ClearLogistics Enhancements

Purchase Order Module 1.0

Installation and Users Guide

First Choice Software, Inc.
4412 Spicewood Springs Road, Suite 701
Austin, Texas 78759

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

Table of Contents

First Choice Software, Inc.
4412 Spicewood Springs Road, Suite 701
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. **Purchase Order Module** allows your administrator to enter purchase orders for goods that your company has ordered from vendors. The interface validates all data, and creates *Replenishment* part requests. The final part of the **Purchase Order Module** is a receive API (stored procedure) that can be called from either Clarify or an RF device that allows the purchase order lines to be received. They are closed automatically when fully received.

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

Creating Purchase Orders

This module provides an interface to create purchase orders. It is a batch procedure that can be called from the Clarify GUI, run from a command line, or batched at a set time each day.

The file *purorder.cbs* contains the purchase order interface. When it is executed, it will create the proper part request headers and details in the Clarify system to represent the purchase orders.

Configuring the Interface

The interface is very configurable. An options file, *init_purorder.dat*, is provided to allow you to set certain fields for each of the purchase orders created by the system. This file is optional. If you do not use it, the interface will use predefined defaults.

To change any of the values in the initialization file, use an ASCII text editor, and change any of the values after the colons. Do not change the labels before the colons. If you do not wish to specify a value, then leave the space after the colon blank. If any one line is left blank, the interface will calculate a default value and use it.

A sample init_purorder.dat file is as follows:

```
USER: sa
PAYMENT_TERMS: Net 15
PRIORITY: Priority 1
STATUS: Awaiting Confirmation
PAYMENT_METHOD: Check
SHIP_VIA: Ground
REPAIR_TYPE: Upgrade Only
REPAIR_STATUS: In QA
FAILURE_CODE: 01 Internal Defect
NOTES: Part request automatically generated by the purchase order
interface.
QUEUE:
CREATE_DATE:
DUE_DATE:
```

The values that can be set are:

- User — This is the user who creates the part requests
- Payment Terms — This is the value to be placed in the payment terms pulldown list

- Priority — Priority of the part request
- Status — The default status of the part request details
- Payment Method — Value for the dropdown code list
- Ship Via — Shipment method dropdown value
- Repair Type — Value for dropdown code list (not usually used for part requests)
- Repair Status — Value for dropdown code list (not usually used for part requests)
- Failure Code — Value for dropdown code list (not usually used for part requests)
- Notes — Optional notes to add to each part request
- Queue — Optional name of a queue to dispatch to. If not supplied, the part requests are not dispatched. If specified, they will be dispatched to the named queue.
- Create Date — When the part requests are created. If left blank, current time is used
- Due Date — When the part requests are expected.

In addition, the names of the files used for the interface can be set. For example, the default name of the input file is *purorder.in*. These file names can be changed by editing the *purorder.cbs* file, and changing the names, which can be found in *the init_vars* method.

Interface Input

The input for the interface is an ASCII file. This file can be created in many ways: Output from a SQL script, external interface program, or ASCII file editing. The name of the input file is *purorder.in*, but that can be modified in the interface program. The format of the interface is ASCII, with one purchase order line item per line of the file. Each line is a pipe-delimited list of the required values for each line. The values required for each line are:

- Purchase Order number
- Purchase Order line item (from 1-n for each purchase order)
- The Clarify part number ordered
- The Clarify mod_level (revision) ordered
- The name of the Clarify inventory location (warehouse) to ship to
- The number of items ordered
- The unit price of each item

All of the above fields are mandatory for each line. However, the revision may be left blank (if revisions are not used for that part number in Clarify), and the unit price can be set to 0 if FIFO costing is not being used.

A sample input file is as follows

```
1237|1|MSWord|6.0|MainWarehouse|5|4.5
1237|2|MSWord|7.0|MainWarehouse|10|.32
1238|1|MSWord|7.0|MainWarehouse|50|54.50
```

Running the Interface

To use the interface, you must insure that the following files are all present in the current directory:

- purorder.cbs
- prheadcr.cbs
- prdetcr.cbs
- purorder.in
- init_purorder.dat (optional)
- clarify.env file (which you create)

To create a clarify.env file, use your favorite editor. A clarify.env file for the interface should look like this:

```
auto_login=TRUE
login_name=<your system admin name here>
db_password=<sa's password here>
db_server=<your DB server name here>
db_name=<your DB name here>
```

The interface can be run from the command line (or a timed execution utility such as *cron*) with the following command:

```
<path>\cbbatch -f prheadcr.cbs -f prdetcr.cbs -f purorder.cbs
               -m purorder.cbs -r order_goods
```

Where: <path> Is the path to the cbbatch program (usually the rulemanger directory of Clarify)

Interface Output

The interface generates several files each time it is run. These files will describe all work, errors, and warnings produced by the interface.

The most important file produced by the interface is called *status.log*. It contains all information about the run of the interface. A sample status file is as follows:

```
Status for Purchase Order Interface

Date of Interface: 04/29/98
Input File Name  : purorder.in

[10:25:37] Interface started.
[10:25:44] Interface completed.

Results of interface:
---------------------
Records processed: 3
Records ignored  : 0

Errors found     : 0
Warnings found   : 0
```

If any errors are found, the file *error.log* will be produced. A sample follows:

```
Errors for Purchase Order Interface
Date of Interface: 04/29/98

[11:04:24] Payment terms 'Bad Payment terms!' is not valid.
```

If any warnings are generated, the file *warning.log* will be produced. A sample follows:

```
Warnings for Purchase Order Interface
Date of Interface: 04/29/98

[11:06:10] Record 1. Part request '75-1' already exists.
             Record added to bad record log.
[11:06:10] Record 2. Part request '75-2' already exists.
             Record added to bad record log.
[11:06:10] Record 3. Part request '76-1' already exists.
             Record added to bad record log.
```

If any of the records in the input file cannot be processed (for any reason), the input line is placed in a file called *badrec.log*. This file can then be edited (to fix the problems), and then used as the input file for another run of the interface. The format of this file is exactly the same as the input file.

NOTE: If you are not using FIFO, modify the code in receive.sql, or just use default receive API (but it doesn't close the part requests automatically, nor does it account for the PO costs in the part transfer record).

receive

SQL Server/Sybase

Included In

receive.sql

Syntax

```
CREATE PROCEDURE receive
                 @dtl_num VARCHAR(20),
                 @from_loc VARCHAR(20),
                 @from_bin VARCHAR(20),
                 @from_good INT,
                 @to_loc VARCHAR(20),
                 @to_bin VARCHAR(20),
                 @to_good INT,
                 @quantity INT,
                 @serial_num VARCHAR(30),
                 @status VARCHAR(80),
                 @carrier VARCHAR(40),
                 @waybill VARCHAR(40),
                 @poss_damage INT,
                 @not_prop_pack INT,
                 @route_to_test INT,
                 @user_name VARCHAR(30),
                 @receive_date VARCHAR(30),
                 @auto_close INT,
                 @use_fifo INT,
                 @gen_tb INT,
                 @ret_val INT OUTPUT AS
```

Oracle

Included In

receiveor.sql

Syntax

```
CREATE OR REPLACE PROCEDURE receive(
                           dtl_num IN VARCHAR2,
                           from_loc IN VARCHAR2,
                           from_bin IN VARCHAR2,
                           from_good IN NUMBER,
                           to_loc IN VARCHAR2,
                           to_bin IN VARCHAR2,
                           to_good IN NUMBER,
```

```
quantity IN NUMBER,
serial_num IN VARCHAR2,
status IN VARCHAR2,
carrier IN VARCHAR2,
waybill IN VARCHAR2,
poss_damage IN NUMBER,
not_prop_pack IN NUMBER,
route_to_test IN NUMBER,
user_name IN VARCHAR2,
receive_date IN VARCHAR2,
auto_close IN NUMBER,
use_fifo IN NUMBER,
gen_tb IN NUMBER,
ret_val IN OUT NUMBER)
```

Description

This API allows the 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**.

The API allows any number of units to be received against the purchase order, including partial receipts, full receipts, or overages. If the part request is created via the **Purchase Order Module** (and FIFO costing is used), the part transaction and FIFO accounting will be performed. If the part request is created in some other manner, standard Clarify costing mechanisms will be used.

The API will automatically close the part request detail line item if it has been received in full. If it is the last open line item of a part request header, the header is also closed. This is controlled by the *auto_close* argument.

A from location (usually an Expense GL account) can be specified in the header. If no location is specified, the API looks for a configuration item with the name of *Receive Expense GL*, and will use that as the from location. If that configuration item is not defined, a hard-coded expense location *EXPGL* will be used.

To use FIFO costing, set the use_fifo argument to 1. To turn off FIFO costing, set it to 0.

Parameters

Parameter Name	Required?	Description
dtl_num	Yes	The part request detail number to receive against
from_loc	No	The inventory location (or expense GL) to receive from
from_bin	No	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
quantity	Yes	How many units were received?
serial_num	No	The serial number received
status	No	New status for part request in Received condition. If left blank, the default status is used
carrier	No	The carrier (site). If filled in, must be a valid site

waybill	No	Waybill number for receive
poss_damage	Yes	Was the unit possibly damaged in transit?
not_prop_pack	Yes	Was the unit not properly packed?
route_to_test	Yes	Should the unit be routed to testing location?
user_name	No	The user performing the receive. 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.
receive_date	No	When was receive done? If blank, current date/time is used
auto_close	Yes	Should the request be closed (and the header too?) if fully received? 1 = Yes, 0 = No
use_fifo	Yes	Is FIFO costing used? 1 = Yes, 0 = No
gen_tb	Yes	Should time bombs be generated for notification? 1 = Yes
ret_val	Yes	This is a return argument containing the error/warning return code.

Returns

Value	Meaning
0	No errors
+1	Receive was OK, but received an overage
-1	The quantity is not a valid number, or is < 0
-2	Cannot receive from same location/bin as the source location/bin
-3	Cannot locate the specified user
-4	Cannot find the specified part request
-5	There are no more units to receive for the request
-6	Cannot have a quantity > 1 and a serial number
-7	Cannot find the conditions for validating the status transition
-8	No transition exists from the current condition to “Received”
-9	The user may not transition from current condition to “Received”
-10	Cannot find the activity string for Receive
-11	Cannot find the specified status
-12	Cannot find the “From” inventory location
-13	Cannot find the “From” inventory bin
-14	Cannot find the “To” inventory location
-15	Cannot find the “To” inventory bin
-16	Carrier site is not found
-17	It is a quantity tracked part with a serial number specified
-18	It is a serialized part, but no serial number is specified
-19	Cannot find default status for “Closed” condition
-20	Cannot find activity string for Closed action
-21	Cannot find the conditions for validating the status transition
-22	No transition exists from the current condition to “Closed”
-23	The user may not transition from current condition to “Closed”
-24	Part request is currently dispatched to a queue
-100 - -199	Error in part transfer. Please see part transfer for more information. For example, an error of –1 in part_transfer will be reported as -101

Examples

- Receive 5 units against part request detail ‘1-1’ to the ‘Austin’ warehouse, bin ‘Receiving’. Use FIFO costing, auto-close, and generate time bombs. Receive is by joe on August 1, 1998, and various other data is set.

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

```
        @ret_val INT

EXEC receive '1-1', '', '', 1, 'Austin', 'Receiving',
             1, 5, '', 'Status', 'Site 42', 'Waybill 32',
             1, 1, 1, 'joe', '8/1/98', 1, 1, 1,
             @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
```

- Receive 1 unit (serial number 1234) against part request detail '1-2' to the 'Austin' warehouse, bin 'Bin 1'. Nothing else is set.

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

EXEC receive '1-2', '', '', 1, 'Austin', 'Bin 1',
             1, 1, '1234', '', '', '',
             0, 0, 0, '', '', 0, 0, 0,
             @ret_val out
SELECT @t_str = CONVERT(CHAR, @ret_val)
PRINT @t_str
```

Configuration Items

There are two configuration items that can be set that will affect the behavior of the receive API. To modify these items, edit the *rcv_config.dat* file with an ASCII editor. Then, for each item, edit the *str_value* field for each item. Finally, import the file as described in the implementation section below.

Default Logistics User

This item controls the user name that is used to log receive operations. If the user name is specified in calls to the receive API, then that user name is used. If that user name is left blank, then the system searches for this configuration item, and uses that name. If this configuration item is not specified, then the API uses a hard-coded user name.

Receive Expense GL

When the receive API is called there are arguments that can be used to specify the "from" account or GL account. If those values are specified, then they are used. If, however, they are left blank, the API searches for this configuration item, and uses the Expense GL account specified. If the configuration item is not added to the system then a hard-coded expense GL is used.

Implementation

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

Requirements

This version of the **Purchase Order Module** requires the following:
Clarify Version: 4.5 or later

Clarify Tools: Data Dictionary Editor (ddeditor)
Clear Basic Batch (cbbatch)
Data Exchange (dataex)

Other Tools: Valid database system (ex: MS SQL Server version 6.5)

Packaging

Purchase Order Module is shipped to you as a zip file.
Example: purorder.1.0.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. Unzip into "c:\clarify\fchoice\logistics" directory.

The following files are provided with this product:

File Name	Purpose
porder	Directory containing the purchase order module code
schema	Directory containing the data model changes for this module
sql_server	Directory containing the stored procedures for SQL Server and Sybase
oracle	Directory containing the stored procedures for Oracle

The *porder* directory contains the following entries:

File Name	Purpose
purorder_user.pdf	This document
purorder.cbs	Clear Basic code for the purchase order interface
prheadcr.cbs	Part request header creation API
prdetcr.cbs	Part request detail creation API
purorder.in	Sample input file for the interface
init_purorder.dat	Sample options file for the interface
rcv_config.dat	Configuration options for the module

The *schema* directory contains the following entries:

File Name	Purpose
log.sch	Modifications to make to the Clarify schema for all of the logistics modules

The *sql_server* directory contains the following entries:

File Name	Purpose
receive.sql	The receive API
trans.sql	The part transfer (helper) API
numsch.sql	Generates the next number for an auto-numbering scheme.

The *oracle* directory contains the following entries:

File Name	Purpose
receiveor.sql	The receive API
transor.sql	The part transfer (helper) API
numschor.sql	Generates the next number for an auto-numbering scheme.

Manual Installation

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

The **Purchase Order Module** files may be installed on any machine that can execute the Clarify data exchange tool (dataex), the database command line tool (such as isql or sqlplus), and the Clarify Clear Basic Batch tool (cbbatch). You will also need to run the Clarify Data Dictionary Editor (ddeditor).

Data Dictionary Modifications

The **Purchase Order 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

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 is a data file that must be imported into the database for the Purchase Order Module to function properly. 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. Import the rcv_config.dat file into the database.

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:

- numsch.sql
- receive.sql
- trans.sql

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
```

Limitations

There is only one known issue with the **Purchase Order Module**.

The current part transfer routine is for quantity only parts. If serialized parts are required, then please contact First Choice for more information.

Performance

There are no known performance issues with the **Purchase Order Module.**

Year 2000

The **Purchase Order 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.

Related Products

First Choice Software, Inc. produces many add-on products to make your Clarify experience more productive and efficient.

First Choice offers several families of products:

Workflow Series

- **Select Sampler** – The **Select Sampler** enhances the capabilities of several Clarify select forms to provide additional selection criteria that are commonly desired. These augmented forms are the Dispatch form, the Assign form, the Select Employees form and the Select Queues form.

- **Commitment Plus** – Allows commitments to be personal or case/subcase based. Control all of your commitments from an improved *My Commitments* form. Fulfill, open, and close commitments from one convenient form.

Administration Series

- **Queue Groups** – Define groups of queues and associate them with users and workgroups. When a user is created or modified, the queues for the user can be automatically assigned to the user.

- **Contact Merge** – This tool allows you to locate and mark contact aliases for a single (master) contact. It then will move all related data (cases, logs…) from the alias contact(s) to the master contact, and will obsolete the alias.

Productivity Series

- **Survey Taker** - This product allows you to define survey questions for your customers. Your employees can then easily perform the on-line survey, gathering useful information for you. Options available for question types, question patterns, defaults, and more. Surveys can be associated with sites, contacts, business organizations, or cases.

Developer Series

- **Clear Basic API Toolkit For ClearSupport –** This toolkit is the essential aid for anyone who wants to extend their Clarify system. Over 40 methods provided , including:

CASE CREATION

SUBCASE CREATION

CASE CLOSURE

Subcase closure

STATUS CHANGE

DISPATCH

ACCEPT/REJECT/FORWARD

YANK

Assign

Move

Log phone note
Log note
Log research note
Log commitment
Fulfill commitment
Initial response

AND MORE!

These methods are fully functioned: they add activity logs and escalation records, just like the Clarify GUI. They can be placed in your ClearBasic code (saving you weeks of effort), and they can be called directly from the Clarify business rule engine – turning your business rules into super business rules.

First Choice Software, Inc.
4412 Spicewood Springs Road, Suite 701
Austin, Texas 78759

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

ClearBasic API Toolkit For ClearQuality – Extend your ClearQuality implementation with this comprehensive toolkit of APIs. Over 20 methods provided , including:

CHANGE REQUEST CREATION

CHANGE REQUEST CLOSURE

STATUS CHANGE (WITH OR WITHOUT TRANSITIONS)

DISPATCH

ACCEPT/REJECT/FORWARD

YANK

Assign
Move
Log note

Duplicate Change Request
Replicate Change Request
Link/Unlink Change Request to Solution

AND MORE!

These methods are fully functioned: they add activity logs and escalation records, just like the Clarify GUI. They can be placed in your ClearBasic code (saving you weeks of effort), and they can be called directly from the Clarify business rule engine – turning your business rules into super business rules.

- **ClearBasic API Toolkit For ClearLogistics –** This toolkit is a must for anyone customizing any of the ClearLogistics family of products from Clarify. The toolkit includes methods for ClearLogistics Field Operations, Order Operations, Spares Manager, and Depot Repair. Over 40 methods provided , including:

PART REQUEST CREATION

PART REQUEST CLOSURE

STATUS CHANGE (WITH OR WITHOUT TRANSITIONS)

DISPATCH

ACCEPT/REJECT/FORWARD

YANK

Assign
Move
Pick/Fulfill/Ship/Receive
Part Transfer
Log Time & Expenses
Dispatch Field Engineer
Log Service Interruption
Create Engineering Change Order
Modify ECO
Create Depot Repair

AND MORE!

These methods are fully functioned: they add activity logs and escalation records, just like the Clarify GUI. They can be placed in your ClearBasic code (saving you weeks of effort), and they can be called directly from the Clarify business rule engine – turning your business rules into super business rules.

- **ClearLogistics Enhancements –** First Choice is proud to announce the availability of five add-on modules that extend the functionality of ClearLogistics Spares Manager. These modules add new functionality necessary for many high volume logistics operations. Each module comes with well-commented source code, documentation, and additional consulting to make the integration and implementation quick and simple. The five modules are:

  FIFO COSTING

  RF/SHIPPING SYSTEM INTEGRATION

  CLEARLOGISTICS – CLEARCALLCENTER INTEGRATION

  Purchase Order Processing

How to Contact Us

For more information about other First Choice Software, Inc. products, or if you have any questions about the **Purchase Order 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