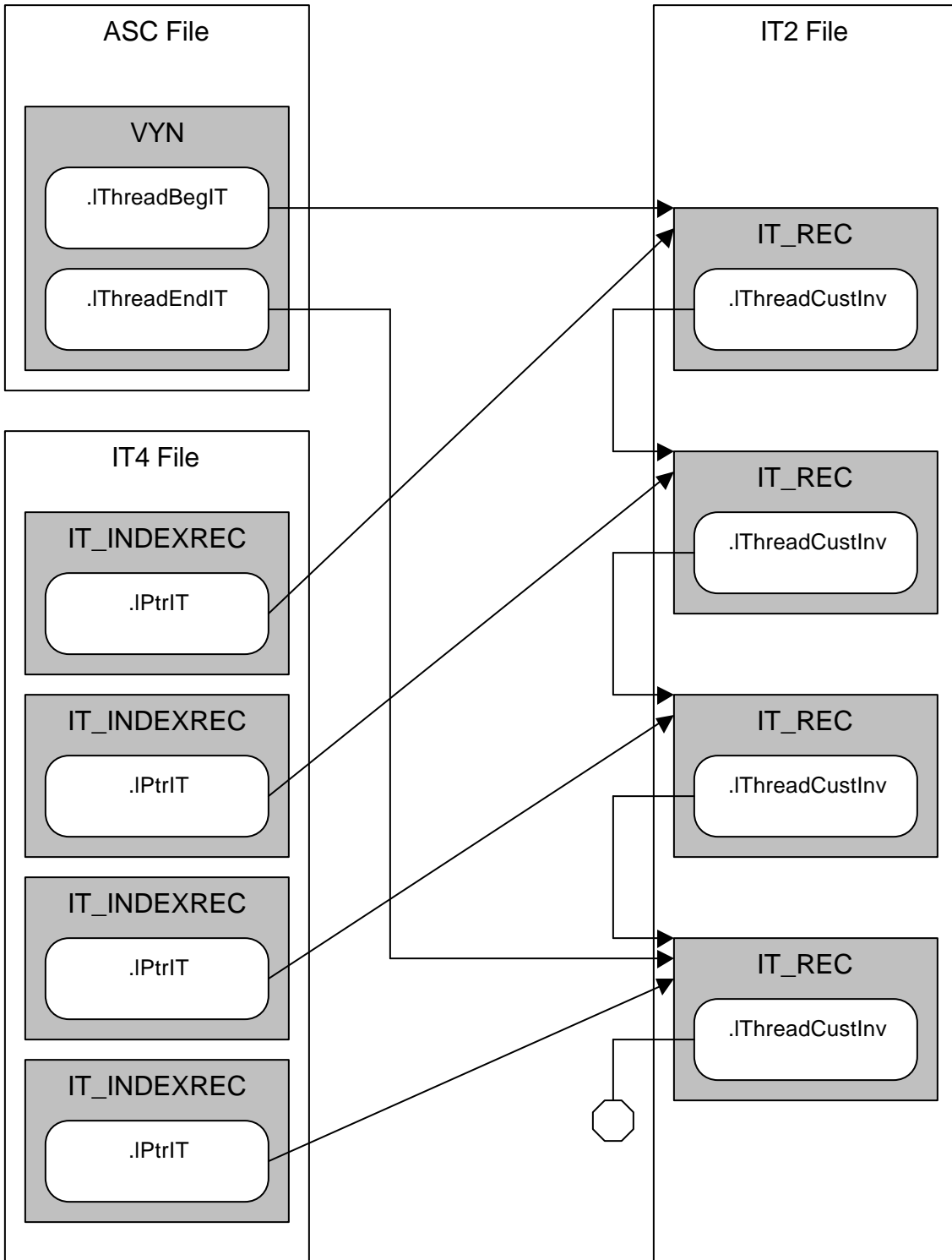
Inventory tracking records are threaded together by internal customer number. The *IT\_REC.IThreadCustInv* field points to the next inventory tracking record that shares the same *IT\_REC.ulCustNum* value. This threading is distinct and unrelated to the inventory threading.

The *IT\_REC.lPtrLookupInfo* field points to the other half of the invoice, which is stored as an inventory lookup record in the IT3 file. The inventory lookup record points back at its twin with *IT\_LOOKUP.lPtrITRec*.

Figure 10 contains an example of the IT2 and IT3 relation of three invoices posted against the same customer. Figure 11 depicts the ASC, IT2, and IT4 relation for a customer with four invoices. (The possibility of additional inline records, as previously denoted by the ellipsis symbol, have been omitted from Figure 11 because of space constraints.)



**Figure 10: Customer Threading Diagram (IT2, IT3)**



**Figure 11: Customer Threading Diagram (ASC, IT2, IT4)**

## 5.0 Invoice Insertion Pseudocode

Create an invoice from one *IT\_REC* and one *IT\_LOOKUP*;

```
SET IT_REC.nJor = 8;  
SET IT_REC.dateJournal = Invoice Date;  
SET IT_REC.dateUsing = Today's Date;
```

FOREACH line item create an (*IT\_LINE*, *IT\_LOOKUP\_LINE*) pair

```
IF the line item is a service item THEN  
    SET IT_LINE.bService flag;
```

```
SET IT_LINE.ulGIAcct = General Ledger Account Number;  
SET IT_LINE.dfQty = Item Quantity (Number Of Units);  
SET IT_LOOKUP_LINE.dPrice = Item Unit Price;  
SET IT_LINE.dfPrice = IT_LOOKUP_LINE.dPrice;  
SET IT_LINE.dfAmt = IT_LINE.dfQty * IT_LINE.dfPrice;  
SET IT_LOOKUP_LINE.sItem = Item Source (The Part Code);  
SET IT_LOOKUP_LINE.sUnit = Item Unit Of Measurement;
```

```
IF you have a Canadian invoice THEN  
    SET IT_LOOKUP_LINE.nGst = GST Tax Code;  
    SET IT_LOOKUP.fGstRate1 = Rate for IT_LOOKUP_LINE.nGst from the SN_REC;  
    SET IT_LOOKUP_LINE.dGSTAmt = IT_LINE.dfAmt * IT_LOOKUP.fGstRate1;
```

```
SET IT_LOOKUP.pstRat = Default PST/SST Tax Percentage from the SN_REC;  
SET IT_LOOKUP_LINE.dSTaxAmt = IT_LINE.dfAmt * IT_LOOKUP.pstRat;
```

```
SET IT_LOOKUP_LINE.sItem = IT_LINE.sSource;  
SET IT_REC.dfInvoiceAmt += IT_LINE.dfAmt ;  
SET IT_REC.dfInvoiceAmt += IT_LOOKUP_LINE.dGSTAmt ;  
SET IT_REC.dfInvoiceAmt += IT_LOOKUP_LINE.dSTaxAmt ;
```

```
SET IT_LOOKUP_LINE.dfAmtIncTax = IT_LINE.dfAmt ;
```

```
SET IT_LOOKUP.nNumLines += 1 ;  
SET IT_LOOKUP.dfSalesTax += IT_LOOKUP_LINE.dSTaxAmt ;  
SET IT_LOOKUP.dfGST1 += IT_LOOKUP_LINE.dGSTAmt ;  
SET IT_LOOKUP.dfTotal += IT_LINE.dfAmt ;
```

```

Get the customer ASC acctrans for the invoice by customer name or number;
IT_REC.nJorNum = The Next Available Journal Number;
IT_REC.ulCustNum = The Internal Customer (from the acctrans);
SET AS STRING IT_LOOKUP.sSource1 = The Next Available Invoice Number;
Create an IT2 record from the IT_REC;
Create an ENTJ record for every unique value of IT_LINE.ulGIAcct;

FOREACH IT_LINE
    SET the proper ENTJ.accRec.amt += all IT_LINE.dfAmt;
    Add the IT_LINE to the IT2 record;

Add the IT2 record to the file image;
Create a new IT3 record from the IT_LOOKUP record;
Create a JIDXREC record;
SET JIDXREC.dateJournal = IT_REC.dateJournal;
SET JIDXREC.nJorNum = IT_REC.nJorNum;
SET JIDXREC.nJor = IT_REC.nJor;
SET JIDXREC.sSource = IT_REC.sSource1;
SET JIDXREC.lPtrIT = IT2 record offset;
SET JIDXREC.bLookupCleared = 0;
SET JIDXREC.bReversed = 0;
SET JIDXREC.bReversal = 0;

FOREACH IT_LOOKUP_LINE
    SET JIDXREC.dfInvoiceAmt += IT_LOOKUP_LINE.dfAmtInclTax;
    SET JIDXREC.dfInvoiceAmt += IT_LOOKUP_LINE.dGstAmt;
    SET JIDXREC.dfInvoiceAmt += IT_LOOKUP_LINE.dSTaxAmt;
    Add the IT_LOOKUP_LINE to the IT3 record;

Add the IT3 record to the file image;
Thread the IT2 and IT3 files;
Add the IT4 record to the file image;

Create a JYD record.
SET JYD.commnt = IT_LOOKUP.sCustNam;
SET JYD.commLen = strlen( IT_LOOKUP.sCustNam );
SET JYD.jf.date = IT_REC.dateJournal;
SET JYD.jf.jorn = IT_REC.nJorNum;
SET JYD.jf.modu = 2;
SET JYD.jf.type = INVOICE;
SET JYD.jf.source = IT_REC.sSource1;
Create a journal record from the JYD record;

Create ENTJ records for the integration accounts;

FOREACH ENTJ
    Create journal entry record with the ENTJ;
    Add the journal entry to the journal record;

```

```
Add the journal record to the file image;

FOREACH journal entry
    Find the GYN for that ENTJ;
    GYN.ytc += ENTJ.accRec.amt;
    Update the GYN journal pointers;

Create a VYD record;
VYD.dat = IT_REC.dateJournal;
VYD.amount = JIDXREC.dfInvoiceAmt;
VYD.source = IT_REC.sSource1;
VYD.dfPreTaxAmt = IT_LOOKUP.dfTotal;
Add the VYD to the acctrans;

Add a journal index to the ASC;
Update the file sizes and index counts in the ASC;

END.
```