ClearLogistics Enhancements

FIFO Costing Module 1.1.1

Installation and Users Guide

Table of Contents

First Choice Software, Inc.
8900 Business Park Drive
Austin, Texas 78759

Web: www.fchoice.com
Phone: (512) 418-2905
Fax: (512) 418-2983

Overview

The *ClearLogistics Enhancement* line of products from First Choice Software, Inc. is designed to add commonly-requested functions to your ClearLogistics system, and to simplify the task of customizing your Clarify System. The **FIFO Costing Module** allows you to perform all of your warehouse functions (including transfers and receipts) using a FIFO (First In First Out) costing paradigm. This enhances the baseline Clarify function of *standard* costing to give you much greater control on how you cost and account for your inventory assets.

The **FIFO Costing Module** integrates seamlessly with both Clarify and the other Logistics modules from First Choice, such as Clear Call Center Integration and RF/Shipping.

What's new in Version 1.1.1

Version 1.1.1 repairs a tiny bug found in the part transfer routine. If, for quantity-tracked parts only, a transfer removes the last inventory in a bin (and the bin is 100% empty of that part), a header flag was not reset. This only causes the bin (with 0 inventory) to show up in the inventory lists, instead of having the row not displayed at all.

The bug is fixed in both trans.sql and transor.sql. Simply replace your file with the newer version, and recompile.

What's new in Version 1.1

Version 1.1 of FIFO costing contains a new piece of functionality for part transfers. The part transfer API now allows you to add more units to an already existing part transfer. See the API reference section for more details.

Standard Costing

Baseline Clarify ships with a singular costing mechanism for inventory parts: standard costing. In this model, each revision of each part can be assigned one (and only one) cost. That is, the cost is the price that you have to pay for this part/revision when you order it from your suppliers.

The standard costing model has some flexibility – each part revision can be costed at a different cost at different times. You may set up, for example, that the cost of a given part is $1.51 in June, but costs us $1.52 for the first two weeks of July.

FIFO Costing

Many Clarify customers have requested an additional method of costing. The basic requirement is that each time a particular part is ordered it may be at a different cost, based on the supplier used, the time of year, the lead time requested, and many other factors.

Thus, the requirement calls for Clarify to be able to record for a given set of parts in inventory the date and time that it was brought into inventory, the quantity of units placed in inventory, the unit cost for them, and (optionally) the source of the parts.

For example, suppose you order a quantity of a part (e.g. a CD Rom) from your supplier each week. On week 1 you order 90 units at a cost of $2.00 each. A week later you call up to order 50 more, and the cost has gone up to $2.50 (due to supply problems). A week later, another 100 are ordered at $2.25 each.

When the units are received (via the First Choice Purchase Order Module, or some other means), the FIFO cost records are created. Then, when the units are transferred out of the warehouse (via a part transfer or a fulfill), they are transacted (or *costed*) using a FIFO mechanism.

So, for example, suppose that we now ship 175 units from the warehouse (to a customer site). In Clarify terms, this would be a fulfill to an expense GL account. Suppose that the shipments were made in quantities of 25. The first 25 would have a total cost of $50.00 ($2.00 for 25 units). The second 25 (and the third) would also have a cost of $50.00. But the fourth group of 25 would have a cost of $55.00 ($2.00 for the first 15 and $2.50 each for the next 10). The next group of 25 would have a cost of $62.50 ($2.50 for 25 units). The next 25 would be $60.00 ($2.50 for 15, and $2.25 for 10). Finally, the last group would have 25 units at $2.25 each for a total of $56.25.

It is important to understand that *specific* units in inventory are not associated with specific costs. Rather, Clarify now remembers how many units exist in inventory at a given price (and from a given source) and will deplete them (and cost them in part transfers) based on the oldest costs being used first. It may be that a specific unit may be transacted into Clarify at a particular cost, and transacted out of inventory at another cost. The whole idea behind FIFO is to keep track of the total inventory cost, and not to associate specific costs with specific units.

FIFO costs are also kept on a *part number basis*. This means, that for a given warehouse, the FIFO totals are kept and used for all revisions of a part number in the warehouse. It is possible to customize FIFO costing to work on a mod_level (revision) basis, but that is not the default behavior.

FIFO costs are also tracked on a warehouse (or inv_locatn in Clarify parlance) basis. They are not tracked by individual inventory bins. So, a unit can be moved into a warehouse in Bin1 (and have the FIFO cost added to the system), but the FIFO cost will be decremented for a unit shipped out of the warehouse from another bin.

Finally, FIFO costs are kept by this module for any movement in or out of a warehouse. Expense GLs do *not* have FIFO costs – only inventory locations. So, any time that a unit is moved into a warehouse or out of one, the FIFO costs are adjusted. If a unit (or units) are transferred from one inventory location to another, the FIFO costs for those units are moved from the first warehouse to the second.

Note: When you have a customer return (if you do not use the First Choice Software Purchase Order Module, or similar function) and it does not have a cost associated with it, the FIFO system will assume that it was recently shipped, and will increment the most recent cost record for that part and that inventory location.

Directions of Transfer

There are three ways in which inventory can be transferred.

1. Into an inventory location. This is typically a receipt (such as from a PO), but can be a transfer from any expense GL (such as building new units from sub-components). The from location will be the expense GL, and the to location will be the inventory location. FIFO costing will do one of two things with a situation such as this:
   - If the part request for the transfer has a value in the x_order_price field, it will create new FIFO costing values for the goods
   - If there is no part request (or no x_order_price value), then the FIFO module will increment the *newest* FIFO record for that part with the quantity for the transfer. This is typically used for customer returns.
2. From an inventory location. This is typically a shipment to a customer, but can be a transfer to any expense GL (such as when you scrap a part). The from location will be the inventory location, and the to location will be a GL account. FIFO costing will consume the proper (oldest) FIFO records from the inventory location.
3. Transfer from inventory location to inventory location. In this situation, the FIFO module will transfer the FIFO records from the from location to the to location.

Sources

The FIFO costing module also (optionally) associates *sources* with each FIFO record. This allows you to track where the units came from, as well as the costs for those units. A new user-defined list FIFO_SOURCE is provided. You may place whatever values you wish in this list. For example, you might want to track the following values for sources: Customer Returns, Vendor Purchased, Scavenged. The values you place in this list are up to you, but the list must always have at least one value in it (even if that value is something such as **Not Used**).

Sources are very valuable for reporting purposes. You can see how many parts have been used from the different sources, and how well they have performed.

Give Backs

The FIFO module ships with a Clear Basic routine to allow part transfer *give backs*. This is a new function in Clarify that allow you to fix up part transfers after the fact.

Suppose that you made a transfer (for a material log, for example) of 5 units of a particular part. After you perform the depot repair you realize that you only used 3 of the units. With base Clarify, you would transfer back the two remaining units.

The new give back method allows you to specify the part transaction id and the number of units to "give back". It will then patch up the part transaction, including the number of units and standard cost. It will also adjust inventory as needed, and will create or destroy FIFO costing records.

This function is completely optional, and may be used or ignored as needed for your site.

Loading FIFO Costing Data

There are three primary mechanisms for loading FIFO costing data:

1. Using the provided interface. This is typically used to initially populate your system.
2. Using the supplied GUI. This is most often used if (somehow) your FIFO costing tables are out of sync with your inventory (usually because someone placed something into inventory without entering it into Clarify).
3. Loading the x_order_price field of a part request, and receiving it into an inventory location with the supplied part_transfer or receive routine (receive routine is part of the First Choice Software Purchase Order Module).

If you wish, you can write your own method (using the provided code as a model) for populating the FIFO costing tables.

Interface

An interface is provided with this package. The interface allows FIFO costing records to be created in Clarify. The interface is provided in the *load_fifo.cbs* file. This file reads a simple ASCII file (a sample is provided with this module), validates the data, and loads the data into the FIFO costing table.

Command Line

The interface is called with the following command line:

```
<path>\cbbatch –f load_fifo.cbs
               -r load_cost
```

Where: <path> is the path to the Clarify rulemgr directory

Note: You must have a valid clarify.env in your current working directory, or specify the logon parameters with the cbbatch command. See the cbbatch documentation for more information.

An example command line follows:

```
C:\apps\clarify\server\60\rulemgr\cbbatch -f load_fifo.cbs
                -r load_cost
```

Input File

The interface requires that you first create an ASCII input file with the FIFO costing data. There is no requirement on how you create the interface, but you must make sure that the input file conforms to the format listed below.

Each FIFO costing record must be on its own line of the interface line (with a CR-LF as a row separator). Each row of the input file contains 6 fields, separated by the pipe character (“|”). A sample input file is provided with the module.
The fields for each row are as follows:

1. Part number - This must be a valid part in the Clarify system.
2. Inventory location – This must be a valid inventory location in Clarify. It must also be ACTIVE.
3. Date of entry – This is the FIFO date/time entered in the Clarify system. If no time is specified, it is assumed to be at midnight for the date given. This field must be a valid date/time in the format required for whatever locale your local system is configured for.
4. Quantity – This is the number of FIFO units to record in the system.
5. Cost – This is the cost for the units recorded in the system.
6. Source – This is the source of the parts. It must be a valid value from the FIFO_SOURCE user-defined list. If you leave this field blank, the default source from that list will be used.

The following is a sample input row for the interface. It specifies that 5 units of MS Excel should have FIFO records at the Austin inventory location. They should be listed for 8AM on November 11th at a cost of $45.56, and should be listed as coming from Source1.

```
MS Excel|Austin|11/11/98 08:00:00|5|45.56|Source1
```

Log Files

The interface produces a variety of log files. The most important is *status.log*. This file details the status of the interface. The following is a sample status.log file:

```
Status for FIFO Cost Interface

Date of Interface: 11/13/98 09:16:12 AM
Input File Name  : fifo.in

[09:16:12] Interface started.
[09:16:12] Interface completed.

Results of interface:
---------------------

Records processed: 2
Records ignored  : 0

Errors found     : 0
```

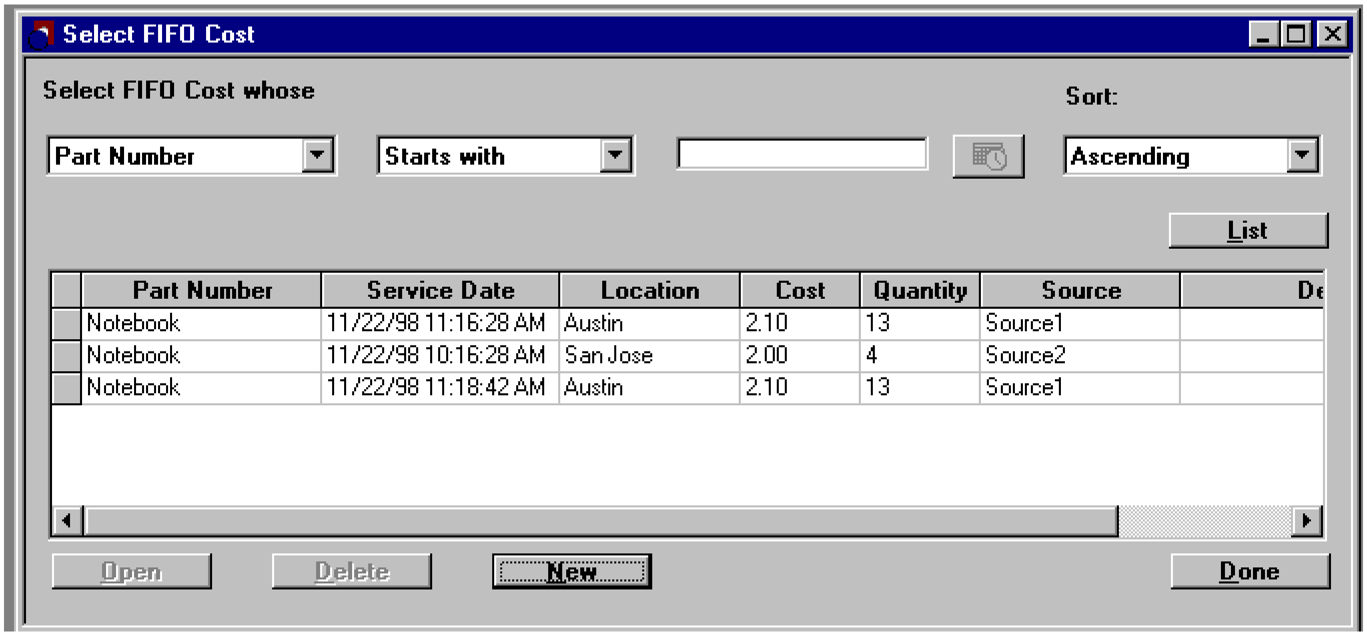
```
Warnings found    : 0
```

In addition, three other log files may be created:

1. error.log – If any fatal errors occur, they will be detailed in this file.
2. warn.log – If any non-fatal warnings occur, they will be listed in this file.
3. badrec.log – If any input rows cannot be processed (for example, for a bad source, or a non-valid cost), the input row will be placed in this file. The badrec.log file can then be edited, and used as input for the interface to load the corrected data.

GUIs

There are two new GUIs provided with this module. The first of them allows you to filter and view the various FIFO costing records. It has a standard Clarify filter, and allows you to open existing FIFO costing records (to edit them), delete records, or create New FIFO Costing records.



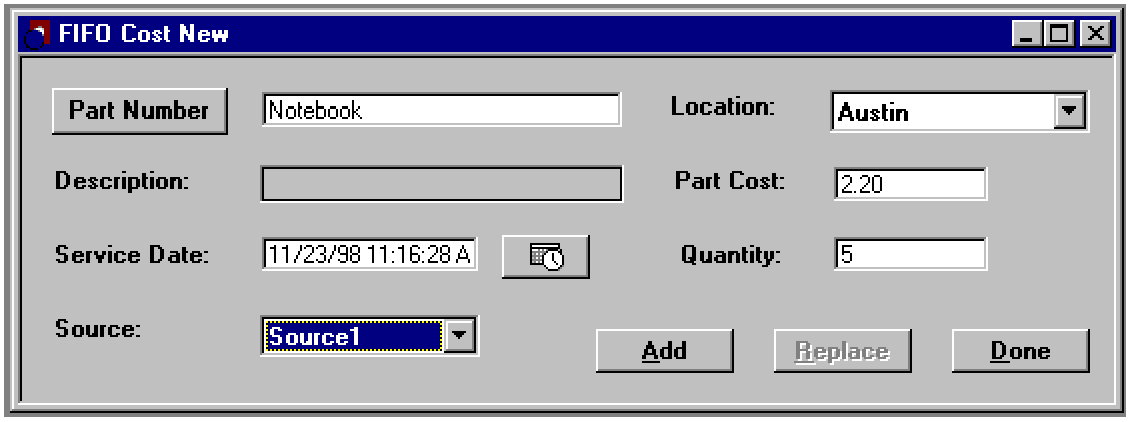
Part Number	Service Date	Location	Cost	Quantity	Source	De
Notebook	11/22/98 11:16:28 AM	Austin	2.10	13	Source1	
Notebook	11/22/98 10:16:28 AM	San Jose	2.00	4	Source2	
Notebook	11/22/98 11:18:42 AM	Austin	2.10	13	Source1	

This form can be called from the Select menu.

The other GUI is used to create (add) new FIFO cost records manually, or edit existing records. For each record you may filter the part number. You may enter as much of the part number as you know and press the Part Number button, or simply press the button to display the Select Part form.

You must select the proper inventory location from the dropdown list, the service date for the FIFO record (you may use the provided date/time button), the part cost (per unit), and the quantity. Finally, you must select the source for the parts from the dropdown list. If you are not using sources, that list will contain only one value.

Pressing the Add button will add a new FIFO record (and display it in the Select FIFO form if it is open). Pressing replace will replace the FIFO record (assuming you've added it, or are editing an existing record).



Part Number: Notebook
Location: Austin
Description:
Part Cost: 2.20
Service Date: 11/23/98 11:16:28 A
Quantity: 5
Source: Source1

Both of these GUIs are (by default) set up in the Policies & Customers application. If you wish, you may move the menu items that call them to any other application or menu.

APIs

The FIFO Costing Module ships with a variety of APIs that you can use to customize your Clarify system with FIFO costs.

The most commonly used is the part transfer API. This API is shipped in both a Clear Basic version, as well as a stored procedure version. Which one you use depends on your needs. They provide similar functionality.

Both APIs can be used in customization, interfaces, and business rules. The most common use for them is to "take over" the function of part transfer form. To illustrate how to do this, a sample code module is provided that performs that function.

The module also ships with a "give back" function. This is described in a section above, and is only shipped as a Clear Basic method. It can be used to "give back" units from a previously committed part transfer.

Finally, there are a number of "helper" stored procedures that are included with this package, and which can be called on their own.

The section at the end of the document details the use of the APIs provided with this module:

Other Implementation Issues

Selection Lists

The FIFO Costing module ships with a series of sophisticated user-defined lists. These are contained in two data exchange files that you import during the installation (see below). You should never have to modify these lists.

FIFO_SOURCE List

This is a list of the sources for the parts recorded in the FIFO system. When you install the product, this list will be created for you. After you import the provided *fifo_source.dat* file, you should use the User-Defined List Editor (in the Policies & Customers application) to add/delete/modify the list of sources as you desire.

The one default value in the list will be "Default Source". If you are not using sources, you may choose to leave this data value in place.

Other Form Changes

There are a couple of optional form changes that you can make with this module:

Part Transfer Form

It is highly recommended that you apply the code in 530.cbs to a user version of the part transfer form. This will call on the part transfer API (Clear Basic version), instead of the default part transfer code.

New Menu Items

When you compile the global module fifo_glob.cbs you will add the following menu items to your Clarify system:

Application	Menu	Item	Description
Clear Logistics	Select	FIFO Cost	Displays the Select FIFO Cost Form
Clear Logistics	New	FIFO Cost	Displays the FIFO Cost Form

Implementation

This section provides detailed requirements, what files are included in this product, installation and other implementation considerations.

Requirements

This version of the **FIFO Costing Module** requires the following:
Clarify Version: 4.5 or later

Clarify Tools:	Data Dictionary Editor (ddeditor) User Interface Editor (uieditor) Clear Basic Compiler (cbex) Data Exchange (dataex)
Other Tools:	Valid database system (ex: MS SQL Server version 6.5)

Packaging

FIFO Costing Module is shipped to you as a zip file.
Example: fifo.1.1.1.zip

Installation Tree

It is recommended that you uncompress the zip file into an fchoice\logistics subdirectory created at the top of the Clarify install tree. For example on NT, should your Clarify server install tree be "c:\clarify", then:

1. Switch to "c:\clarify" directory.
2. Create an "fchoice" directory if necessary.
3. Switch to "fchoice" directory.
4. Create a "logistics" directory if necessary.
5. Create an "fifo" directory if necessary.

6. Unzip into “c:\clarify\fchoice\logistics\fifo” directory.

The following files are provided with this product:

File Name	Purpose
code	Directory containing the Clear Basic source code
docs	Directory containing the documentation
files	Directory containing the data exchange files to import
forms	Directory containing the forms for the module
inter	Directory containing the FIFO Costing interface
oracle	Directory containing the stored procedures for Oracle
schema	Directory containing the data model changes for this module
sql_server	Directory containing the stored procedures for SQL Server and Sybase

The *code* directory contains the following entries:

File Name	Purpose
fifo.dir	The directives file to compile the Clear Basic code
530.cbs	Code to put FIFO functionality on the part transfer form
fifo1970.cbs	Code for the Select FIFO form
fifo1971.cbs	Code for the FIFO Costing form
fifo_glob.cbs	Global code for the FIFO Costing module
give_back.cbs	Code for the give_back API
select.cbs	Code for the Select dropdown lists
trans.cbs	Code for the part transfer API

The *docs* directory contains the following entries:

File Name	Purpose
fifo_user.pdf	This file

The *files* directory contains the following entries:

File Name	Purpose
fifo_source.dat	Data exchange file containing the user-defined list of FIFO sources
select.dat	Data exchange file containing the user-defined list for selection
selfifo.dat	Data exchange file containing the user-defined list for FIFO selection

The *forms* directory contains the following entries:

File Name	Purpose
1970.dat	New form for the Select FIFO function
1971.cbs	Clear Basic code for the FIFO Cost form

The *inter* directory contains the following entries:

File Name	Purpose
load_fifo.cbs	Load FIFO cost interface
fifo.in	Sample input file for the FIFO cost interface

The *schema* directory contains the following entries:

File Name	Purpose

log.sch	Schema modifications for the First Choice Logistics Modules

The *sql_server* directory contains the following entries:

File Name	Purpose
build_id.sql	Builds id_numbers for part transactions
check_trans.sql	Validates logistics status transitions
log_help.sql	Assorted helper routines
numsch.sql	Generates the next number for an auto-numbering scheme.
trans.sql	Part Transfer API

The *oracle* directory contains the following entries:

File Name	Purpose
build_idor.sql	Builds id_numbers for part transactions
check_transor.sql	Validates logistics status transitions
log_helpor.sql	Assorted helper routines
numschor.sql	Generates the next number for an auto-numbering scheme.
transor.sql	Part Transfer API

Manual Installation

Note: It is highly recommended that you first install the **FIFO Costing Module** on a test system and get familiar with it before installing it on a production system.

The **FIFO Costing Module** files may be installed on any machine that can execute the Clarify data exchange tool (dataex) and the database command line tool (such as isql or sqlplus. You will also need to run the Clarify Data Dictionary Editor (ddeditor), and the User Interface Editor (uieditor)

Data Dictionary Modifications

The **FIFO Costing Module** requires a number of changes be made to the data schema. If you have already modified the schema for another logistics module, you may skip this step.

Note: You should make all of the schema changes in the log.sch file, no matter how many of the logistics modules you are implementing. Several of the modules reference the tables for other modules, even if you are not using the second module.

New Tables/Views

The following new tables are added to the Clarify system for these modules:

Table Name	Table Number	Purpose
work_order	3530	Work order table
work_order_moved	3531	Tracks parts moved for a work order
fifo_cost	3532	Tracks fifo costs for parts at inv locations
fifo_cost_view	3533	Joins fifo cost tables

batch	3534	Tracks part requests as a group for shipping
container	3535	A box shipped from an inventory location
cont_dtl_qty	3536	Tracks how many of a specific part request are included in a container
container_view	3537	Joins container tables
batch_def	3538	Tracks batch options
inv_sum	3539	Summary table of part revision totals at a given inventory location
trans_fifo_dtl	3540	Tracks FIFO cost details for a part transfer for a give back

If you are already using any of the above table numbers, you may modify the log.sch file and change them to other, unused numbers in the user-defined range (2000-4999). You may also have to search in the sql, cbs, and dat files for the table numbers and replace them as well.

New Fields Added to Existing Tables

The following fields are added to existing Clarify tables:

Table Name	Field Name	Data Type	Length
inv_bin	x_shippable	Long Integer	N/A
part_num	x_shippable	Long Integer	N/A
part_num	x_kit	Long Integer	N/A
part_num	x_prodclass	Variable String	20
part_num	x_category	Variable String	20
demand_hdr	x_po_num	Variable String	20
demand_hdr	x_ship_all	Long Integer	N/A
demand_dtl	x_order_price	Float	N/A

New Fields Added to Existing Views

The following fields are added to existing Clarify views:

Table Name	Source Table	Field Name	New Field Name
parts_view	inv_bin	x_shippable	x_shippable
parts_view	inv_bin	active	inv_active
parts_view	inv_locatn	active	loc_active
partnum_view	part_num	x_shippable	x_shippable
partnum_view	mod_level	replaces_date	replaces_date

New Relations Added to Existing Tables

The following relations are added to existing Clarify tables:

Table Name	Relation Name	Cardinality
case	case2work_order	OTOP
contr_itm	contr_itm_p2demand_dtl	OTOP
contr_itm	contr_itm_s2demand_dtl	OTOF
contract	contract_p2demand_hdr	OTOP
contract	contract_s2demand_hdr	OTOF

demand_dtl	demand_dtl2batch	MTO
demand_dtl	demand_dtl2cont_dtl_qty	OTM
demand_dtl	demand_dtl_p2contr_itm	OTOP
demand_dtl	demand_dtl_s2contr_item	OTOF
demand_hdr	demand_hdr_p2contract	OTOP
demand_hdr	demand_hdr_s2contract	OTOF
inv_bin	inv_bin2batch_def	OTM
inv_locatn	inv_locatn2work_order	OTM
inv_locatn	inv_locatn2inv_sum	OTM
inv_locatn	inv_locatn2fifo_cost	OTM
inv_locatn	inv_locatn2container	OTM
inv_locatn	inv_locatn2batch	OTM
mod_level	mod_level2work_order	OTM
mod_level	mod_level2inv_sum	OTM
mod_level	mod_level2work_order_moved	OTM
mod_level	mod_level2batch_def	OTM
user	user2batch_creator	MTO
user	user2batch_closer	MTO
work_order	work_order2case	OTOF

Updating the Data Dictionary

To make the required schema changes:

1. Start the Data Dictionary Editor.
2. Choose *Save To File* from the *File* menu.
3. Type **log.new** and press the *OK* button.
4. Edit the *log.new* file (it will be stored in the DD Editor directory) and make the changes listed in the *log.sch* file.
5. Choose *Apply Changes* from the *Actions* menu.
6. Select the *log.new* file.
7. When asked, press *Continue*.
8. Review the results presented. If any errors, fix them and repeat steps 5-7.
9. If there are no errors, press the *Apply Changes* button.
10. On the next form press the *Upgrade* button.
11. When the upgrade completes, press *Done*.
12. Exit the Data Dictionary Editor

Import Data Files

There are several data files that must be imported into the database for the **FIFO Costing Module** to function properly. Edit the files as needed (and described earlier in this document). The data file is imported with the following command:

```
<path>\dataex –user_name <user>
            -password <pass>
            -db_server <serv>
            -db_name <db>
            -imp <file>
```

Where:

<path>	Is the path to the dataex program
<user>	Is the system administrator user
<pass>	Is the system administrator password
<serv>	Is the database server name

<db>	Is the database name
<file>	Is the form file (.dat extension)

For each file that is imported, dataex should report 0 errors and 0 warnings.

1. select.dat
2. selfifo.dat
3. fifo_source.dat

Note: You should edit the configuration file to set your desired options before importing it into the database.

Note: The slash between <*path*> and *dataex* should be a forward slash for UNIX systems.

Note: If the <*file*> is not in the current directory, it must be preceded by the path to the directory it is in.

Note: Each file should import with 0 errors and 0 warnings. If there are any errors or warnings, the dataex.mes file should be investigated for the reason.

Compiling the Stored Procedures

There are some SQL files that must be compiled for this module. They are:

- build_id.sql
- numsch.sql
- log_help.sql
- check_trans.sql
- trans.sql

Note: You must compile the files listed above first, and in the order shown. Otherwise, when you compile the latter files you will see errors (of routines that do not yet exist).

Sybase and SQL Server Database

To compile the stored procedures for the module, you must place the stored procedures from sql directory on a machine that has the *isql* program available.

To compile the changes:

1. Edit each of the stored procedure files. Change the line that reads "use <db>" (near the top of the file). Replace the <db> with the name of your database. Save the changes.
2. Compile the changes with the command:

```
isql -Usa -P<pass> -S<serv> < <file>
```

where:

<pass>	Is sa's password
<serv>	Is the name of the database server
<file>	Is the name of the stored procedure file

If you see a series of numbers as output, the stored procedure has compiled properly. If there are any error messages, you must fix whatever is wrong and recompile.

An example of a successful run is as follows:

```
1>  2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15> 16> 17> 18> 19> 20> 21> 22> 23> 24>
    25> 26> 27> 28> 29> 30> 31> 32> 33> 34> 35> 36> 1> 2> 3> 4> 5> 6> 1> 2> 3> 4> 5> 6> 7>
    8> 9> 10> 11> 12> 13> 14> 15> 16> 1> 2> 3> 4> 5> 6> 7> 1> 2>
```

Oracle Database

To compile the stored procedures for the toolkit, you must place the two stored procedures (from the sql directory) on a machine that has the *sqlplus* program available.

To compile the changes:

1. Compile the changes with the command:

   sqlplus sa/<pass>@<sid> @<file>

where:

<pass>	Is sa's password
<sid>	Is the name of the database SID
<file>	Is the name of the stored procedure file

If you see a series of statements listing successful compilations and creations, the stored procedure has compiled properly. If there are any error messages, you must fix whatever is wrong and recompile.

An example of a successful run is as follows:

```
SQL*Plus: Release 3.3.2.0.2 - Production on Tue Dec 09 20:21:33 1997

Copyright (c) Oracle Corporation 1979, 1994.  All rights reserved.


Connected to:
Oracle7 Workgroup Server Release 7.3.2.3.1 - Production Release
With the distributed option
PL/SQL Release 2.3.2.3.0 - Production


Procedure created.

No errors.

Procedure created.

No errors.

Grant succeeded.


PL/SQL procedure successfully completed.


Synonym created.


Grant succeeded.


PL/SQL procedure successfully completed.


Synonym created.

Disconnected from Oracle7 Workgroup Server Release 7.3.2.3.1 - Production Release
With the distributed option
PL/SQL Release 2.3.2.3.0 - Production
```

Importing the User Interface Forms

The forms provided with this module must be imported into the database with the data exchange tool. These form files are imported with the following command:

```
<path>\dataex -user_name <user>
               -password <pass>
               -db_server <serv>
               -db_name <db>
               -imp <file>
```

Where:

<path>	Is the path to the dataex program
<user>	Is the system administrator user
<pass>	Is the system administrator password
<serv>	Is the database server name
<db>	Is the database name
<file>	Is the form file (.dat extension)

Note: The slash between *<path>* and *dataex* should be a forward slash for UNIX systems.

Note: If the *<file>* is not in the current directory, it must be preceded by the path to the directory it is in.

Note: Each file should import with 0 errors and 0 warnings. If there are any errors or warnings, the dataex.mes file should be investigated for the reason.

Note: You may not need to import all of the forms. See the Implementation Overview section above for more information.

The forms that should be imported with data exchange for this module are:

- 1970.dat
- 1971.dat

One other form that you may wish to add for this module is #530 – the Part Transfer Form. If you do want to have the GUI part transfers work with the provided routine, you must save a new user version of this form. The new form 530 is not provided in data exchange format because there are not significant changes in the form, only in the Clear Basic code. You must save a new user version of this form in the database with a user version of *log1.0.* To do this, perform the following steps:

1. Start the User Interface Editor.
2. Choose *Forms…* from the *Select* menu.
3. Change the first dropdown list from *Title* to *ID*.
4. Type “530” in the text box and press the *List* button.
5. Select the form number and press the *Open* button.
6. Choose *Save As…* from the *File* menu.
7. Type **log1.0** and press *Save*.
8. Exit the User Interface Editor.

The forms that should have new user versions saved are:

- 530 – Only if you want to have the GUI part transfer form work with FIFO costing

After all of the form files have been imported successfully, the new forms must be added to the proper resource configuration(s) in UI Editor. You must decide which user or groups of users should see the new versions of the imported forms.

To add the forms to the resource configuration(s):

1. Start the User Interface Editor.
2. Choose *Resource Configs…* from the *Select* menu.
3. Press the *List* button.
4. Select the resource configuration you want to modify and press the *Open* button.
5. Type **log1.0** in the text box to the right of the starts with dropdown list and press the *List* button.
6. Select the forms you wish to add to the resource configuration from the left grid control, and press the *Copy>>* button.
7. If any confirmation dialogs appear, press the *OK* button to continue.
8. Press the *Replace* button. After the save is completed, press the *Done* button.
9. Repeat steps 4-8 for any other resource configurations you wish to modify.
10. Exit the User Interface Editor.

Compiling the ClearBasic Code

You will now compile the supplied ClearBasic code against the forms you have imported.

When Clarify starts up, it runs the routine 'initialize_app' (if one exists). Only one routine with this name will be run on client start-up. The module <module> contains an 'initialize_app' routine for this application. If some other custom application has a version of this routine, only one of these versions will execute. It is REQUIRED that you merge any existing version of 'initialize_app' into the version in module <module>. Then, replace the original version of 'initialize_app' with the merged version and recompile it. This will prevent the original version from overwriting the merged version if you recompile your original code, and it will ensure that all your custom applications will still be executed.

Not doing this as specified may cause this application or previous customizations to function improperly.

To compile the code modules, make sure that the code modules (*.cbs) and the directives files are located in the same directory. Enter the following command to compile the code:

```
<path>\cbex -dir <dir_file> \
            -batch      \
            -overwrite
```

Where:

<path>	is the path to the cbex program.
<dir_file>	is the name of the directives file

The directives files that must be compiled for this module are:

- fifo.dir

Note: You should edit the directives file before you use it to compile code. You may not need to compile all of the source modules for your implementation. To not compile a file, place a single quote (under the double quote on most keyboards) at the beginning of the line.

When the compilation is completed, you should look at the *cbex.log* file to check on the status of the compilation. There should be two rows for each file compiled that start with the word **SUCCESS**. If there are any failures (there shouldn't be), check the above instructions to make sure that your compilation environment is set up correctly.

Note: The fifo_glob.cbs file contains code to add new menu items to Clarify. If you already have a globals module that contains an *App_Initialize* method, you must add the code from fifo_glob.cbs to that method and NOT compile fifo_glob.cbs. This is due to the fact that Clarify may only have one application initialization method.

Note: If you are not using FIFO costing on the part transfer form, you should comment the line that compiles the 530.cbs module.

Note: The ClearBasic compiler requires that the user who performs the compilation be a member of a resource configuration that includes the forms compiled against. Example: If user *sa* compiles the CB code, he must be a member of a resource configuration that holds the new forms.

Note: The slash between <*path*> and *cbex* should be a forward slash for UNIX systems.

After you have finished compiling the code, you are ready to use the module.

Note: Any users who are currently running on the system should either stop Clarify and start it up again, or choose the *Update Desktop* menu item on the *Desktop* menu.

Limitations

There are no known issues with the **FIFO Costing Module**.

Performance

There are no known performance issues with the **FIFO Costing Module.**

Year 2000

The **FIFO Costing Module** was designed to be Year 2000 compliant. All date/time comparisons are performed using database-supplied routines. All date/time storage is in the Clarify database using date/time fields. No Year 2000 issues are expected.

How to Contact Us

For more information about other First Choice Software, Inc. products, or if you have any questions about the **FIFO Costing Module** product, please contact us at:

First Choice Software, Inc.
8900 Business Park Drive
Austin TX 78759
(512) 418-2905
support@fchoice.com
www.fchoice.com

Trademarks

The following are trademarks of First Choice Software, Inc. and may not be used without the express written consent of First Choice Software, Inc.

- Workflow Series
- Administration Series
- Productivity Series
- Developer Series
- Interface Series
- Select Sampler
- Commitment Plus
- Attachments Anywhere
- Zipcode Reverse Directory
- Queue Groups
- Contact/Site Merge
- Survey Taker
- Clear Basic API Toolkit For ClearSupport
- Clear Basic API Toolkit For ClearQuality
- Clear Basic API Toolkit For ClearLogistics

API Reference

The next section of this document lists the available APIs in the **FIFO Costing Module**. Each API is documented with sections for:

- File Included In
- Syntax
- Description
- Parameters
- Return Values
- Examples

part_transfer (CB)
part_transfer_br (CB)

Included in

trans.cbs

Standard API

```
Public Function part_transfer(part_num As String, _
                              mod_level As String, _
                              domain As String, _
                              quantity As Long, _
                              serial_num As String, _
                              from_loc As String, _
                              from_bin As String, _
                              from_good As Boolean, _
                              to_loc As String, _
                              to_bin As String, _
                              to_good As Boolean, _
                              user_name As String, _
                              trans_date As String, _
                              ref_id As String, _
                              notes As String, _
                              gen_time_bomb As Boolean, _
                              fifo_flag As Integer, _
                              update_cost As String, _
                              update_source As String, _
                              trans_id As String, _
                              std_cost As String) As Integer
```

Business Rule API

```
Public Sub part_transfer(part_num As String, _
                         mod_level As String, _
                         domain As String, _
                         quantity As Long, _
                         serial_num As String, _
                         from_loc As String, _
                         from_bin As String, _
                         from_good As String, _
                         to_loc As String, _
                         to_bin As String, _
                         to_good As String, _
                         user_name As String, _
                         trans_date As String, _
                         ref_id As String, _
                         notes As String, _
                         gen_time_bomb As String, _
                         fifo_flag As Integer, _
                         update_cost As String, _
                         update_source As String, _
                         trans_id As String, _
                         std_cost As String)
```

Description

These are the Clear Basic versions of the part transfer APIs. These APIs emulate the Clarify part transfer mechanism. The part/revision/domain are specified (the part to be transferred). Also specified are the from location/bin and the to location/bin. The quantity and serial number of the parts to be transferred are specified, as are some additional information (reference ID, transfer date, and notes).

As in Clarify, the transfer can be from good or bad stock and can be to good or bad stock. Also, time bombs (for escalation/notification) can be specified, or not.

The FIFO flag has three possible values:

- 0 – No FIFO costing used
- 1 – FIFO costing is used
- 2 – Use the cost specified in the update_cost field

The update cost field is the cost per unit of the transfer, and is used in two cases: 1) if the preceding field is set to 2, this is the cost of the transfer. 2) If the flag is set = 1, then this is the cost of the transfer, and FIFO records will be created. This is only valid for transfers from Expense GL accounts.

The update source is also used only when the flag = 1 and it is a transfer in from an expense GL. In this case, the update source is the source placed in the FIFO costing record.

Note: If you are transferring (in from an expense GL) using FIFO costs, and don't wish to create a new FIFO record (but rather wish to increment the most recent FIFO record), just leave the update_cost field blank ("").

The final two arguments are output arguments that return the transaction ID of the part transfer, and the standard cost (total) for the transfer.

The APIs also check part authorization levels for auto-replenishment, and will automatically work with that engine.

Adding to an existing part transfer:

The part transfer APIs now have the ability to increase the number of units in a previously transacted part transfer. For example, suppose that a part transfer of 5 units has been made. The transaction ID is '42'. But you realize after the fact that the transfer really was for 8 units. You can create a new part transfer, or, using this feature, you can "fix" the previous part transfer.

To do this, specify (in the trans_id argument), the transaction ID of the previous part transfer. In the quantity field, specify how many units to add to the transfer. The fifo_flag, update_cost, update_source, and gen_time_bomb parameters are also used for the additional units. All other arguments are ignored. The from/to locations and bins, as well as the part number/mod_level are determined from the existing part transfer.

Note: This new feature only works with transfers of quantity tracked parts.

Parameters

Parameter Name	Required?	Description
part_num	Yes	The part number transferred
mod_level	Yes	Revision of the part
domain	Yes	The domain of the part

quantity	Yes	How many to transfer
serial_num	No	The serial number (if serial tracked) to transfer
from_loc	Yes	The inventory location (or GL account) to transfer from
from_bin	Yes	The bin to transfer from (if from location)
from_good	Yes	Is the transfer from "good" stock?
to_loc	Yes	The inventory location (or GL account) to transfer to
to_bin	Yes	The bin to transfer to (if to location)
to_good	Yes	Is the transfer to "good" stock?
user_name	No	Who performed the transfer. If blank, current user is used
trans_date	No	When was transfer performed? If blank, current time is used
ref_id	No	Optional reference ID of transfer
notes	No	Optional notes about the transfer
gen_time_bombs	Yes	Should time bombs be generated?
fifo_flag	Yes	Should we use FIFO costing? 0 = No, 1 = Yes, 2 = Use the cost in the next field.
update_cost	No	If the fifo_flag = 2, this is the per-unit cost for the transfer. Not commonly used. If the fifo_flag = 1, it is the FIFO cost of the transfer (only valid for transfer in from expense GL).
update_source	No	If the fifo_flag = 1, and this is a transfer in from an expense GL with an update cost specified, this is the source of the FIFO parts.
trans_id	Yes	Output argument with the transaction ID of the part transfer
std_cost	Yes	Output argument with the standard cost of the part transfer

Returns

Value	Meaning
0	No errors
-1	To location/bin and from location/bin cannot be the same
-2	Cannot transfer quantities < 1
-3	Cannot transfer quantities > 1 if a serial number is specified
-4	Cannot locate the specified part number
-5	Part is serial-tracked, but no serial number was specified
-6	No part revision record found for specified revision
-7	Cannot find the "from" location
-8	Cannot find the "from" bin
-9	Cannot find the "to" location
-10	Cannot find the "to" bin
-11	Cannot find the specified user
-20	Serial number is found in inventory, but not at the "from" location/bin
-21	Cannot make inventory go negative by transferring too many parts from a location/bin
-22	Cannot find the activity string for "Transfer"
-23	Supplied update cost is not numeric
-24	There are not enough FIFO units available to perform the transfer
-25	For transfers into inventory (from expense GL), there are no FIFO records available to increment the count
-40	Invalid value for fifo_flag specified.
-41	You are attempting to add units to an existing part transfer, but the part transfer is for a serialized part
-42	You are attempting to add units to an existing part transfer, but the supplied transaction ID is not found

Examples

- Transfer 20 units of Accounting 101 from an expense GL account to “Bin 1” in Austin. All stock is good. Use FIFO costing.

```
Dim ret_int   As Integer
Dim trans_id  As String
Dim std_cost  As String

ret_int = part_transfer(“Accounting”, “101”, “QuanityDomain”, 20, _
          “”, “EXPGL”, “”, True, “Austin”, “Bin 1”, True, _
          “”, “”, “”, “”, True, 1, “”, “”, trans_id, std_cost)
```

- Transfer MS Word 7.0, serial number 555666 from Austin Bin 2 to San Jose Bin “Receiving”. The inventory was good, but is now marked as bad (for Quality Control). Fred did the transfer on July 30th at 5:13 AM. Add notes and a ref ID, and don’t generate a time bomb. Don’t use FIFO costing.

```
Dim ret_int   As Integer
Dim trans_id  As String
Dim std_cost  As String

ret_int = part_transfer(“MS Word”, “7.0”, “Product”, 1, “555666”, _
                        “Austin”, “Bin 1”, True, “San Jose”, _
                        “Receiving”, False, “fred”, _
                        “7/30/98 5:13:00”, “Ref ID3”, _
                        “Some notes”, False, 0, “”, “”, trans_id, _
                        std_cost)
```

- Transfer 15 units of the Notebook part from expense GL ‘EXPGL’ to Austin Bin 2. The inventory was good, but is now marked as bad (for Quality Control). Fred did the transfer on July 30th at 5:13 AM. Add notes and a ref ID, and don’t generate a time bomb. Use FIFO costing, and specify the cost and source.

```
Dim ret_int   As Integer
Dim trans_id  As String
Dim std_cost  As String

ret_int = part_transfer(“Notebook”, “”, “Product”, 15, “”, _
                        “EXPGL”, “”, True, “Austin”, _
                        “Bin 2”, False, “fred”, _
                        “7/30/98 5:13:00”, “Ref ID3”, _
                        “Some notes”, False, 1, “.34”, “MySource”,
                        trans_id, std_cost)
```

- Transfer a part via a business rule.

```
cbbatch –f trans.cbs –r part_transfer_br –as “MS Word” –as “7.0”
        -as “Product” -al 1 –as “555666” –as “Austin” –as “Bin 1”
        -as “True” –as “San Jose” –as “Receiving” –as “False”
        -as “fred” –as “7/30/98 5:13:00” –as “Ref ID3”
        -as “Some notes” –as “False” –al 1 –as “” –as “” –as “” –as
         “”
```

- Add 4 units to part transfer ‘42’. Generate a time bomb, and use FIFO costing.

```
Dim ret_int   As Integer
Dim trans_id  As String
Dim std_cost  As String
```

```
trans_id = “42”
ret_int = part_transfer(“”, “”, “”, 4, “”, _
                        “”, “”, True, “”, _
                        “”, True, “”, _
                        “”, “”, _
                        “”, True, 1, “”, “”,
                        trans_id, std_cost)
```

Part Transfer (SP)

SQL Server/Sybase

Included In

trans.sql

Syntax

```
CREATE PROCEDURE part_transfer
             @from_loc VARCHAR(20),
             @from_bin VARCHAR(20),
             @from_good INT,
             @to_loc VARCHAR(20),
             @to_bin VARCHAR(20),
             @to_good INT,
             @part_num VARCHAR(30),
             @mod_level VARCHAR(10),
             @domain VARCHAR(40),
             @quantity INT,
             @serial_num VARCHAR(40),
             @user_name VARCHAR(30),
             @ref_id VARCHAR(20),
             @notes VARCHAR(60),
             @trans_date VARCHAR(30),
             @wo_num VARCHAR(80), /* Not currently used. */
             @use_fifo INT,
             @update_cost VARCHAR(20),
             @gen_tb INT,
             @ret_val INT OUTPUT AS
```

Oracle

Included In

transor.sql

Syntax

```
CREATE OR REPLACE PROCEDURE part_transfer(
                            from_loc IN VARCHAR2,
                            from_bin IN VARCHAR2,
                            from_good IN NUMBER,
                            to_loc IN VARCHAR2,
                            to_bin IN VARCHAR2,
                            to_good IN NUMBER,
                            part_num IN VARCHAR2,
                            mod_levell IN VARCHAR2,
                            domain IN VARCHAR2,
                            quantity IN NUMBER,
                            serial_num IN VARCHAR2,
                            user_name IN VARCHAR2,
                            ref_id IN VARCHAR2,
                            notes IN VARCHAR2,
                            trans_date IN VARCHAR2,
                            wo_num IN VARCHAR2, /* Not currently used. */
                            use_fifo IN NUMBER,
                            update_cost IN VARCHAR2,
                            gen_tb IN NUMBER,
                            ret_val IN OUT NUMBER)
```

Description

This API allows the user to transfer goods into a warehouse, out of a warehouse, or among bins in one or more warehouses. The API emulates the part transfer form in ClearLogistics (and if you are using FIFO Costing, you should call on this API from that form instead of the default actions).

The API allows the user to specify the from location (or expense GL), the to location (or expense GL). If the parts are good or bad (for both from and to). The user also specifies that part number/revision/domain (to identify the part), as well as the quantity transferred (and the serial number if it is a serialized part).

The work order number is always left blank (it is only used by the First Choice Software **Work Order Module**). The reference ID and notes are text that can be recorded, just like the Clarify GUI. The trans_date allows the date for the transfer to be set.

The Update FIFO flag has three possible values:

- 0 – No FIFO costing used
- 1 – FIFO costing is used
- 2 – Use the cost specified in the update_cost field

The update cost field is the cost per unit of the transfer, and is only used if the preceding field is set to 2. The final input argument allows time bombs to be generated or not. Time bombs are the mechanism that business rules use to know that an operation has occurred.

The final argument is a return code.

Clarify system to receive units against a part request. The part request may be opened in many different manners, including via the Clarify GUI, via the First Choice **Purchase Order Module**, or via the First Choice **Clear Call Center Integration Module**.

Parameters

Parameter Name	Required?	Description
from_loc	Yes	The inventory location (or expense GL) to receive from
from_bin	Yes	The bin (if the receive is from an inventory location) to receive from. If an expense GL is used, this should be empty
from_good	Yes	Is the inventory received from a good source? 1 = Yes, 0 = No
to_loc	Yes	Where is the inventory received to. This is an inv location
to_bin	Yes	The bin to receive to. If inventory bins are not used, this may be left blank
to_good	Yes	Is the inventory received to a good dest? 1 = Yes, 0 = No
part_num	Yes	The part number to transfer
mod_level	Yes	The revision to transfer (blank if you do not use revisions)
domain	Yes	The domain of the part number
quantity	Yes	The number of units transferred
serial_num	No	Serial number transferred (if serialized)

user_name	No	The user performing the transfer. If this is left blank, the API looks up the user name stored in the configuration item Default Logistics User. If this is not found, a hard-coded user name is used.
ref_id	No	An optional reference ID
notes	No	Optional notes
trans_date	No	Date/time of transfer. If blank, current date/time is used
wo_num	No	Leave this blank
use_fifo	Yes	0 = No FIFO costing. 1 = FIFO Costing. 2 = Use next argument for the unit cost
update_cost	No	Unit cost for the transfer
gen_tb	Yes	1 = Generate time bomb. 0 = Do not generate time bomb
ret_val	Yes	This is a return argument containing the error/warning return code.

Returns

Value	Meaning
0	No errors
-1	Cannot transfer from a location to itself
-2	Cannot transfer < 1 unit
-3	Cannot have quantity > 1 if it is serialized
-4	Cannot locate specified part number
-5	Part is serialized and no serial number supplied
-6	Cannot find the part revision
-7	“From” location is not valid (or is an expense GL)
-8	“From” bin is not valid
-9	“To” location is not valid (or is an expense GL)
-10	“To” bin is not valid
-11	The specified user is not valid
-120	A serialized part already exists, but is not in the specified “From” location
-121	The transfer would cause the “From” location/bin to have a negative quantity
-122	Cannot find the transfer activity string
-123	Cannot build the transaction string
-124	There are not enough FIFO records in the database to perform the transfer

Examples

- Transfer 5 units of MS Word 6.0 from Bin 1 to Shipping in the ‘Austin’ warehouse. Use FIFO costing.

```
DECLARE @t_str   VARCHAR(10),
        @ret_val INT

EXEC part_transfer ‘Austin’, ‘Bin 1’, 1, 'Austin', ‘Shipping, 1,
                   ‘MS Word’, ‘6.0’, ‘Quantity’, 5, ‘’, ‘’, ‘’,
                   ‘’, ‘’, ‘’, 1, ‘’, 1, @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
```

- Move 20 units in a bin of a part from good inventory to bad inventory. Do not use FIFO costing, and do not generate time bombs. Mark the transfer as performed by ‘Joe’, on August 1.

```
DECLARE @t_str   VARCHAR(10),
        @ret_val INT
```

```
EXEC part_transfer 'Austin', 'Bin 1', 1, 'Austin', 'Bin 1, 0,
                   'MS Word', '6.0', 'Quantity', 20, '', 'Joe', '',
                   '', '8/1/98', '', 0, '', 0, @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
```

- Move a unit with serial number '1234' from an Expense GL to an inventory location. Put in a reference ID and Some notes. Do not use FIFO costing, and do not generate time bombs. Mark the transfer as performed by 'Joe', on August 1.

```
DECLARE @t_str   VARCHAR(10),
        @ret_val INT

EXEC part_transfer 'EXPGL, '', 1, 'Austin', 'Bin 1, 1,
                   'Serial Part', '1.0', 'Product', 1, '1234',
                   'Joe', 'Ref ID', 'Notes', '8/1/98', '',
                   0, '', 0, @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
```

Give Back Units on a Part Transfer

Included In

give_back.cbs

Standard API

```
Public Function give_back(trans_id As String, _
                          quantity As Long, _
                          std_cost As String) As Integer
```

Business Rule API

```
Public Sub give_back_br(trans_id As String, _
                        quantity As Long)
```

Description

These APIs allow the user to “give_back” units on a part transfer. Suppose, for example, that a part transfer was made on 100 units, and it was then realized that only 80 units were needed. A new part transfer could be made for those 20 units, but there are places in Clarify where that is not practical, because there is only one field available to hold the part transfer.

These routines patch up the part transfer as if it had been made on the smaller number of units. It will fix the part_trans record, as well as adjusting inventory, and FIFO costs (if needed).

Parameters

Parameter Name	Required?	Description
trans_id	Yes	The part transaction to give back units for
quantity	Yes	The number of units to give back
std_cost	Yes	An output argument with the new standard cost for the part transaction

Returns

Value	Meaning
0	No errors
-1	Invalid number of units supplied
-2	Cannot find the specified part transaction
-3	The give back quantity is larger than the quantity remaining on the part transaction
-4	Not enough FIFO records (at the to location) to give back
-5	The give back would drive the quantity at the to bin to a negative quantity

Examples

- Give back four units on part transaction 42.

```
Dim ret_int   As Integer
Dim std_cost  As String

ret_int = give_back(“42”, 4, std_cost)
```

- Give back the same via a business rule.

```
cbbatch –f give_back.cbs –r give_back_br –as “42” –al 4
```

Build an ID Number

Oracle

Included In

build_idor.sql

Syntax

```
CREATE OR REPLACE PROCEDURE build_id_number (the_seq IN VARCHAR2,
                            auto_num OUT VARCHAR2,
                            ret_int OUT NUMBER)
```

SQL Server/Sybase

Included In

build_id.sql

Syntax

```
CREATE PROCEDURE build_id_number @the_seq VARCHAR(30),
                 @auto_num VARCHAR(80) OUTPUT,
                 @ret_int INT OUTPUT AS
```

Description

This API allows the user to build a new ID number. It assumes that Clarify already has an autonumbering sequence for the desired entity. The API generates all facets of the autonumber, including counters, dates, and padding – just like the Clarify GUI. It insures that no two rows can have the same autonumber, and returns the generated ID number.

Parameters

Parameter Name	Required?	Description
the_seq	Yes	A valid autonumber sequence
auto_num	Yes	The returned ID number
ret_int	Yes	This is a return argument containing the error/warning return code.

Returns

Value	Meaning
0	No errors

Examples

- Generate a new container number

```
DECLARE @t_str   VARCHAR(10),
        @ret_val INT,
        @id_num VARCHAR(80)

EXEC build_id_number ‘Container’, @id_num, @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
PRINT @id_num
```