Openbravo POS

Integration and Development Guide

Redhuan D. Oon (RED1)
Rebel Leader - The ADempiere Project • September 11, 2011

(Update November 2012 - read Epilogue at the end - and my new cave is now in http://sf.net/p/red1)

Cockpit view of the World Trade Center towers seen from the flightpath of American Flight 11

source: wikipedia

"I don't think Arab terrorists from Saudi Arabia can carry out this highly sophisticated operation with such success."

Dr. Mahathir Mohamad,

http://chedet.cc/blog/?p=595

"The Middle East has a lax mentality on the necessity for punctuality and few wear watches. Tardiness is not a sign of disrespect and does not warrant an apology unless the tardiness is excessive (more than 90 minutes)."

*source: www.fas.org/irp/agency/army/**arabculture**.pdf*

"This just shows, what goes around comes around, even to the US."

Bobby Fischer,

(while hiding in Manila)

source: http://www.rense.com/general17/chessgrandmaster.htm

Sponsored by

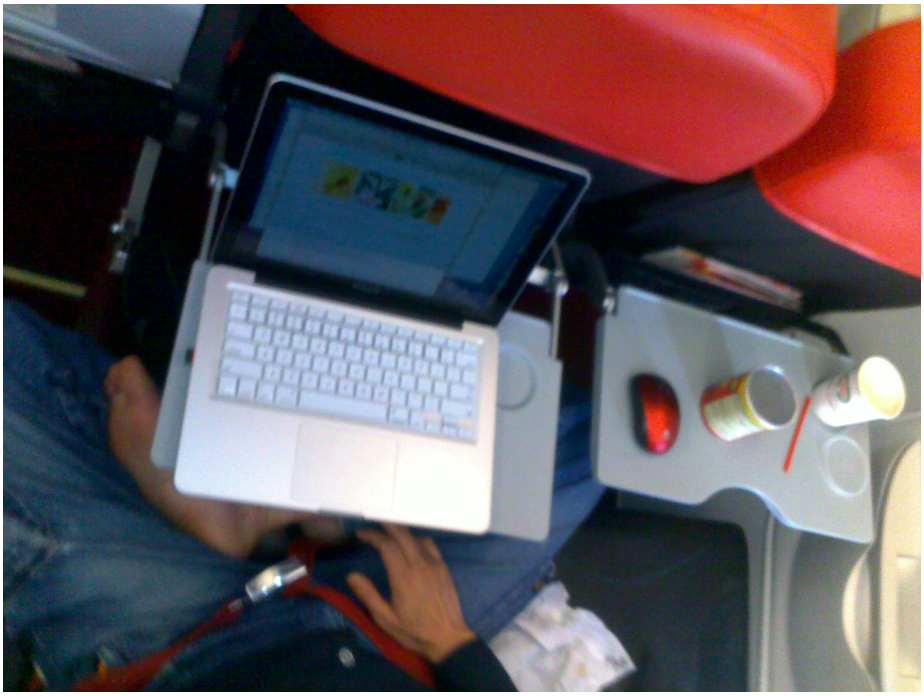
BANGLADESH

DISCLAIMER

This introductory disclaimer was written onboard flight AK662 from Kuala Lumpur bound for ClarkField Manila on the morning after **September 11**. Choosing that date theme couldn't escape me, with all the mass media coverage for the 10 year anniversary. It triggered my brain cells to reflect and draw a learning parallel about software development been in crisis and prone to disasters. It helps me appreciate the greatest potential for often impossible odds by going Open Source, of which is anarchic and *taleban*-ridden in its *realpolitik*. It is not meant to read further into the sad tragedy on that fateful day. Our prayers and thoughts are with those who lost their lives and their loved ones as well as the survivors who are known to suffer invariably from it. As Allah shall gather all Mankind onto that Single Day where Truth shall be finally revealed.

Enjoy in peace!

red1
Also in hiding, momentarily at Dorm 902A, Asia Pacific College, surrounded by pretty *pinoys*,
Magallanes Village,
Makati City, MANILA.



Onboard Flight AK662 bound for ClarkField. As you can see, I booked the whole row.

Seats for the angels on my right and left shoulders.

CREDITS

Credits must be attributed to sources or reference of works that is made public and openly shared for the benefit of the Open Source world. Firstly I have to give it to *Openbravo POS* or formerly *TinaPOS* for been such a crisp POS application written in my favorite painful Java. Second is to the Openbravo project itself for their informative wiki that helped me recover the deprecated v2.2 of the POS application and avoid their possible commercialisation uncertainty. Then it is to Packt Publishing UK that gave me a complimentary copy of the *ADempiere 3.6 CookBook* by Ajit Kumar. From there, it leads me to the *Apache ActiveMQ* project itself of which I am also equally indebted to. In the CookBook itself I could gather tell-tale signs that it has in some parts refer to the already published wiki pages of Carlos Ruiz (GlobalQSS, Colombia) who is always exposing code secrets lavishly, and I will be lying if in any of my works there has no touch of the master himself. Usually he or Low Heng Sin sometimes gives me an ever slight tap in the right direction that would have been a disaster if I did not correct my flight path in time.

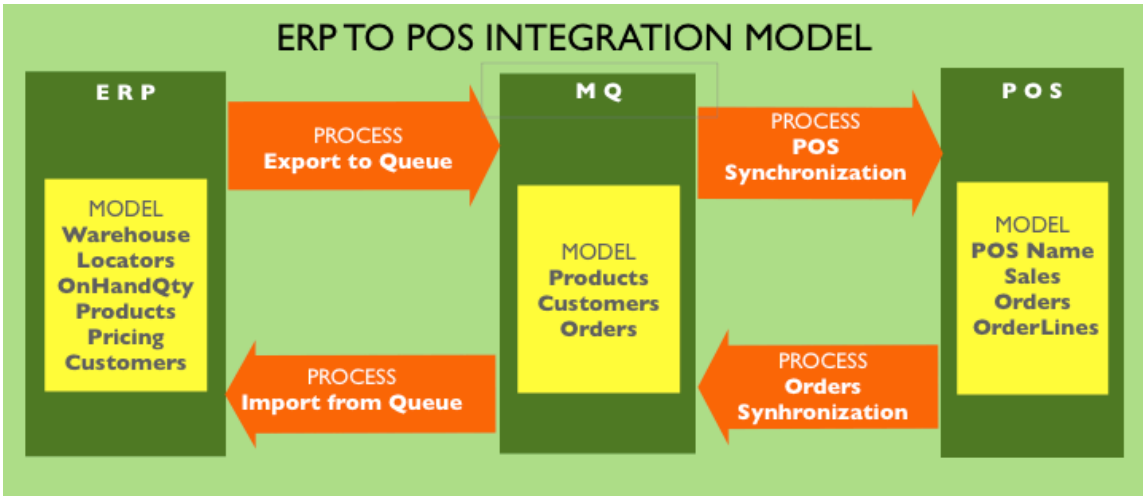
As usual I always give thanks and my gratitude to the global community of users, testers, screamers, lurkers and rare contributors, as without them, I would have not found inspiration nor purpose to carry on living, or fighting to the death, depending on the occasion. I am also glad to work more closely with 'banym' or Dominik Zajac (BayCIX GmbH, Munich) of whom I hope to work side by side in Munich on my next task of harnessing the Jenkins build server he setup so quickly for our benefit in this project.

TABLE OF CONTENTS

Integration Guide

2 limbs within 2 parts

OR, WHY THIS IS NOT A SUICIDE MISSION



This is Pashtun Territory. Mission Accomplished. Proceed to page 50.

Who are you and what do you want?

This 2 section guide is part one of a twin part rendition that first show how the integration of Openbravo POS is done, and packaged into ADempiere 361, and later in a future part II will port it to iDempiere, the OSGi project of ADempiere, and tested within a Jenkins build and quality assurance server. In this first section, if you are a newbie convert, we welcome you to understand and setup the POS and ADempiere integration at one go. In the next, if you are a suicidal developer, we welcome you to blow the code up.

Why Openbravo POS?

Openbravo POS is a standalone Point of Sales application software that can operate without any backend ERP. That means you can use it right away on a station at a sales location. Away from radar. It is not affected or stoppable by any backend, that may have itself gone down or crashed.

It can operate quickly as it processes minimal document types within a very light *Derby* database. It has marvelous features such as split payment option and split invoice for a single sale. It has restaurant table arrangement. It has stock movement capture during sales and import from ERP. It is written in Java and of course the mother of all reasons, it is Open Source and free to extend and improve.

Why integrate to an ERP?

Without a backend, it will be limited and poor in its data environment. It will not have products nor customer updates unless you key them in manually and repetitively for each POS station. You may replicate but it is best to centralised from an ERP backend. Integrating to an ERP allows the POS sales orders to pass to a more capable and powerful system that can perform financial analysis, measure business efficiency and update operational requirements planning such as inventory replenishment and distribution. The ERP can also allocate each POS station as a virtual locator in its *Warehouse Locators* network.

How to integrate?

This is the crucial question and it is a most interesting one. Technology advances are abound all over the web. There is real-time *synchronous* web services as well as offline *asynchronous* messaging mechanisms. In our integration we are taking the asynchronous approach. After all we are not as perfect-timed as *the hijackers*, nor are we not going to test this in Afghanistan. But it is more of been pragmatic. A POS station is not an ATM banking machine. It need not be requesting constantly about the latest product price strategic changes or brag back its latest sales conquests. If it does, it will be a very busy chatline and not much time is left doing POS retailing or giving some peace to the head to think. In other words, we need *loose coupling*. That allows a POS to sit tight and operate freely from any server linkage or dependency. Central command may go down but it can still fly.

Any information it needs can be updated in a scheduled, periodic basis. A logical schedule can be daily, at night when the station is closed. The figures it updates ought to be granular enough such as end-of-day sales, or total quantity per product. A retail outlet or even a manufacturer hedge its prices on a daily basis rather than minute to minute, so as to maintain sanity. To the accountant, they need to budget before an event. How many? Where do they go? What are their routes? Targets? How much the tickets? What about airport security? What preparation and training? How

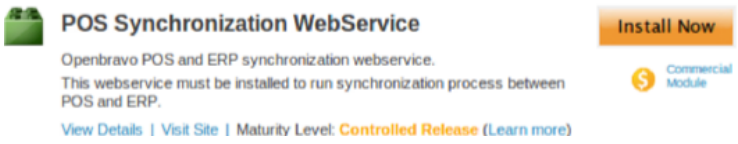
long the practice before due? How many errors before perfection? And how the heck do you get them to all show up on time? Planning is not easy. Even if you plan way before, and keep it simple, and take into account most common contingencies. As *Murphy's Law* says,'whatever can go wrong, WILL go wrong'.

Smart management allows for failures and mistakes but knows how to deal with them. Otherwise it will be called the *Military* instead where failure is not an option. And they are known for the biggest mistakes. Too regimented. Too much *chain of command*. A big ego within. Not agile enough. Recipe for disaster.

Mistakes are a norm. They are anticipated and prepared for within good design questions. How much stock levels you carry? How much do you want to update? What are your selling cycles? You can plan the next day stocks, you can plan the picking on the warehouse floor. You can inform the carrier routes. Your truck driver will go nuts if you issue too many *destination* changes before he even has time to review his *map*. He will miss his *target* if he has to think too much. He needs routine. For perfect routines are regular and has certainty. You need that to practice till it is burnt into your subconscious.

Then there are infrastructure questions. How good is your broadband? How reliable? What is the guaranteed SLA? What if there is an outage? How do you backup? How do you restore from backup? How long can you go down (while waiting for backup to restore and gaps recovered) before your business goes down?

Why ADempiere?



Openbravo POS is at version 2.3 at the time of planning this. It is already showing signs of commercial flight. There are dollar signs posted in its plugin page. Its POS synching is now a plugin. I would recommend Openbravo ERP if you do need commercial vendor support as ADempiere's support is more of a communal and open one that might not fit well with some end-users. More of this issue in the next section.

Out of religious extremism, I am with the *freedom fighter* community. I am fanatic that freedom will ensure eternity and angelic quality. Commercial interests has limitations in todays *free lunch* no-holds barred Internet heaven (Virgins not included).

ADempiere has been a community fork since 2006. It has given good practice to many of its advocates. We learnt the code day by day. There are 3.4 million lines of it. No one can memorise them, left alone master the code's flow or intent. But through practice and experience, its seniors gained the flying hours needed to fly further. Particularly on a route that no one else has taken.

The ERP Bazaar has its own direction, according to its biggest givers - Carlos Ruiz and Low Heng Sin. It is going OSGi. They are equally concerned about stability and compatibility as much as technology. It is now the iDempiere project. It goes for maximum possible quality and visibility in its open source nature. It also needs to be well tested and documented better. It also means a singular vision and direction. There is no democracy capable in dictating nor distracting the mastermind's head. He knows what is needed. He knows when, how, who, what, and why.

iDempiere?

That is for the future Part II. After this POS integration is working nicely, we are set to see if we can port it as another module within the *OSGi framework*. We can maintain it well and easy by creating tests for it to run off *Jenkins* build server that can track the whole OSGi suite. OSGi platform allows other modules from ADempiere such as *Libero Manufacturing* to possibly join in as another equal citizen with the other plugins in the air. They can attempt takeoff, fail, crash and burnt itself out of the way.

The *Jenkins* incorporation is a culminating vision to provide build and testing automation accessible remotely where each change of code is automatically tested and verified almost instantly. It means savings in thousands of man hour dollars.

September 11 in *Mission Impossible* likeness demonstrates an astounding spectacle that is said to be a result of a group of madmen. They joined a flying school, learnt how to fly, fly a large boeing, hijack it first time on the appointed times, and taking it at high speed in the air, searching for a target only by human vision, and marshaling the plane hands-on right into dead centre on first fly pass. Here I am trying to figure out something much much simpler, And all my attempts are riddled with mistakes, bugs and failures. I have greatest help from other gurus, abundant online material and a stubborn heart that prays, "*My God, my God, why have you forsaken me?*".

Fortunately, Open Source allows for the dumbest bomb-strapped developers to excel.

DOWNLOAD PLAN

Follow these instructions to the letter my brother. Deviate and you can end up in *Hell*. You can download and run Openbravo POS right away from its website or install what is done in the stated repository at the SourceForge project.

I- Original Openbravo POS, select according to your platform within this link:
http://sourceforge.net/projects/openbravopos/files/Openbravo%20POS/Openbravo%20POS%202.30.2/

For newbies, It is best to get the Openbravo POS installer first because it is a great self-installer to get your POS up and running in no time. At least play around with it to get a feel of the controls and cockpit. Also it can be a good comparison with my modified version just in case you think it is a bug, sending yourself to hell and blaming your guru here for that. After enough flight simulation, you can crash it and proceed below.

Links to Softwares

For true-git *Freedom Source*, go to

A- Compiled POS application This is modified version the original **I** above.
http://sourceforge.net/projects/adempiere/files/openbravoPOS/openbravopos_2.30.2_bin.zip/download 5.1Mb

B- openbravoPOS jar This is to replace just the jar at your original **A** above without replacing everything

http://sourceforge.net/projects/adempiere/files/openbravoPOS/openbravopos.jar/download 260.2kb

C- ADempiere patch for silentsetup
http://sourceforge.net/projects/adempiere/files/openbravoPOS/openbravoMQ.jar/download

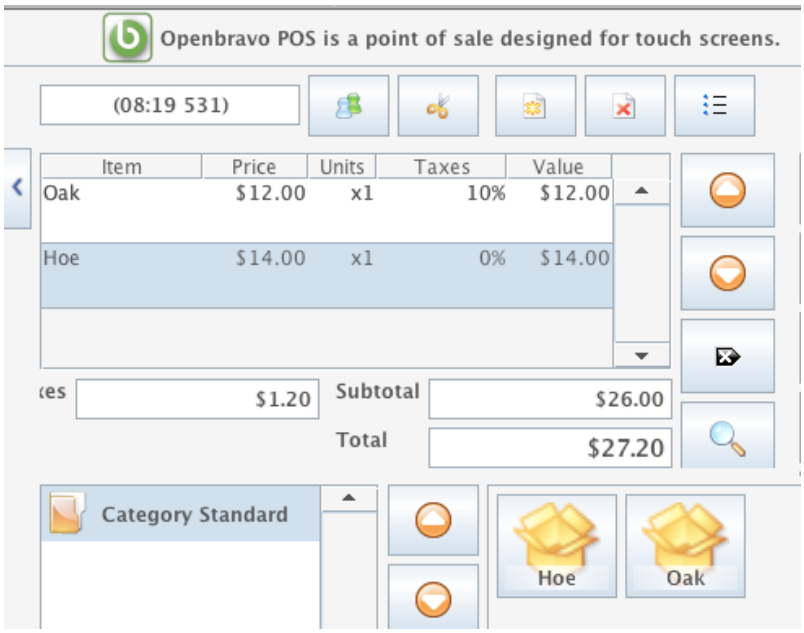
D- Migration scripts
http://sourceforge.net/projects/adempiere/files/openbravoPOS/migration.zip/download

There is some minor inadequacy with the present ADempiere version which needs upgrading of its present activeMQ-core-5.5.0.jar in the tools/lib folder of the source. Instructions on what to do are given later during *Execution Plan*.

EXECUTION PLAN

Reminder: My brother, *Hell* is real. Especially when you are trying out something that is free. So memorise this or go back to your *cave*. It is not advisable to be too lazy to *Read The Fundamentalist Manual*. You may become spiritually imbalanced to interpret any of the scriptures.

Firstly with the installer, blow it up and you will find a **start.bat** or **start.sh** inside. Run that and you will get to this pop-up. It is checking for you if you have created a database. This POS comes with adapters for *Derby, HSQl, MySQL, Postgres* and *Oracle* flavours. To make life easier for you, it pops-up this question so that you can say Yes and allow it to automatically create the Derby database and populate new *virgin* tables. (Later for those who survive this suicide mission, *Your guru* here will then show you where and how the scripts define the DB contents and config properties)



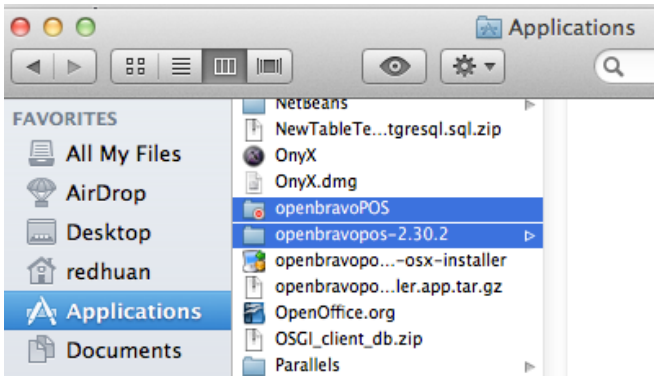
If you launch it, you can get to define some simple products such as *Hoe* and *Oak* and at the Sales menu try to order something.

If you don't want to trouble yourself with taking off and landing, but go straight to your targets, you can proceed to the next lesson which will eventually guide you to the integration plan and execution that quickly and easily import all the products from the ERP country. Much like *autopilot*.

Placing the new binary

Put the modified binary **A- Compiled POS application** in your root folder and give it a simpler name other than the Openbravo installed one so that you can distinguish which tower you are heading to and not knock into the same one your dear brother just did.

Ok, we just have to give you a visual below just in case that sounds confusing without a visual cue.



The simpler named one 'openbravoPOS' is the modified one. The longer named one is the installer one. And below it is the installer. No, don't touch that. You already did blow it up. Don't blow up again. Wait till your guru is one thousand miles away.

A later alternative is to just replace with the **B- openbravoPOS jar** above only after the first time you have used **A**.

Replacing the database

Now, you have to destroy your existing **openbravo-database** by deleting it. (The database folder is usually under the user home). You may rename it if you have a need to use the old database. This will fool control tower to think that you have no *transponder* database and asking to reinstall the database and with it the modified one that has my modified scripts. After you start the POS, you can click on the **Maintenance** button on the left panel and see at the bottom screen the appearance of two *angels*. You might want to kneel down and give thanks.

The *POS synchronization* button is to import the Products (with Prices and store qty just for this POS terminal) and Customers lists. The *Orders synchronization* is for exporting the Sales from this terminal back to the remotely operated ERP.

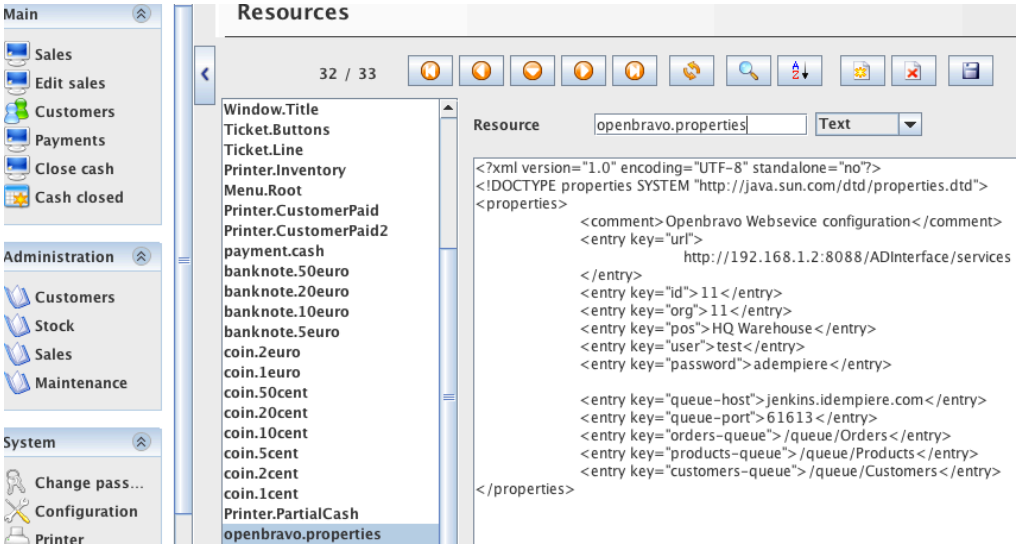
Upgrading ADempiere

But remember about the 5.0.0 jar that needs upgrading? You can take the solution on page 17 or upgrade your own source taught in page 40. You have to do this first before migrating step next.

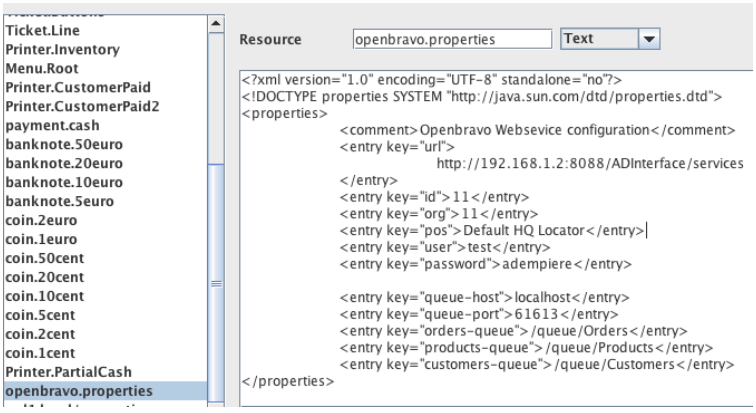
Migration of ADempiere

Next chapter guru here will show you what we mean by remotely operated. Meanwhile we still have some more work to do in migrating the present ADempiere ERP before it is ready to work with the POS app. We need to apply a patch to it and some migration scripts. You can do that with **C- ADempiere patch for silentsetup** and **D- Migration scripts** (same manner as most ADempiere modifications. I also briefly described it in the next section but please refer to online resource such as www.adempiere.com if you are still lost. Don't call 911. It may be busy.

Below is the screenshots of the Maintenance > Properties > **openbravo.properties** page. Note that unlike the original openbravoPOS you first installed, this modified one has such properties restored. The properties are given by openbravo Wiki in the previous 2.2 release which they deprecated.



I have added more properties to reuse it fully for the integration settings. Ensure that the **pos** locator name is set to "HQ Warehouse" for your maiden flight. Note that your **queue-host** is at the remote **jenkins.idempiere.com**. You may modify them to suit your activeMQ and *paths* . For example here below has the *queue-host* set to *localhost*.But seriously, just use the *Afghan* link line given to you by *Osama*. You may have no time left.



INTEGRATION PLAN

The ERP Application or server can be entirely remote to the POS or be anywhere in the world that has broadband Internet access. As long as the POS and ERP system are connected to a web-enabled line, we have a perfect operational environment. But with the asynchronous idea, we need not be that perfect or alive all the time, or at the same time. The Internet or ERP server can be momentarily down and the POS or POSes can still operate, doing sales and handling normal day to day business. When the Internet or ERP is back on, then can the POS update itself with the latest products and pricing and inventory figures from the ERP central command. Now, with ActiveMQ or messaging technology even the POS can update without the ERP end to be up. As long as the ERP has deposited or updated the ActiveMQ (message queue) service which is somewhere else, again even remotely situated with its own web access. The ActiveMQ service from the Apache project is a MOM or *message-oriented middleware*. It has its own Derby database that can store the messages queue and thus it can also go under the radar and shut off. When it is turned on again, its messages are still there. That is called *persistence*. This allow the service to experience disruption or stopping due to electrical outage. If it has booted up again, it won't need to be updated again by either the POS or ERP end. However that persistence option is controllable as some messaging strategies may not want persistence. At the moment I have turned it on within the code. Reason been that many POS stations may take a longer time to access it.

The ready-made demo ActiveMQ service is actually somewhere in Bavarian country (Munich to be exact, compliments from Dominik of **BayCIX GmbH**). It will be freely accessible as the URL in the next screen shows. If heaven can wait, you may change the setting to 'localhost'. But you have to download from **Apache ActiveMQ** website and run it yourself in your own PC. It is a very simple task to do that, a no-brainer. At the URL on the right you can call up the admin page and go to *Customers* link and see actual data. This is a fantastic feature of ActiveMQ as now you have a bird's eye view of your communication to and fro both ends. You can also see how many are reading the queues and I have arranged the queues according to clear separate name paths: Products, Customers, Orders.

INTEGRATION EXECUTION

Before we can apply the package and migration script of the integration module we have to upgrade the ADempiere application to **ActiveMQ 5.5.0**. That is done from inside the ADempiere source and explained in the next section for Developers (page 40 - Upgrading ActiveMQ 5.5.0). For those who can't wait to be in paradise, I will upload ready-made **ADempiere zip** with an **ExpDat dump jar** but in *PG18* flavour. (Sorry we run out of that virginal *Oracle* flavour).

Complete binary for ADempiere

ADempiere.zip -

http://sourceforge.net/projects/adempiere/files/openbravoPOS/Adempiere_360LTS.010.zip/download

Packages.zip -

http://sourceforge.net/projects/adempiere/files/openbravoPOS/packages.zip/download

ExpDat.dmp (Postgres) -

http://sourceforge.net/projects/adempiere/files/openbravoPOS/ExpDat_pos.jar/download

Now we do the actual migration on the ERP side. With your ADempiere already set up, you can place the patch along with the contents of **packages.zip** i.e. /packages/.. :

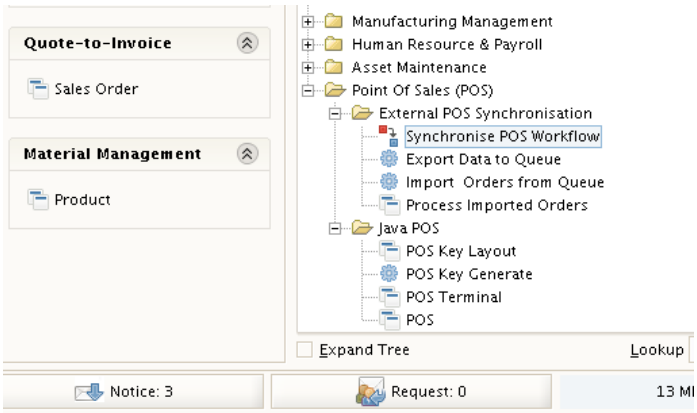
<ADempiere_Home>/packages/openbravoPOS/lib/**openbravoMQ.jar**

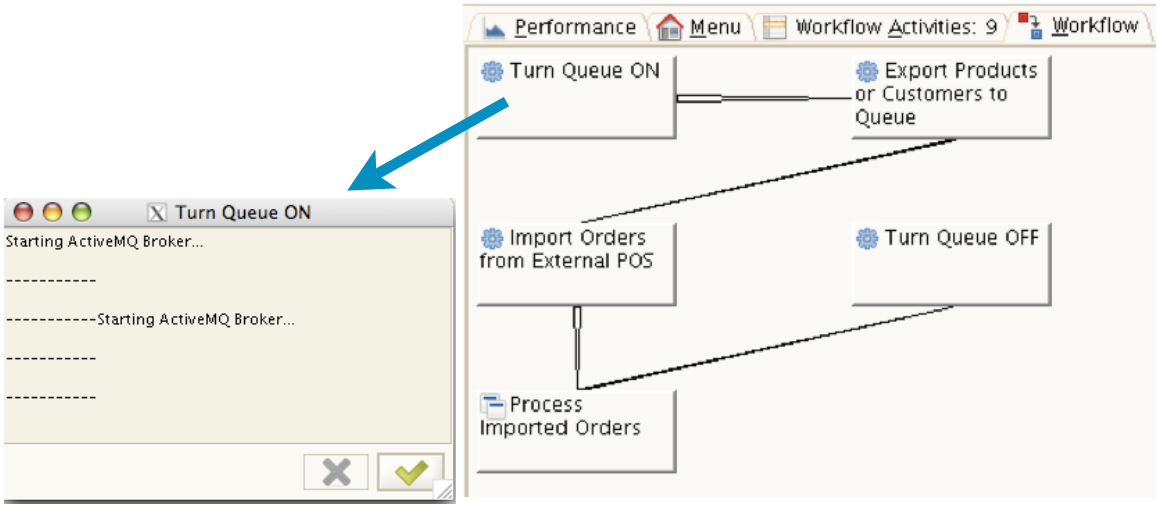
Then **RUN_silentsetup** and wait for a while for it to end in SUCCESS.

Then you have to go to your ADempiere database editor and run the contents of **/migration** folder. (You need not do this if you took the complete binary with Expdat.dmp that has everything applied). The SQL scipts in there will change the ADempiere menu to show the following.

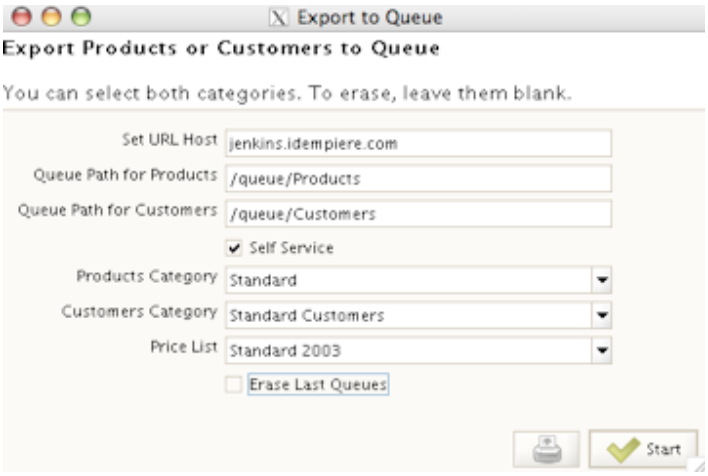
Menu Workflow for POS Synch

If you click on the first item, you can see that an idiot bungle-proof workflow is given to you. Notice that they are joined by flow lines.





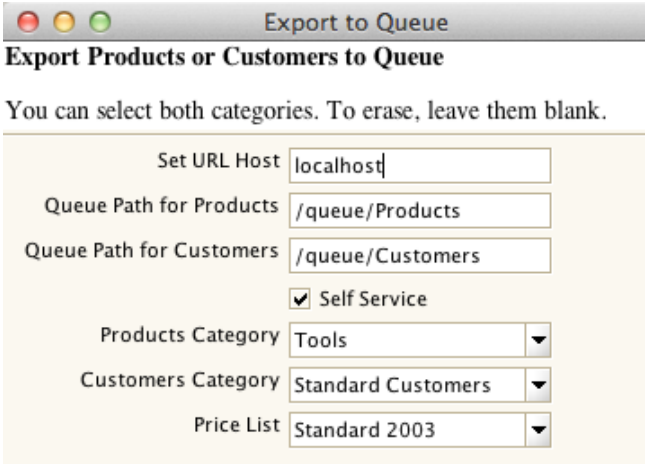
In your case using the default, without your own ActiveMQ, you do not need the Queue to start. (If you need your own ActiveMQ then refer to the suicide note in the next section).



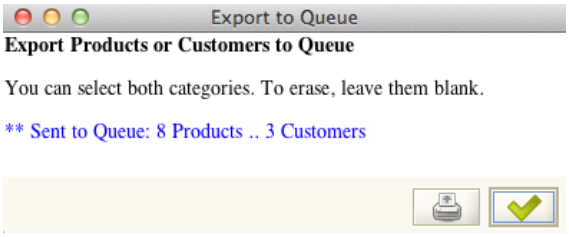
Now we come to the first proof of concept of the whole system. Which is the 'Export .. to Queue'. By pushing this button it will open up this screen on the right:

Settings for Exporting to Queue

As you can see, it is preset to the ActiveMQ URL host out in bavarian country.You can still use that or your own by replacing that with *localhost*. That is shown in the next image below. I will recommend you select *Tools* for *Products Category* as there are more interesting products there.



You have to choose a price-list or else no products will fire off. The Self Service box is to ensure that only such products are exported. *Pull it*. You will see a pop for Products and also a pop for Customers. Both sets of data are now sent to the queue.

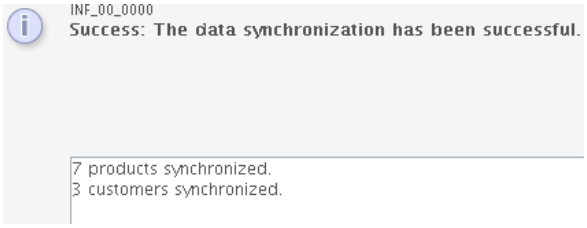


The export success display is showing that 8 items of the product list is sent and 3 customers information too. The product information comprises of POS Locator Name (matching the POS unique ID), Product Name, List Price (as selling price in the POS), Limit Price (as purchase price), and the Storage OnHandQty (for that POS Locator Name).

Storage Quantity By POS Location

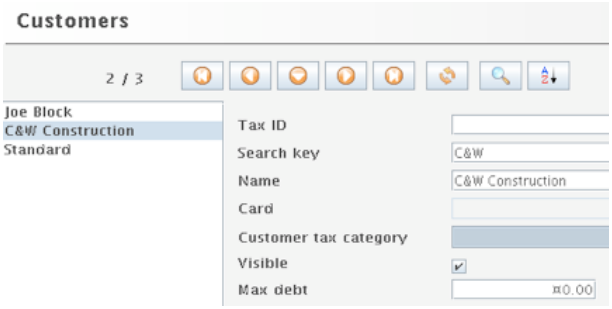
Now, within those 8 products, one of them are for two locators. Thus, 7 will be for the HQ Warehouse and 1 is for another locator. We are using 'warehouse locators' as POS stations. So now we have to show that our 'HQ Warehouse' only receive those 7 and not double that belongs to another store. Ensure that your **openbravo.properties** pos is set to **HQ Warehouse**.

Reminder: The ERP has just exported the data to an ActiveMQ service for depositing and not directly to the POS station. Later in the Developer Guide we will examine what happens under the hood.

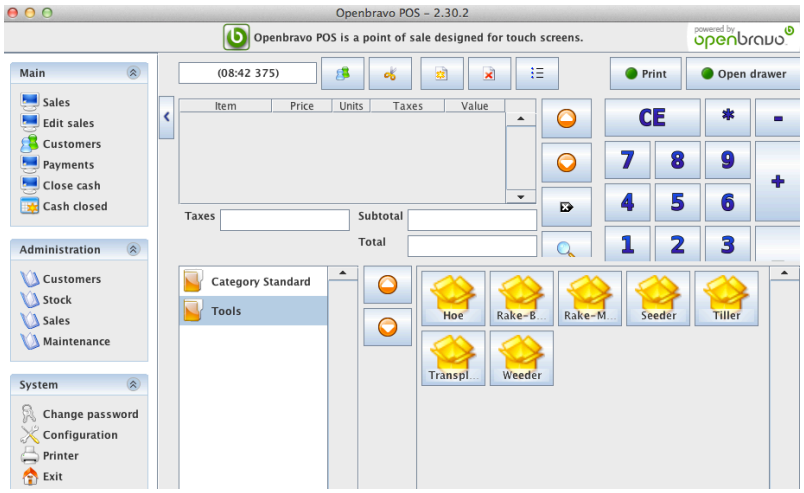


Go to your Openbravo POS and click on the *POS synchronization* button. Output shown here. The response is successful and the info box shows matching data received. Both of these data are now within the POS client database.

Go to the Customers item on the menu and do a default search to see for yourself.

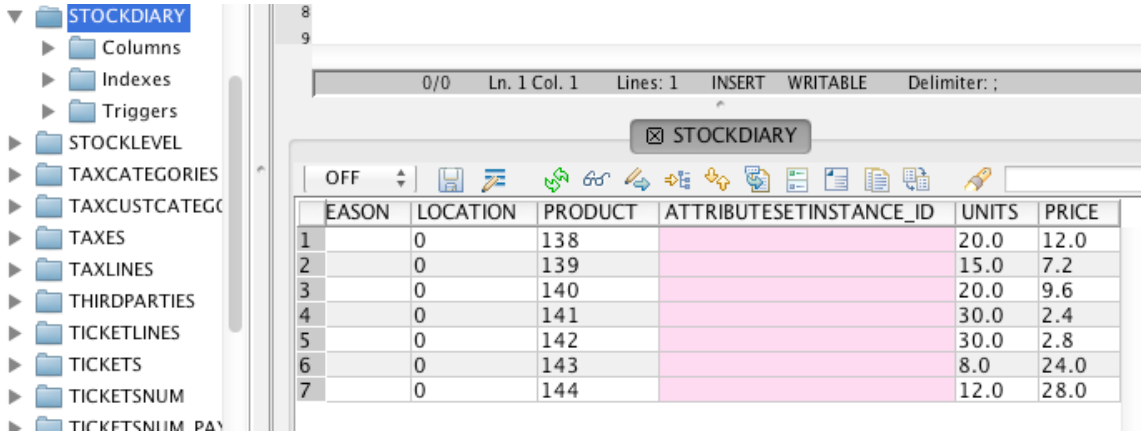


But there is more. See below. Go to the *Sales* screen. Click on the *Tools* category. You will see the 7 items imported into the Stock tables and ready for action.



Examining Database for Stock Movement

But there is more*. Go get a Derby database editor and peek into the database. To be exact look inside the *StockDiary* table. That keeps track of the POS stock movement.



Later we will peer into the StockDiary again to see if it does track stock movement.

We will next try making a sales for a known customer. Then we export the sales to the message queue. Then we ask headquarters to import that queue into the ERP. That will make a close loop on our integration. Do note that as inventory is integrated where in the final analysis the ERP will get upated by way of the Import Orders, that all its POS locators' inventories will reflect the POS locations inventories levels.

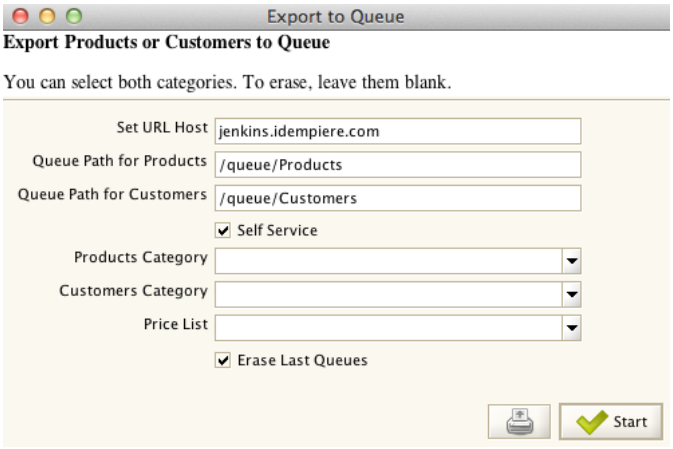
We have now gone deep enough into *Afghanistan*. Nothing should be impossible from here on. Go pat your *imam* on his back and pass around some *hashish* to smoke while admiring this free masterpiece.

* I heard from a friend that *But* or *And there is more* is a Steve Jobs trademark style of presenting his latest Apple product on the big stage. I sometimes speak on a big stage presenting the Open Source project and like to copycat such cliches.

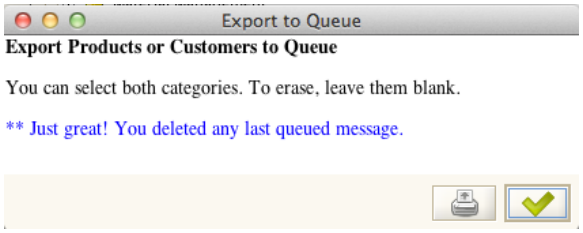
Erasing the Queues

We are not allowing the POS stations to consume the queue set so that multiple stations can keep taking the same queue information. The sensible way to control this is for the ERP centre to do that. For example look at this ActiveMQ admin panel. It is showing that the twin set of both Products and Customers queues have content. After the POS stations have confirmed they got the latest information, we call up the ERP Export to Queue Process and check the 'Erase Last Queues' box while leaving the two categories selection blank:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued
Customers	1	0	2	1
Orders	0	0	1	1
Products	1	0	1	0



After this process is carried out, a pop-up box will indicate if it is successful or not.



You can then see in the Queue Admin page to confirm that the queue set is indeed gone.

If there are more queues that you forgot to delete then you have to repeat this process. Usually there is only one queue set sent as each queue is a single long string that contains all the records needed. You should delete an unused queue set before exporting a fresh set.

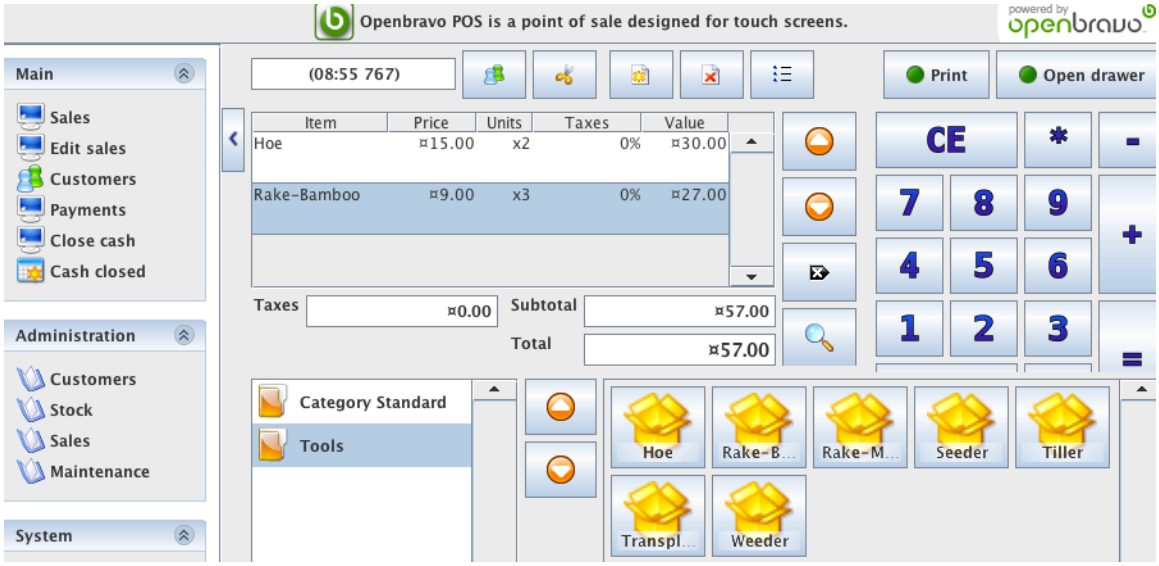
Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued
Customers	0	0	2	2
Orders	0	0	1	1
Products	0	0	1	1

MAKING A SALES

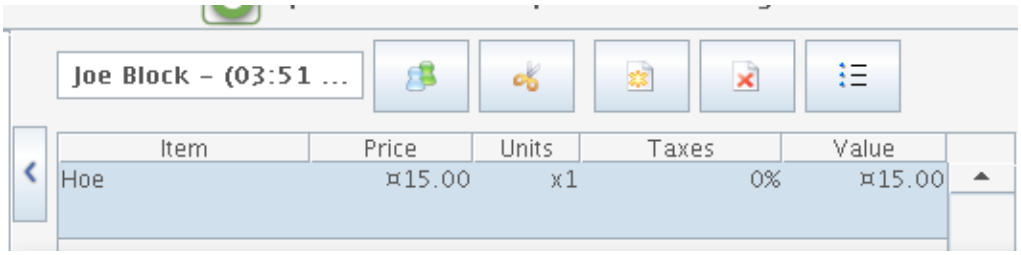
Click on **2** on the number-pad and click on **Hoe**.

Click on **3** on the number-pad and then click on **Rake-Bamboo**.



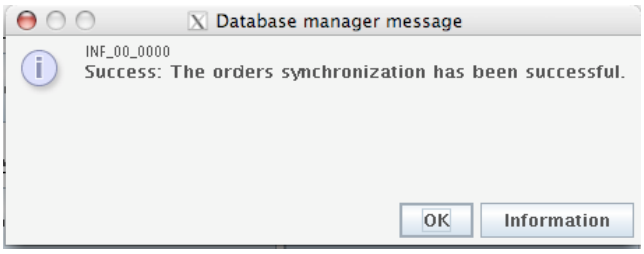
Click on = (the equal sign) on the right-side of the number-pad to checkout the sales. Accept the dollar notes, return the change and click OK.

Note that in my case above I did not click on a Customer to assign to. But in another sales, I did do that as shown below. To get the *Customers list*, click on the persons icon next to the field. Then select for example **Joe Block**.



ORDERS SYNCHRONIZATION

Now we are ready to go to the POS Maintenance menu to do *Orders synchronization*. Click on it and very quickly a message will pop out as seen here.



Click on the information button and it will indicate how many *orderlines* are created. In our case it is two.

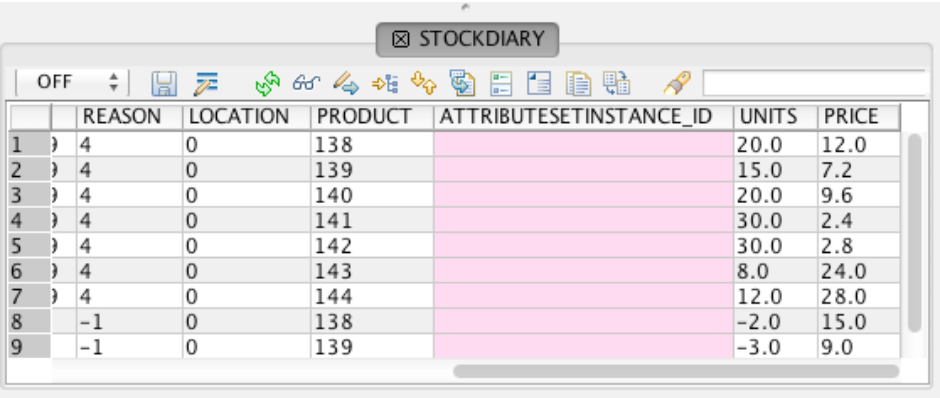
Remember that you need not do this synchronisation at every sales transaction. You have to devise a replication strategy be it on an hourly, daily or any other regulated time basis. The reason is that the other side has to synch their actions too. They have their preferred importing processing times or a particular format that makes sense to their operations. For example the HQ may want all POS stations sales in by midnight and processed overnight, so that the next morning they have a summarised sales performance report on the discussion or accounts controller table. They wouldn't want to see a huge stack of reports that is redundant. Also you may need to stage the operations as there may be mistakes during synching and you may find it hard to recover or fall back if your timings are too closed together.

Vice versa is also true. You may want your POS channels to pick up new stock information nightly and be prepared with the new pricing and stock levels the next morning before start of day. Again you would need to allow time for pricing changes to be displayed on the items sold. If such information comes in every other minute, there is bound to be confusion to both sales assistants and customers.

Thus such a replication that is stored and forward or pre-planned is more pragmatic and manageable. Also you need not have a 24X7 availability on your broadband. It can operate during off peak hours. Your technical or IT team managing the queue and batching of queues can review if there are mistakes or bugs and recover back in time without disrupting business or been disrupted by screams from the sales floor.

Remember the **StockDiary** table? Let us get back to that and see what is happening within it. We also wish to check the StockCurrent table. They are all shown below.

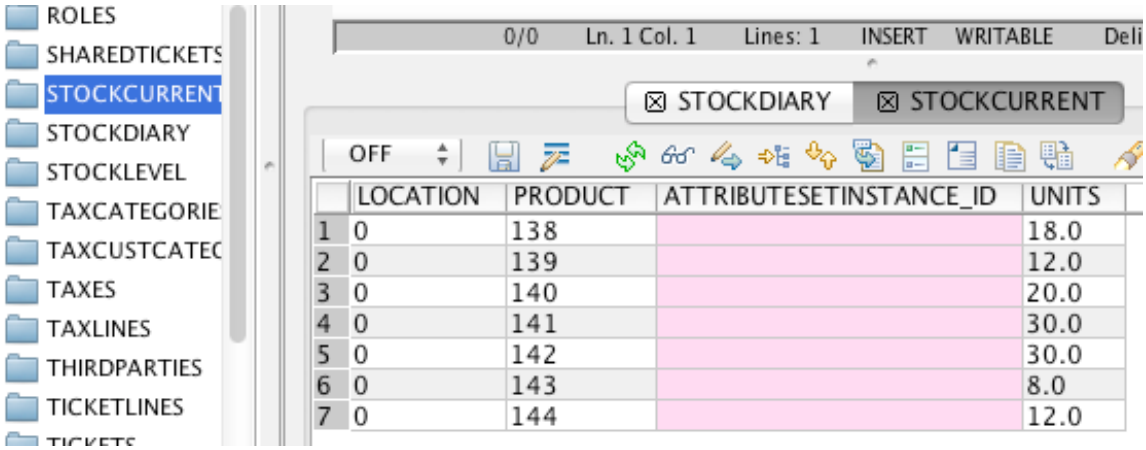
Checking Sales impact on Stock Movement


STOCKDIARY

OFF

	REASON	LOCATION	PRODUCT	ATTRIBUTESETINSTANCE_ID	UNITS	PRICE
1	4	0	138		20.0	12.0
2	4	0	139		15.0	7.2
3	4	0	140		20.0	9.6
4	4	0	141		30.0	2.4
5	4	0	142		30.0	2.8
6	4	0	143		8.0	24.0
7	4	0	144		12.0	28.0
8	-1	0	138		-2.0	15.0
9	-1	0	139		-3.0	9.0

You can see above that the Product IDs concerned (138 and 139) are tracked as to the release of 2 items and 3 items respectively. Below the **StockCurrent** table shows the nett inventory counts.


ROLES
SHAREDTICKETS
STOCKCURRENT
STOCKDIARY
STOCKLEVEL
TAXCATEGORIE
TAXCUSTCATEC
TAXES
TAXLINES
THIRDPARTIES
TICKETLINES

0/0 Ln. 1 Col. 1 Lines: 1 INSERT WRITABLE Deli

STOCKDIARY STOCKCURRENT

OFF

	LOCATION	PRODUCT	ATTRIBUTESETINSTANCE_ID	UNITS
1	0	138		18.0
2	0	139		12.0
3	0	140		20.0
4	0	141		30.0
5	0	142		30.0
6	0	143		8.0
7	0	144		12.0

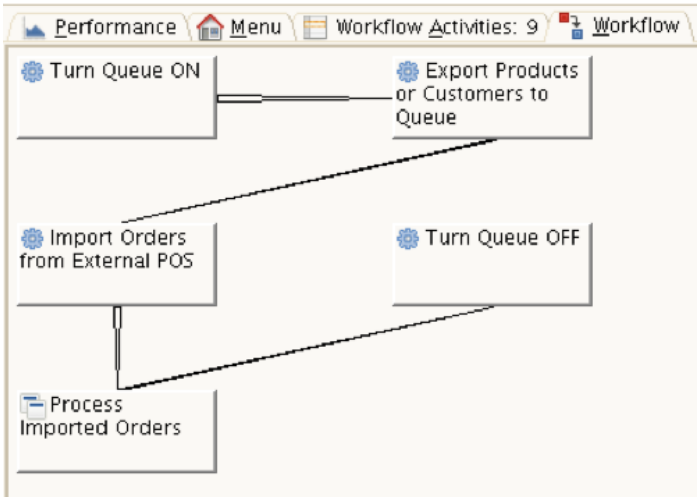
ID 138 is at 18 units from previously 20 and ID 139 is at 12 from previously 15.

Why I am showing this from the database and not from the Openbravo POS is because Openbravo did not implement the code to show it! It is fine because I have made note on this in the source I modified and uploaded. It is a gap that can be fulfilled on our next flying mission.

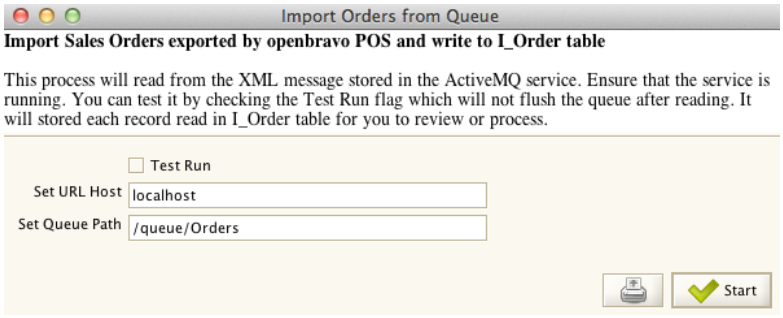
Onwards, we are now ready to get back to HQ and import from these POS station's queue.

IMPORTING ORDERS INTO ERP

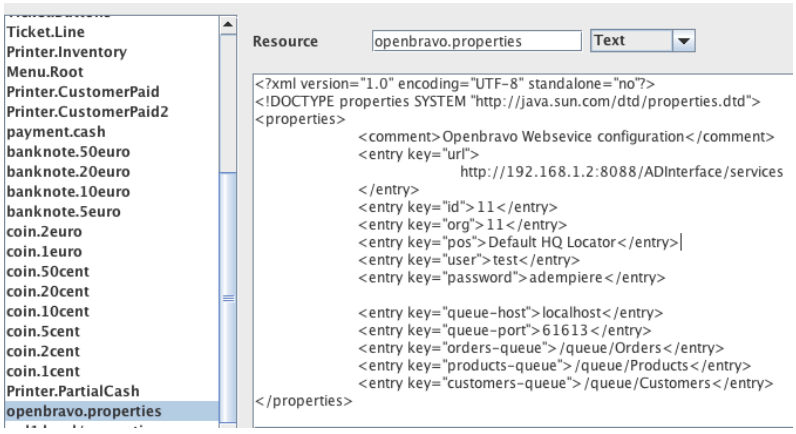
At the ERP application, we return back to the synchronisation menu:



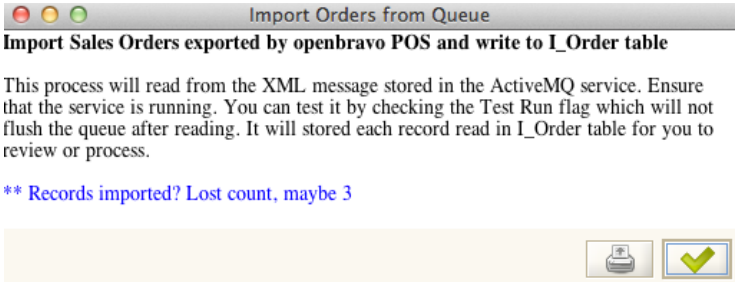
And call up the Import Orders from External POS process (change URL Host to yours):



I have put a test run control box so that the queue is not deleted after reading. This is useful if you are doing initial testing. During live operations this may not be used. If you still do, then you have to kill the queue from the ActiveMQ admin web page. Also note what is your URL Host. It has to be the same as the one the POS sent to. Similar with the Queue Path. Here is the **openbravo.properties** page to refresh your memory:



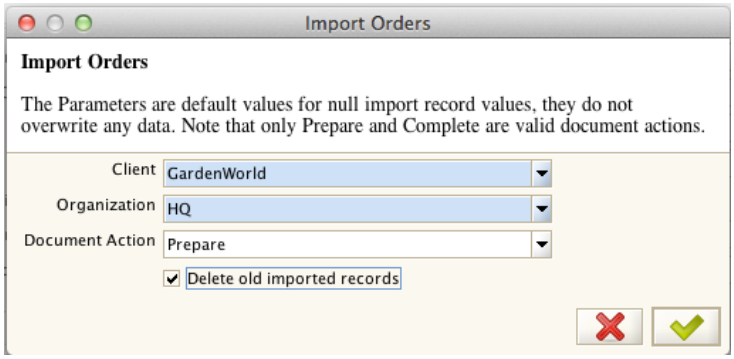
After clicking OK to the Import Orders from External Process, you should have a successful pop-up, something like this:



Don't believe a word it says. In our test case we should have 2, not 3 though. I was using a different screenshot from another test. Yes we do lose the package when handling too many images. What more with wires, straps, liquids, and code devices. From here we go to the *Process Imported Orders* step. You will find the *orderlines* in order (this is from another sample test to prove that the Customer header information is replicated nicely):

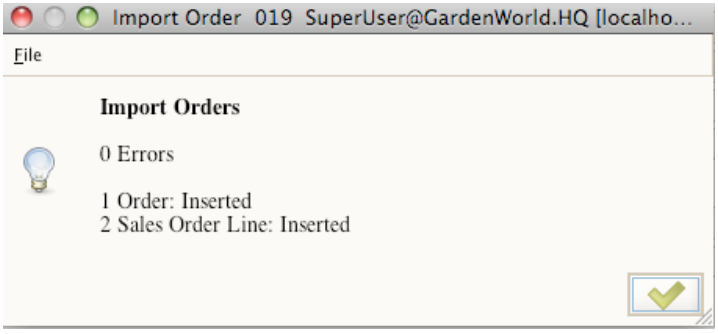
Order	Import Order	Imported	Order	Business Partner Key	Sales Order Line	Import Error Message
	1000052	☑	46_2011-09-08 00:00:00	JoeBlock	46 - 2011-09-0...	
	1000053	☑	47_2011-09-08 00:00:00	C&W	47 - 2011-09-0...	

Click on the process button and you will have a dialog box asking:



If you select *Complete* instead of *Prepare*, the POS Orders will process accordingly including update Invoice, Payment and Shipment tables.

Click OK and you should get the image here:



Depending your choice above, inventory status will change. You can check that out yourself from the *View InfoProduct* for the items processed. Note that **Prepare** will reserve the stock and affect the *Available* quantity, and **Complete** of a Sales Order will remove the *OnHand* quantity.

CLOSURE

I remembered long winding conversations within the community regarding POS and its integration. It was such a hot topic and some projects come and go tackling this seemingly infamous item. It is regarded as a killer app as many businesses operates from a Point of Sales. POS is a frontline app which doubles up as the most visible starting point of an ERP. When *POSterita* came into the scene it was with its own POS interface, which I must add, did make a hit (there goes a good pun). When *Compiere* left the scene, it has a poor looking POS interface but many try to toss it around hoping it gets better. When replication needs were looked at using web services and message queues, again lots of chatter in cyberspace.

My discovery is that this has become a simple task (for the experts that is) but they prefer not to disclose or contribute a final masterpiece well integrated fairly and squarely so as to continue the open source journey, where past wheels are not reinvented but stood upon for greater heights. But to be fair to them, I suspect that they are either too busy or disorganised with too much stitching work. This is a vicious circle. In the long run they will be more bogged down and their value dwindling as someone else will contribute and grab the flag of been the first to do so.

I don't mean that mine is the best piece of work. On the contrary it is lousy, but I did it. As the creator of *Linux*, Linus Torvalds in the famous classic debate with the original author of *Mimix*, Prof. Tanenbaum (which Linus took his source from), pointed out that it is more effective to release early rather than release properly.

With this contribution sponsored by SYSNOVA, Bangladesh, we hope that no longer conversations go around the usual questions of *whodunit* and '*are you contributing it?*' A framework source is all out there as I introduce in the next section and will continue to update it as there have been a rush of interest for such work. There is now hope to evolve swifter upon such shoulders of *Openbravo*, *Compiere*, *ActiveMQ* and other numerous contributors. I am focussing more on developing nations' needs around this particularly now here in Manila where there is a prospective pilot project for a large retail network of 33 outlets that is looking at using this solution to cover customer loyalty and micro-financing redemption.

Here I am denying the kudos, as I am convinced that innovation do happens elsewhere. We must evolve to sustain the eco-system collectively and always respecting the contributors and helping them in kind. If only contributors are to go to paradise, at least it won't be a lonely place.

FAQS

As this is a growing project, I maintained an online resource to capture its progress. I have also constructed a FAQ page to gather those recurring questions and doubts. Please refer to the link given below and and submit your comments there which will be edited and updated where relevant (minus the comedy). Meanwhile I paste the present list of likely questions I can think of below.

http://red1.org/adempiere/viewtopic.php?f=31&t=1371

1. **How are the queue messages formed?**
Answer: Each synchronisation process whether it be from the POS or from the ERP is stored as a single queue message or actually a single XML string. In that string, is made up of <detail> tags where each detail is a single record, be they an order or product.

2. **How would the system know which record belongs to which model?**
Answer: At the moment we are using different queue paths, i.e. for Products it is http://localhost:61613/Products and for Customer info it is ../Customers. Also in the code it checks for <DocType> to be the right model i.e. M_Product. So there is double measure to ensure the record is not taken wrongly. Even if it does, it should break into an exception and not proceed.

3. **What if the ERP server is offline or the web access is down or unstable?**
Answer: The beauty of ActiveMQ approach is that it is asynchronous, meaning that if either the POS or ERP host is down, there is no worry of getting the queue delivered in the end. Once either one is up and does a synch with the Queue server, the message is there for consumption.

4. **What if the Queue server is down?**
Answer: Then either ends cannot send or collect anything and have to wait for it to be up.

5. **What happens to the queued messages that was not consumed when the Queue goes down?**
Answer: There are not gone but stored in a disk cache database based on Derby at the queue server. They are still present if we are using PERSISTENT=TRUE setting when sending the message. At the moment we are using such a setting in the header property.

6. **What if we have many POS clients all over needing to access the same ERP export queue? Would it disappear after the first POS client consumed the queue?**
Answer: We are removing the removal of the queue for POS clients. The queue can only be removed by the ERP Server. Thus we leave the Products or Customers or any info for the POS stations on until all of them have taken it. In future we can incorporate an auto

ACK (acknowledgment) that ensures all POS clients have synched with the ERP info before removing it.

7. **How scalable is this thing?**
Answer: This may be manageable to whatever degree we want solely at the ActiveMQ server (with some code change at the ERP side). We have not tested in a multi POS and real life remotely spread environment. But due to the maturity of ActiveMQ and its many features, we can extend the system to cover high performance requirements. At the moment you can use this to test the concept and even apply it to some 10 POS stations quietly.

8. **How different is this from the previous work on replication and integration here and here.**
Answer: The work done here is simple and limited but more up-to-date. It is simpler because we are using the Stomp broker that has minimal code to achieve very powerful build-up of functionality within ActiveMQ service. Thus it can be extended when needed. It is simple but limited because we are not making an end to end replication but just a definite scope of synching POS orders to the ERP end and taking products, inventory and customer info from the ERP side onto the POS. It is more up-to-date because besides the Stomp approach, we are referring to latest openbravo POS for 2.3 and not 2.2. The ERP side is also referring to ADempiere 361 of CarlosRuiz which is itself on same page with iDempiere by Hengsin.

9. **Why should we refer to this instead of previous works?**
Answer: Firstly you should ask if this is what you want. This project is as its title says is for 'integrating openbravo POS to ADempiere ERP' and not 'replication' proper. If so, then this is the right place as it is more clearly well commented in its code work and well documented here as well as in a complete PDF guide. The guide will also advice openly and freely how developers can take this and extend it further. With red1.org and his peers (Carlos Ruiz, HengSin and others) there is absolutely no information hiding. There is also the best practice compliance standards imposed on his peers that all code must be (a) English language based in namespace, (b) backward compatible or reported where it is not, (c) well tested and reviewable.

10. **What about POSterita?**
Answer: Sadly it is not contributed to the standards required. Now no one is contributing it further nor maintaining it. Also here we are concerned about the synch here and not the POS per say.

10.1. **What about Openbravo ERP? Why not use it instead of ADempiere ERP?**
Answer: Yes that is logical since Openbravo effectively introduced new plugin framework using Pentaho data tool which necessitates doing that synching from Openbravo ERP itself. ADempiere does not have that.This means in order for ADempiere to do the same, we have to close this gap with Openbravo also, which is beyond this Proof of Concept's scope. If you wish to go completely Openbravo, then that path is recommended. If

you wish to stick to ADempiere, this is what we got. However we are community FOSS wheras Openbravo is commercial FOSS. There is a catch eventually.

11. **Can we also contribute code work to the project?**
Answer: We heard this many times. Just do it. We will find you and acknowledge you in if you did. But what we find is also (if there are such contributions) that work done is not to our standard of quality nor compliant to best practice. In those cases we do not respond nor wish to get involve with any controversy that will result if we do.

12. **What if I cannot contribute accordingly but I can sponsor more work with money?**
Answer: That is also nice, and you can write to us personally to negotiate the terms. However we must insist that our work is strictly FOSS and no code or info hiding is allowed. One of the issues with POSterita mentioned was that. You need not worry about trade secret client data as that is not code. Even client private configs is not code but metadata. We also give you back brand recognition when we publish you as the sponsor as the case here where SYSNOVA, Bangladesh is our main sponsor.

13. **What if we need this for our own commercial environment? Can we purchase any guarantee or support?**
Answer: This project is under clauses of GPL which like any other license (including proprietary) has no guarantee or liability for risks. However if you require paid implementation and support services, we can recommend you those that we know personally to have delivered responsible and accountable service known within our community:

THIS PEACE ENDS HERE.
NEWBIES, DO NOT CROSS THIS LINE.
HEAT SEEKERS ONLY.
==========

Note to the CIA: There is no hidden code in this document. But I do love caves and only got to stay in a farm. I have published its geo-location coordinates just to prove to that I got nothing to hide. In fact I am trying to get everyone to know what I am doing. That is why I do Open Source.

Development Guide

How POS integration was done

AND HOW YOU CAN CONTINUE THE MISSION

DISCLAIMER

Some assumptions I made about code, licensing and approach may be wrong. This POC is part of a *throwaway* approach not unheard of in *agile* and e*xtreme programming* where the end is desired first before a beginning becomes conclusive. Thus to frustrate stopping too much, sometimes wild assumptions are made. They are expected to be less critical than the objective itself and can always be corrected later. This hopes to avoid the seeming fate of many other contributions not seeing the light of day, where spending more time working on a sufficiently helpful documented wholesome prototype is better than caring code to perfection.

The language I used in this section is also a lot more cryptic. No, it is not Arabic nor Tagalog. It is just my way of assuming that you know where and how the software is usually encoded. That means, if you are too lazy to read up or do more research than push your joystick in front of endless computer games, you might consider changing your religion.

I have maintained some sort of logging of details as I worked through the code here: http://red1.org/adempiere/viewtopic.php?f=29&t=1356 . As I progressively commit the code to the SVN repositories, I put comments as descriptively as possible. I have dispensed of a more thorough discipline of issuing trackers as I regard this more of a POC (proof of concept) task as the challenge does not possess the luxury of helpful direct documentation. There also exist competing Web Services approach to connect the POS to the ERP as presently used by Openbravo ERP. The previous effort by Carlos Ruiz and Trifon Trifonov as stated in the Sponsorship page also reported some show stopper issue with the licensing. It is hoped that a quick release of a proven working prototype is more beneficial to the community than a finished product that may never show up.

I also benefited tremendously from hindsight after certain tasks were exhausted. I will attempt to share them whenever possible without disrupting the readability of the document. I believe some findings can be more beneficial in someone else's mind.

PICKING THE SOURCE

Continuing from last section, here are two more links for browsing through the source code that I am working on. They are the sources for the packages on page 12.

E- Source for B

http://adempiere.svn.sourceforge.net/viewvc/adempiere/branches/POS_ActiveMQ/

F- Source for C

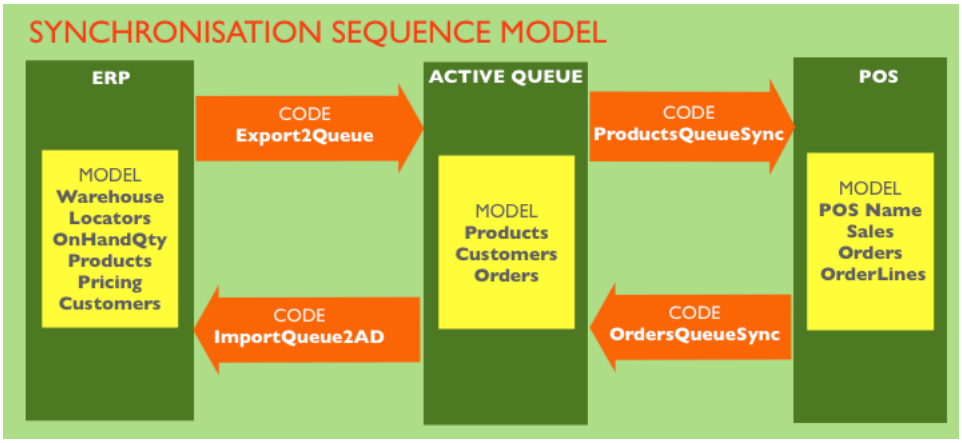
http://adempiere.svn.sourceforge.net/viewvc/adempiere/branches/FitnesseOBpos/

The link in **E** is the Openbavo POS source that I modified and retain it as a separate project. I am putting to the community to deliberate if this is a good idea to fork. Consider this a case forward for other contributors to make an informed judgement.

The patch source for ADempiere is so named *FitnesseOBpos* because it is intended via the *Gui* testing *Aspect4J* combined with the simple *Fitnesse* testing engine, which I am leaving it for later once I get to Munich to work more closely with Dominik on the Jenkins server. As is the usual experience, every time we open the hood into a project's source code we can find bugs and gaps, and Openbravo POS does not escape such mortality.

Code Cycle

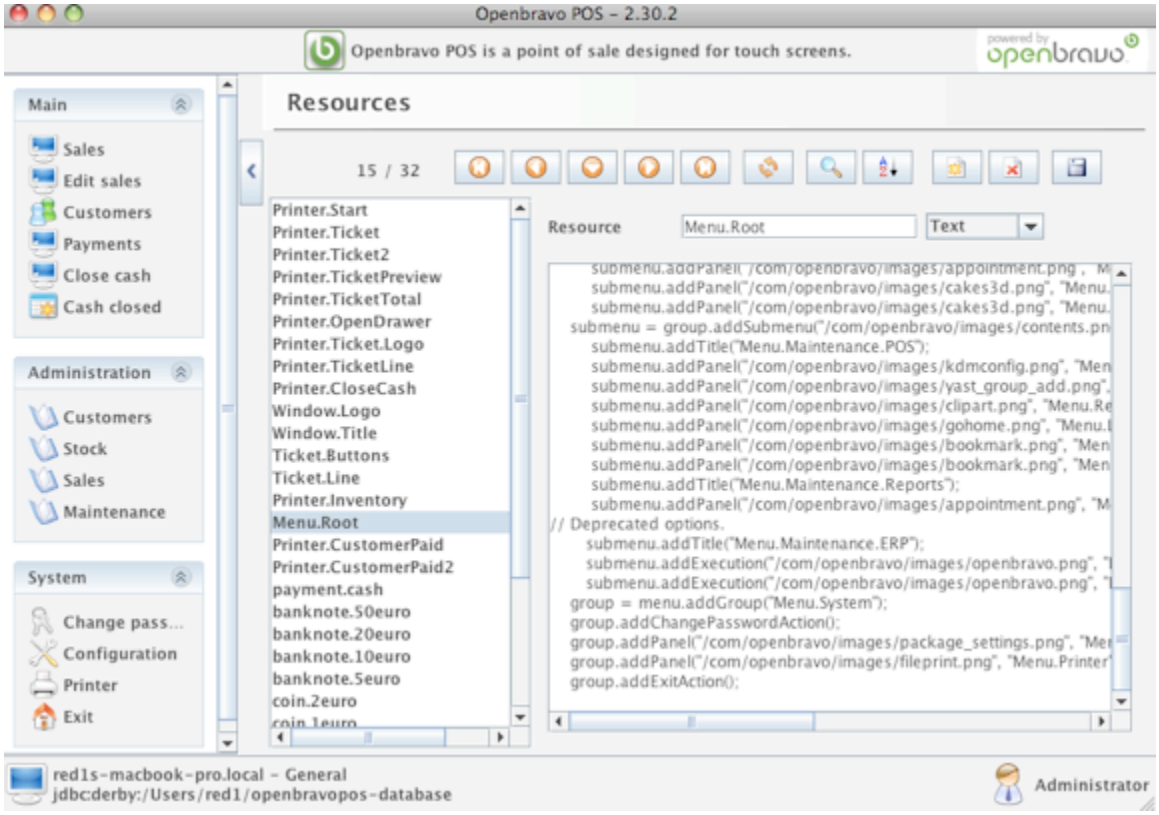
A good way to understand which code is which and doing what is to use this big map of the rugged *Pashtun* territory before going in. You will come across the mentioned code classes and



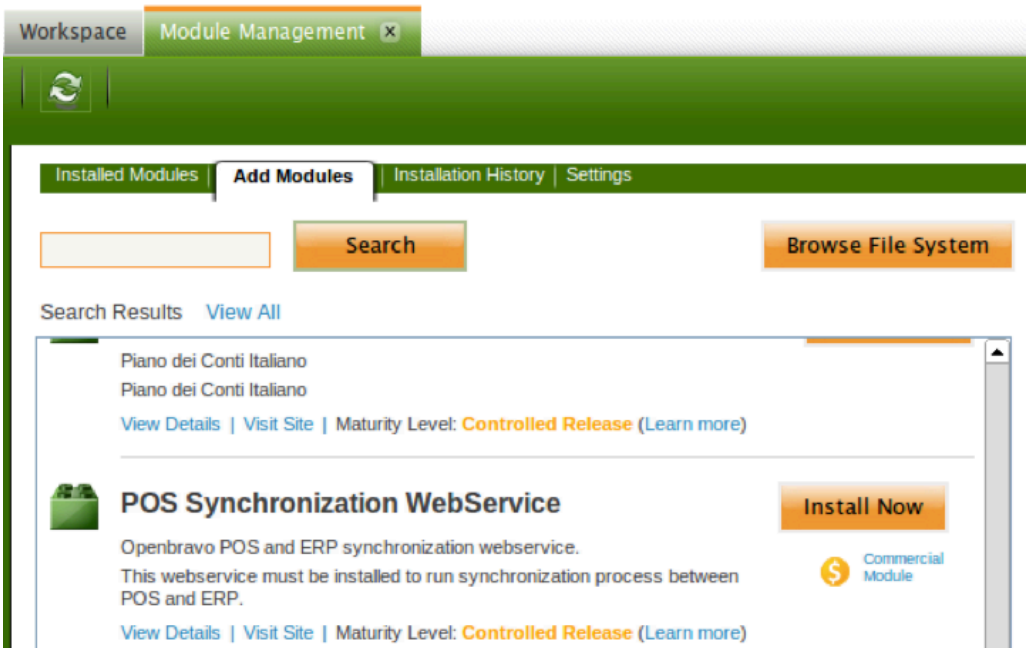
their model data as you proceed. In a later chapter there is a hi-level review of this diagram.

VERSION DEPRECATION

As I explained before, the present POS version is 2.3 and I am using the synching measures of 2.2. Thus Openbravo deprecated such measures as seen here below:



Note the remarks **//Deprecated Options** on the right panel towards the bottom. I uncommented the deprecation for the next three lines, and that restored the menu items under Maintenance for the two synching processes. Upon investigation at the Openbravo website I came across their new method of activating such an option (see next image).



You can see that it is embedded into the ERP Application and this seems a bit too tightly coupled for us to apply its current synching version to ADempiere. It is using the more direct realtime application of Web Services. Unless ADempiere also adopt the similar plugin approach and adjust its Web Services accordingly, it will not be modular within its own framework. Thus in effect, when I take the previous version 2.2 code for synching with ActiveMQ method, I have to fork the current 2.3 version as there are changes since 2.2 that affects it. That fork is now uploaded to our SVN as **/branches/POS_ActiveMQ**. Later we shall go through some of the gaps that are mostly changes to the data model of the POS. But I did try to apply minimal changes to avoid wider version gap.

You can also notice the $ sign with the words *Commercial Module*. If the synchronising module is sold and eventually protected from free download, this may drive more interest to the ADempiere integration as freebies are allergic to the word 'fee' instead of 'free'. Thus this lends credence to the argument for a technical fork by necessity as allowed by the GPL clause. As I have not got the chance to install Openbravo ERP and try that option I will reserve further comment as to what is the direction and options here. I welcome further feedback from other parties to enlighten us. Nevertheless, from my study of their code, I am happy with what I find. The POS is

quite handy and lightweight and resourceful. It can go a long way in the hands of good Open Source contributors.

The reason why I am choosing ActiveMQ messaging service over web services is due to the complexity of the later over the former for a quick POC. ActiveMQ also allow more robust handling of events as the POS or ERP can be offline as long as the MQ service is online which is very light and convenient. The object passing in XML format is also more easier to handle and allow further anticipated scalability of data model design.

Version Gap in POS

Between v2.2 and v2.3 there are gaps as to its coding for synchronisation and its data model. I have to reverse engineer (mostly guess work from what the conflict with the tables were) and patch the code to get through in reading the data for synchronising. Mostly the trouble spots revolve around the handling code such as `DataLogicSales` usage, `TicketInfo` model use which breaks methods such as `loadTicket` the new v2.3 has Attributes field as:

```
SELECT L.TICKET, L.LINE, L.PRODUCT, L.ATTRIBUTESETINSTANCE_ID, L.UNITS,
L.PRICE, T.ID, T.NAME, T.CATEGORY, T.VALIDFROM, T.CUSTCATEGORY, T.PARENTID,
T.RATE, T.RATECASCADE, T.RATEORDER, L.ATTRIBUTES FROM TICKETLINES L, TAXES T
WHERE L.TAXID = T.ID AND L.TICKET = ? ORDER BY L.LINE
```

as compared to the previous v2.2 reference:

```
SELECT L.TICKET, L.LINE, L.PRODUCT, L.UNITS, L.PRICE, T.ID, T.NAME, T.CATEGORY,
T.CUSTCATEGORY, T.PARENTID, T.RATE, T.RATECASCADE, T.RATEORDER, L.ATTRIBUTES
FROM TICKETLINES L, TAXES T WHERE L.TAXID = T.ID AND L.TICKET = ?
```

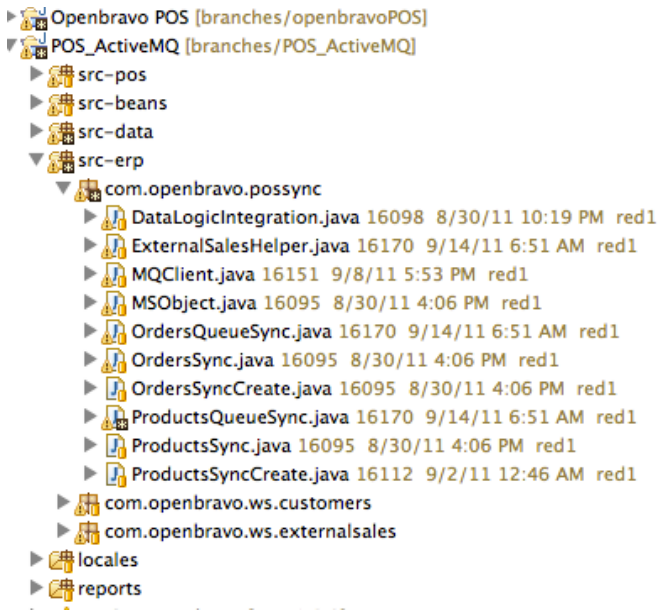
Lack of good ORM

The archaic JDBC calls is a big nuisance as you have to really count backwards to guess which data type or value it is really reading in some model handling. An example is below:

```
Object[] diary = new Object[7];
 diary[0] = new Double(p.getPriceBuy()); //UUID.randomUUID().toString();
 diary[1] = now;
 diary[2] = diff > 0.0 ? MovementReason.IN_MOVEMENT.getKey()
                       : MovementReason.OUT_MOVEMENT.getKey();
 diary[3] = warehouse;
 diary[4] = p.getID();
 diary[5] = UUID.randomUUID().toString(); //new Double(diff);
 diary[6] = new Double(diff);//new Double(p.getPriceBuy());
```

XML STRING PASSING

Some weeks were spent studying the issues and the code involved and thus came to the conclusion that we can reuse the present POS helper classes but switch the final commit of messages as an ActiveMQ connection. I copied and refactored from *OrdersSync.java* and ProductsSync.java as **OrdersQueueSync.java** and **ProductsQueueSync.java** to do my changes in there. More of this in a later chapter.



As shown in the Eclipse Package Explorer view above, my new branch for OpenbravoPOS is POS_ActiveMQ. In the **src-erp** folder, **com.openbravo.possync** package, I copied the Sync javas by adding the *Queue* word to it.

The idea of XML string in the message queue is borrowed from Ajit Kumar's *CookBook* which gave me the sample code for the initial handshake with an *ActiveMQ*. (please refer to the book, pages 192/3).

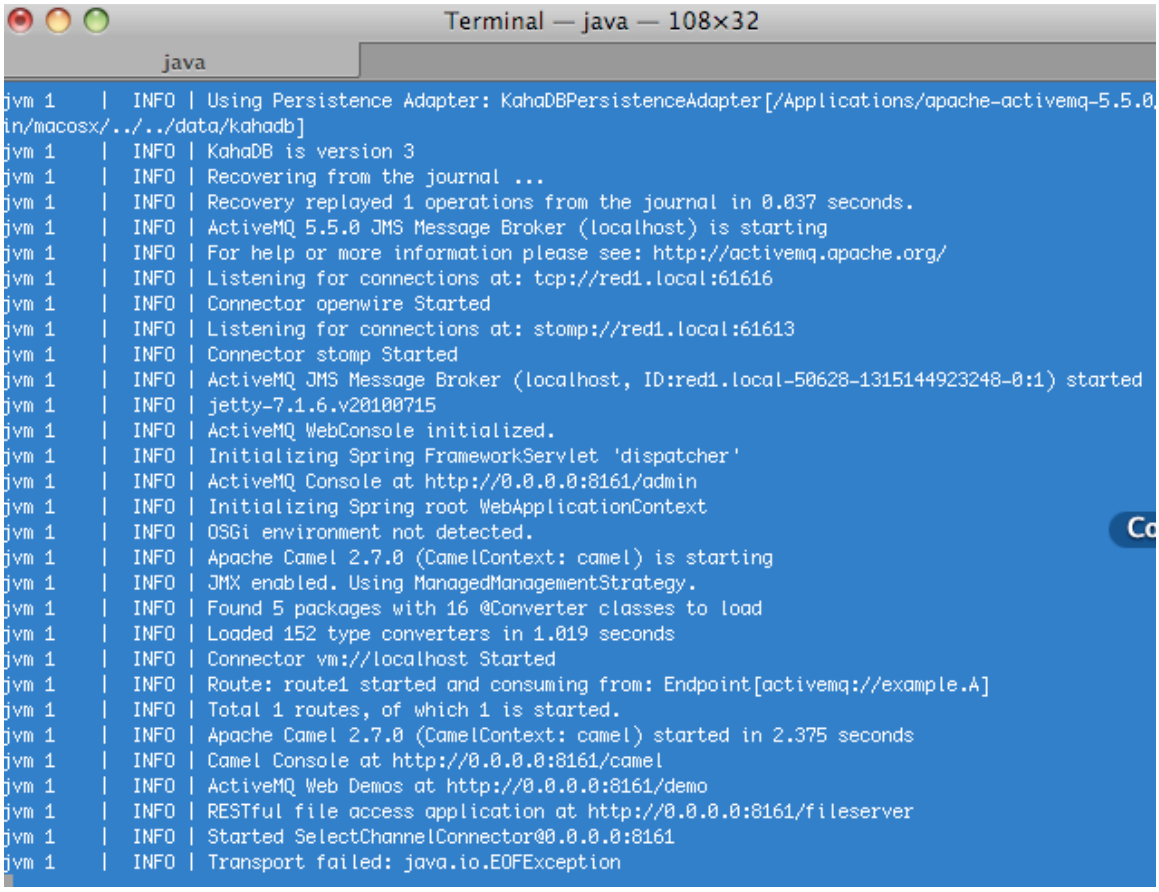
The ERP side of exporting data for synching with the POS is a simple SQL query behind an AD Process with parameter selection by the user to decide what type of data is to be sent to the POS.

STARTING OWN ACTIVEMQ

If you have downloaded your own *ActiveMQ* from the *Apache* project website you can start it within ADempiere by clicking on that first icon shown in the workflow. But you need to activate the Stomp protocol within its **activemq.xml** like this:

```
<transportConnector name="stomp" uri="stomp://0.0.0.0:61613"/>
```

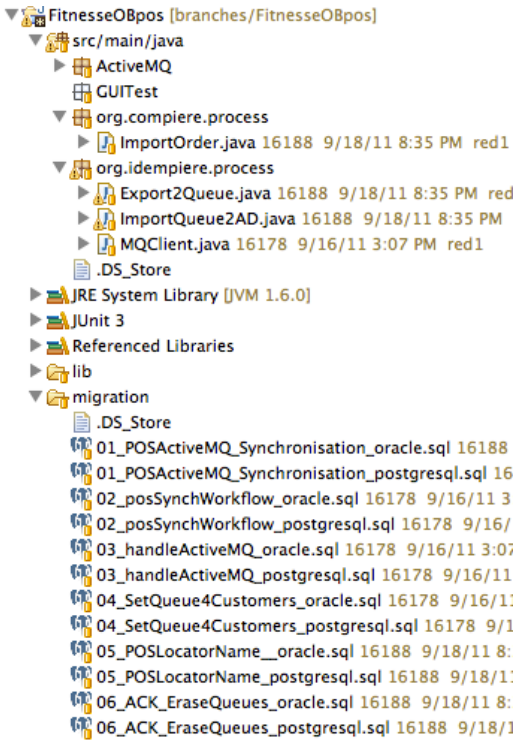
If it does not start from ADempiere's workflow icon, or there is a permissions conflict issue, you can manually start it at a terminal box with '**activemq start**' or '**activemq console**'. With the *activemq console* command for your *black ops* you should see descriptive prompts inside the terminal box:



ADEMPIERE SIDE

All the modifications needed on the ADempiere ERP side is housed in a separate project also under the ADempiere SVN repository as

https://adempiere.svn.sourceforge.net/svnroot/adempiere/FitnesseOBpos



The **ActiveMQ** package contains the two simple tests I picked up from the *CookBook* which refers to the ActiveMQ online resource. They are meant for a quick test of sending and receiving the queue during development work. I used them whenever I think something is wrong with my actual queue handshake code.

The **org.idempiere.process** package has the actual process code used in the ADempiere menu to do both exporting the selected data model out and importing the XML queued messages in. The MQClient helper class is to manage the ActiveMQ process and settings.

The **org.compiere.process** has the **ImportOrder** overriding the ADempiere main version as there is a gap which is explained elsewhere.

The **migration** folder contains all the necessary changes to the ADempiere *Application Dictionary*. The scripts are generated using the *Log Migration* tool in the *Preferences* window with the Dictionary Maintenance option turned on. That means *Centralized ID Management* is applied where needed.

The **migration** folder is zipped and uploaded along with other important artifacts of the project.

Upgrading ActiveMQ 5.5.0

Now, ADempiere is already using *ActiveMQ* for its *Replication Module* but its *jar* is at version 5.0.0. So we have to upgrade that to the latest version that has the **Stomp** broker which our message communication uses. In order to do that, you have to do the following modification to your source and compile a binary again.

1. Replacing the jar in <source>/**tools/lib**. This jar is put up at SVN below.
   http://sourceforge.net/projects/adempiere/files/openbravoPOS/activemq-all-5.5.0.jar/download

2. Modify the *.classpath* and *build.xml* files according to the diff patch below:

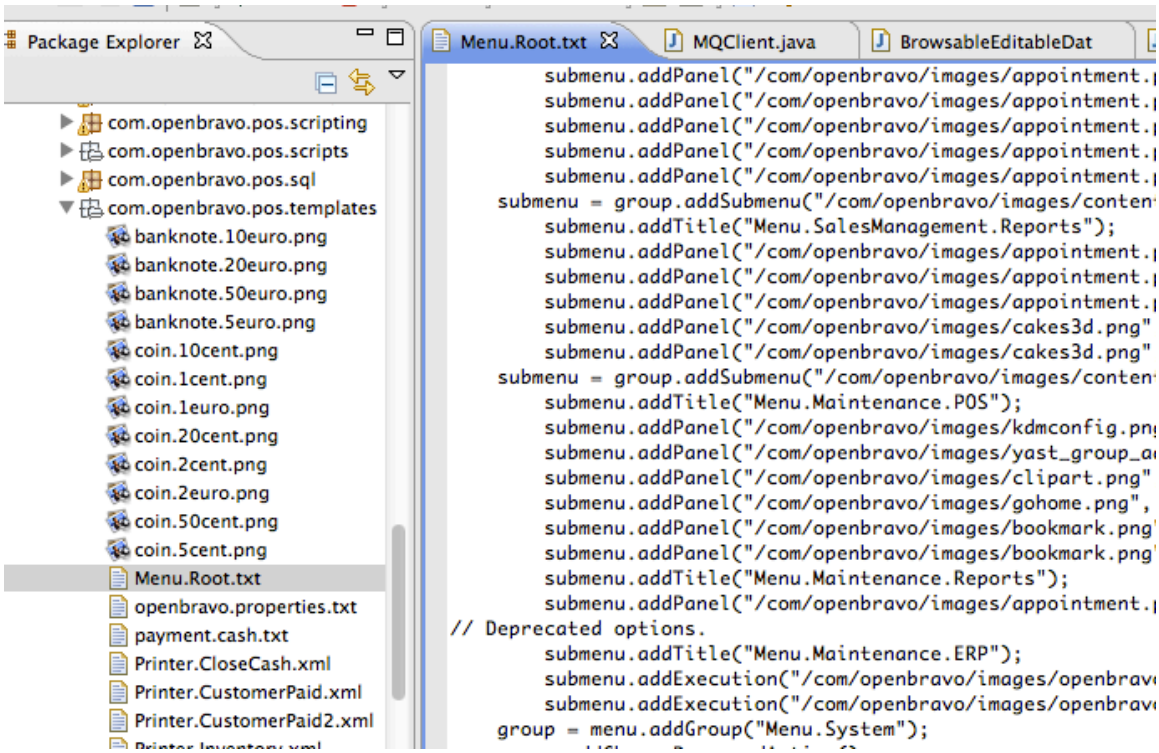
```
diff -r d2b91ab0a8c1 .classpath
--- a/.classpath    Thu Aug 04 23:53:00 2011 -0500
+++ b/.classpath Sun Sep 18 12:33:21 2011 +0800
@@ -20,7 +20,7 @@
        <classpathentry kind="src" path="webCM/src/main/servlet"/>
        <classpathentry kind="src" path="migration/src"/>
        <classpathentry kind="src" path="posterita/posterita/src/main"/>
-       <classpathentry exported="true" kind="lib" path="tools/lib/activemq-core-5.0.0.jar"/>
+       <classpathentry kind="lib" path="tools/lib/activemq-all-5.5.0.jar"/>
        <classpathentry exported="true" kind="lib" path="tools/lib/ant-commons-net.jar"/>
        <classpathentry exported="true" kind="lib" path="tools/lib/ant.jar"/>
        <classpathentry exported="true" kind="lib" path="tools/lib/ant-launcher.jar"/>
diff -r d2b91ab0a8c1 tools/build.xml
--- a/tools/build.xml       Thu Aug 04 23:53:00 2011 -0500
+++ b/tools/build.xml       Sun Sep 18 12:33:21 2011 +0800
@@ -271,7 +271,7 @@
                        <patternset refid="manifest.exclude" />
                </zipfileset>
                <!-- JMS -->
-               <zipfileset src="lib/activemq-core-5.0.0.jar">
+               <zipfileset src="lib/activemq-all-5.5.0.jar">
                        <patternset refid="manifest.exclude" />
                </zipfileset>
                <!-- C3P0 connection pool -->
@@ -434,7 +434,7 @@
                        <patternset refid="manifest.exclude" />
                </zipfileset>
                <!-- JMS -->
-               <zipfileset src="lib/activemq-core-5.0.0.jar">
+               <zipfileset src="lib/activemq-all-5.5.0.jar">
                        <patternset refid="manifest.exclude" />
                </zipfileset>
                <!-- C3P0 connection pool -->
diff -r d2b91ab0a8c1 tools/lib/activemq-core-5.0.0.jar
Binary file tools/lib/activemq-core-5.0.0.jar has changed
```

Now you can rebuild with <**source**>**utils_dev/RUN_build** and <**binary**>**RUN_setup** or silentsetup as done in the previous section (page 17).

DEMYSTIFYING POS SOURCE

Using the Menu.Root properties

As seen in the earlier chapter of XML String Passing, The Openbravo POS code is forked POS_ActiveMQ. We now begin to examine more closely its architecture and tools. The *Menu.Root* properties within Openbravo POS is a handy configurator of the application not unlike ADempiere's use of the *App Dictionary*. It allows the menu and its contents or properties to be applied at runtime.



Remember at page 15, during the launching of the POS Maintenance screen we view its Properties page. This is now looking at it at the source level, which is housed in the **templates** folder. You can see the Deprecated options at the bottom right. The whole group for these sync buttons selection is thus this:

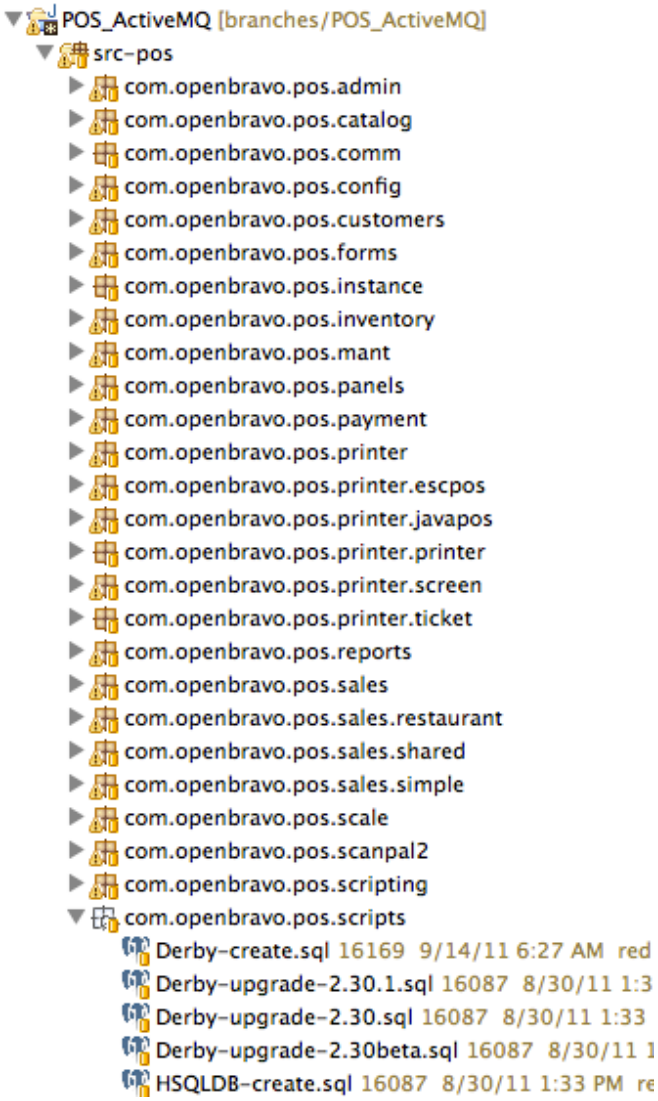
```
// Deprecated options.
        submenu.addTitle("Menu.Maintenance.ERP");
        submenu.addExecution("/com/openbravo/images/openbravo.png","Menu.ERPProducts",
                                                "com.openbravo.possync. ProductsSyncCreate");

         submenu.addExecution("/com/openbravo/images/openbravo.png","Menu.ERPOrders",
                                                "com.openbravo.possync. OrdersSyncCreate");
```

These long lines define the menu items' appearance and what java classes it triggers namely **ProductsSyncCreate** and **OrdersSyncCreate**.

If you remember earlier I stated that I duped these classes from their originals. So I replaced the original names here. This is where we inform the POS that we be taking over from them.

Creating openbravo.properties

Remember at that page 15 we are looking at the openbravo.properties page called from the Menu.Root. In the official Openbravo POS version such a page does not exist and we are advised how to create them by clicking on the new button and typing the contents there.



Similar to the concept of *Compiere's AD*, such settings define the application menu properties but how do you get that into the database during setup? In ADempiere we also have to input it directly at the System GUI. For this POS, it can be done via its DB creation scripts. Referring back to the package of the template source, move some folders up within the **src-pos** parent and you can see that as shown on the bottom right. It has db creation scripts for all 4 popular databases: **Derby, HSQL, MySQL, PostgreSQL** and **Oracle**.

Open up the **Derby-create.sql** and search for **openbravo.properties** which is already done the additional insert:

```
INSERT INTO RESOURCES(ID, NAME, RESTYPE, CONTENT) VALUES('31',
'openbravo.properties', 0,
$FILE{/com/openbravo/pos/templates/openbravo.properties.txt});
```

As you can see it is defining an item inside the **Resources** table. This tells us that the Resources icon in the Maintenance menu leads to what is inside the Resources table. And this script creates a new entry that we wanted. And it picks up its content from the same location as the Menu.Root file which is

```
templates/openbravo.properties.txt
```

So now you can go to the **templates** folder to fetch and open up that file which is restored but extended too.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
   <comment>Openbravo Websevice configuration</comment>
   <entry key="url">
          http://192.168.1.2:8088/ADInterface/services
   </entry>
   <entry key="id">11</entry>
   <entry key="org">11</entry>
   <entry key="pos">HQ</entry>
   <entry key="user">test</entry>
   <entry key="password">adempiere</entry>

   <entry key="queue-host">jenkins.idempiere.com</entry>
   <entry key="queue-port">61613</entry>
   <entry key="orders-queue">/queue/Orders</entry>
   <entry key="products-queue">/queue/Products</entry>
   <entry key="customers-queue">/queue/Customers</entry>
</properties>
```

From here you will get an idea on how to extend the POS system particularly from its main menu structure.

The *openbravo.properties* is extended to send parameters (*entry key*) pertaining to the use of the ActiveMQ service. The earlier WebService configuration is still intact as seen above which points to **/ADInterface/services**. This means that we should be able to make use of a competing service at the same time.

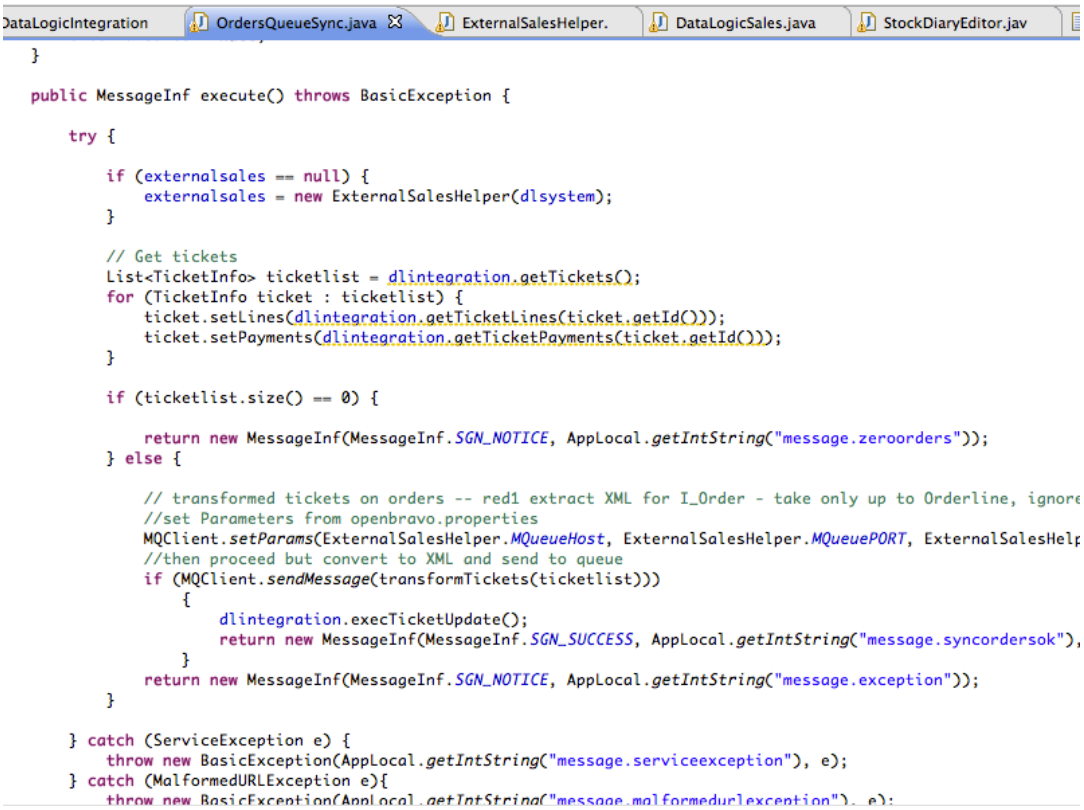
SYNCH IN ACTION

In this next topic you will see into the ExternalSalesHelper class where the params are fetched for the POS system.

```
public class ExternalSalesHelper {

    private ExternalSalesImpl externalSales;
    private WebServiceImpl externalCustomers;
    public static String MQueueHost;
    public static int MQueuePORT;
    public static String OrdersQueue;
    public static String ProductsQueue;
    public static String CustomersQueue;
    private String m_sERPUser;
    private String m_sERPPassword;
    public static String m_iERPId;
    public static String m_iERPOrg;
    public static String m_iERPPos;

    /** Creates a new instance of WebServiceHelper */
    public ExternalSalesHelper(DataLogicSystem dlsystem) throws BasicException, ServiceEx

        Properties prop = dlsystem.getResourceAsProperties("openbravo.properties");
        if (prop == null) {
            throw new BasicException(AppLocal.getIntString("message.propsnotdefined"));
        } else {
            String url = prop.getProperty("url");
            if (url == null || url.equals("")) {
                throw new BasicException(AppLocal.getIntString("message.urlnotdefined"));
            } else {
                MQueueHost = prop.getProperty("queue-host");
                MQueuePORT = Integer.parseInt(prop.getProperty("queue-port"));
                OrdersQueue = prop.getProperty("orders-queue");
                ProductsQueue = prop.getProperty("products-queue");
                CustomersQueue = prop.getProperty("customers-queue");
```

This partial mugshot gives you the idea what is happening. All the new params are passed to *public static variables* to be used during queuing and eventual synching.

Then the actual synching is done by the classes referred earlier. Below is the execute method implementation which infiltrates with the XML Queue passing. This should give you a tremendous head-start on your mission. You can refer to the methods called by this class and compare with the original class. You will note that we avoid introducing any new pattern so as to allow a clean synch POC to happen, except for covering the missing gaps which I am explaining elsewhere. Remember that I duplicated this whole class from the original before modifying it so as to avoid contaminating what is done before and destroying important evidence.



Note that these classes are generated by the WebServices toolset from Java as shown in the remarks sections of the classes involved:

```
This file was auto-generated from WSDL
```

Next we take a good look at the XML creation code that is applied for all the XML strings.

Products XML String

The product xml string is used by the ERP system to send out its list of products according to a selected **Category**, and **PriceList**. It is formed by the class method **Export2Queue** which is a *AD process* class. That class also double up to send the Customers XML. Part of the method for products fetching is given below:

```
private String ProductXML(List<MProduct> query, MProductCategory prodcat) {
  try {
          StringWriter res = new StringWriter();
          XMLStreamWriter writer = XMLOutputFactory.newInstance().createXMLStreamWriter(res);
                writer.writeStartDocument();
                    writer.writeStartElement("entityDetail");
                    writer.writeStartElement("type");
                    writer.writeCharacters("openbravoPOS");
                    writer.writeEndElement();

                    for (MProduct product:query) {
                              //get pricing details from Product's ProductPrice
                              int anyPrice = new Query(Env.getCtx(), MProductPrice.Table_Name,
                                        MProductPrice.COLUMNNAME_M_PriceList_Version_ID
                                        + "=? AND "+MProductPrice.COLUMNNAME_M_Product_ID+"=?" ,null)
                                        .setParameters(pPriceListVersionID, product.getM_Product_ID())
                                        .count();
                              //above checks if any exist before fetching them. If not, default should be zero.
                              if (anyPrice==0) continue;
                                        MProductPrice price = new Query(Env.getCtx(),MProductPrice.Table_Name,
                                                  MProductPrice.COLUMNNAME_M_PriceList_Version_ID+
                                                  "=? AND "+MProductPrice.COLUMNNAME_M_Product_ID+"=?", get_TrxName())
                                                  .setParameters(pPriceListVersionID, product.getM_Product_ID())
                                                  .first();

                              //get Storage Qty based on POS ID / OrgValue (this assumes that we cannot be sending
                              Product that has no Storage
                              int anyStorage = new Query(Env.getCtx(), MStorage.Table_Name,
                                        MStorage.COLUMNNAME_M_Product_ID+"=?", get_TrxName())
                                        .setParameters(product.getM_Product_ID())
                                        .count();
                              if (anyStorage==0) continue;
          //the count will determine more XML details according to each storage (POS ID/Org) as separate detail
                              List<MStorage> stores = new Query(Env.getCtx(), MStorage.Table_Name,
                                        MStorage.COLUMNNAME_M_Product_ID+"=?",null)
                                        .setParameters(product.getM_Product_ID())
                                        .list();

                              for (MStorage store:stores){
                                        //get Warehouse Name from Store Locator to match to POS Locator Name
                                        MLocator loc = new Query(Env.getCtx(),MLocator.Table_Name,
                                                  MLocator.COLUMNNAME_M_Locator_ID +"=?",get_TrxName())
                                                  .setParameters(store.getM_Locator_ID())
                                                  .first();
                                        count++;
                                         writer.writeStartElement("detail");

                                        writer.writeStartElement("DocType");
                                            writer.writeCharacters(MProduct.Table_Name);
                                            writer.writeEndElement(); //</DocType>

                                        //Warehouse Name as POS name
                                        writer.writeStartElement("POSLocatorName");
                                        writer.writeCharacters(loc.getWarehouseName());
                                        writer.writeEndElement();

                                        writer.writeStartElement("ProductName");
                                        writer.writeCharacters(product.getValue());
                                        writer.writeEndElement();
```

```
            //Storage Data - QtyOnHand
            writer.writeStartElement("QtyOnHand");
            writer.writeCharacters(store.getQtyOnHand().toString());
            writer.writeEndElement();

            //ProductCategory model
                    writer.writeStartElement(MProductCategory.COLUMNNAME_M_Product_Category_ID);
            writer.writeCharacters(Integer.toString(product.getC_TaxCategory_ID()));
            writer.writeEndElement();
            writer.writeStartElement("CategoryName");
            writer.writeCharacters(prodcat.getName());
            writer.writeEndElement();

            writer.writeStartElement(MProduct.COLUMNNAME_M_Product_ID);
            writer.writeCharacters(Integer.toString(product.getM_Product_ID()));
            writer.writeEndElement();

          //Tax model i.e. Name, Percentage? (Rate is complex in AD)
          MTaxCategory tax = new Query(Env.getCtx(),MTaxCategory.Table_Name,
                    MTaxCategory.COLUMNNAME_C_TaxCategory_ID+"=?",null)
                              .setParameters(product.getC_TaxCategory().getC_TaxCategory_ID())
                              .first();
                    writer.writeStartElement(MTaxCategory.COLUMNNAME_C_TaxCategory_ID);
                    writer.writeCharacters(Integer.toString(tax.getC_TaxCategory_ID()));
                    writer.writeEndElement();
                    writer.writeStartElement("TaxName");
                    writer.writeCharacters(tax.getName());
                    writer.writeEndElement();

                    writer.writeStartElement(MProduct.COLUMNNAME_UPC);
                    writer.writeCharacters(product.getUPC());
                    writer.writeEndElement();

            //Pricing info
                    writer.writeStartElement(price.COLUMNNAME_PriceList);
                    writer.writeCharacters(price.getPriceList().toString());
                    writer.writeEndElement();

                    writer.writeStartElement(price.COLUMNNAME_PriceLimit);
                    writer.writeCharacters(price.getPriceLimit().toString());
                    writer.writeEndElement();

                    writer.writeEndElement(); //</detail>
          // if there exist Storage and exist Price
                    }
  // Product
}
```

POS Locator Name handling

Note the fetching of the *Warehouse Name* from the *Stores* via its *Locator*. This name is to match with the *POS Locator Name*. In the ERP, such Warehouse names are organised prominently particularly in the **Product Info View** panel where you can see the alternative Stores quantity information. That panel then act as a quick lookout albeit virtually of all your POS stores. This means effective centralised managing of the entire POS network. The handling of this at the ImportOrder side was lacking but I managed to patch that up.

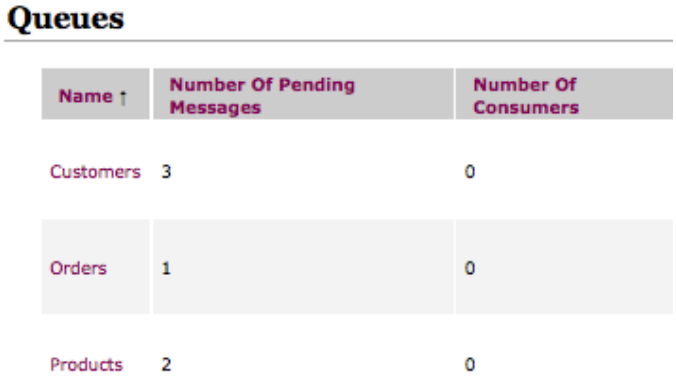
Note also that the Tax Model is lacking as it is also not implemented at the ImportOrder side. However I did not solve this but it can be similarly solved as the locator. More input advice from accounting minds are also needed to make a better patching of this.

A resulting sample XML string is as follows:

```
<?xml version="1.0" ?><entityDetail>
<type>openbravoPOS</type>
<detail><DocType>M_Product</DocType>
<POSLocatorName>HQ Warehouse</POSLocatorName>
<ProductName>Hoe</ProductName>
<QtyOnHand>20</QtyOnHand>
<M_Product_Category_ID>107</M_Product_Category_ID>
<CategoryName>Tools</CategoryName>
<M_Product_ID>138</M_Product_ID>
<C_TaxCategory_ID>107</C_TaxCategory_ID>
<TaxName>Standard</TaxName>
<UPC></UPC>
<PriceList>15</PriceList>
<PriceLimit>12</PriceLimit>
</detail>
```

Below we show the ActiveMQ admin page again for reference on how you can get access the above XML in the heat of action.

Click on the Queues link of your ActiveMQ admin page. Then click on the Products link within it. At the message page, click on the shown message and it will display the above XML at the bottom of the page.



Repeat reminder about the use of this XML within our integration. The *POS Locator Name* should correspond to the **pos** entry key in *openbravo.properties* which is *HQ Warehouse* in order for it to find its right POS station.:

<entry key="pos">HQ Warehouse </entry>

Then only can we do the **POS synchronization** accurately for all stores**.** Each POS station will pick up only its own locator inventory information so as to allow the central ERP host to organise the entire POS network inventory movements. This is one of the most important objectives of a replicated or synchronised POS integration.

Orders XML String

A sample of the exported Orders from the POS is as follows:

```
<?xml version="1.0" ?>
<?xml version="1.0" ?>
<entityDetail>
<type>I_Order</type>
<detail>
<DocTypeName>POS Order</DocTypeName>
<AD_Client_ID>11</AD_Client_ID>
<POSLocatorName>HQ Warehouse</POSLocatorName>
<DocumentNo>2</DocumentNo>
<DateOrdered>2011-09-16 12:35:07.599</DateOrdered>
<ProductValue>Seeder</ProductValue>
<QtyOrdered>2.0</QtyOrdered>
<PriceActual>30.0</PriceActual>
<TaxAmt>0.0</TaxAmt></detail>
</entityDetail>
```

As noted in last topic about some gap, the warehouse locator in ADempiere will not be able to synch which locator this is from as the corresponding I_Order data reference is M_Warehouse_ID instead of its name and ImportOrder code has to do a lookup first. This gap is now solved.

Multiple POS Locators Looping

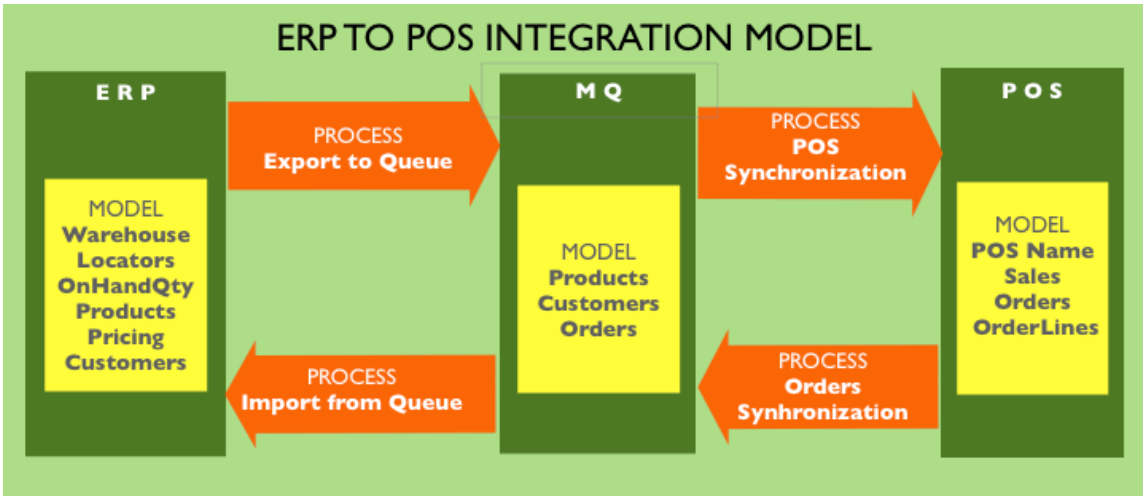
During ImportQueue2AD, the code only reads a single message queue, meaning it only will handle a single POS station's queue, and then ACK or erase it. If there are many POS stations sending back their orders of the day, there will be multiple queues according to each locator. For 33 stations, there will be 33 queues showing up on the MQ Admin page. To read all of them means you will have to run the Import from Queue process 33 times. Thus we have to tell the process that we have 33 queues to read and it has to read all of them rather than just once.

One idea to handle this is to have a batch counter at another queue-path, such as /queue/count to store a running counter. Each POS when depositing its message to the queue shall first read that counter, increment it, and then write back to it. In that way at the end of the day, the counter will give the total number of POS locators having done that.

The ERP server will thus, when picking up the Orders use the counter figure to do a looping sequence according to the number of times indicated in the counter. It can also compare with a control figure that checks how many POS locators are there in the network. This will be interesting for me to do with the *taleban* I mean students here at the college.

10,000 FEET VIEW

You can also see this helicopter view at the start of the development guide on page 33 which basically dupes the orange arrow boxes with exact java class names. Here the boxes are named according to what the end user sees when they open up each application stack. It is good to review and revise this model for the developer to get a grasp, within one page. Particular that the mind must have been bombarded by so much alien concepts the last few pages.



The processes carry data between the 3 remotely located stacks and they worked asynchronously.

The ERP on the left is ADempiere and the middle stack is the ActiveMQ and the right is the Openbravo POS application. The right can be many separate POS clients, but the left and middle are usually singular.

The **model** in each stack shows the relevant data values that are used during synchronisation. The **processes** are operated by only the both left and right stacks respectively. Export to Queue and Import from Queue belongs to the ERP, whereas POS Synchronization and Orders Synchronization belong to the POS client.

The middle ActiveMQ service do not do any of those processes shown. It is just a holder of the passed information within the queue paths' data model. MQ is accessed by URL links with controlled ports for added security from both ends. It has to be alive only when any end is talking to it. It can be brought down when not in use. It will not lose its data. Both ends need not be up at the same time. That is why it is called asynchronous.

Embedded Journalist in War Zone

Just kidding. It is an embedded online media actually. (I am just trying to fill up this really empty bunker looking spot on this page). I made a youtube movie that demonstrates the whole integration within a single song. It can also give you a helicopter view by showing in action the whole process from setup to direct hits on remote targets. I embed the link below. It was done while I was halfway through the project though. So you might not see some cooler stuff that I have finally done. I would appreciate it if you click on the 'like' button there to heat it up. It might help me launch more goodness to the world. At least you can make me look good. Then I make you look good too.

http://www.youtube.com/watch?v=TwXyK1KoO-M

ISSUES

Missing in Action

There are a few spots that I have discovered that do not implement some needed things inside the POS. One is explained earlier and here there is a more glaring gap. Even if you use v2.2 as it is from Openbravo, you may not get the Stock view. This raise my suspicion that perhaps there is some hidden code somewhere. Anyway, leave that to the *CIA*. We do not want clutter in here and lost the end which is now clearly within sight. The most important one is the stock qty view from the StockCurrent and StockDiary tables. As I shown in the setup section, such tables and its correct values exist as synchronised but the code to send them to the display is not there. The class concerned is under the src-pos /

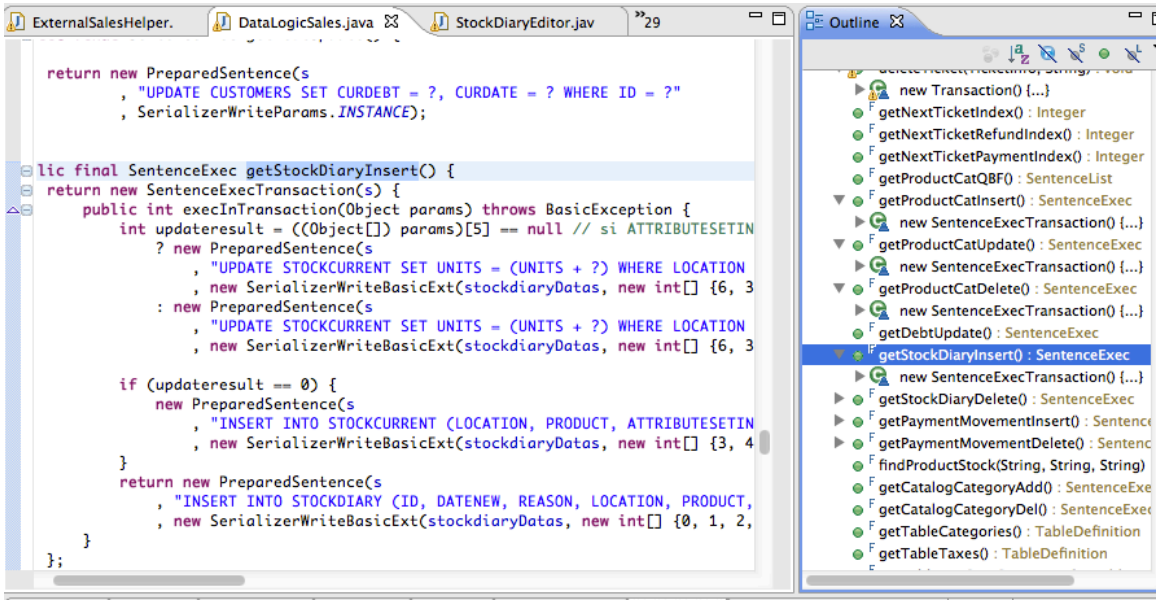
```
openbravo/pos/inventory/StockDiaryEditor.java
```

I committed to SVN some comments and below is the diff to indicate what I have investigated so far. If you want to access the actual commit you can refer to

```
Revision: 16169
http://adempiere.svn.sourceforge.net/adempiere/?rev=16169&view=rev
```

```
    public void refresh() {
@@ -324,16 +325,19 @@
                    attsetid = null;
                    attsetinstid = null;
                    attsetinstdesc = null;
+                   units = null;
                    jproduct.setText(null);
                    m_jcodebar.setText(null);
                    m_jreference.setText(null);
                    jattributes.setText(null);
+                   m_junits.setText(units);
                } else {
                    productid = prod.getID();
                    productref = prod.getReference();
                    productcode = prod.getCode();
                    productname = prod.toString();
                    attsetid = prod.getAttributeSetID();
+// units -- Need to get from StockDiary (is not implemented by open-
bravo POS in this version)
                    attsetinstid = null;
                    attsetinstdesc = null;
                    jproduct.setText(productname);
```

The above needs a helper method such as the one I snapped below from **DataLogicSales**. You can see in the *Outline* view on the right that the **getStockDiary** has handling methods but not for getting the StockDiaryView. This is the place to start if you wish to continue the mission.



The use of PreparedSentence and BaseSentence also indicated that this POS use its own pattern to handle its direct JDBC SQL calls. Not unlike Compiere's which we inherited. But it is a good pattern to learn and can add up to the learning curve in mastering this code.

No Tax

True to rugged territory, we do not implement tax during the synch of orders from the POS. The *TaxAmt* do come over but at the ImportOrder side we do nothing with it. This is because ADempiere's tax data model is strange. It does not set the tax percentage at the Customer or Product table but at the **Sales Order table** to lookup a **Tax table**. Thus, we need the accountant to tell us exactly what that means and how should we map the tax amount. What should the Tax ID setting be as at the OrderLine, some lookup from that determines the percentage set at the Tax table.

At the POS side the objects possible are

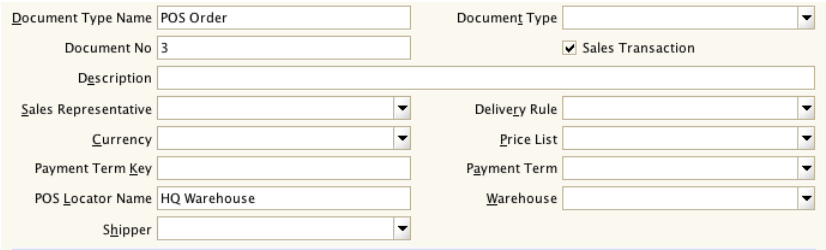
`getTaxLines(), getTaxLine(tax), getTaxes(), getTax()`

At the moment we are just passing the last object as a *TaxAmt*.

Import Order Warehouse ID

During the Import Order at ADempiere's side, there are two issues at least. I noticed that it does not accept *Warehouse Locator Value* but *M_Warehouse_ID*. This is non-applicable for imports to work as IDs are hard to synchronise. The locator name should be a unique text name that users can easily identify for example the actual location name of the Store. Passing IDs does not work and the importer has to manually set the Warehouse name for each POS station. This gap is now resolved within the FitnesseOBPOS package and committed and working well in my runs.

I made AD changes so that the ImportOrder tab has a new field called **POS Locator Name**:



The code snip I introduced in **ImportOrder.java** to process the new field is as follows:

```
//Get Warehouse ID from WarehouseName matching POS Locator Name
sql = new StringBuffer ("UPDATE I_Order o "       //
          + "SET M_Warehouse_ID=(SELECT M_Warehouse_ID FROM M_Warehouse w WHERE w.Name=o.PosLocatorName"
          + " AND o.AD_Client_ID=w.AD_Client_ID) "
          + "WHERE M_Warehouse_ID IS NULL AND I_IsImported<>'Y'").append (clientCheck);
          no = DB.executeUpdate(sql.toString(), get_TrxName());       //        Warehouse by POS Name
          if (no != 0)
                    log.fine("Got POS to set Warehouse=" + no);
```

Secondly is during synching the POS stations has their own running sales document no., which ADempiere during import will stop at as there is a constraint against repeat DocumentNo. This can be solved by configuring that field in Sales Order to be comprised of the POS sales no. plus the POS station ID, plus the date. This gap will be resolved in later commits.

POST MORTEM

Well, hopefully no one got killed in this mission as no one likes to clean up their body parts. I did not get to go more than what I been documenting here and online at my forum. There is more to the POS application than is possible within this short scope of proving a good enough integration for others to follow and most important make productive use of the POS and ERP right away. I saw the body I mean code part that can send the product image together during synching. But that will need some investigation whether you can put the images into the luggage I mean message queue. There can be workaround if luggage is not possible such as by sending a cargo or FTP in from the client. The XML then just pass the filename of the image to attach at the POS.

Steven Sackett of Adaxa, Melbourne gave me two calls over something else. One call he made before getting on the plane to Berlin for the ADempiere conference, and the second call was the day he came out from a small surgery on his return from Berlin. He told me about their own experience making integration to the TinaPOS (early days of this POS) and that he may have come across the issue with *returned goods*.

There are also questions from few other parties in Europe on which POS to choose between the JavaPOS and this standalone POS. The advantage with this POS seems very clear if you want to have remote POS stations and prefer asynchronous messaging connectivity. But it still has much room to explore and improve upon, the whole reason why we fly Open Source.

My constant advice is to always collaborate and share the lessons, improvements and save everyone unnecessary costs of maintenance and focus on our core business, i.e. increasing a user pool, better, quicker branding, writing more higher quality code with great documentation and higher worth consulting.

Thank you for flying with Open Source. Statistics have shown it is more safer than all other modes of software distribution put together.

red1
1316 hrs, 16.09.11
14°34′00″N and 121°01′58″E

ALBUM

Somehow someone out in cyberspace miss my album photos (Peter M Austin that is), so ok, I do have some in my mobile phone from last Saturday (after finishing the most of this POS guide) where Karma rewarded me with a great day out with Joe Gene and his lovely wife, Jing.

We had lunch at a place called Sonya's Garden. With natural surroundings where the garden provided all the food on the table.

Below, I couldn't find this mysterious Sonya at her all-natural garden bed, so I waited for any fair lady that might chance along.

Sonya's place is at a lovely highlands location called Tagaytay Taal that has a volcano dormant for 300 years.

We stopped by a roadside traditional cafe (with coconut atap tops) to get a good sight of the volcano. There it is, that small hilltop popping out in the middle of the huge crater lake, sprawled far below us. The cat perched on the balcony doesn't seem to care of the sheer drop to the plateau surrounding it.

And there is more.

Here is from my earlier visit to the college pulling it off on a big screen in a big hall with big audio link from my Macbook.

Great audience. In a big hall.

Great food. Look at that crab trying to get to my chillis.

Loud music.

Hm, actually, I could hear them from my dorm, so I went to have a word with them. And i asked them if they can play my song that I wrote instead.

EPILOGUE - UPGRADED TO OSGI PLUGIN

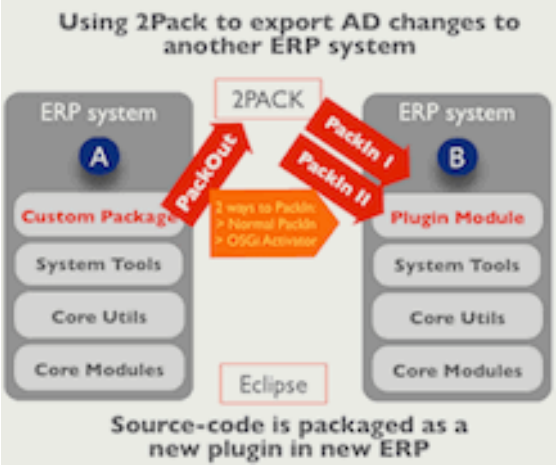
This recent year has been remarkable, as iDempiere v1.0.a just got released with a boom I mean bang. See http://red1.org/adempiere/viewtopic.php?f=29&t=1712.

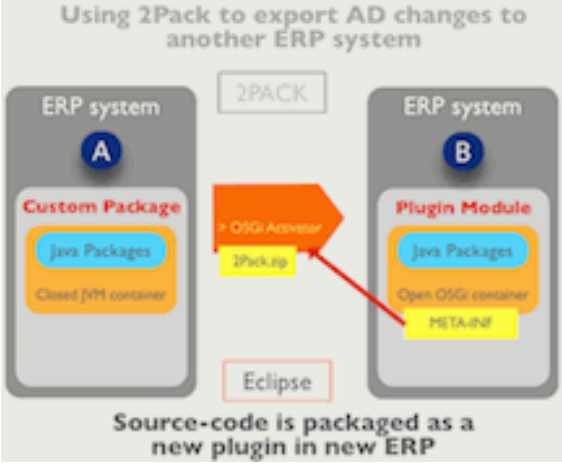
For that occasion I have successfully migrated this Openbravo POS integration's ERP handling side into an OSGi plugin. See http://red1.org/adempiere/viewtopic.php?f=29&t=1713.

This has some reason to incite excitement:

1. You no longer need to use customization.jar approach to bring in code changes particularly those that bear the same namespace. Just deploy the new plugin containing the same code into the OSGi stack. Its declared *Extension*s will exposed the named processes to the ERP system. Likewise to disengage you merely stop that plugin and it will fall back to its old similarly named processes if any. And in OSGi you can stop the plugin without landing the plane!

2. DB metadata changes are now done via 2Pack and can be deployed when that plugin first runs!





Enjoy your new cockpit

- red1 , November 14th, 2012. Kuang, Selangor, Malaysia.

Errata - I also found out from further investigation that the Stock Diary really works. It is just a matter of clicking on the right reports on the OpenbraboPOS dashboard. Well, see what happened when you do not have sufficient flying experience?