Case Study: Error/Log/Email Messaging System

Paul Guggenheim

Paul Guggenheim & Associates

About PGA

- Working in Progress since 1984 and training Progress programmers since 1986
- Designed seven comprehensive Progress courses covering all levels of expertise including - The Keys to OpenEdge®
- Author of the Sharp Menu System, a database driven, GUI pull-down menu system.
- **White Star** Software Strategic Partner
- **ProStar** Partner and Consultant
- **AppPro** Partner
- Major consulting clients include Chicago Metal Rolled Products, Eastern Municipal Water District, Eaton Corporation, Foxwoods Casino, International Financial Data Services, Montana Metal Products, National Safety Council, Preferred Podiatry, Plymouth Tube, Stanley Engineering, Tower Automotive and Tyson Foods.
- Head of the Chicago Area Progress Users Group

Client Objectives

- Create a messaging alert system.
- Record messages in log file.
- Optionally email messages to members of an email group.
- Add to existing programs without significant modification.
- Setup database rules that determine when messages are sent.

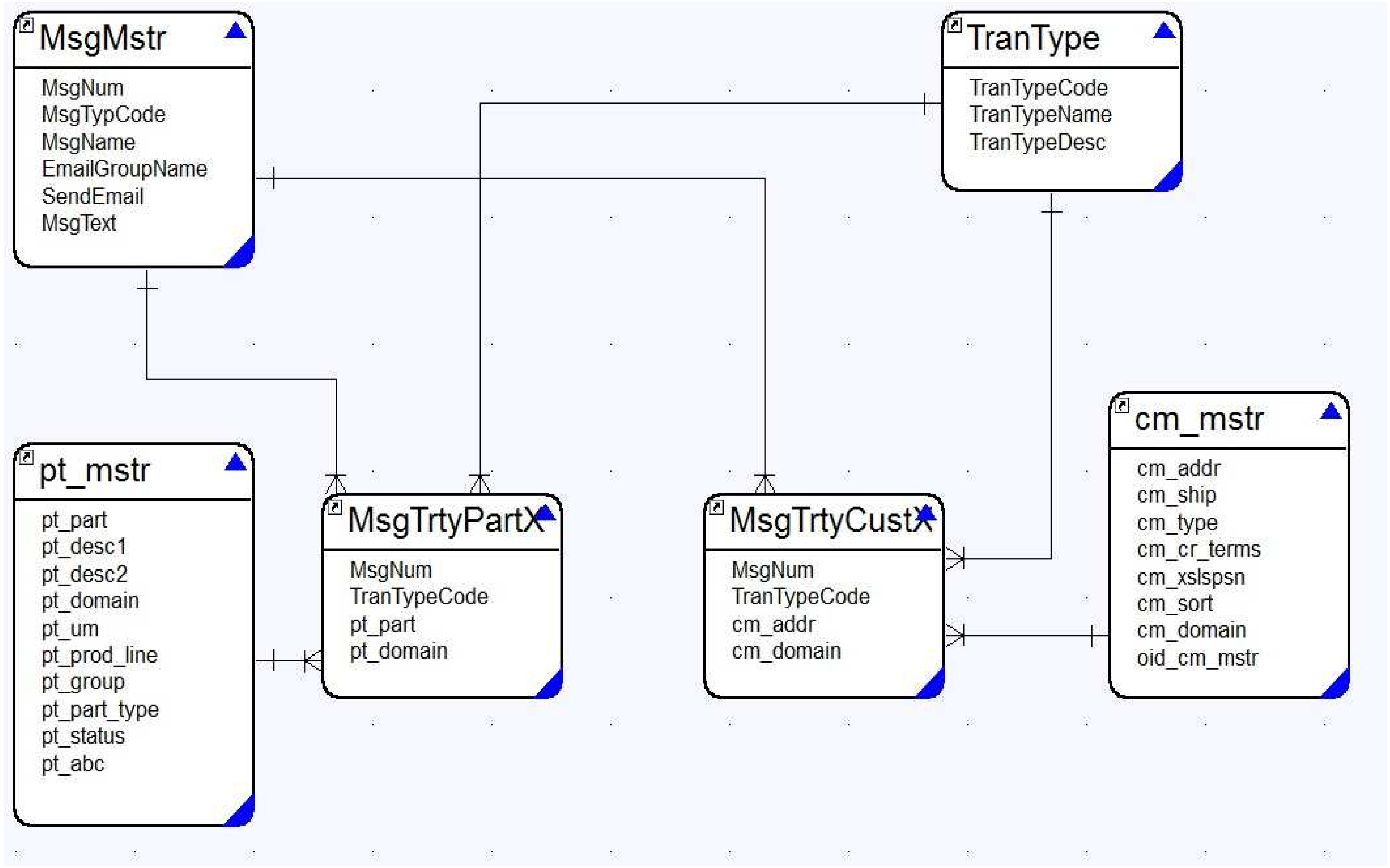
Client Objectives (cont.)

- Setup database rules that determine when messages are sent.
  - Part Availability
  - Customer Orders
  - Transaction errors

Database Design

- Three Database Views
  - Message-Customer-Part-Transaction Type
  - Email-Group-User
  - Email History

Message-Customer-Part-Transaction Type



Message-Customer-Part-Transaction Type

- Two cross reference tables are used to determine:
  - Which parts for specific transaction types should create a particular message. (MsgTrtPart)
  - Which customers for specific transaction types should create a particular message. (MsgTrtCust)

Message-Customer-Part-Transaction Type

- Examples:
  - When an inventory adjustment transaction occurs which drops the on hand quantity below 50 for component part X100, then issue warning message “Part X100 below safety stock level”.
  - Issue a notification message when an order acknowledgement occurs for customer ABC.

Message-Customer-Part-Transaction Type

- Perform Text Substitution for Message Alerts

^msgnum	-	message number
^domain	-	domain
^site	-	site
^group	-	group
^callingpgm	-	calling program
^pgmstack	-	program stack
^userid	-	userid
^msgtype	-	message type
&1 to &9	-	user defined

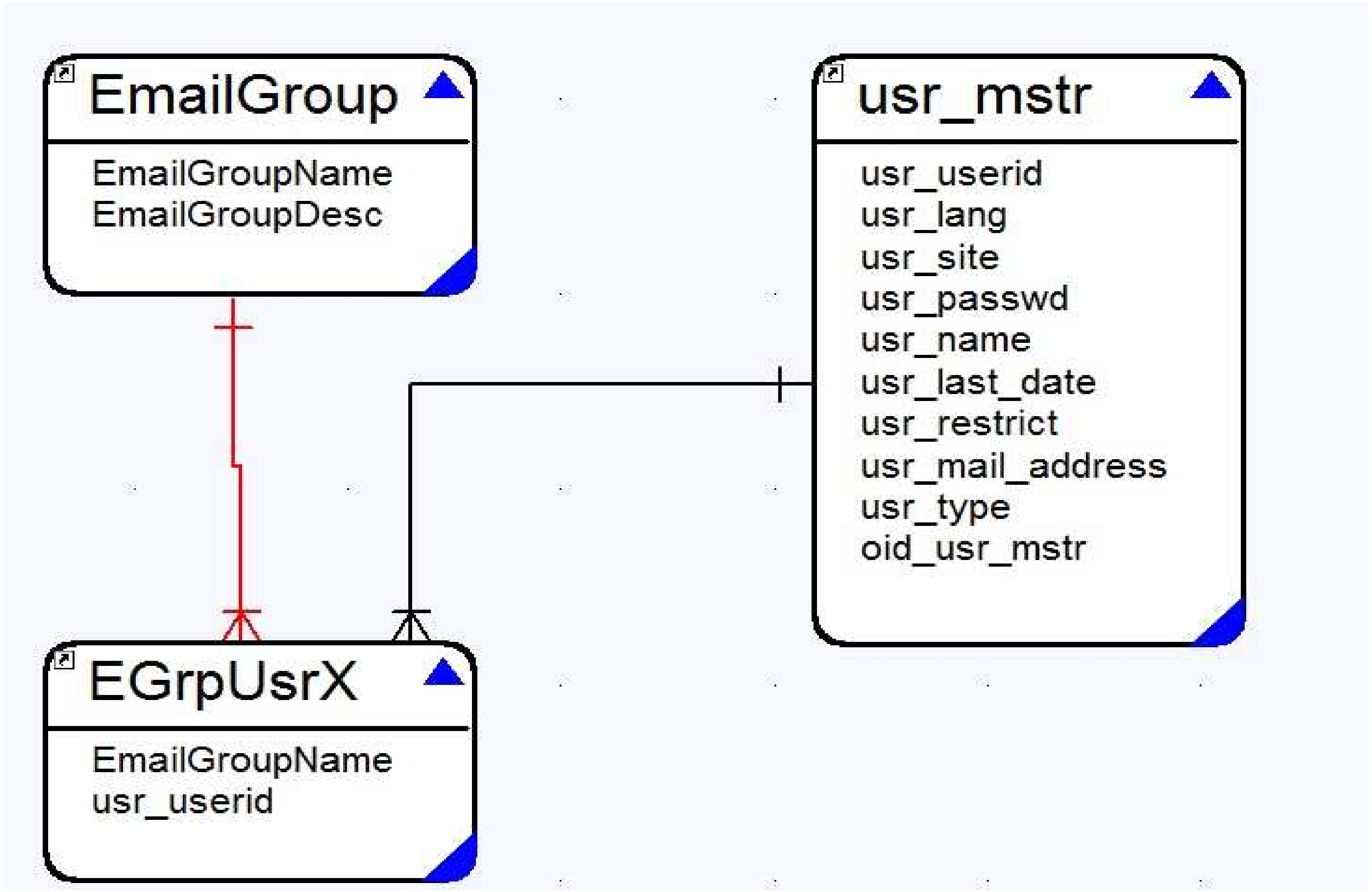
Perform Text Substitution for Message Alerts

- Example:

```
Invalid part number &6 in site ^site
  for transaction type &3.
```

- Use the substitute function to insert various parameters into a given message.

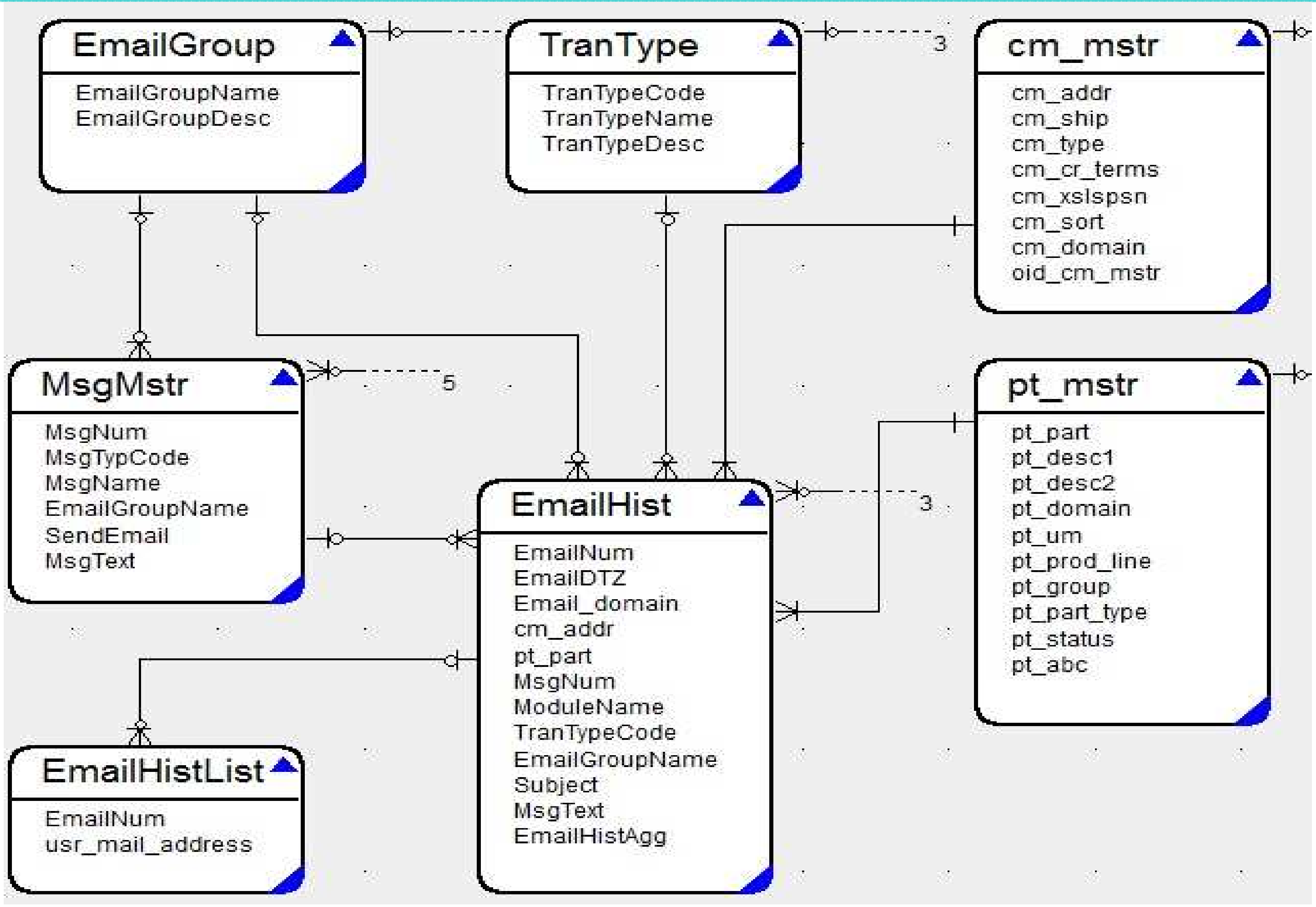
Email-Group-User



Email-Group-User

- An email group may contain many users.
- A user may belong to more than one email group.
- The cross reference table EGrpUsrX is used to model that relationship.
- An email group may be associated with a particular message.

Email History



Email History

- In addition to the log file capturing every message generated, an email history table is used to record every email sent per message.
- History records may be tracked by:
  - Email group and User
  - Transaction Type
  - Customer and Part Number

Tools of the Trade

- Business Rules Engine
- Subscribe/Publish
- Log-Manager System Handle

Business Rules Engine

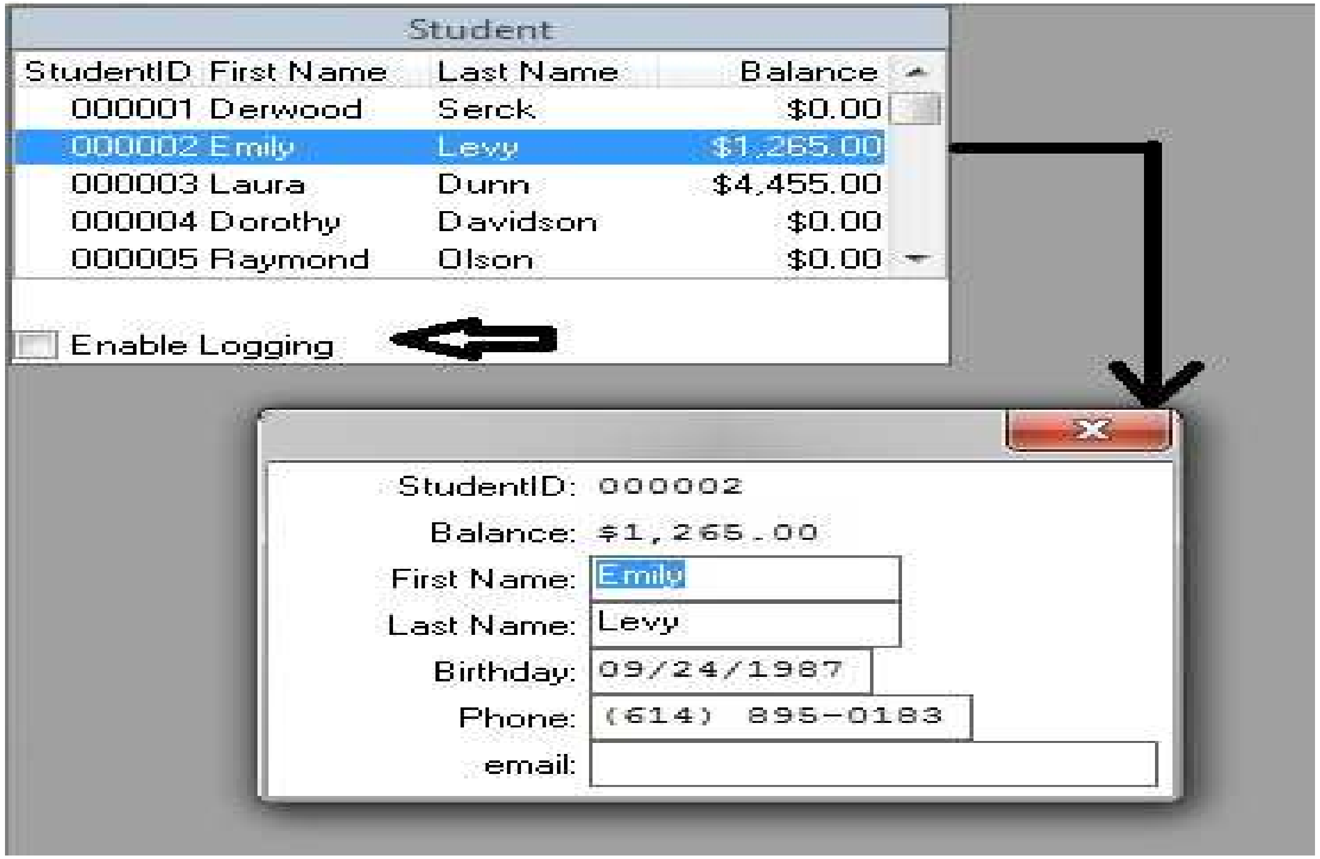
- When an event occurs, we will use the information from the Message-Customer-Part-Transaction Type database design to determine if a message should be generated.
- Use the MsgTrtyPartX if a part is passed, and use the MsgTrtyCustX if a customer is passed for the Business Rules Engine.

Subscribe/Publish-Named Events

- A named event is an internal procedure.
- The subscribe statement enlists procedures to access named events through the publish statement.
- The publish statement runs the internal procedure but doesn't raise an error if the internal procedure is not subscribed or does not exist.
- This makes the publish statement more flexible and independent than the run statement. This concept is *loose coupling*.

Subscribe/Publish Example

- This example allows the user to choose whether auditing is enabled.


Student

StudentID	First Name	Last Name	Balance
000001	Derwood	Serck	$0.00
000002	Emily	Levy	$1,265.00
000003	Laura	Dunn	$4,455.00
000004	Dorothy	Davidson	$0.00
000005	Raymond	Olson	$0.00

☐ Enable Logging

StudentID:	000002
Balance:	$1,265.00
First Name:	Emily
Last Name:	Levy
Birthday:	09/24/1987
Phone:	(614) 895-0183
email:	

Subscribe/Publish Example

- The update student program, updstud.p runs studlog.p persistently.
  - The program contains the internal procedure studentchanged which receives the student record buffer as a parameter and records output to the studlog.txt file.
- The enable logging toggle-box subscribes/unsubscribes the studentchanged event in studlog.p to the current procedure.

Subscribe/Publish Example

```
/* studlog.p - append to student log file whenever
   a student record is changed */

procedure studentchanged:
   define parameter buffer student for student.
   output to studlog.txt append.
   put unformatted "Student: " student.studentid "
   " sfirstname " " slastname
                   " was changed on " today " at "
   string(time,"HH:MM:SSam") skip.
   output close.
end.
```

Subscribe/Publish Example

```
on value-changed of logging
do:
   assign logging.
   if logging then
   subscribe procedure loghandle
  to "studentchanged" in this-procedure.
   else
   unsubscribe procedure loghandle to
  "studentchanged" in this-procedure.
end.
```

Subscribe/Publish Example

```
on default-action of b1
do:
  do on endkey undo,leave with view-as dialog-box
  with 1 column:
    find current student exclusive-lock.
    display studentid balanceamt.
    update sfirstname slastname bday phone email.
    b1:refresh().
    publish "studentchanged" (buffer student).
  end.
end.
```

Subscribe/Publish Example

Studlog.txt file contains:

```
Student: 2 Emily Levy was changed on 05/04/14 at  8:13:52pm
Student: 4 Dorothy Davidson was changed on 05/04/14 at  8:14:20pm
```

Multi-User Log File Write Issue

- What happens when more than one user tries to write to the same file at the same time?
  - It doesn’t matter if the program uses:
    - **`output to studlog.txt.`**
    - **`output to studlog.txt append.`**
- Only one user’s write command is written to the output file.

Multi-User Log File Write Issue

- Running crtmsg1.p and crtmsg2.p around the same time.
- The second program started takes precedence at writing to the output file, ***`log1.txt`***.

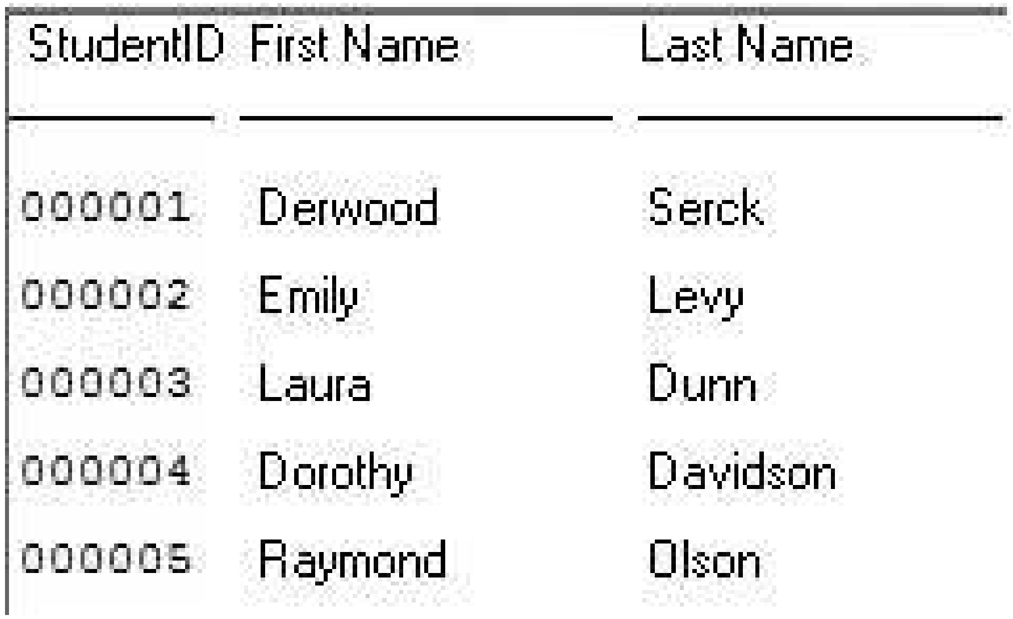
Multi-User Log File Write Issue

```
/* crtmsg1.p - output a message using the output
   to statement */
def var i as int.
def stream log.

output stream log to log1.txt append.
for each student i = 1 to 5:
  display studentid sfirstname slastname.
  put stream log studentid " " sfirstname " "
  slastname skip.
  pause i.
end.
output stream log close.
```

Multi-User Log File Write Issue

- crtmsg1.p output:



```
StudentID  First Name        Last Name
_________  ______________    ______________

000001     Derwood           Serck
000002     Emily             Levy
000003     Laura             Dunn
000004     Dorothy           Davidson
000005     Raymond           Olson
```

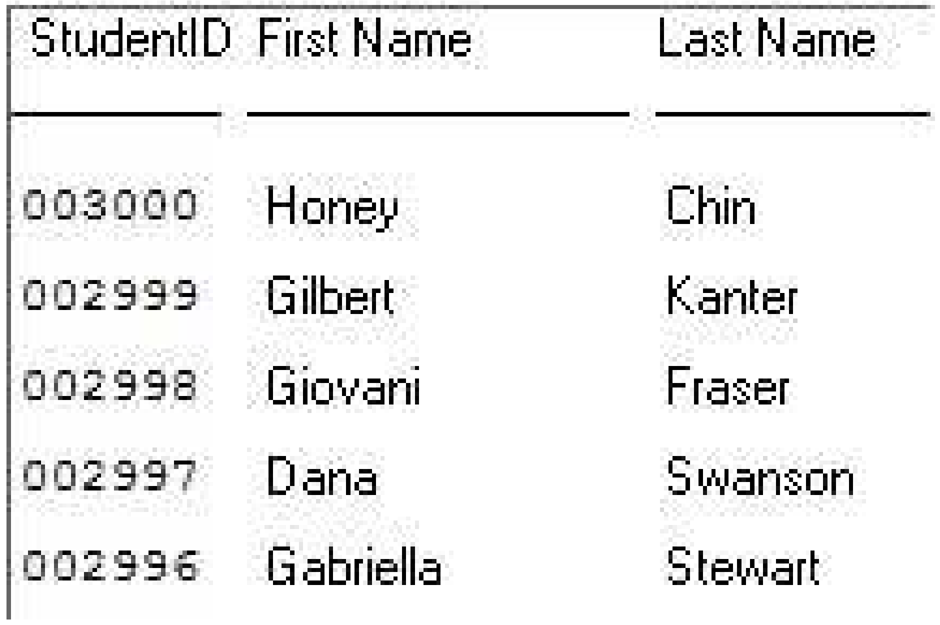
Multi-User Log File Write Issue

```
/* crtmsg2.p - output a message using the output
   to statement */
def var i as int.
def stream log.

output stream log to log1.txt append.
for each student by studentid desc i = 1 to 5:
  display studentid sfirstname slastname.
  put stream log studentid " " sfirstname " "
   slastname skip.
  pause i.
end.
output stream log close.
```

Multi-User Log File Write Issue

- crtmsg2.p output:



```
StudentID  First Name        Last Name
_________  ______________    __________

003000     Honey             Chin
002999     Gilbert           Kanter
002998     Giovani           Fraser
002997     Dana              Swanson
002996     Gabriella         Stewart
```

Multi-User Log File Write Issue

- What is the solution?

Log-Manager System Handle

- OpenEdge offers the Log-Manager system handle for its logging infrastructure. It provides:
  - Standardized reporting of run-time activity
  - Logging diagnostic data for troubleshooting problems.
  - It's also the solution to the multi-user log file write issue.

Multi-User Log File Write Solution

```
/* crtmsg3.p - output a message using the
log-manager */
def var i as int.

log-manager:logfile-name = "logmgr1.txt".
for each student i = 1 to 5:
  log-manager:write-message(string(studentid)
       + " "
       + sfirstname
       + " "
       + slastname).
  pause i.
end.
```

Multi-User Log File Write Solution

```
/* crtmsg4.p - output a message using the
log-manager */
def var i as int.

log-manager:logfile-name = "logmgr1.txt".
for each student by studentid desc i = 1 to 5:
  log-manager:write-message(string(studentid)
     + " "
     + sfirstname
     + " "
     + slastname).
  pause i.
end.
```

Multi-User Log File Write Solution

Combined output for logmgr1.txt:

```
Logging level set to = 2
No entry types are activated
1 Derwood Serck
2 Emily Levy
3 Laura Dunn
Logging level set to = 2
No entry types are activated
3000 Honey Chin
2999 Gilbert Kanter
4 Dorothy Davidson
2998 Giovani Fraser
5 Raymond Olson
2997 Dana Swanson
2996 Gabriella Stewart
```

Log-Manager - Log File Format

- DateTime-TZ - [14/05/12@20:10:19.004-0500]
- Process ID - P-006448
- Thread ID - T-007656
- Logging Level – 1
- Execution Environment – 4GL
- Log Entry Type - 4GLMESSAGE
- Message Text - This part Left Nostril Inhaler is nothing to sneeze at.

Log-Manager Components

- Components used in this example:
  - LogFile-Name attribute
  - Write-Message() method
- Other Useful Components
  - Clear-Log() method
  - Log-Entry-Types() method

Log-Manager Components

- The Clear-Log() method clears all messages existing in the current client log file and leaves the file open for writing.
- The Log-Entry-Types() method is a comma-separated list of one or more types of log entries to write to the log file.
  - There are many Log-Entry-Types that may be specified.
  - A logging level (0-4) may be optionally specified for each Log-Entry-Type in the list.

Log-Entry-Types() Example

```
/* crtmsg5.p - output message using the log-manager */
def var i as int.
def var ok as log.

log-manager:logfile-name = "logmgr1.txt".
ok = log-manager:CLEAR-LOG ().

log-manager:LOG-ENTRY-TYPES = "4GLTrace,4GLTrans,QryInfo:3".
for each student i = 1 to 5:
  assign student.address2 = "Suite " + string(i).
  log-manager:write-message(string(studentid)
     + " "
     + sfirstname
     + " " + slastname).
  pause i.
end.
```

QryInfo Beginning Output

```
QRYINFO      Query Plan:
  C:\workspaces\oe112\pga\crtmsg5.p line 9
QRYINFO      QueryId: 11722928
QRYINFO      Type: FOR Statement
QRYINFO      Client Sort: N
QRYINFO      Scrolling: N
QRYINFO      Table: dbaschool.student
QRYINFO        Indexes: studentId
```

QryInfo Ending Output

```
QRYINFO     Query Statistics:
   C:\workspaces\oe112\pga\crtmsg5.p line 9
QRYINFO     QueryId: 11722928
QRYINFO     DB Blocks accessed:
QRYINFO      dbaschool : 11
QRYINFO     DB Reads:
QRYINFO      Table: dbaschool.student : 5
QRYINFO      Index: student.studentId : UNAVAILABLE
QRYINFO     dbaschool.student Table:
QRYINFO      4GL Records: 6
QRYINFO      Records from server: 6
QRYINFO       Useful: 6
QRYINFO       Failed: 0
QRYINFO      Select By Client: N
```

4GLTrans Output

```
4GLTRANS      BEGIN TRANS 28
    [C:\workspaces\oe112\pga\crtmsg5.p @ 9]
APPL          1 Derwood Serck
4GLTRANS      END TRANS 28
    [C:\workspaces\oe112\pga\crtmsg5.p @ 13]
4GLTRANS      BEGIN TRANS 29
    [C:\workspaces\oe112\pga\crtmsg5.p @ 9]
APPL          2 Emily Levy
4GLTRANS      END TRANS 29
    [C:\workspaces\oe112\pga\crtmsg5.p @ 13]
```

❑

❑

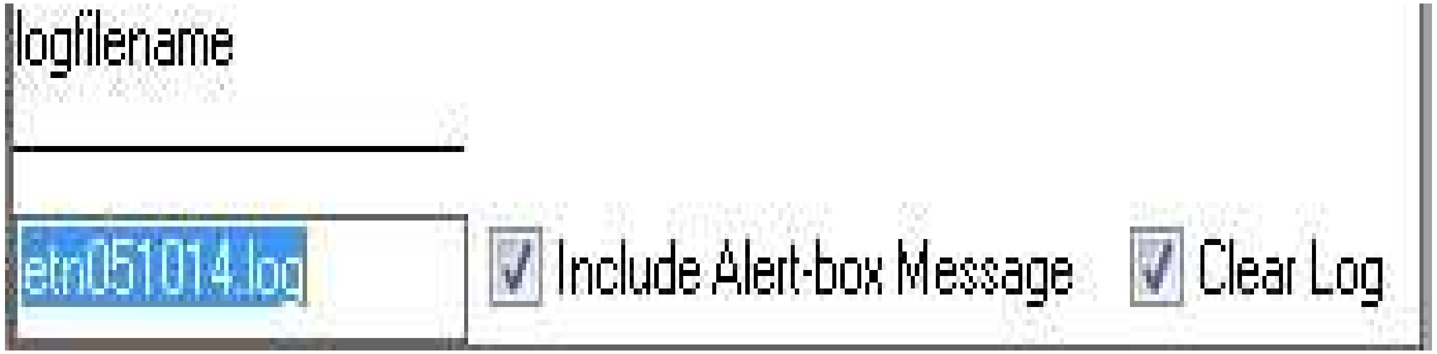
❑

Managing Log Files

- -logthreshold – Use this startup parameter to to specify the file size at which OpenEdge rolls over (renames and saves) log files.
- -numlogfiles – Specify the number of rolled over log files to keep on disk at any one time, for ABL session, including the current log file.

Case Study Demonstration

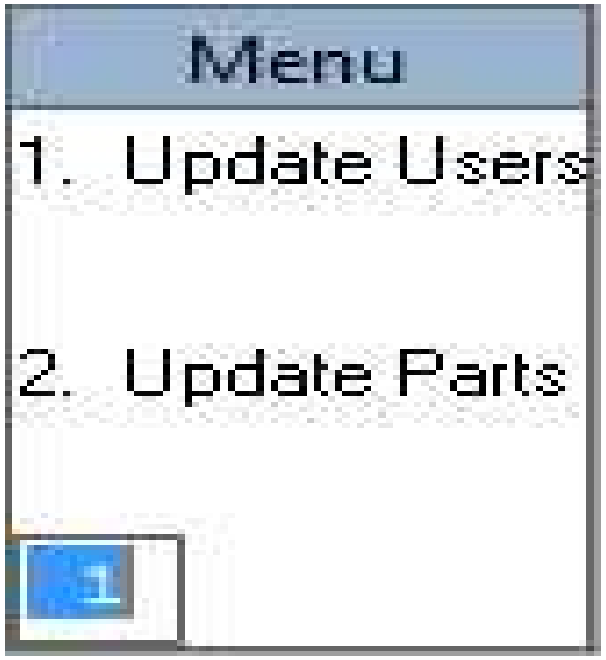
Case Study Demonstration



logfilename

etn051014.log

☑ Include Alert-box Message ☑ Clear Log



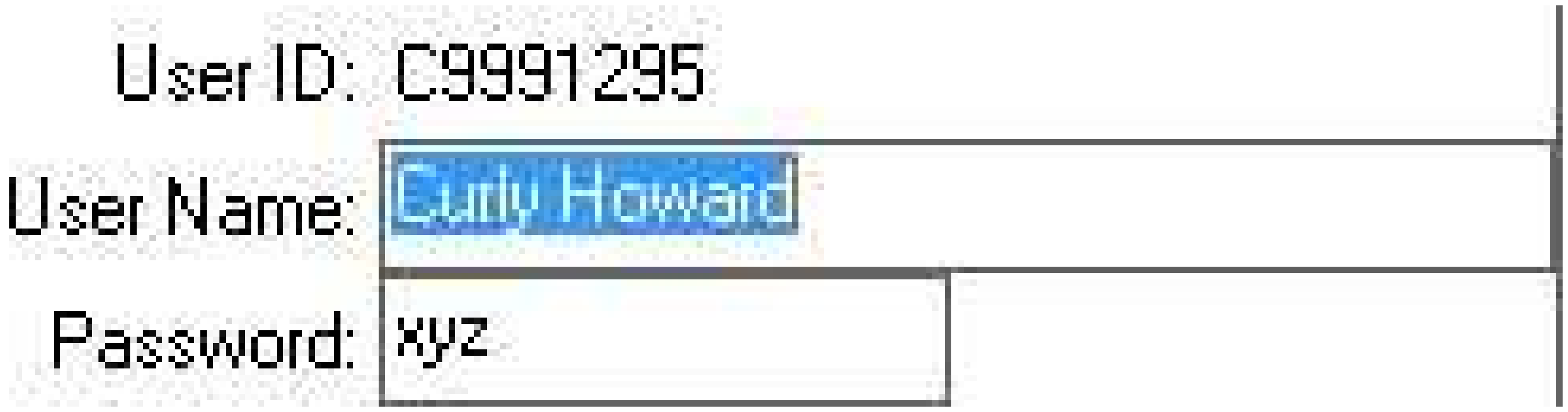
Menu

1. Update Users

2. Update Parts

1

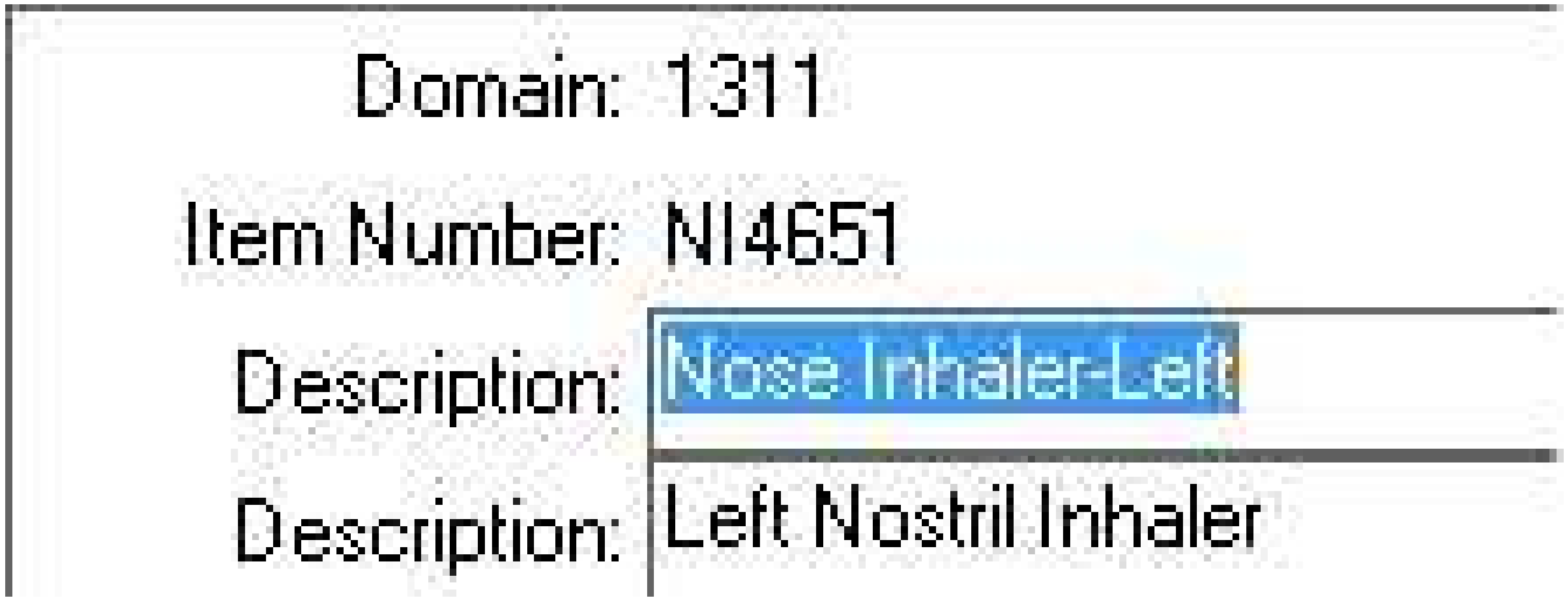
Case Study Demonstration



User ID:	C9991295
User Name:	Curly Howard
Password:	xyz

- Publish message 1 before update.
- If user name matches “*howard*” then log message number 2.

Case Study Demonstration



Domain: 1311

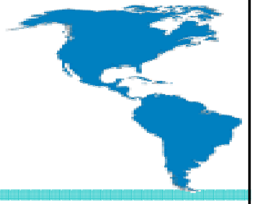
Item Number: NI4651

Description: Nose Inhaler-Left

Description: Left Nostril Inhaler

- Publish message 1 before update.
- If the part number matches "*46*" then log message number 3.

Summary



- An messaging alert system is useful and not that difficult to implement in a legacy based OpenEdge application.
- Use a business rules based engine, subscribe/publish and log-manager concepts to implement the alert system.
- The Log-Manager system-handle is useful for many instances in an application and is helpful for debugging applications.

Questions