Đại Học Quốc Gia Thành Phố Hồ Chí Minh

Đại Học Khoa Học Tự Nhiên Thành Phố Hồ Chí Minh

Khoa Công Nghệ Thông Tin

Bộ Môn Công Nghệ Phần Mềm

Phạm Thế Hùng – Nguyễn Khuyến

**Phần Mềm Quản Lý Khách Hàng Thân Thiết**

Khóa Luận Cử Nhân Tin Học

Thành Phố Hồ Chí Minh năm 2010

Đại Học Quốc Gia Thành Phố Hồ Chí Minh

Đại Học Khoa Học Tự Nhiên Thành Phố Hồ Chí Minh

Khoa Công Nghệ Thông Tin

Bộ Môn Công Nghệ Phần Mềm

Phạm Thế Hùng - 0612177

Nguyễn Khuyến - 0612193

**Phần Mềm Quản Lý Khách Hàng Thân Thiết**

Khóa Luận Cử Nhân Tin Học

Giảng Viên Hướng Dẫn :

Thầy Trần Hiển Đạt

Niên Khóa 2006 – 2010

**Nhận xét của giảng viên hướng dẫn**

**Nhận xét của giảng viên phản biện**

**Lời Cảm Ơn**

✍🕮🖎

Lời đầu tiên chúng em xin cảm ơn Khoa Công Nghệ Thông Tin Trường Đại Học Khoa Học Tự Nhiên đã tạo cơ hội cho chúng em thực hiện khóa luận.

Chúng em xin gởi lời cảm ơn chân thành nhất tới Thầy Trần Hiển Đạt, Thầy đã hướng dẫn rất tận tình cho chúng em trong suốt khóa luận vừa qua.

Bên cạnh đó chúng em xin gởi lời cảm ơn đến tất cả quý Thầy Cô của Khoa đã tận tình giảng dạy và trang bị cho chúng em kiến thức vững chắc góp phần rất lớn vào việc nghiên cứu thành công khóa luận này.

Cuối cùng,chúng em gởi lời cảm ơn sâu sắc đến cha mẹ và gia đình đã nuôi nấng dạy dỗ chúng em đến ngày hôm nay cũng như động viên giúp đỡ chúng em trong suốt quá trình thực hiện khóa luận.

Dù đã cố gắng hết sức để thực hiện khóa luận tuy nhiên những thiếu sót là không thể tránh khỏi vì vậy mong nhận được những lời chỉ bảo và góp ý của Thầy Cô cũng như những chia sẽ của tất cả các bạn.

Thành Phố Hồ Chí Minh, tháng 7 năm 2010

Nhóm sinh viên thực hiện

Phạm Thế Hùng – Nguyễn Khuyến

Mục Lục

[Chương 1 : Tổng Quan 15](#_Toc261098761)

[1.1 Yêu cầu thực tế và lý do thực hiện đề tài : 15](#_Toc261098762)

[1.2 Hiện trạng của đơn vị xây dựng phần mềm : 15](#_Toc261098763)

[1.3 Mục tiêu khóa luận 16](#_Toc261098764)

[1.4 Hướng tiếp cận của khóa luận : 16](#_Toc261098765)

[1.5 Các ưu điểm và khuyết điểm của hệ thống : 16](#_Toc261098766)

[1.6 Nội dung thực hiện của đề tài : 17](#_Toc261098767)

[Chương 2 : Cơ Sở Lý Thuyết 18](#_Toc261098768)

[2.1 ISO 8583 là gì? 18](#_Toc261098769)

[2.2 Cấu trúc thông điệp 18](#_Toc261098770)

[2.2.1 Chỉ thị thông điệp (Message Type Indicator – MTI) 18](#_Toc261098771)

[2.2.1.1 Phiên bản ISO8583 18](#_Toc261098772)

[2.2.1.2 Lớp thông điệp : 19](#_Toc261098773)

[2.2.1.3 Chức năng của thông điệp 19](#_Toc261098774)

[2.2.1.4 Gốc thông điệp (Message Origin) 20](#_Toc261098775)

[2.2.2 Bitmap 21](#_Toc261098776)

[2.2.3 Thành phần dữ liệu (Data Element) 24](#_Toc261098777)

[2.3 Wire Protocol 25](#_Toc261098778)

[2.4 Thiết kế thông điệp cho giao dịch cộng điểm 27](#_Toc261098779)

[2.4.1 MTI (Message Type Indicator) : 27](#_Toc261098780)

[2.4.2 Field No : 3 Processing code 28](#_Toc261098781)

[2.4.2.1 The transaction Type Code(TTC): 28](#_Toc261098782)

[2.4.2.2 Account Type Code (ATC) chỉ ra loại của tài khoản. 28](#_Toc261098783)

[2.4.3 Field 48 28](#_Toc261098784)

[2.4.4 Field 61 29](#_Toc261098785)

[2.4.5 Message sample : 29](#_Toc261098786)

[2.4.5.1 Message request: 29](#_Toc261098787)

[2.4.5.2 Message Response : 29](#_Toc261098788)

[2.5 JPOS cách thức xử lý ISO8583 31](#_Toc261098789)

[2.5.1 ISOMsg & Co 31](#_Toc261098790)

[2.5.2 Packing và Unpacking 34](#_Toc261098791)

[2.6 Transaction là gì? 38](#_Toc261098792)

[2.6.1 Các tính chất của Transaction 39](#_Toc261098793)

[2.6.1.1 Atomic 39](#_Toc261098794)

[2.6.1.2 Consistency 39](#_Toc261098795)

[2.6.1.3 Isolation 39](#_Toc261098796)

[2.6.1.4 Durable 39](#_Toc261098797)

[2.6.2 Transaction trong JPos 1.6.4 39](#_Toc261098798)

[2.6.2.1 Participant 39](#_Toc261098799)

[2.6.2.2 Pha 41](#_Toc261098800)

[2.6.2.3 Mở rộng vấn đề : 43](#_Toc261098801)

[2.6.2.4 Transaction Context: 43](#_Toc261098802)

[Chương 3 : Thiết Kế Chương Trình 44](#_Toc261098803)

[3.1 Thiết kế kiến trúc của chương trình : 44](#_Toc261098804)

[3.1.1 Thiết kế tổng quát 44](#_Toc261098805)

[3.1.2 Chi tiết của kiến trúc 45](#_Toc261098806)

[3.1.2.1 Database module: 45](#_Toc261098807)

[ Org.hcmus.dao.package: 46](#_Toc261098808)

[ Org.hcmus.dao.idao 47](#_Toc261098809)

[ Org.hcmus.dao.lms 47](#_Toc261098810)

[ Org.hcmus.bus 47](#_Toc261098811)

[3.1.2.2 Participant Module: org.hcmus.participant 48](#_Toc261098812)

[3.1.2.3 Utilily Module: org.hcmus.util 48](#_Toc261098813)

[3.1.2.4 Listener Module : org.hcmus.lms 48](#_Toc261098814)

[3.2 Mô hình hóa nghiệp vụ 48](#_Toc261098815)

[3.2.1 Actors 49](#_Toc261098816)

[3.2.2 User Cases 49](#_Toc261098817)

[3.2.3 Đặc tả Usercase : 50](#_Toc261098818)

[3.2.3.1 Đặc tả usercase Add Point 50](#_Toc261098819)

[3.2.3.2 Thực thi chương trình 51](#_Toc261098820)

[3.2.3.3 GateWay (Multi Channel) 54](#_Toc261098821)

[Chương 4 Chương 4 : 58](#_Toc261098822)

[Chương 5 Phụ Lục : 59](#_Toc261098823)

[5.1 Cách Thức Tổ Chức Biên Dịch Trong JPos 59](#_Toc261098824)

[5.1.1 Tổ chức thư mục của JPos Project 59](#_Toc261098825)

[5.1.2 Cơ chế biên dịch của JPos 60](#_Toc261098826)

[5.1.3 Thêm vào hoặc loại bỏ module. 62](#_Toc261098827)

[5.1.4 Manifest File 63](#_Toc261098828)

[5.1.5 Build.xml và build.properties 63](#_Toc261098829)

[5.2 Cài đặt JPos trên window 64](#_Toc261098830)

[5.2.1 Yêu cầu : 64](#_Toc261098831)

[5.2.2 Thiết lập các biến mội trường 64](#_Toc261098832)

[5.2.3 Test các gói cài đặt 66](#_Toc261098833)

[5.3 Sử dụng JPOS trên window. 69](#_Toc261098834)

[5.4 Test Q2 73](#_Toc261098835)

[5.5 Cài đặt JPOS trên LINUX 79](#_Toc261098836)

[5.5.1 Yêu cầu 79](#_Toc261098837)

[5.5.2 Cài đặt các thành phần 79](#_Toc261098838)

[5.6 Sử dụng JPOS trên LINUX 92](#_Toc261098839)

[5.7 QSP & Q2 94](#_Toc261098840)

[5.7.1 QSP 95](#_Toc261098841)

[5.7.2 Cấu hình. 97](#_Toc261098842)

[5.7.2.1 Cấu hình QSP 97](#_Toc261098843)

[5.7.2.2 Object. 98](#_Toc261098844)

[5.7.2.3 Configuration Object 99](#_Toc261098845)

[5.7.2.4 Reconfiuration Object 100](#_Toc261098846)

[5.7.2.5 Truy xuất đối tượng 102](#_Toc261098847)

[5.7.2.6 Runable Object 102](#_Toc261098848)

[5.7.2.7 JMX support 103](#_Toc261098849)

[5.7.2.8 <Task> 104](#_Toc261098850)

[5.7.2.9 Channel 105](#_Toc261098851)

[5.7.2.10 <filter> 107](#_Toc261098852)

[5.7.2.11 <mux> 107](#_Toc261098853)

[5.7.2.12 <server> 108](#_Toc261098854)

[5.7.2.13 <sequencer> 109](#_Toc261098855)

[5.7.3 Q2 110](#_Toc261098856)

[5.7.3.1 QBean description 112](#_Toc261098857)

[5.7.3.2 Q2 Dynamic class loading 113](#_Toc261098858)

[5.7.3.3 Q2 Script 114](#_Toc261098859)

[5.8 Space 115](#_Toc261098860)

[5.9 Đặc tả Usercase 116](#_Toc261098861)

[5.9.1 Đặc tả usercase Subtract Point 116](#_Toc261098862)

[5.9.1.1 Tóm tắt: 116](#_Toc261098863)

[5.9.1.2 Dòng dự kiện 116](#_Toc261098864)

[5.9.1.3 Các yếu cầu đặc biệt 117](#_Toc261098865)

[5.9.1.4 Trạng thái hệ thống khi bắt đầu thực hiện usercase. 117](#_Toc261098866)

[5.9.1.5 Trạng thái của hệ thống sau khi thực hiện usercase 117](#_Toc261098867)

[5.9.1.6 Điểm mở rộng : 117](#_Toc261098868)

[5.9.2 Luồng Sự Kiện Cho usercase Subtract Point 117](#_Toc261098869)

[5.9.3 Đặc tả usercase Redemption Point 118](#_Toc261098870)

[5.9.3.1 Tóm tắt: 118](#_Toc261098871)

[5.9.3.2 Dòng dự kiện 118](#_Toc261098872)

[5.9.3.3 Các yếu cầu đặc biệt 119](#_Toc261098873)

[5.9.3.4 Trạng thái hệ thống khi bắt đầu thực hiện usercase. 119](#_Toc261098874)

[5.9.3.5 Trạng thái của hệ thống sau khi thực hiện usercase 119](#_Toc261098875)

[5.9.3.6 Điểm mở rộng : 119](#_Toc261098876)

[5.9.4 Luồng Sự Kiện Cho Usercase Redemption Point 120](#_Toc261098877)

[5.9.5 Đặc tả usercase Balance Inquiry 120](#_Toc261098878)

[5.9.5.1 Tóm tắt: 120](#_Toc261098879)

[5.9.5.2 Dòng dự kiện 120](#_Toc261098880)

[5.9.5.3 Các yếu cầu đặc biệt 121](#_Toc261098881)

[5.9.5.4 Trạng thái hệ thống khi bắt đầu thực hiện usercase. 121](#_Toc261098882)

[5.9.5.5 Trạng thái của hệ thống sau khi thực hiện usercase 121](#_Toc261098883)

[5.9.5.6 Điểm mở rộng : 121](#_Toc261098884)

[5.9.6 Luồng sự kiện Balance Inquiry 122](#_Toc261098885)

[5.10 Bảng đặc tả cho các thông điệp được thiết kế 122](#_Toc261098886)

[5.10.1 Subtract Point Transaction 122](#_Toc261098887)

[5.10.1.1 Tổng Quát 122](#_Toc261098888)

[5.10.1.2 Đặc tả chi tiết 123](#_Toc261098889)

[5.10.2 Balance Inquiry Transaction 125](#_Toc261098890)

[5.10.2.1 Tổng quát 125](#_Toc261098891)

[5.10.2.2 Đặc tả chi tiết 126](#_Toc261098892)

[5.10.3 Redemption Transaction 128](#_Toc261098893)

[5.10.3.1 Tổng quát 128](#_Toc261098894)

[5.10.3.2 Đặc tả chi tiết 129](#_Toc261098895)

[5.10.4 Đặc tả cho các trường thông điệp chung : 131](#_Toc261098896)

[5.10.4.1 Field No : 35 Track 2 Data 131](#_Toc261098897)

[5.10.4.2 Field No : 39 Response code 132](#_Toc261098898)

[5.10.4.3 Filed No : 41 Card Acceptor Terminal ID “TID” 132](#_Toc261098899)

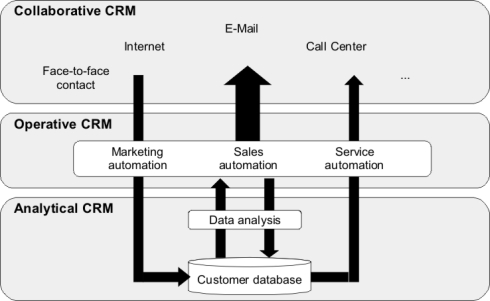
[5.10.4.4 Field No 42 Card Acceptor Identify Code “MID” 132](#_Toc261098900)

[5.10.4.5 Field 61 132](#_Toc261098901)

[5.10.4.6 Conditional Code 132](#_Toc261098902)

Lời Mở Đầu

Ý thức được tầm quan trọng của các hoạt động hướng khách hàng (KH), ngày càng nhiều doanh nghiệp quan tâm hơn đến các ứng dụng quản trị quan hệ khách hàng (CRM). Để đạt được giá trị lâu dài của giải pháp CRM, yêu cầu kiến trúc chiến lược phải gồm toàn bộ các hoạt động kinh doanh, dịch vụ liên quan đến khách hàng của doanh nghiệp và trên mức độ toàn doanh nghiệp.



Hình 1‑1

Hình trên biểu diễn một kiến trúc CRM được cho là thành công hiện nay.

Tại các doanh nghiệp Việt Nam, chiến lược CRM đầy đủ gặp rất nhiều khó khăn. Việc đánh giá vị trí hiện tại của doanh nghiệp liên quan đến giá trị, lòng trung thành và độ hài lòng của khách chưa đầy đủ vì thiếu thông tin khách hàng. Giá trị khách hàng phần lớn được đánh giá phiến diện theo doanh số trong khi lòng trung thành và độ hài lòng của khách hàng thì gần như không đánh giá được. Việc thiết lập các mục tiêu về khách hàng phần lớn mang tính mơ hồ, không rõ ràng. Đa phần doanh nghiệp đều không quan tâm đến việc chỉ ra các yêu cầu về con người: kỹ năng, văn hóa, tổ chức, trách nhiệm, quyền hạn…Các yêu cầu về dữ liệu khách hàng thì hoàn toàn thiếu và không thống nhất, dựa trên dữ liệu rời rạc hiện có của các phòng ban riêng rẽ. Ví dụ, một doanh nghiệp phân phối hàng tiêu dùng thì dữ liệu khách hàng đại lý gần như chỉ có dữ liệu cơ bản (tên khách hàng, địa chỉ, doanh số, dữ liệu kế toán). Trong khi CRM cần nhiều hơn rất nhiều các thông tin này để xây dựng lại mối quan hệ kinh doanh với các khách hàng đã mất hoặc xây dựng các kế hoạch marketing hay chính sách bán hàng, dịch vụ...

Vì những nhược điểm được chỉ ra ở trên vấn đề xây dựng một kiến trúc CRM là thiết thực và lâu dài cho doanh nghiệp. Trong khuôn khổ của khóa luận chúng ta không quan tâm tất cả các giải pháp xây dựng kiến trúc CRM mà thay vào đó chúng ta quan tâm đến một nhánh nhỏ trong CRM là khách hàng thân thiết. Và việc xây dựng hệ thống quản lý khách hàng thân thiết(LMS) trở thành vấn đề đang được doanh nghiệp quan tâm hiện nay.

Sau cái nhìn tổng quát về CRM và thấy được vị trí của LMS trong mô hình CRM. Chúng ta đề ra một số giải pháp cho hệ thống LMS. Các giải pháp của LMS cũng không lạc khỏi mục tiêu đã đề cập ở trên là làm cách nào để giữ các khách hàng tốt nhất của bạn.Một số câu hỏi được đặt ra và bằng việc trả lời các câu hỏi này chúng ta có cái nhìn sâu hơn về LMS.

* Tại sao khách hàng rời bỏ chúng ta:

-Có phải do chất lượng sản phẩm của bạn chưa tốt?

-Thông tin truyền thông phát triển khiến cho khách hàng có nhiều thông tin hơn về sản phẩm và sự cạnh tranh của thị trường ngày càng khóc liệt?

-Chế độ quan tâm của bạn đối với khách hàng thân thiết chưa tốt?

* Đánh giá lợi ích khi giữ được khách hàng thân thiết :

-Khách hàng thân thiết đồng nghĩa với cơ hội bán hàng.

-Lợi ích được thấy rõ dựa trên số sản phẩm khách hàng thân thiết tiêu thụ.

-Lợi nhuận cho mỗi lần mua bán đối với một khách hàng thân thiết.

* Khi bạn xây dựng một hệ thống khách hàng thân thiết bạn sẽ được những lợi ích gì ?

-Làm cho khách hàng ngày càng thân thiết hơn.

-Nắm được giá trị của của mỗi khách hàng thân thiết.

-Sự giới thiệu của khách hàng thân thiết cho các khách hàng mới.

-Làm cho họ quay trở lại với sự tiêu tụ cao hơn lần trước.

-Hiểu rõ khách hàng hơn nữa và kích thích nhu cầu của họ.

-Kich thích khách hàng mà không cần giảm giá.

-Theo dõi và đo lường việc mua sắm của khách hàng.

-Tăng khả năng chiến đấu với các sản phẩm khác.

Từ những điều đã được đề cập phía trên khóa luận tập trung vào xây dựng một hệ thống quản lý khách hàng thân thiết thông qua thẻ khách hàng thân thiết do doanh nghiệp phát hành.

Khóa luận này gồm 4 chương có nội dung như sau :

**Chương 1** : Giới thiệu bối cảnh, lý do thực hiện đề tài cũng như các giải pháp hiện thời có liên quan từ đó rút ra hướng tiếp cận của khóa luận để thực hiện đề tài.

**Chương 2** : Trình bày sơ lược các lý thuyết về chuẩn ISO8583, lý thuyết về JPOS Framework các chi tiết kỹ thuật liên quan đến đề án.

**Chương 3** : Phân tích và thực hiện hệ thống quản lý khách hàng thân thiết dựa trên lý thuyết đã đề ra ở trên.

**Chương 4** : Thực nghiệm,đánh giá và tổng kết : nêu đánh giá về toàn bộ đề tài , trình bày các kết quả đã đạt được cũng như hạn chế của đề tài từ đó đề xuất ra những hướng nghiên cứu trong tương lai.

# : Tổng Quan

## Yêu cầu thực tế và lý do thực hiện đề tài :

Trong vài năm tới Việt Nam của chúng ta sẽ có chính phủ điện tử, chính phủ điện tử ra đời sẽ giúp hoàn chỉnh hệ thống thương mại điện tử của Việt Nam vì vậy thương mại điện tử trở thành mảnh đất màu mở và giàu tiềm năng hiện nay. Cùng với xu hướng đó quản lý mối quan hệ khách hàng cụ thể là quản lý khách hàng thân thiết được các doanh nghiệp đặc biệt quan tâm . Hệ thống quản lý này có thể áp dụng từ những nhà sách nhỏ muốn giữ chân bạn đọc của mình cho đến các hệ thống siêu thị lớn kích thích khả năng mua sắm của người tiêu dủng.

Các phần mềm quản lý khách hàng thân thiết trên thị trường bắt đầu đã xuất hiện trong vài năm trở lại đây điển hình là hệ thống quản lý khách hàng thông qua thẻ khách hàng thân thiết ở các siêu thị như BigC và Nguyễn Kim tuy nhiên các phần mềm này phần lớn là mã nguồn đóng và được thực hiện cụ thể nhầm vào một đối tượng doanh nghiệp nhất định vì thế chi phí thực hiện là rất lớn và khả năng tái sử dụng thấp.Vì thế việc xây dựng một hệ thống quản lý khách hàng thân thiết mã nguồn mở trở thành nhu cầu thực tế được quan tâm và khả năng phát triển dựa trên sự đóng góp của cộng đồng cho các phần mềm mã nguồn mở rất hứa hẹn.

## Hiện trạng của đơn vị xây dựng phần mềm :

Phần mềm được xây dựng trong phạm vi của một khóa luận ở trường đại học.Các gói chương trình được nhóm sinh viên thực hiện dưới sự hướng dẫn của giảng viên.Các trang thiết bị dùng để thực hiện và kiểm thử dưới sự giúp đỡ của một số công ty chuyên về thương mại điện tử ,nhà trường tạo mọi điều kiện tốt nhất cho sinh viên khảo sát thực tế cũng như phát triển dự án.

## Mục tiêu khóa luận

Khóa luận gồm 3 mục tiêu chính :

* Thứ nhất tìm hiểu về chuẩn ISO8583 cách thức gởi và nhận thông điệp ISO8583.
* Thứ hai : tìm hiểu về JPOS framwork.
* Thứ ba : Sử dụng JPOS framework gởi và nhận các gói tin ISO8583 từ đó ứng dụng xây dựng nên hệ thống quản lý khách hàng thân thiết dựa vào thẻ khách hàng thân thiết do doanh nghiệp phát hành.

## Hướng tiếp cận của khóa luận :

Khóa luận tập trung vào xây dựng giải pháp server cho hệ thống LMS. Hệ thống server sẽ nhận các gói dữ liệu gởi theo chuẩn ISO8583 từ các thiết bị đọc mã vạch hoặc các máy POS terminal hay thậm chí là từ thiết bị di động có hỗ trợ phần mềm đọc mã vạch. Sau đó hệ thống sẽ tiến hành thực hiện các tác vụ có liên quan như cộng điểm khi mua hàng trừ điểm khi trả hàng hoặc các tác vụ khác như kích hoạt thẻ sau đó lưu trữ vào database và trả kết quả về cho client.

Hệ thống hỗ trợ khách hàng quản lý truy vấn thông tin tài khoản của mình thông qua giao diện web. Quản lý thống kê các tài khoản khách hàng, thống kê việc mua bán của khách hàng nhầm giúp các doanh nghiệp đưa ra chuyến lược kinh doanh hợp lý.

## Các ưu điểm và khuyết điểm của hệ thống :

**Ưu điểm** : Hỗ trợ các thiết bị có sẵn như điện thoại di động có khả năng đọc barcode.Các kênh truyền được thiết kế độc lập nhờ một module trung gian giúp cho việc khả chuyển các kênh truyền phục vụ cho việc mở rộng bài toàn sau này.

**Khuyết điểm** : Khả năng bảo mật thấp do mã nguồn mở.Việc ứng dụng vào thực tế cần có sự đầu tư của các doanh nghiệp.

## Nội dung thực hiện của đề tài :

Đề tài được thực hiện theo từng bước sau :

* Tìm hiểu về hệ thống và thực trạng của các giải pháp quản lý khách hàng thân thiết hiện nay.
* Phân tích và định hình bài toán quản lý khách hàng thân thiết đề ra mục tiêu và nhu cầu của người dùng cũng như các chức năng mà hệ thống hỗ trợ.
* Tìm hiểu về chuẩn ISO8583 và các phần mềm hỗ trợ gởi nhận thông điệp ISO8583 trên nền tảng cả Java và .Net.
* Tìm hiểu về JPOS framework và cách sử dụng JPOS framework để gởi nhận thông điệp theo chuẩn ISO8583
* Tìm hiểu về việc quản lý transaction sử dụng JPOS framework.
* Tổ chức xây dựng cơ sở dữ liệu cho bài toán.
* Phân tích và thiết kế tài liệu cũng như mô hình cho bài toán thông qua ngôn ngữ UML cũng như đề ra kiến trúc cho hệ thống và quy trình phát triển hệ thống.
* Tiến hành thực thi dự án.
* Kiểm thử và sửa lỗi
* Đóng gói
* Viết tài liệu về dự án.

Vừa rồi chúng ta vừa lướt qua tổng quan của dự án. Chương tiếp theo chúng ta sẽ bàn sâu hơn về các chi tiết kỹ thuật cũng như cơ sở lý thuyết của dự án.

# : Cơ Sở Lý Thuyết

## ISO 8583 là gì?

ISO8583 là một tiêu chuẩn cho các thẻ giao dịch tài chính bằng cách trao đồi các thông điệp được chỉ định.

## Cấu trúc thông điệp

Một thông điệp ISO8583 được chia làm các phần sau:

-Chỉ thị thông điệp (Message Type Indicator)

-Một hay nhiều bitmap chỉ tới các thành phần dữ liệu trong thông điệp.

-Thành phần dữ liệu hay trường các dữ liệu.

### Chỉ thị thông điệp (Message Type Indicator – MTI)

Một MTI chứa các thành phần sau :

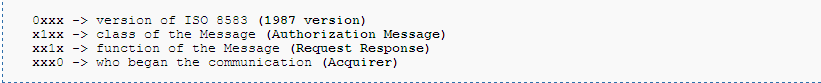
-Phiên bản ISO8583

-Lớp thông điệp

-Chức năng thông điệp

-Gốc của thông điệp

Những thành phần này được đại diện bằng một trường gồm 4 chữ số.

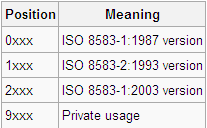


Hình ‑

#### Phiên bản ISO8583

Vị trí đầu tiên của MTI chỉ định phiên bản cho thông điệp của chuẩn ISO8583.

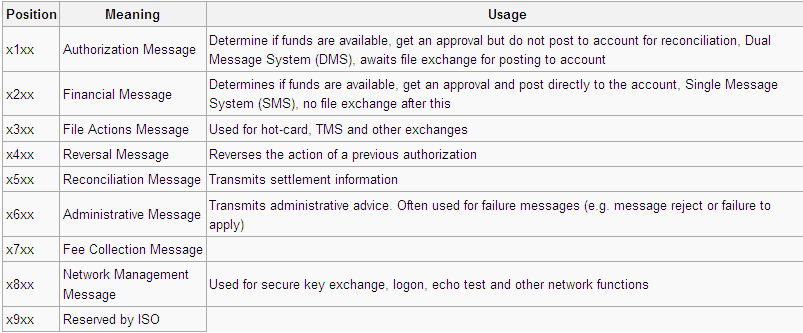
Cụ thề sau:



Hình ‑

#### Lớp thông điệp :

Vị trí thứ 2 của MTI chỉ ra mục đích của của thông điệp.



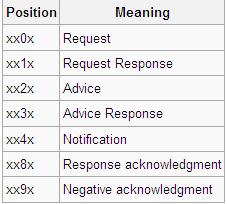
Hình 2‑3

Chúng ta xem xét ví dụ sau :

x**4**xx chỉ ra mục đích sử dụng cho thông điệp này.Tình huống khi một thông điệp được gởi đi xác nhận mã thẻ nhưng sau một khoãng thời gian nào đó cho trước (Timeout) mà vẫn chưa được trả lời xác nhận. Thì vị trí thứ 2 có giá trị 4 trong x**4**xx chỉ ra rằng đây là một thông điệp gởi trả của việc xác nhận trước đó.

#### Chức năng của thông điệp

Vị trí thứ 3 của MTI chỉ ra chức năng của thông điệp định nghĩa cách mà thông điệp sẽ chạy trong hệ thống.



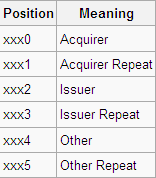
Hình 2‑4

Xem xét xx**0**x . Request là một chức năng từ cuối tới cuối (end to end) .Ví dụ từ nhà chấp nhận thẻ(các ngân hàng) cho tới các nhà phát hành thẻ(các nhà phát hành thẻ Vista,MasterCard).

Trong khi đó xx2x Advice là một chức năng điểm tới điểm(point to point). Ví dụ từ terminal(POS devide hay máy ATM chẳng hạn) tới nhà chấp nhận thẻ, từ nhà chấp nhận thẻ ra đường mạng từ đường mạng tới nhà phát hành thẻ việc chuyển đổi được bảo đảm trên mỗi liên kết.

#### Gốc thông điệp (Message Origin)

Vị trí thứ tư trong MTI chỉ ra nguồn của thông điệp trong chuỗi chi trả.

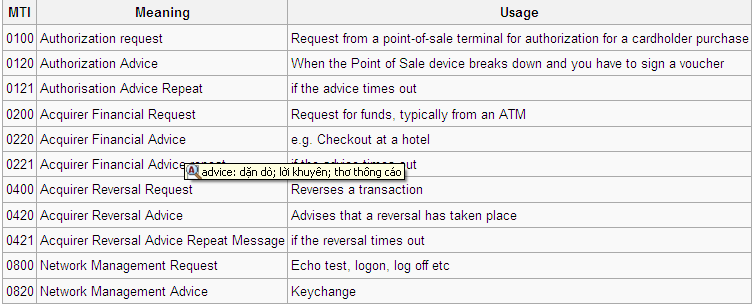


Hình ‑

* Ví dụ về MTI

Sự kết hợp các vị trí của một MTI chỉ ra ý nghĩa đầy đủ của MTI

Hình dưới chỉ ra một số ví dụ để xem xét.



### Bitmap

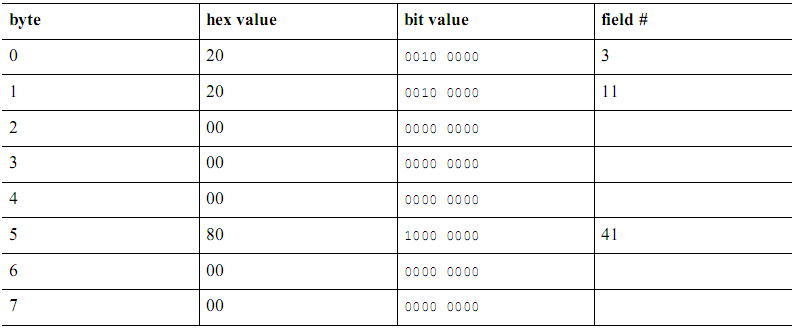
Bitmap là một trường hay là một trường con của thông điệp chỉ ra các thành phần dữ liệu được trình bày ở những nơi khác trong thông điệp.

Một thông điệp ít nhất có một bitmap được gọi là Primary Bitmap (Bitmap chính) chỉ định các thành phần dữ liệu từ số 1 tới số 64. Một seconary bitmap(Bitmap thứ hai) chỉ định các trường dữ liệu từ 65 cho đến 128 và cũng có thể có bitmap thứ 3 chỉ định các trường dữ liệu từ 129 cho tới 192(Bitmap thứ 3 này rất ít sử dụng).

Bitmap có thể được truyền bằng 8 byte nhị phân hoặc 16 kí tự thập lục phân trong bảng mã ASCII hoặc tập kí tự [[1]](#footnote-2)EBCDIC.

Một trường được chỉ định khi mà bit đại diện cho trường đó bật là true trong bitmap.Ví dụ byte x82 chuyển qua giá trị nhị phân “1000 0010” chỉ ra rằng thông điệp chứa trường số 1 và số 7.

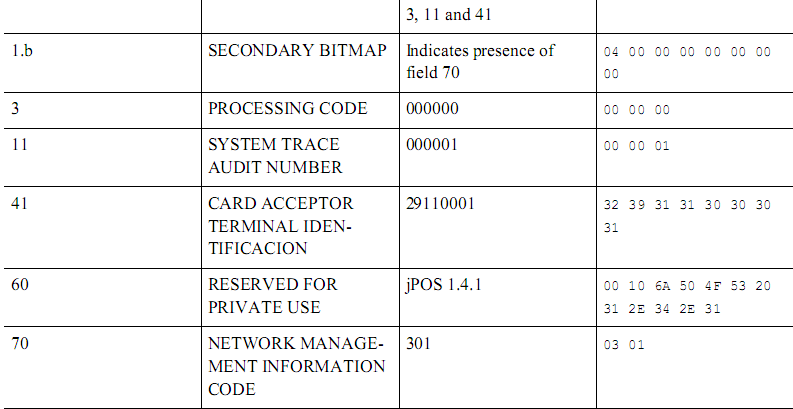
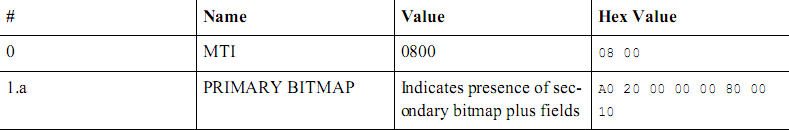
Xem xét Primary Bitmap sau:



Hình 2‑6

Primary bitmap (2020000000800000) chứa các bit 1 bật ở các vị trí 3,11,41(xem bit value hình trên) cho chúng ta biết rằng thông điệp chứa các trường 3,11,41.

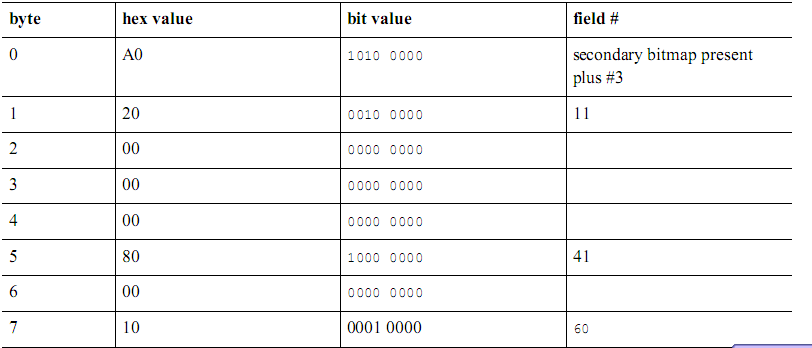
Xem xét thông điệp có bitmap và các trường như sau:



Hình ‑

Chúng ta xem xét Primary Bitmap và Seconary Bitmap.

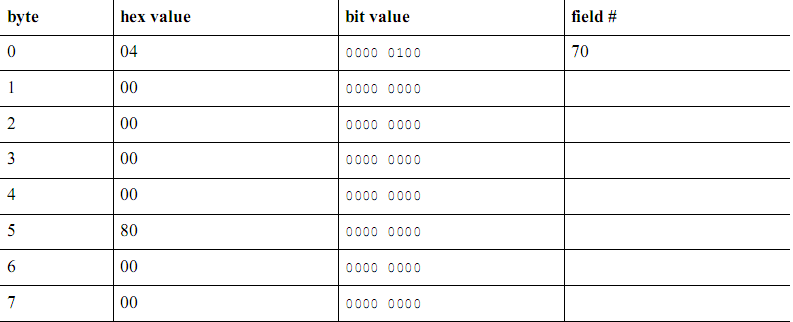
* Primary Bitmap



Hình 2‑8

***Bit đầu tiên trong chuỗi bit value của Primary Bitmap bật lên thành 1 cho biết thông điệp có chứa Seconary Bitmap.***

* Seconary Bitmap :



Hình 2‑9

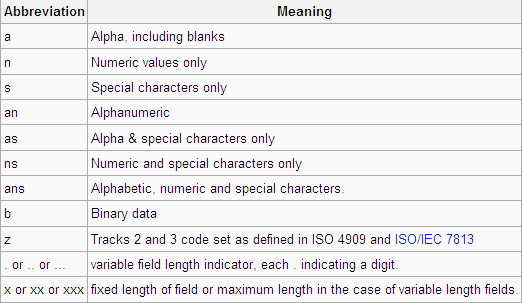
Chuyển giá trị thập phân về giá trị nhị phân rồi tiến hành xác định vị trí của các bit bật lên thành 1 để xác định chỉ số của các trường tương tự như Primary Bitmap.

### Thành phần dữ liệu (Data Element)

Thành phần dữ liệu (Data Element) hay còn được gọi là trường dữ liệu (Field) chứa các trường dữ liệu cá nhân và mang các thông tin giao dịch.

Có tới 128 trường dữ liệu trong chuẩn ISO8583 phiên bản 1987. Và lên đến 192 trường dữ liệu cho các phiên bản về sau.

Mỗi trường dữ liệu thì tuân theo một chuẩn định dạng để chỉ định nội dung của các trường và độ dài của trường đó.



Hình 2‑10

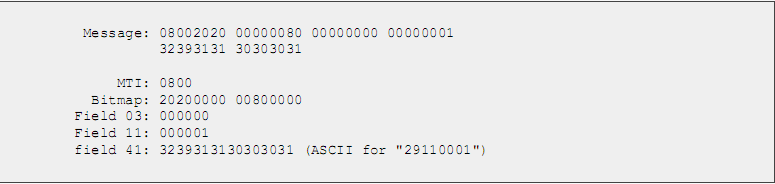
Chiều dài của các trường được chia làm 2 dạng :

* Chiều dài cố định (fixed length) kí hiệu x hay xx hay xxx đó là chiều dài tối đa mà trường đó có thể chưa.(xxxx chỉ dùng cho chuẩn ISO8583 phiên bản 2003).Các trường có chiều dài cố định thường không biểu diễn chiều dài trong nội dung thông điệp.[[2]](#footnote-3)
* Chiều dài có thể biến đổi (variable length) kí hiệu ‘**.’** hay ‘**..’** hay **‘…’** Mỗi ‘**.’** đại diện cho một chữ số. Các chiều dài có thể biến đổi trong quá trình biểu diễn phải thể hiện chiều dài trước nội dung thông điệp.

Nội dung của các trường có thể là số,chữ hoặc là các kí hiệu đặc biệt hoặc là sự kết hợp của số, chữ và các kí hiệu đặc biệt.

Một điểm chúng ta cần lưu ý là bạn có một trường là số (numeric field) nhưng bạn có thể biểu diễn nó bằng một chuỗi các kí tự ASCII hay EBCDIC.

Xem xét ví dụ sau :



Hình 2‑11

Trong ví dụ của chúng ta thấy truờng số #03 biểu diễn theo kiểu BCD sử dụng 3 byte thập lục phân với giá trị của nó là “00 00 00”.Tương tự đối với trường #11 giá trị là “00 00 01” được biểu diễn bởi 3 byte thập lục phân giá trị “00 00 01”.Trường số #41 được biểu diễn gồm 8 byte (32 39 31 31 30 30 30 31) thập lục phân thể hiện 8 kí “29110001” trong bảng mã ASCII.

## Wire Protocol

Sau khi chúng ta đã có thông điệp ISO8583 chúng ta phải vận chuyển nó trên đường truyền sử dụng một vài giao thức giao tiếp như (TCP/IP, UDP/IP, X.25, SDLC, SNA, ASYNC, QTP, SSL,…..)

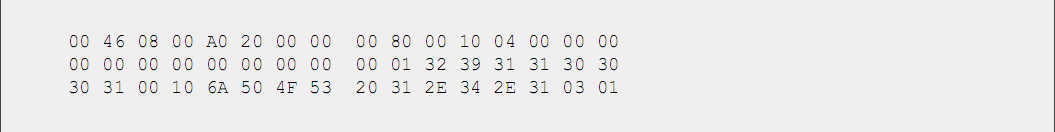
Nhưng các phương thức này không phải là một phần của thông điệp ISO8583 vì thế các nhà sản xuất và các doanh nghiệp có thể tùy chọn phương thức giao tiếp phù hợp với họ.

Một số giao thức(Đặc biệt là là các giao thức cũ) thường cần thêm header để cung cấp thông tin cho router. Trong khi một số khác thì cần có trailer.

Vì vậy một chuỗi thông điệp truyền trên đường mạng (stream-based) thì thường gồm các phần sau :

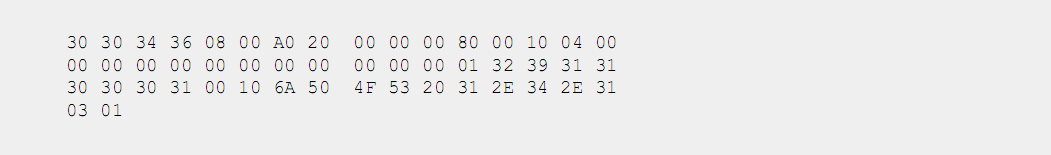
* Header(Có thể không có)
* ISO8583 Message(Bắt buộc)
* Trailer(Có thể không có)

Với phương thức TCP/IP sử dụng 2 byte để chỉ ra chiều dài của chuỗi truyền trên đường truyền.Vì thế một chuỗi truyền 46 byte có thể truyền như sau :



Hình 2‑12

Tuy nhiên đây chỉ là một cách trong việc chỉ ra chiều dài chuỗi truyền.Chúng ta có thể chọn cách truyền chiều dài của chuỗi truyền dưới dạng ASCII như sau :



Hình 2‑13

Trong đó 0046 🡪 30 30 34 36 (Giá trị ASCII của 0046)

Sau khi đã nắm rõ về thông điệp ISO8583 chúng ta tiến hành thiết kế thông điệp cho bài toán :

## Thiết kế thông điệp cho giao dịch cộng điểm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field No | Data Element Name | Attribute | Request | Respone | Note |
|  | MTI | n 4 | M | M | HEX |
|  | Bitmap | b 8 |  |  | HEX |
| 3 | Processing code | n 6 | M | M | HEX |
| 4 | Amount Transaction | n 12 | M |  | HEX |
| 35 | Track 2 Data | z..37 | M |  | HEX |
| 39 | Response Code | an 2 |  | M | ASCII |
| 41 | Card Acceptor Terminal ID | ans 8 | M | M | ASCII |
| 42 | Card Acceptor Identification Code | ans 15 | M | M | ASCII |
| 48 | Reserse Private | ans…999 |  | M | Length of field (Hex) and Content of field (ASCII). |
| 61 | Reverse Private | ans …999 |  | M | Length of field (Hex) and Content of field (ASCII). Use Tag Length value for each message. |

Bảng đặc tả cho cách trường :

### MTI (Message Type Indicator) :

|  |  |  |
| --- | --- | --- |
| Request | Respone | Description |
| 0200 | 0210 | Add Point |

### Field No : 3 Processing code

Processing code chia làm 3 trường nhỏ :

* Transaction Type code,AN2
* Account Type Code 1,AN2
* Account Type Code 2,AN2 (skip : default : 00)

#### The transaction Type Code(TTC):

TTC được sử dụng với MTI như là một khóa duy nhất để định nghĩa tác vụ cần thực hiện.

* Request

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0200 | 40 | Add Point |

* Response

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0210 | 40 | Add Point |

#### Account Type Code (ATC) chỉ ra loại của tài khoản.

|  |  |
| --- | --- |
| ATC | Description |
| 70 | Loyalty Card |

### Field 48

|  |  |  |
| --- | --- | --- |
| Tag | Length | Description |
| FF51 | 04 | Điểm lấy từ giao dịch |
| FF52 | 04 | Tổng điểm |

### Field 61

|  |  |  |
| --- | --- | --- |
| Tag | Length | Description |
| FF01 | Chiều dài nội dung cho mỗi thông điệp | Nội dung cùa mỗi thông điệp gởi về cho client. |

### Message sample :

#### Message request:

**Header** : 6000030000 🡺 5 byte (hex)

**MTI** : 0200 🡺 2 byte (hex)

**Bitmap** : 3000000020C00000 🡺 8 byte (Hex)

**F3** : 407000 🡺 3 byte (Hex)

**F4** : 000000200000 🡺 6 byte (hex)

**F35** : 379604215000000046D140610114680300000000 🡺 20 byte ( hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

Total Message length : 5 + 2 + 8 + 3 + 6 + 20 + 8 + 15 = 67 🡺 2 byte (hex) 0043

0043600003000002003000000020C00000407000000000200000379604215000000046D1406101146803000000003232322020202020322020202020202020202020202020

#### Message Response :

**Header** : 6000000003 🡺 5 byte (hex)

**MTI** : 0210 🡺 2 byte (hex)

**Bitmap** : 2000000000C10008 🡺 8 byte (Hex)

**F3** : 407000 🡺 3 byte (Hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

**F48**: 0014FF510430303032FF520430303238 🡺 16 byte

**F61** :

* Message 1st:

**Tag** : FF01 => 2 byte hex

**Length 1st message** : 20 => 0x14 => 1 byte

**Message content 1st** : 31302042414e204e48414e20414f205448554e20

(Value : 10 BAN NHAN AO THUN)

* Message 2nd :

**Tag** : FF02 => 2 byte

**Length 2nd message** : 18 => hex 0x12 => 1 byte

**Message 2nd content**: 32302042414e204e48414e20584520444150

(Value : 20 BAN NHAN XE DAP)

**Total F61 length** : 2 + 1 + 20 + 2 + 1 + 18 = 44 + 2 = 46 ( 2byte to send 0044)

0044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

**Total Message length** : 5 + 2 + 8 + 3 + 8 + 15 + 16 + 46 = 103 🡺 2 byte (hex) 0067

0067600000000302102000000000C1000840700032323220202020203220202020202020202020202020200014FF510430303032FF5204303032380044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

***Các đặc tả cho trường thông điệp của các tác vụ khác chúng ta có thể xem trong phần phụ lục.***

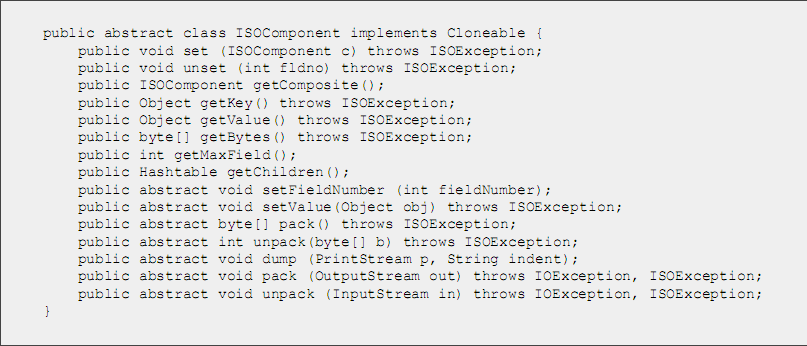
## JPOS cách thức xử lý ISO8583

Phần này tập trung mô tả cách thức JPOS xử lý thông điệp ISO8583.

### ISOMsg & Co

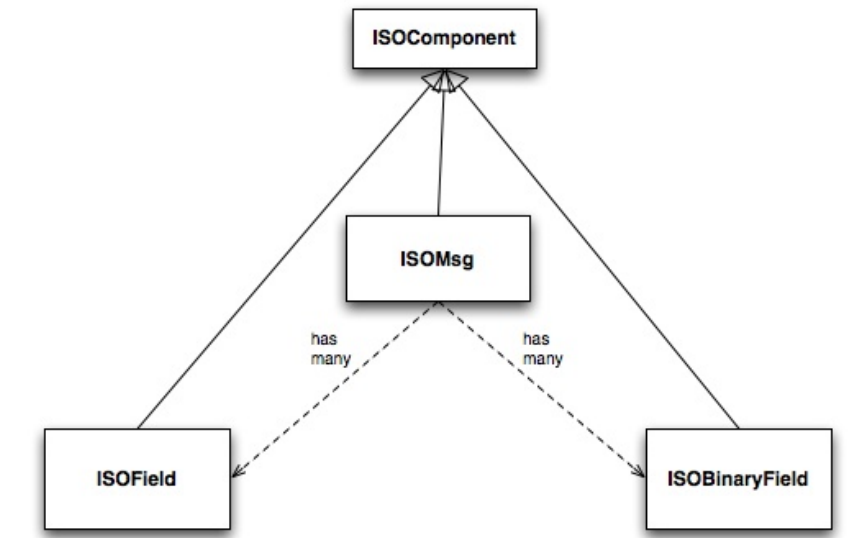
Đại diện cho một thông điệp ISO8583 JPOS sử dụng đối tượng ISOMsg.

ISOMsg,ISOField,ISOBitmapField,ISOBinaryField và bất cứ Custom Field nào của ISO8583 đều hiện thực từ lớp ISOComponent. Chúng ta xem xét các phương thức của lớp ISOComponent sau :



Hình 2‑14

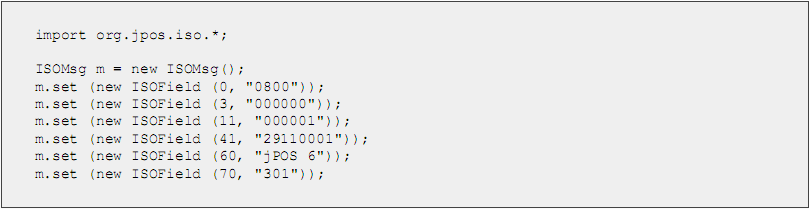
Chúng ta theo dõi diagram sau để xem cách mà ISOComponent tương tác với các lớp đối tượng khác :



Hình 2‑15

Các lớp đối tượng quản lý thông điệp ISO8583 được thiết kế theo mẫu Component như hình trên.(Chi tiết về mẫu Component chúng ta có thể tham khảo tại trang sách ).

Chúng ta có thể sử dụng phương thức ISOComponent.set(ISOComponent) để tạo một message ISO8583 như sau :



Hình 2‑16

Đoạn mã nguồn tạo mới một thông điệp ISO8583. Sau đó khởi tạo MTI với giá trị 0800(MTI được xem là trường số 0 trong thông điệp 8583).

Sau đó thiết lập giá trị cho các trường 3,11,41,60,70.

Tuy nhiên để rút gọn và code dễ đọc hơn JPOS cung cấp các phương thức tiện dụng như sau :

ISOMsg.setMTI(String)

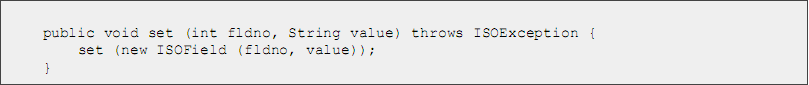
Phương thức trên thiết lập MTI cho message ISO8583.

Và

ISOMsg.set(int fieldNumber,String fieldValue)

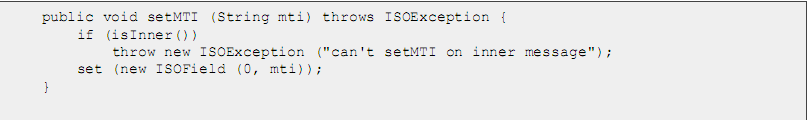
Phương thức ISOMsg.set thiết lập trường giá trị (tham số fieldValue) cho trường ở vị trí được chỉ định (tham số fieldNumber).

Thực sự bên trong các phương thức này bao bọc lại các phương thức còng kềnh phía bên trên :



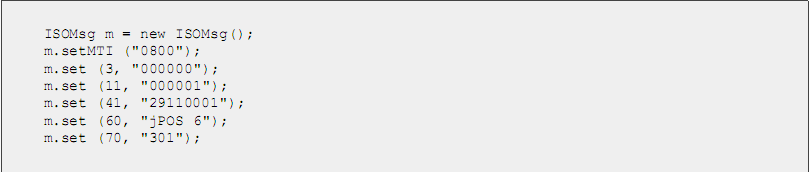
Hình ‑

Tương tự



Hình 2‑18

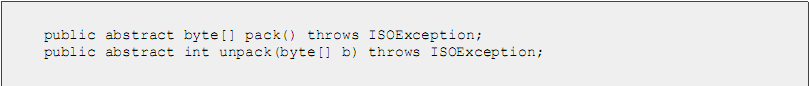
Vì thế ISO8583 message ban đầu có thể viết lại như sau :



Hình ‑

### Packing và Unpacking

JPOS có hai phương thức khá hữu dụng :

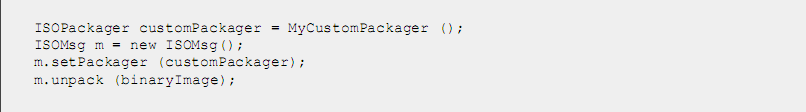
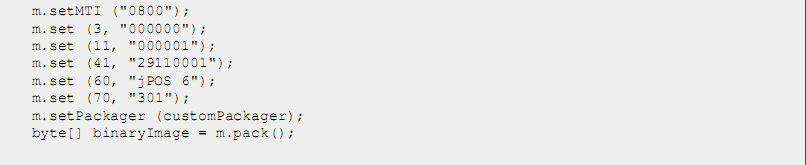


Hình 2‑20

Phương thức pack() trả về một mảng byte[] chứa giá trị đại diện của một thành phần cho trước( Thành phần này có thể là một field hoặc là toàn bộ message).

Phương thức unpack thì thực thi ngược lại tức là chuyển đổi một mảng byte[] trở thành giá trị của một field hoặc toàn bộ message.

Do mỗi thông điệp chứa rất nhiều trường trong đó. Các trường được cấu hình theo những cách rất khác nhau (fix-length ASCII ,fix-length HEX, var length ASCII, var length HEX hoặc thậm chí ASCII kết hợp với HEX) do đó mỗi message đều được chỉ ra rằng các trường của mình được mô tả như thế nào thông qua packager. Packager đơn giản là một bản mô tả về message và nhờ bản mô tả này JPOS có thể packing/unpacking theo bản mô tả đó.

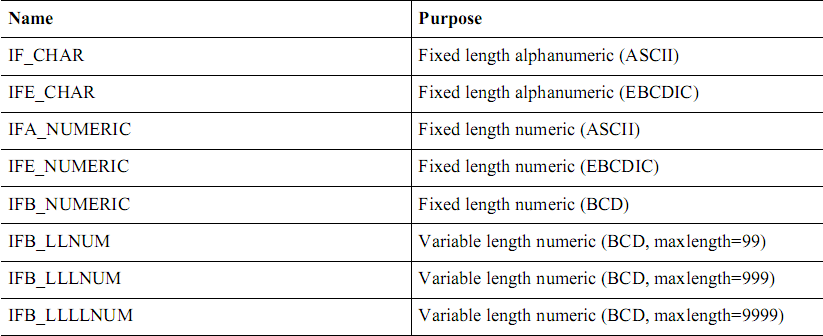


Hình 2‑21

Ví dụ trên chỉ cho chúng ta cách thức JPOS packing và unpacking message thông qua packager.

Như đã nói ở trên mỗi package mô tả cho các trường của thông điệp ISO8583 mỗi trường trong packager đại diện bằng đối tượng ISOFieldPackager.

JPOS thiết kế đầy đủ các ISOFieldPackager và chúng ta hầu như không cần phải viết lại.

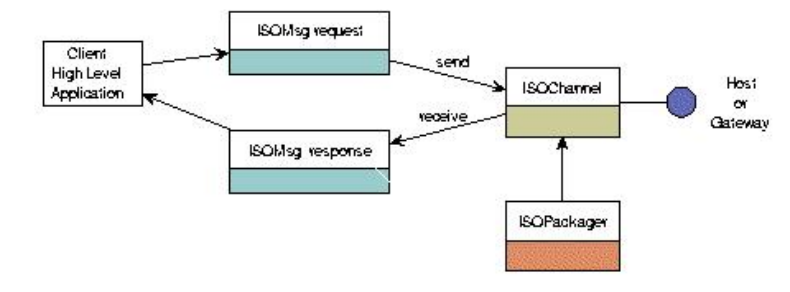


Hình 2‑22

Quản lý Wire Protocol với ISOChannel

JPOS sử dụng lớp đối tượng ISOChannel để quản lý Wire Protocol.

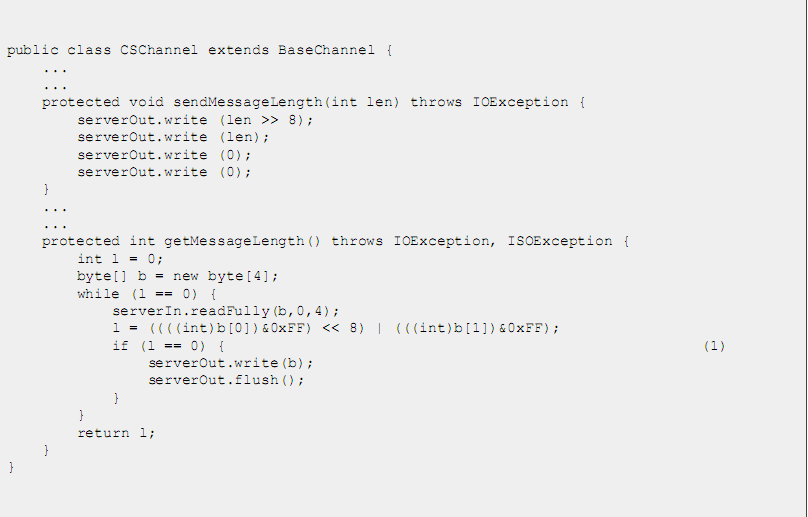
ISOChannel được sử dụng để gởi và nhận thông điệp ISOMsg.



Hình 2‑23

JPOS hổ trợ hầu hết các loại channel thông dụng. Hầu hết các channel trên nền tảng TCP/IP được mở rộng từ lớp org.jpos.iso.BaseChannel. Các channel này đơn giản là override 2 phương thức sendMessageLength và getMessageLength.

Xem xét CSChannel (org.jpos.iso.channel.CSChannel) sử dụng 2 byte để chỉ ra chiều dài của thông điệp ISO 8583 công với header của nó và 2 byte dự trữ cho những mục đích mở rộng sau này.



Hình 2‑24

Bây giờ chúng ta lướt qua các channel mà JPOS hỗ trợ :

|  |  |
| --- | --- |
| Channel Name | Wire Protocol |
| CSChannel | * LL LL 00 00 [header] ISO-DATA * LL LL 2 byte chỉ ra chiều dài của [Header] công với ISO8583 image * 00 00 để dành cho tương lai. * Header (lựa chọn). * ISO-DATA-8583 image (byte thô) |
| NACChannel | * LL LL [TPDU] ISO-DATA * LL LL 2 byte chỉ ra chiều dài của [TPDU] cộng với ISO8583 image * TPDU (Transport protocol data unit)[Lựa chọn] * ISO-DATA-8583 image (byte thô) |
| NCCChannel | * LL LL [TPDU] ISO-DATA * LL LL 2 byte chỉ ra chiều dài của [TPDU] cộng với ISO8583 image(Khác với NACChannel ở chỗ 2 byte này được gởi theo kiểu BCD) * TPDU (Transport protocol data unit)[Lựa chọn] * ISO-DATA-8583 image (byte thô) |
| ASCIIChannel | * LLLL[Header]ISO-Data * LLLL 4 byte ASCII chỉ ra chiều dài [Header] và ISO-DATA * Header (Lựa chọn) * ISO-DATA – 8583 image(byte thô) |
| RawChannel | * LL LL LL LL [header] ISO-DATA * LL LL LL LL chỉ ra chiều dài của [Header] và ISO-DATA * ISO-DATA -8583 Image(Byte thô) |

## Transaction là gì?

Chúng ta từng nghe nhắc đến rất nhiều lần về transaction mà đặc biệt là transaction trong database.Vì vậy bài viết chỉ nêu lên những nét tổng quát và chủ yếu nhất của một transaction đồng thời đi sâu vào việc quản lý transaction trong JPos 1.6.4 .

Transaction là một chuỗi các hành động được thực thi trong đó mỗi hành động phải **nguyên tử**.Đưa ra quyết định về sự thành công cũng như thất bại của tất cả các hành động.(Việc đưa ra khái niệm dựa trên hiểu biết của tác giả về transaction)

Để một chuỗi các hành động được gọi là một transaction thì chuỗi hành động đó phải đảm bảo 4 tính chất sau :

* Atomic (Nguyên tử)
* Consistency (Nhất quán)
* Isolation (Độc lập)
* Durable(Duy trì)

### Các tính chất của Transaction

#### Atomic

Một hành động được gọi là nguyên tử nghĩa là nó không thể phân chia ra những hành động con bên trong nó nữa.

Ví dụ : Xét ví dụ về hai hành động sau : Tính điểm trung bình của một sinh viên và tính điểm trung bình của một khóa học trong trường đại học.Chúng ta xem xét hành động thứ hai : tính điểm trung bình của một khóa học mà điểm trung bình của một khóa học thì nó bao gồm điểm trung bình của các sinh viên. Do đó chúng ta nói rằng hành động thứ hai không đạt tiêu chí atomic. Còn hành động thứ nhất thì đáp ứng được tiêu chí này. Việc xét các tiêu chí dựa trên ngữ cảnh của bài toán.

#### Consistency

Giữa các chuỗi hành động đôi lúc cần trao đổi dữ liệu qua lại cho nhau. Tính nhất quán thể hiện ở chỗ các dữ liệu trao đổi qua lại giữ các hành động phải được nhất quán một cách chính xác về giá trị.

#### Isolation

Sự độc lập giữa các transaction thể hiện ở chỗ các tài nguyên (cả tĩnh và động) **chỉ được** truy xuất đúng trong transaction của nó.

#### Durable

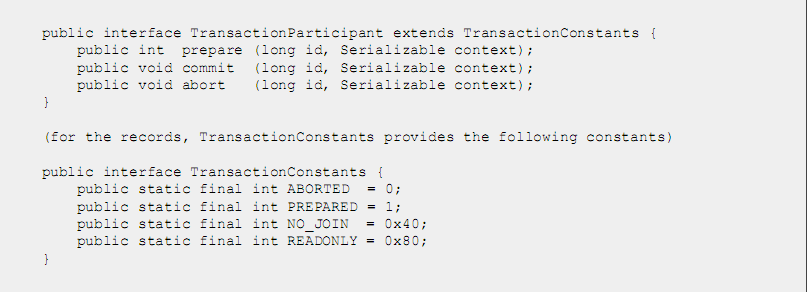
Thuộc tính này chỉ ra rằng khi một sự kiện xảy ra trên một hành động của transaction thì transaction manager sẽ đưa ra quyết định cho hành động đó hoặc thậm chí **đưa ra quyết định** cho toàn bộ các hành động của transaction hoặc các hành động có liên quan.

### Transaction trong JPos 1.6.4

#### Participant

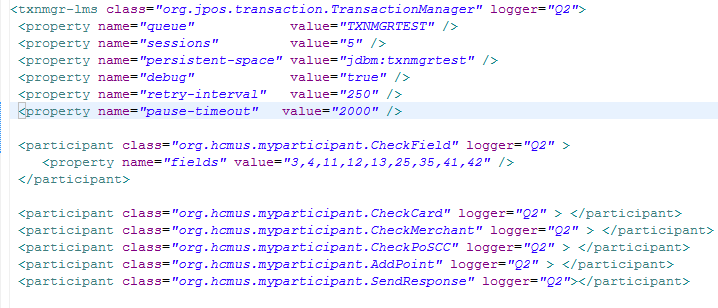
Transaction trong Jpos 1.6.4 mang đầy đủ các tính chất của một transaction.

Các hành động atomic sẽ được được gọi là **participant**. JPos cung cấp các interface để chúng ta thực thi các hành động này .



Hình 2‑25

Để khai báo một transaction trong Q2 đơn giản là tạo một file xml để cấu hình cho transaction đó trong sau đó lưu lại trong thư mục deploy. Một điều đáng chú ý trong việc cấu hình transaction là thuộc tính class của thẻ gốc phải chỉ tới lớp ***org.jpos.transaction.TransactionManager***.



Hình 2‑26

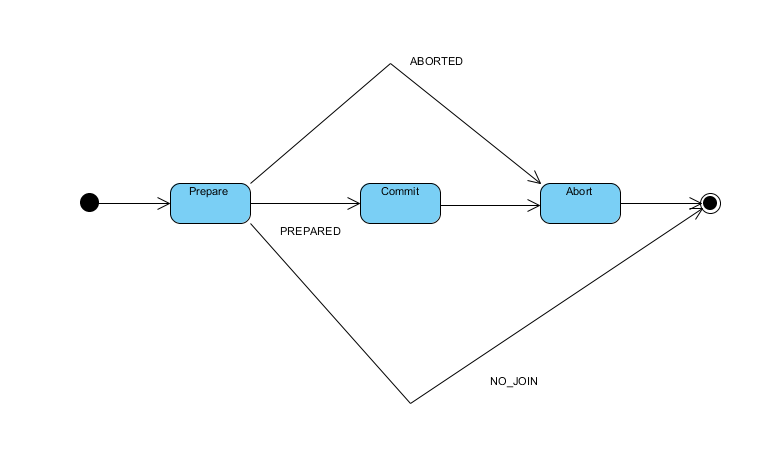
Các participant được cấu hình sẽ thực thi interface TransactionParticipant.

#### Pha

Trước khi bàn về việc quản lý và đưa ra quyết định cho các participant của jpos . Chúng ta xem xét các pha mà một participant tham gia vào. JPos chia ra làm 3 pha chính Prepare, Commit và Abort.

|  |  |  |
| --- | --- | --- |
| Name | Value | Description |
| ABORTED | 0 | Participant trong pha Prepare. Participant sẽ chuyển qua pha Abort. |
| PREPARED | 1 | Participant đã được prepare và và kiểm tra tính prepared cho các paricipant khác. |
| NO\_JOIN | 0x40 | Báo cho bộ quản lý transaction biết rằng participant này không tham gia vào pha commit cũng như abort. |
| READ\_ONLY | 0x80 | Báo cho bộ quản lý transaction biết rằng participant có không sửa đổi các dữ liệu được duy trì vì thế việc lưu lại một bản sao của nó là không cần thiết. |

Xem xét state chart sau :



Hình 2‑27

TransactionManager sẽ thực hiện nhiệm vụ điều khiển các Transaction bằng cách gọi phương thức prepare của tất cả các Participant.Nếu tất cả chúng đều trả về PREPARED thì transaction sẽ chuyển đến pha commit và tại đây phương thức commit của tất cả các participant sẽ được gọi.

Nếu một trong các participant trả về ABORTED thì transaction sẽ chuyển đến pha abort. Và tại đây phương thức abort của tất cả các participant sẽ được gọi.

#### Mở rộng vấn đề :

Khi một participant trả về NO\_JOIN để báo cho bộ quản lý transaction biết rằng participant đó không muốn tham gia vào quá trình transaction nữa. Tuy nhiên trong các trường hợp recovery thì transaction vẫn phải gọi phương thức commit và abort của nó. Vì vậy developer phải chuẩn bị để nhận phương thức commit hay abort của một (unknow transaction).

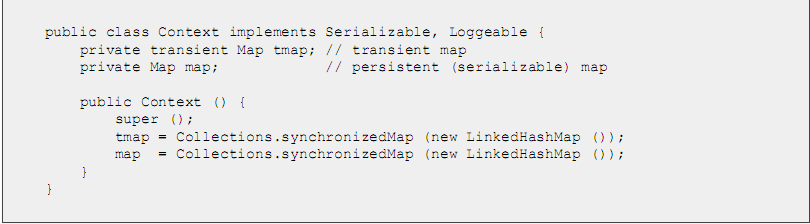
Source code của bạn sẽ kiểm tra long id hoặc Seriablizable context sau đó làm cái tác vụ mà bạn muốn.

#### Transaction Context:

Để đảm bảo cho tính chất Isolation và Consitency. Transaction dùng đối tượng Context để quản lý dữ liệu dùng chung cho các participant cũng như nhất quán dữ liệu đó xuyên qua các participant.

Trong context sử dụng 2 đối tượng **map : persitent map và trancient map** để quản lý và duy trì các đối tượng dùng chung .

Trancient map đưa cho bạn khả năng để đặt các đối tượng động vào trong context như socket connection, jdbc connection, thread, manage object.



Hình ‑

# : Thiết Kế Chương Trình

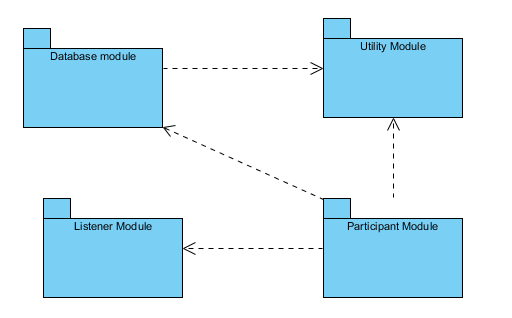
Chương trình tập trung vào việc thiết kế và thực thi chương trình.

## Thiết kế kiến trúc của chương trình :

### Thiết kế tổng quát

Hệ thống được thiết kế gồm 4 module chính : database module, participant module, utility module and listener module.

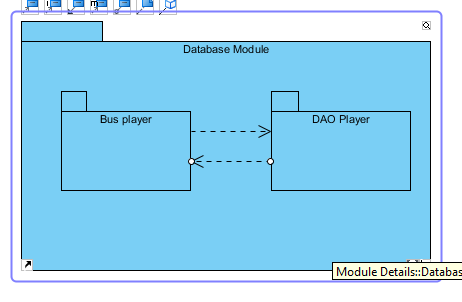
* Database module: Bao gồm tất cả các lớp và interface thực hiện liên kết và truy xuất cũng nhau tạo các lớp ánh xạ các bảng của database và thực thể của lớp. Module được thiết kế với kiến trúc 1 tier 2 player. Trong đó lớp DAO player được thiết kế theo mẫu Abstract Factory.Chi tiết cho module này được trình bày bên dưới.
* Participant module: Bao gồm tất cả các lớp liên quan đến nghiệp vụ và luồng của transaction.
* Utility module: Định nghĩa các lớp dùng chung hỗ trợ cho việc lấy giá trị của thông điệp ISO8583 và các lớp dùng chung khác cho tất cả các module đã đề cập ở trên.
* Listen module : Đây là module đơn giản nhất lắng nghe kết nối của client đồng thời đưa kết nối đó vào hàng đợi để xử lý.



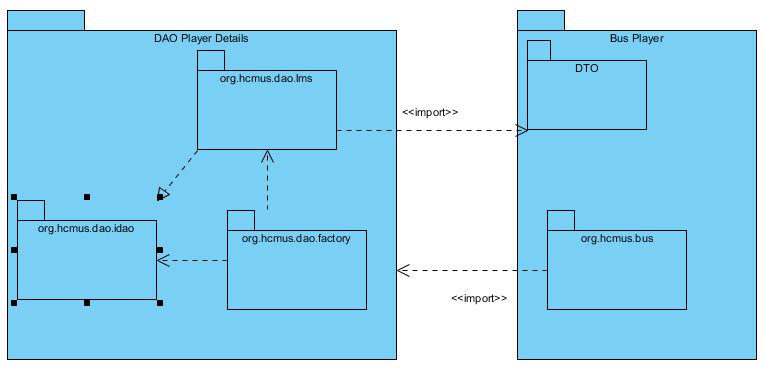
Hình ‑

### Chi tiết của kiến trúc

#### Database module:



Hình ‑



Hình ‑

### Org.hcmus.dao.package:

SQLDAOFactory

LMSDAOFactory

|  |  |
| --- | --- |
| Class | Description |
| LMSDAOFactory | Đây là lớp trừu tượng. Khi tạo một cơ sở dữ liệu mới bạn phải kế thừa từ lớp này. |
| SQLDAOFactory | Đây là một thể hiện của SQL database. |

### Org.hcmus.dao.idao

|  |  |
| --- | --- |
| Interface | Description |
| IJPOS\_Card | Đây là interface cho lớp card. |
| IJPOS\_Customer | Đây là interface cho lớp customer |
| IJPOS\_Gift | Đây là interface cho lớp gift |
| IJPOS\_Log\_Exchange | Đây là interface cho lớp log exchange. |
| IJPOS\_Log | Đây là interface cho lớp log |
| IJPOS\_Merchant | Đây là interface cho lớp merchant |
| IJPOS\_PoSCC | Đây là interface cho lớp PoSCC . |
| IJPOS\_Task | Đây là interface cho lớp task . |

### Org.hcmus.dao.lms

Tất cả các lớp trong package này thực thi từ interface trong ***org.hcmus.dao.idao***.

|  |  |
| --- | --- |
| Class | Description |
| JPOS\_CardDAO | Đây là lớp thiết kế cho card |
| JPOS\_CustomerDAO | Đây là lớp thiết kế cho customer |
| JPOS\_GiftDAO | Đây là lớp thiết kế cho for gift |
| JPOS\_Log\_ExchangeDAO | Đây là lớp thiết kế cho log exchange |
| JPOS\_LogDAO | Đây là lớp thiết kế cho log |
| JPOS\_MerchantDAO | Đây là lớp thiết kế cho merchant |
| JPOS\_PoSCCDAO | Đây là lớp thiết kế cho PoSCC |
| JPOS\_TaskDAO | Đây là lớp thiết kế cho task |

### Org.hcmus.bus

Tất cả các lớp trong package này thực hiện nghiệp vụ cho các lớp tương ứng trong ***org.hcmus.dao.lms*** package.

|  |  |
| --- | --- |
| Class | Description |
| JPOS\_CardBUS | This is class for card class. |
| JPOS\_CustomerBUS | This is class for customer class. |
| JPOS\_GiftBUS | This is class for gift class. |
| JPOS\_Log\_ExchangeBUS | This is class for log exchange class. |
| JPOS\_LogBUS | This is class for log class. |
| JPOS\_MerchantBUS | This is class for merchant class. |
| JPOS\_PoSCCBUS | This is class for PoSCC class. |
| JPOS\_TaskBUS | This is class for task class. |

#### Participant Module: org.hcmus.participant

|  |  |
| --- | --- |
| Class | Description |
| AddPoint | Tác vụ cộng điểm. |
| CheckField | Kiểm tra các các field gởi đến. |
| CheckCard | Kiểm tra Card gởi đến.(Ngày hết hạn, sự tồn tại của Card) |
| CheckMerchant | Kiểm tra các thiết bị quét mã card. |
| SendResponse | Trả kết quả về cho client. |

#### Utilily Module: org.hcmus.util

|  |  |
| --- | --- |
| Class | Description |
| Constants | All constants are defined here. |
| MLSConfig | Lấy database configuration từ xml file |
| MessageHelper | Hỗ trợ việc lấy thông tin đón nhận từ client |

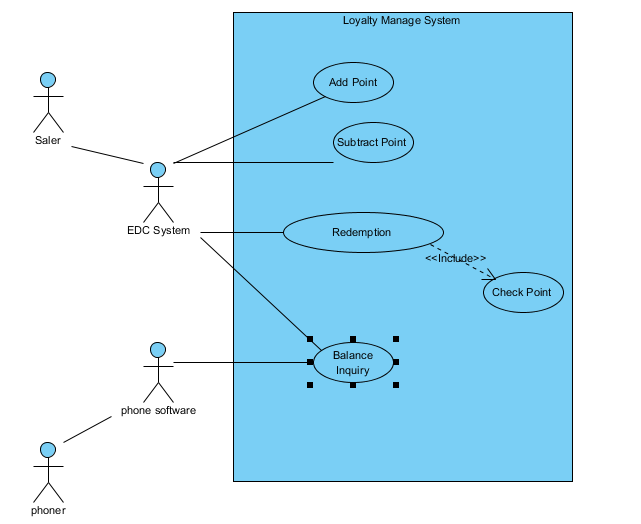
#### Listener Module : org.hcmus.lms

This is a small package but it is very important.

|  |  |
| --- | --- |
| Class | Description |
| LMSISOListener | Nhận thông điệp từ client. |

Sau khi đã thiết kế kiến trúc hệ thống chúng ta tiến hành mô hình hóa nghiệp vụ hệ thống.

## Mô hình hóa nghiệp vụ



Hình ‑

### Actors

|  |  |
| --- | --- |
| Actor | Actor Description |
| Saler | Nhân viên bán hàng |
| EDC System | Các thiết đọc mã thẻ và chuyển mã thẻ lên server dưới dạng thông điệp ISO8583. |
| Phoner | Người sử dụng điện thoại |
| Phone Software | Thiết bị điện thoại có hỗ trợ độc barcode và có phần mềm chuyển thông điệp dưới dạng ISO 8583 lên server. |

### User Cases

|  |  |  |
| --- | --- | --- |
| User Case ID | UserCase Name | UserCase Description |
| UC\_LMS\_01 | Add Point | Cộng điểm khi khách mua hàng. |
| UC\_LMS\_02 | Subtract Point | Trừ điểm khi khách trả hàng. |
| UC\_LMS\_03 | Redemption | Đổi điểm lấy quà tặng |
| UC\_LMS\_05 | Banlance Inquiry | Truy vấn số điểm hiện có. |

### Đặc tả Usercase :

#### Đặc tả usercase Add Point

##### Tóm tắt

Usercase này nhầm cộng điểm vào thẻ của khách hàng đã mua hàng ứng với số tiền mà họ đã mua.

##### Dòng dự kiện

###### Dòng sự kiện chính :

Sự kiện này bắt đầu khi người bán hàng dùng máy EDC quét mã thẻ của khách hàng đồng thời mã tác vụ thực hiện là cộng điểm.

Hệ thống yêu cầu máy EDC cung cấp thông tin về : Mã thẻ, mã tác vụ,số tiền của khách hàng đã mua.

Hệ thống tiến hành kiểm chứng các giá trị được chuyển lên.

Hệ thống cộng điểm và tiến hành lưu lại vào log trên database.

Thông báo giao dịch thành công và trả thông tin quảng cáo cho người dùng.

###### Các dòng sự kiện khác

* Không gởi đủ các trường của thông điệp. Khi EDC chuyển lên không đủ các trường mà hệ thống yêu cầu thì hệ thống sẽ trả về người dùng thông báo lỗi các trường của thông điệp bị lỗi.
* Mã thẻ đã hết hạn sử dụng : khi EDC chuyển lên mã thẻ đã hết hạn sử dụng. Hệ thống sẽ trả về cho người dùng lỗi mã thẻ đã hết hạn sử dụng.
* Mã thẻ không tồn tại trong hệ thống : EDC chuyển lên mã thẻ chưa được đăng kí. Hệ thống sẽ trả về mã lỗi thẻ không tồn tại trong hệ thống.
* Mã thẻ chưa được kích hoạt : EDC chuyển lên mã thẻ chưa được kích hoạt. Hệ thống sẽ trả về mã lỗi thẻ chưa được kích hoạt trong hệ thống.
* TID và MID chưa được đăng kí trên hệ thống : Nếu EDC chuển lên mã TID và MID chưa được đăng kí trên hệ thống thì hệ thống sẽ báo TID và MID không tồn tại.

##### Các yếu cầu đặc biệt

Không có

##### Trạng thái hệ thống khi bắt đầu thực hiện usercase.

Hệ thống server đã được kích hoạt và lắng nghe trên port đã cấu hình.

##### Trạng thái của hệ thống sau khi thực hiện usercase

Nếu usercase thành công thì giá trị điểm của thẻ khách hàng được cập nhật vào database. Ngược lại giá trị không thay đổi.

##### Điểm mở rộng :

Không có.

#### Thực thi chương trình

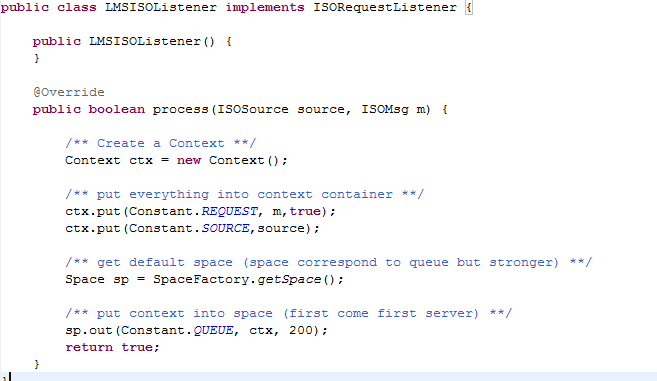
Khi một thông điệp gởi đến chương trình theo đúng kênh truyền(channel) và cổng(port) mà server đang lắng nghe (Việc cấu hình và đăng kí server với QSP và Q2 xem phần phụ lục QSP và Q2). Server sử dụng đối tượng thực thi từ Interface ISORequestListener trong đó có 2 tham số :

* Một là ISOSource : đây thực chất là một socket nhận nhiệm vụ trả message về cho client sau khi đã xử lý.
* Hai là ISOMsg : đây là thông điệp nhận từ client.

Tuy nhiên vì một bussiness thực hiện qua rất nhiều các tác vụ con (participant) nên thông điệp nhận được này cần được xử lý như một transaction để có thể đối phó tốt khi có lỗi xảy ra hay các luồng sự kiện phụ xảy ra.

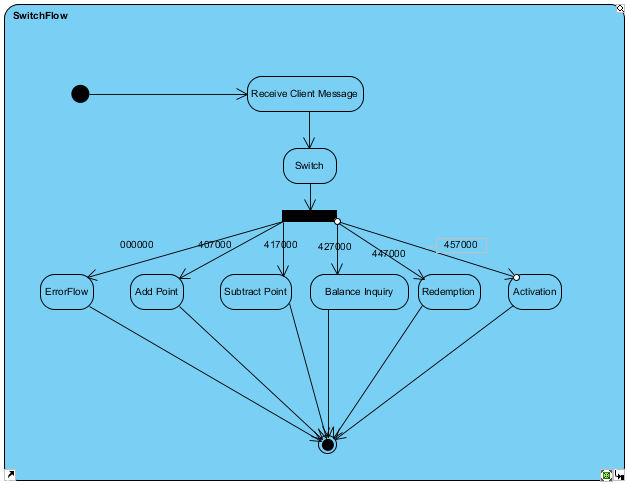
Như đã đề cập trong phần cơ sở lý thuyết về transaction phía trên. Chúng ta cần sử dụng một đối tượng Context. Đối tượng này nhầm mục đích duy trì và độc lập giá trị giữa các transaction với nhau.

Sau đó chúng ta sẽ đặt Context này vào trong Space (Xem phần phụ lục về Space) để đảm bảo tính tuần tự khi xử lý cũng như chuyền tay Context này qua TransactionManager quản lý.



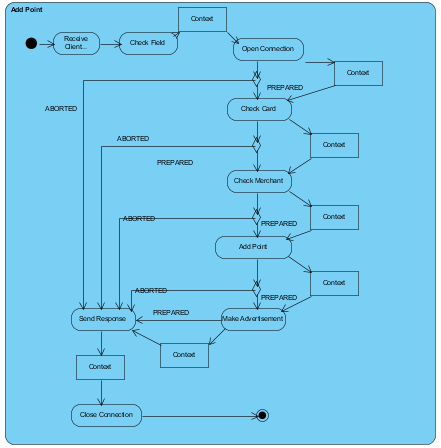
Hình 3‑5

Khi một Context chuyển qua TransactionManager thì Context đó sẽ được TransactionManage lấy ra từ trong Space sau đó khởi tạo một transaction. Và cuộc đời của Transaction bắt đầu từ đây. Context này sẽ  ‘lăn’ qua tất cả các (Atomic Bussiness của Transaction mà trong JPOS gọi là participant ). Và Participant đầu tiên mà Context phải trãi qua đó là Switch làm nhiệm vụ chụp bắt và kiểm tra MIT và trường số 3 (Processing Code) của message chứa trong Context để biết tác vụ cần thực hiện. Khi đã xác định được tác vụ cần thực hiện thì switch sẽ forward thông điệp trên qua đúng tác vụ xử lý cho thông điệp đó.



Hình 3‑6

Sau khi về đúng với các tác vụ cần thực hiện Context sẽ lăn tiếp qua các participant trong tác vụ đó.

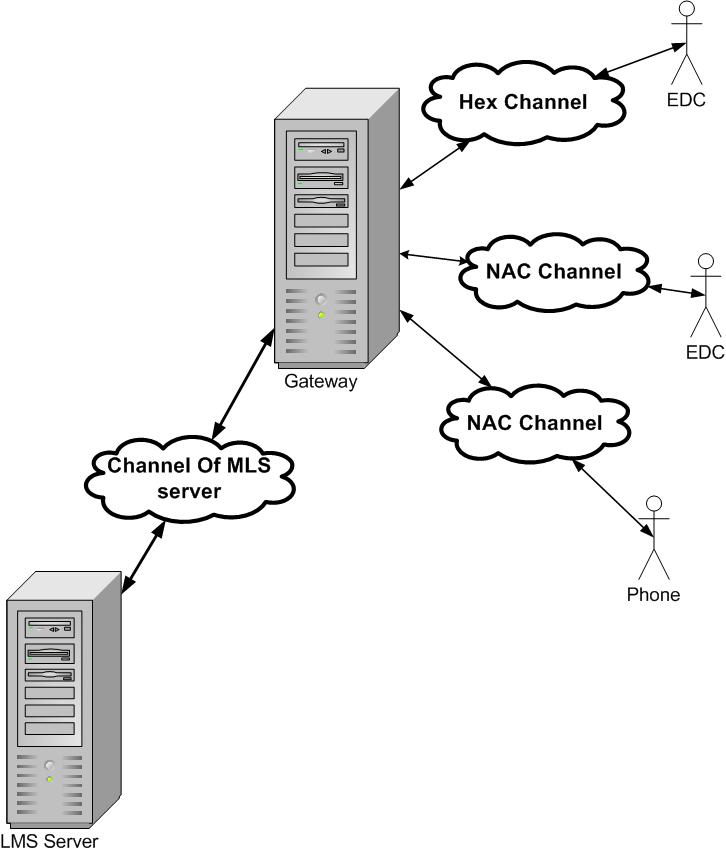


Hình 3‑7

Việc đặc tả cho các usercase khác xem thêm trong phần phụ lục.

#### GateWay (Multi Channel)

Như đã đề cập ở trên thông điệp truyền trên đường truyền theo các kênh truyền khác nhau. Có kênh truyền sử dụng header có kênh truyền thì không sử dụng, có kênh truyền gởi thông điệp theo dạng Hex lại có kênh truyền gởi thông điệp theo dạng ASCII.Cụ thể đối với thiết bị di động ở xa, đường truyền tương đối yếu hơn so với các thiết bị EDC đặt tại chỗ kết nối trực tiếp với hệ thống như một mạng LAN vì thế điện thoại thường sử dụng channel với mã HEX để truyền thông điệp nnhưng EDC lại chọn mã ASCII để truyền.



Hình 3‑8

Vì vậy việc đề xuất ra một gateway đóng vai trò trung gian và chuyển đổi các kênh truyền độc lập với server là một điều hết sức khả thi.Đó là lý do mà gateway ra đời.

Gateway sẽ nhận các thông điệp trên các kênh truyền do client chuyển đến chuyển kênh truyền đó phù hợp với kênh truyền của Server và sau đó nhận thông điệp theo kênh truyền của Server và trả thông điệp đó về theo kênh truyền của client.

Xét một một chuỗi thông điệp sau :

0043600003000002003000000020C00000407000000000200000379604215000000046D140610114680300000000**3232322020202020322020202020202020202020202020**

0038600003000002003000000020C00000407000000000200000379604215000000046D140610114680300000000**000000010000000000000010**

Chuỗi thông điệp đầu được cấu hình dưới channel torng đó trường số 41 và trường số 42 của thông điệp được gởi dưới dang ASCII vì thế chiều dài tổng cộng của thông điệp là 67 byte (0x43).

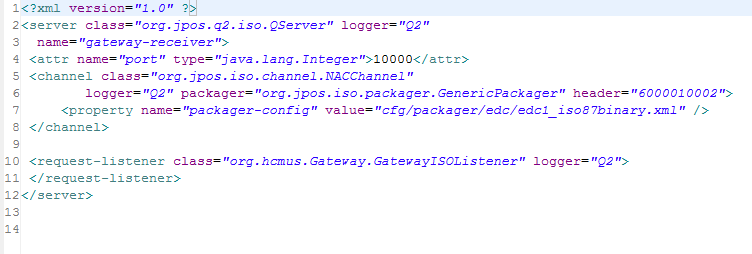
Xét chuổi thông điệp thứ hai trong đó trường số 41 và trường số 42 được gởi theo kiểu HEX vì thế độ dài của thông điệp ngắn hơn chỉ có 56 byte(0x38).

Nhiêm vụ của Gateway là sẽ nhận chuỗi thông điệp thứ 2 sau đó chuyển chuỗi thông điệp thứ 2 về đúng định dạng của chuổi thông điệp thứ nhất và gởi lên cho server.

**Để làm được việc này thì Gateway tiến hành như sau :**

Giá trị đầu vào của Gateway đó là các packager của Server và của mỗi client với các kênh truyền khác nhau.

Gateway thiết kế các server(Server của Gateway) với các port khác nhau để đón các thông điệp trên các kênh truyền khác nhau. Vì thế bài toán đa kênh truyền trở thành bài toàn đa port.



Hình 3‑9

Hình trên biểu diễn việc cấu hình cho một server lắng nghe trên port 10000 với kênh truyền edc1\_iso87binary.xml. Và thông điệp đến trên server này sẽ được lắng nghe như sau :

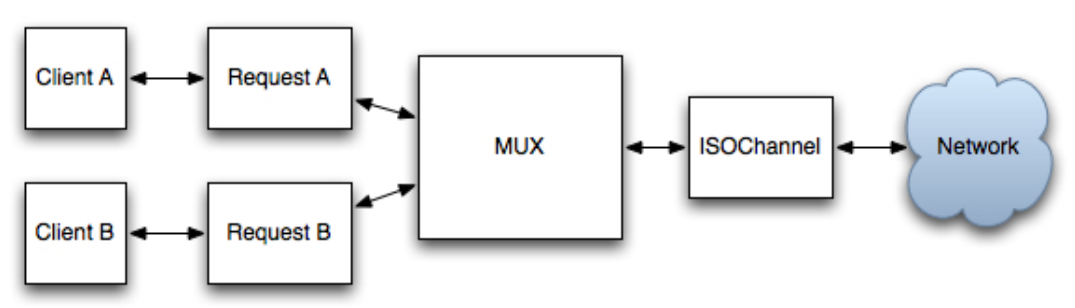
ISOMsg response = sender.Request(msg);

source.send(response);

Sử dụng đối tượng Sender để gởi thông điếp đến server đồng thời chờ thông điếp trả về sau đó gởi thông điệp trở về cho client.

**Xem xét đối tượng Sender**.

Sender đơn giản là một đối tượng MUX được cấu hình trong file xml và đăng kí với Q2 server. MUX bảo đảm rằng một thông điệp dưới client sẽ chuyển thông điệp lên server một cách tuần tự và nhận kết quả trả về một cách tuần tự. (lưu ý MUX được dùng ở client không được dùng như một server)



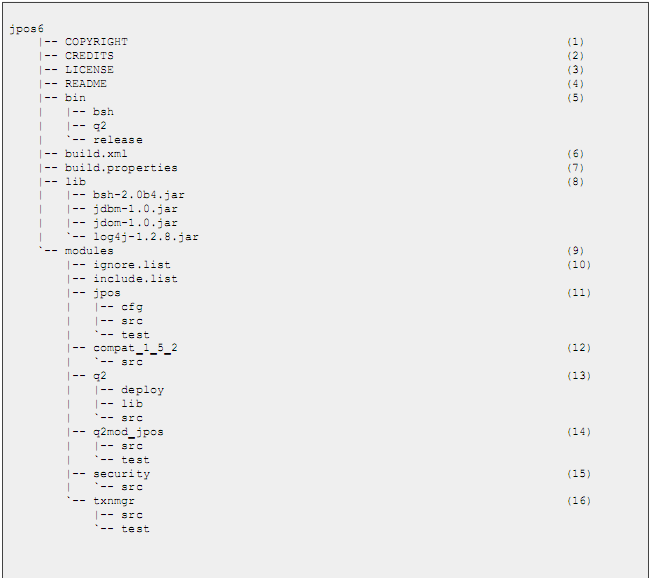
Hình ‑

# : Thực Nghiệm Và Đánh Giá Kết Quả

# Phụ Lục :

## Cách Thức Tổ Chức Biên Dịch Trong JPos

### Tổ chức thư mục của JPos Project



Hình ‑

COPYRIGHT : JPos’ copyright và Software copyright

CREDITS : Danh sách những người tạo nên JPos

LICENSE : Bản quyền GNU và AGNU

README : Thông tin để biên dịch JPos

BIN : Chứa mã Unix/Cygwin dùng để khởi chạy Q2

build.xml : File cấu hình hỗ trợ biên dịch sử dụng Apache Ant

build.properties : Định nghĩa các thuộc tính cho file cấu hình ở mục (6)

lib : Các thư viện liện được cung cấp để chaỵ project

modules : Chứa các module cần thiết cho quá trình biên dịch. Chúng ta có thể thêm các module khác do chúng ta tự định nghĩa vào đây.

File ignore.list cũng như include.list giúp file cấu hình ant biết được những module nào cần loại bỏ cũng như là cần biên dịch.

Jpos module là module chính của project chứa đựng các thứ khác liện quan tới những gói thao tác trên thoông điệp ISO8583

Compat\_1\_5\_2 : chứa đựng các thành phần thừa kế trong các phiên bản cũ. Trong các phiên bản hiện tại ít được mở rộng.

Q2 : Chứa gói Q2.

Q2 sử dụng một dịch vụ tên là QBean.QBean có thể sử dụng cho nhiều mục đích khác nhau.Thực hiện cấu hình và thực hiện đa tương tác với JPos component.

Security : Chứa các gói liên quan đến bảo mật

Txnmgr : Chứa các gói liên quan đến việc quản lý phiên giao dịch.

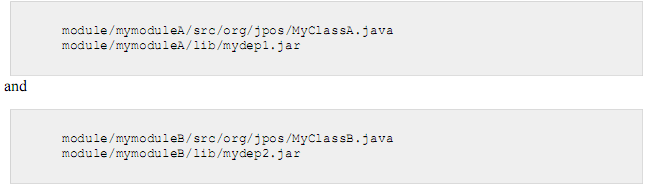
### Cơ chế biên dịch của JPos

Trong bất kì module nào của JPos cũng chứa đựng các thư mục sau. Module chứa một vài thư mục sử dụng trong quá trình thực thi(runtime directory) ví dụ như **deloy**.Q2 quét vào thư mục này để khởi chạy cũng như là dừng các dịch vụ (QBean). Trong khi đó thư mục **cfg** chứa các file cấu hình lúc đang thực thi (runtime configuration) và các file định nghĩa các thuộc tính. Thư mục **cfg/lib** chứa các file jar được cung cấp để chạy các ứng dụng. Thư mục test dung đề chứa các Unit test.

JPos sẽ liên kết tất cả các file trong các module (bao gồm các module mặc định của JPos và các Module khác được chỉ ra trong **include.list** file) . Sau đó tự động copy chúng vào thư mục **build/module** trong quá trình biên dịch.

Xem xét ví dụ sau :

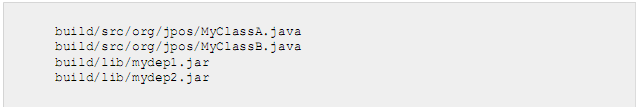
Chúng ta có moduleA và ModuleB chứa các file sau:



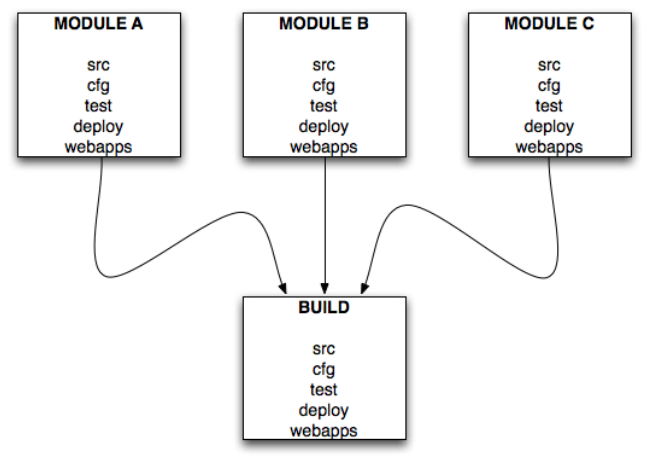
Hình 5‑2

Khi chúng ta gọi lệnh ant để biên dịch JPos thì bước đầu tiên nó sẽ sao chép:

Tất cả các file \*.java trong thư mục moudule/\*\*/src 🡺 build/src và moudule/\*\*/cfg/lib hoặc module/\*\*/lib 🡺 build/lib



Hình ‑



Hình 5‑4

**Khởi chạy JPos:**

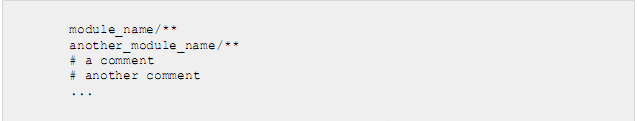
Vào thư mục build chứa file jpos.jar vừa mới biên dịch gõ lệnh sau:

java –jar jpos.jar

Có một Unix Shell Script (Chúng ta có thể chạy shell script này trên Window bằng Cygwin hay có thể dựa vào mẫu có sẵn tạo ra một file batch (\*.BAT) chạy trên window) và rồi called Q2 trong bin/q2 để khởi động Q2.

### Thêm vào hoặc loại bỏ module.

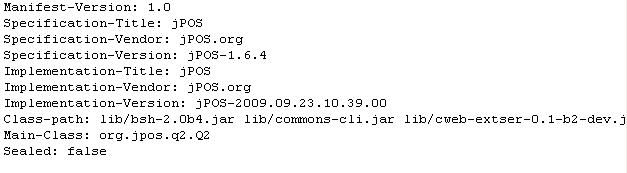
Chúng ta có thể thêm các module và gỡ bỏ các module cho quá trình biên dịch của JPos một cách dễ dàng thông qua module/include.list và module/ignore.list[[3]](#footnote-4)



Hình ‑

### Manifest File

JPos tự sinh ra một file Manifest chứa thông tin của ứng dụng trong ***build/classes/manifest.mf***

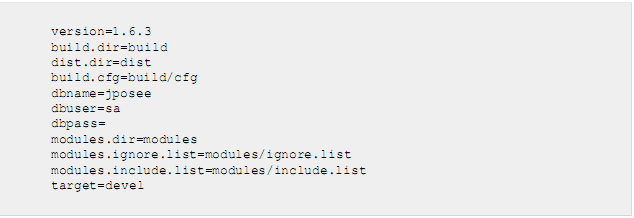


Hình ‑

### Build.xml và build.properties

Build.xml là một file cấu hình sử dụng Apache Ant để hỗ trợ cho việc thực thi ứng dụng[[4]](#footnote-5).

File build.properties cấu hình các thư mục sẽ được thực thi trong quá trình biên dịch đã trình bày ở trên. File build.xml sẽ sử dụng các thư mục được cấu hình trong build.properties dưới dạng các biến được định nghĩa sẵn.



Hình ‑

## Cài đặt JPos trên window

### Yêu cầu :

* Cài đặt Sun Java JDK 1.5 hoặc cao hơn : <http://java.sun.com/javase/downloads/index.jsp>
* Cài đặt Apache Ant 1.7.0 hoặc cao hơn : <http://ant.apache.org/>
* Cài đặt SVN client:
* CommandLine - <http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91>
* GUI Version - <http://tortoisesvn.net/downloads>
* OS : Window XP hoặc cao hơn. (Phần trình bày cài đặt trên windown XP)

### Thiết lập các biến mội trường

Nhấp phải vào My Computer chọn Properties chọn tab Advanced chọn Enviroment variable.

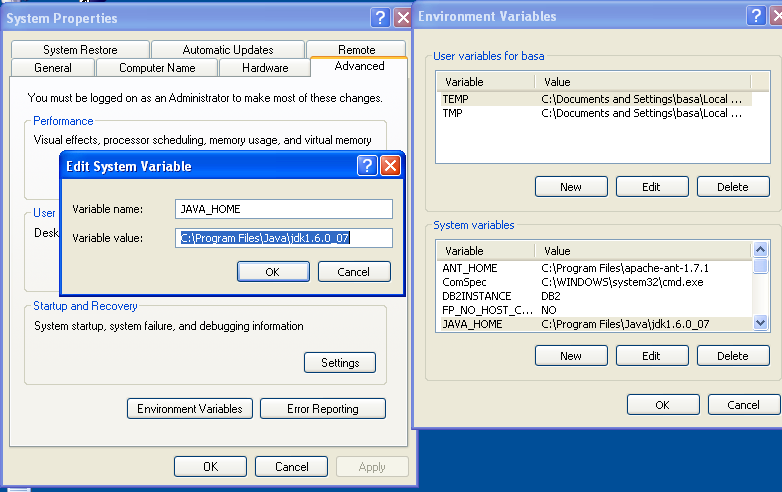
* Cài đặt JAVA\_HOME:

Mục đích là chỉ ra môi trường JDK nhầm sử dụng trong command line và sử dụng khi biên dịch trong các IDE như Eclipse hoặc NetBean.

Trong group : System variable chọn New

Variable name : JAVA\_HOME

Variable value : [Đường dẫn tới thư mục JDK cài đặt][[5]](#footnote-6)

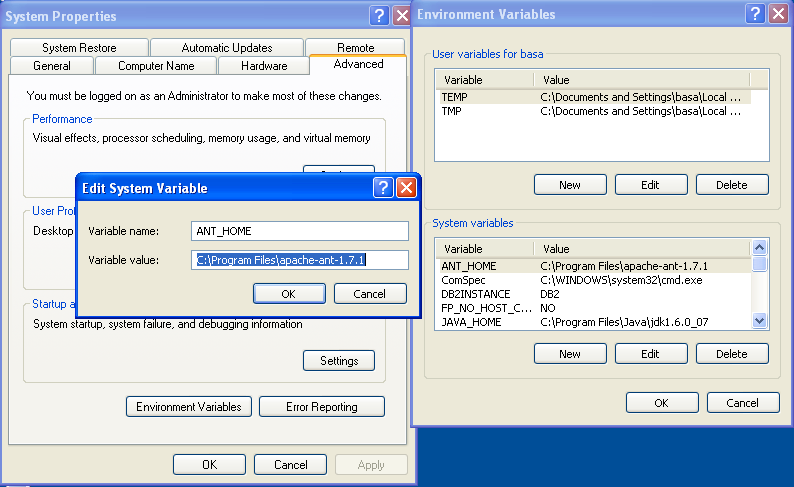


Hình ‑

* Cài đặt ANT\_HOME

Sau khi giải nén gói Apache Ant 1.7.x copy thư muc vừa giải nén C:\Program File[[6]](#footnote-7).

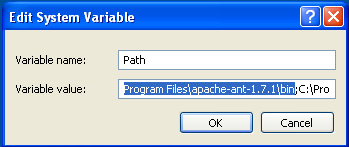
Tiến hành cài đặt biến môi trường tương tự JAVA\_HOME



Hình ‑

* Cấu hình path

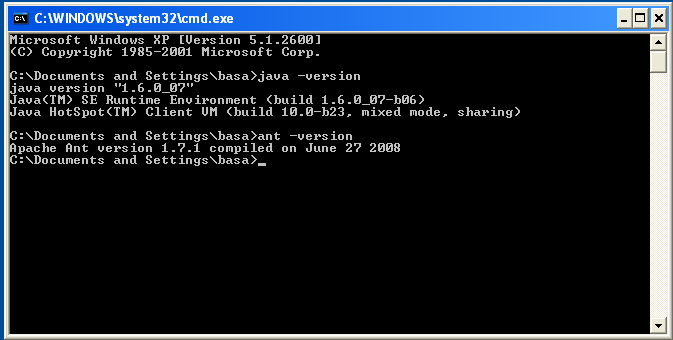
Trong system variable chon path chọn edit thêm vào đường dẫn chỉ tới thư mục **bin** của gói Apache Ant 1.7.x .Ví dụ: ;C:\Program Files\apache-ant-1.7.1\bin



Hình ‑

### Test các gói cài đặt

Vào command line gõ java –version sau đó là ant –version. Nếu cài đặt tốt sẽ được hình sau:



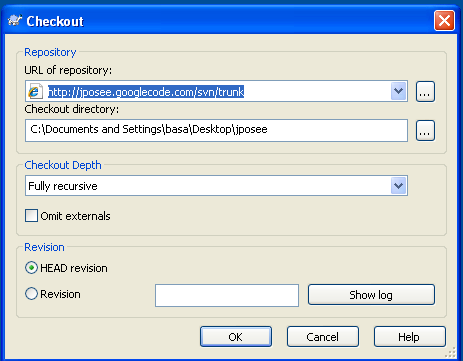
Hình 5‑11

Download các gói Jpos

Dowload Jpos 1.6.4 trực tiếp :

<http://sourceforge.net/projects/jpos/files/jpos/1.6.4/jpos-1.6.4.zip/download>

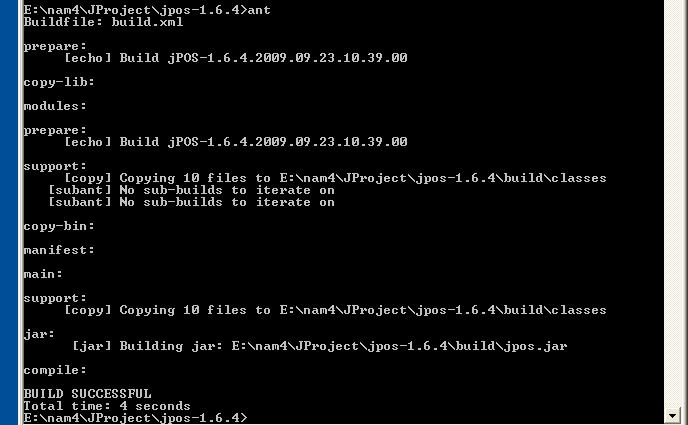
Check out Jpos ee với SVN Tortoisie : <http://jposee.googlecode.com/svn/trunk>



Hình 5‑12

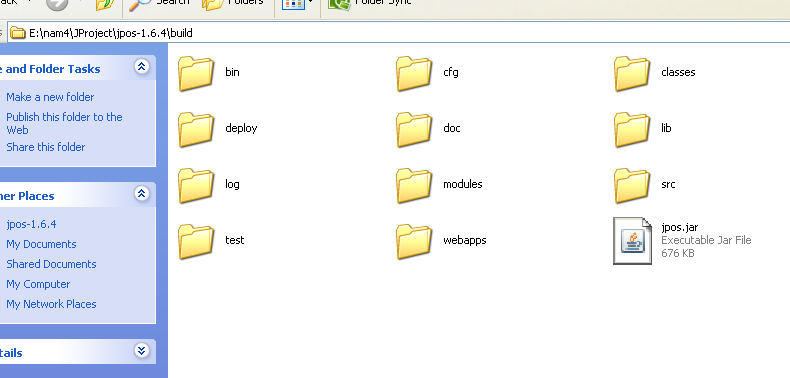
Biên dịch JPOS

Vào thư mục JPOS 1.6.4 gõ lệnh ant



Hình 5‑13

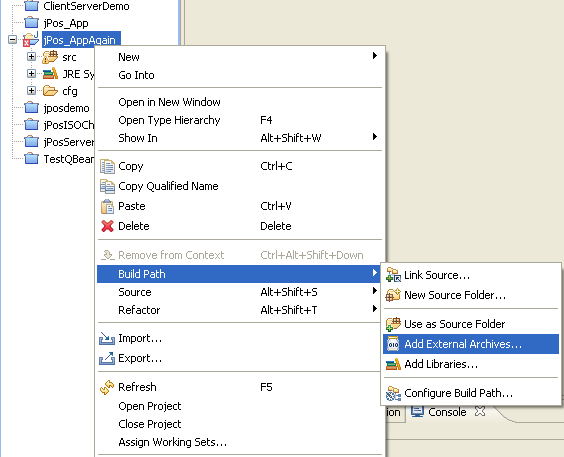
File jpos.jar sau khi biên dịch.



Hình ‑

## Sử dụng JPOS trên window.

Sử dụng Jpos với các IDE lập trình dưới dạng library.



Hình 5‑15

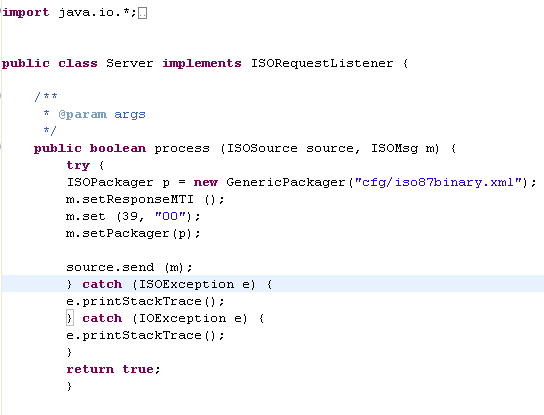
Link tới file jpos.jar vừa biên dịch.

Thực hiện test cơ bản với Jpos.

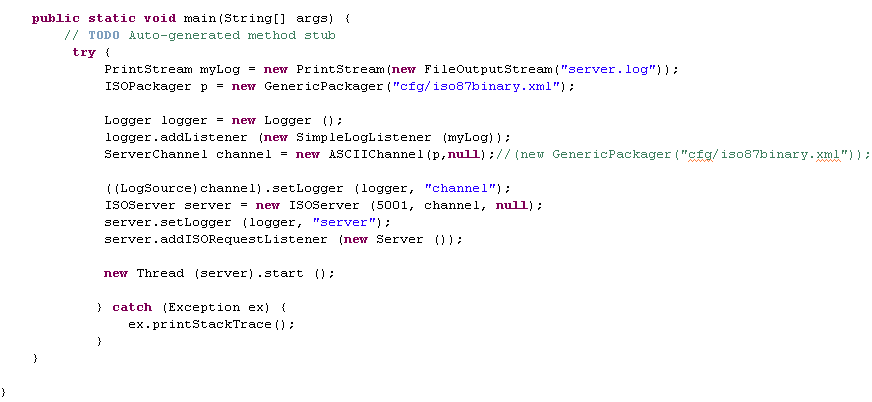
Gởi nhận gói tin ở mức cơ bản giữa client và server.

Có thể tạo 2 project hoặc đơn giản là tạo một project cho cả client và server.

Tạo file server.java có nội dung như sau:



Hình ‑



Hình 5‑17

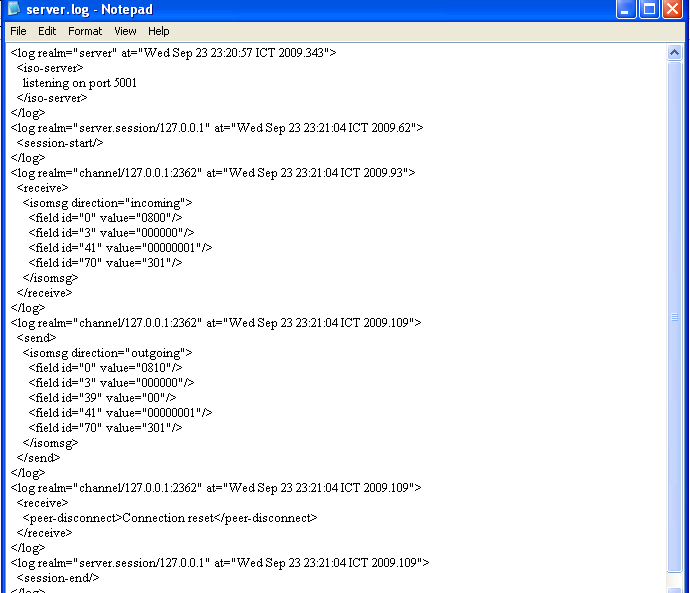
Và file client.java có nội dung như sau:



Hình 5‑18

Mục đích của việc test : client sẽ gởi một Message chứa các trường 3,41,70 và server sau khi nhận được thông điệp sẽ thêm trường số 39 vào và trả lại cho client.

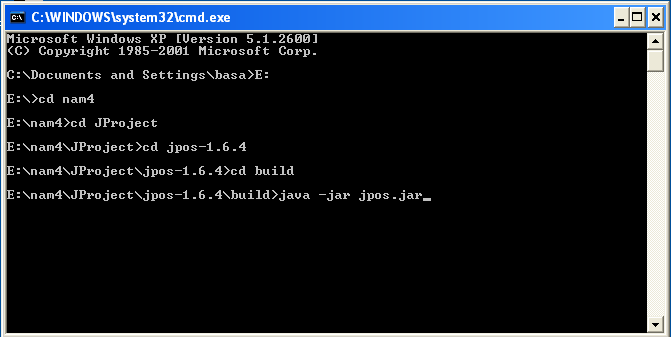
Chạy file server.java sau đó chạy file client.java nếu thành công chúng ta sẽ nhận được kết quả là file server.log sau :



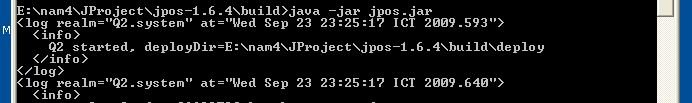
Hình ‑

## Test Q2

Đầu tiên chúng ta sẽ vào thư mục build của gói Jpos 1.6.4 chạy lệnh **java –jar jpos.jar** nếu thành công thì chúng ta sẽ thấy thông báo Q2 started như hình.



Hình ‑



Hình 5‑21

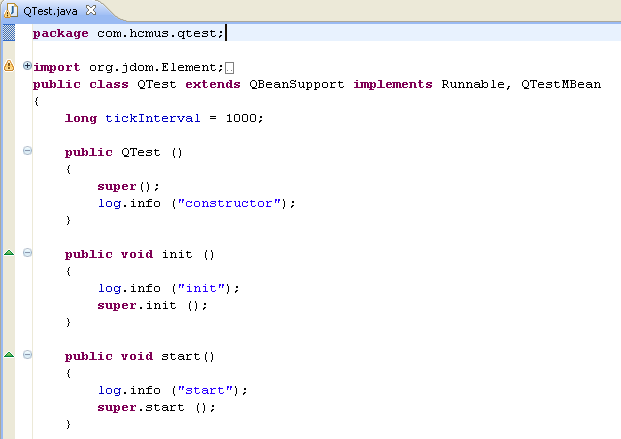
Bây giờ chúng ta sẽ viết một ứng dụng nhỏ để test Q2.

Bây giờ chúng ta sẽ giả lập để cứ sau mỗi 5s thì Q2 sẽ xuất ra màn hình console một con số đếm tăng dần.

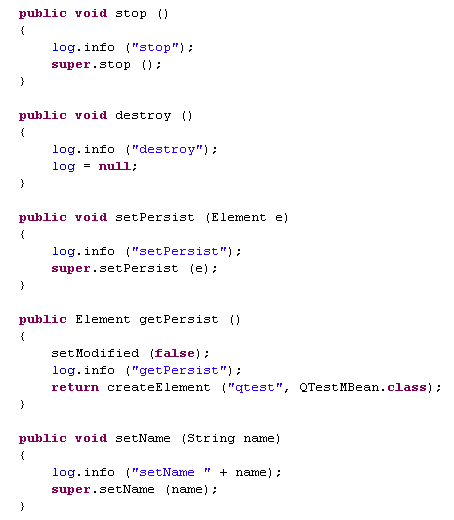
Tạo project TestMBean chẳng hạn.

Tạo gói com.hcmus.qtest

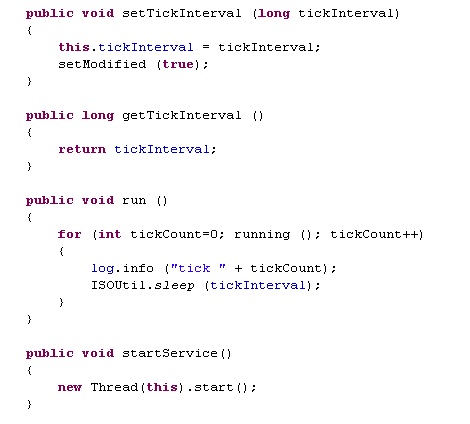
Tạo file QTest.java như sau:



Hình ‑

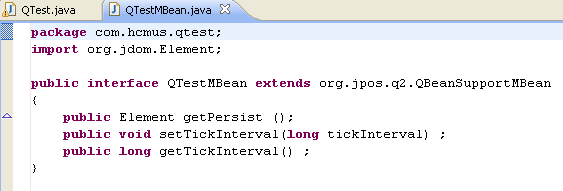


Hình ‑



Hình 5‑24

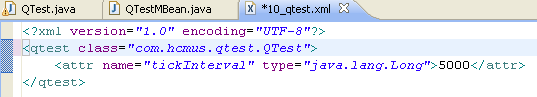
Và lớp QTestMBean.java có nội dung như sau:



Hình 5‑25

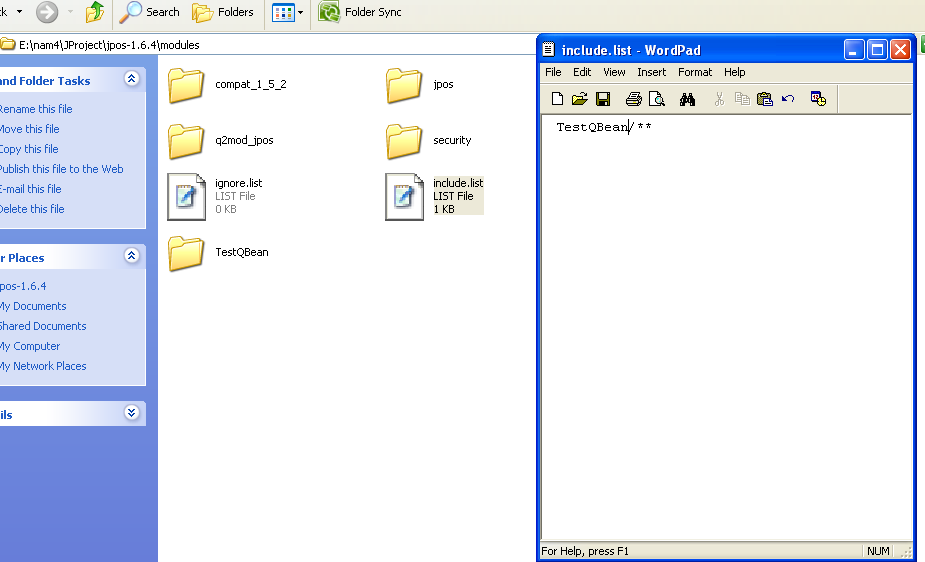
Tạo thư mục **deploy** ngang hàng với thư mục **src**

Tạo file 10\_qtest.xml trong thư mục deploy có nội dung như sau:



Hình 5‑26

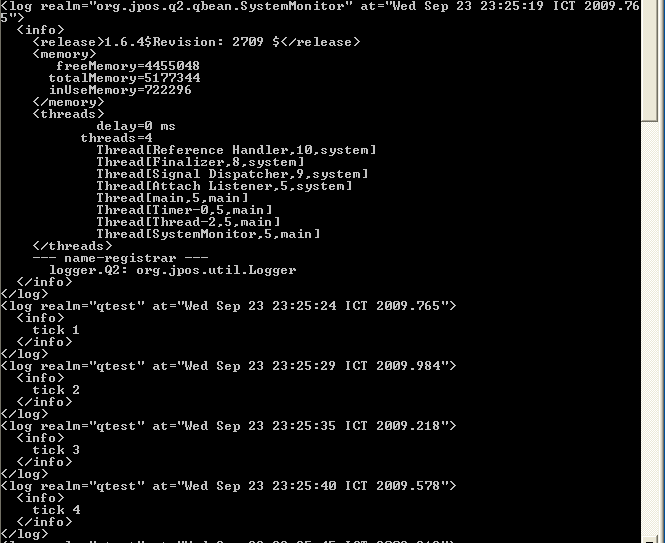
Copy project QTestMBean vừa tạo vào thư mục module trong gói jpos1.6.4 .Mở file **inlude.list** thêm vào dòng **TestQBean/\*\***



Hình 5‑27

Trở lại thư mục chứa gói jpos gõ lệnh ant để build lại file build.xml.

Sau khi build xong chúng ta sẽ chạy lại Q2 vào thư mục chứa file jpos.jar gõ lệnh java –jar jpos.jar. Nếu mọi chuyện đều tốt đẹp chúng ta sẽ có kết quả như hình.



Hình ‑

## Cài đặt JPOS trên LINUX

### Yêu cầu

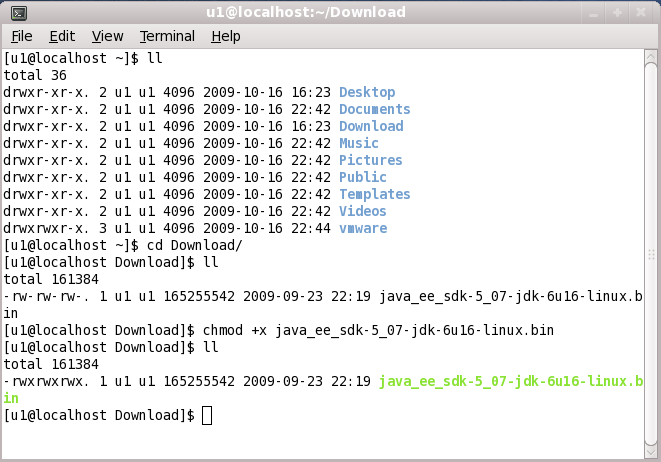
* Cài đặt Sun Java JDK 1.5 hoặc cao hơn : <http://java.sun.com/javase/downloads/index.jsp>
* Cài đặt Apache Ant 1.7.0 hoặc cao hơn : <http://ant.apache.org/>
* Cài đặt SVN client:
* OS : LINUX (phần trình bày được cài đặt trên Fedora core 11)

### Cài đặt các thành phần

* Cài đặt JDK

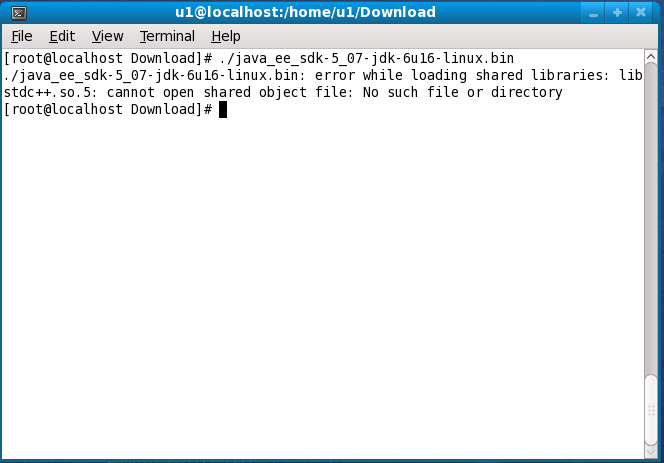
Download JDK for Linux : file java\_ee\_sdk-5\_07-jdk-6u16-linux.bin

Sau khi download về thực hiện change mode execute cho file java\_ee\_sdk-5\_07-jdk-6u16-linux.bin



Hình 5‑29

Thực hiện execute file java\_ee\_sdk-5\_07-jdk-6u16-linux.bin



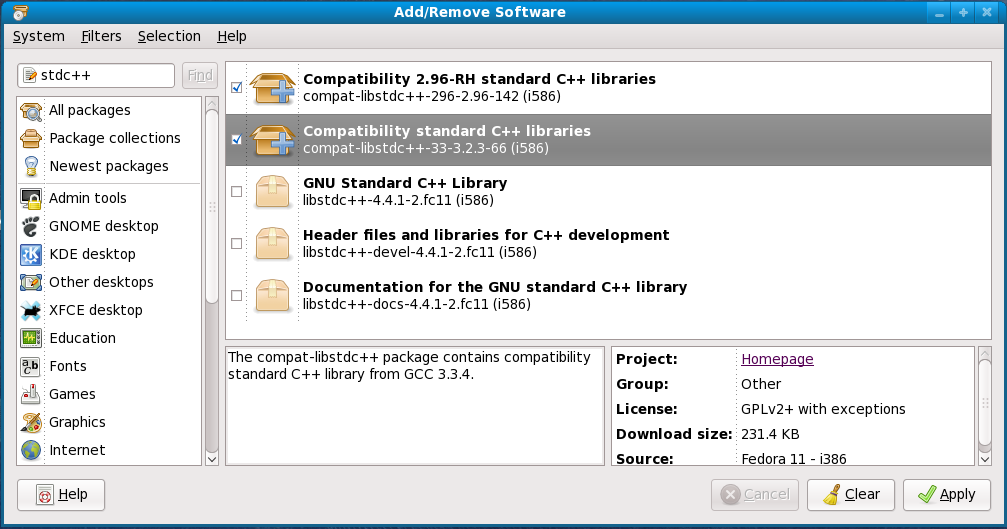
Hình 5‑30

Nếu xuất hiện lỗi này là do chưa cài đặt thư viện bổ sung

* Cài thư viện bổ sung

Bao gồm 2 gói :

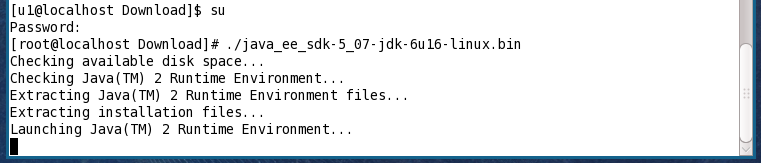
* compat-libstdc++-296-2.96-142.i586
* compat-libstdc++-33-3.2.3-66.i586



Hình 5‑31

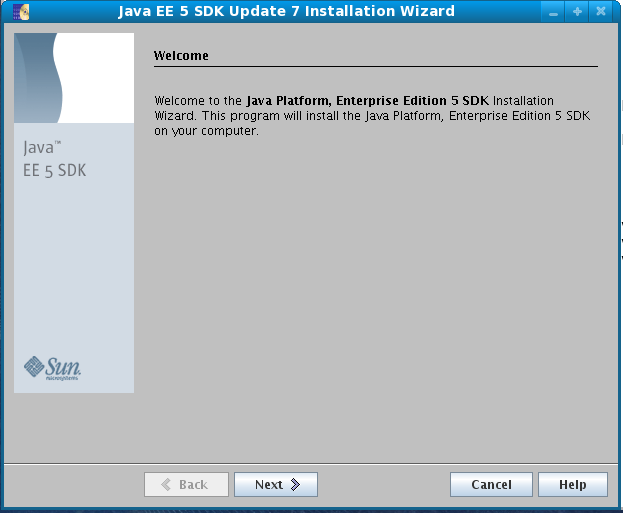
Trong quá trình cài đặt nếu có yêu cầu nhập mật khẩu của root thì cứ nhập vào

Thực thi lại việc cài đặt jdk với quyền của root



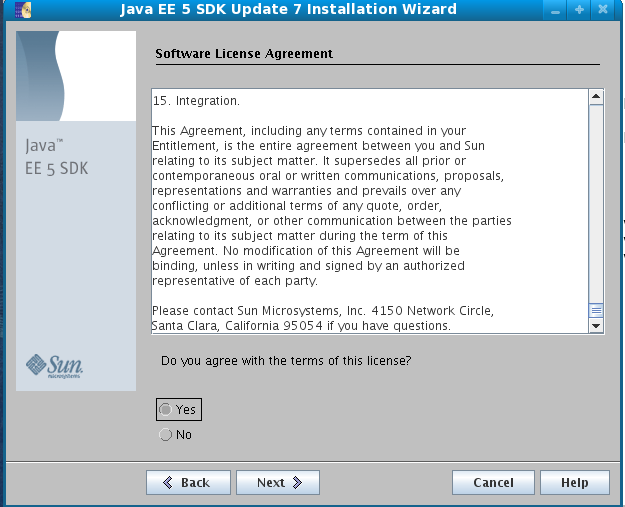
Hình 5‑32

Khi đó ta sẽ có giao diện thực thi việc cài đặt JDK tương tự như window



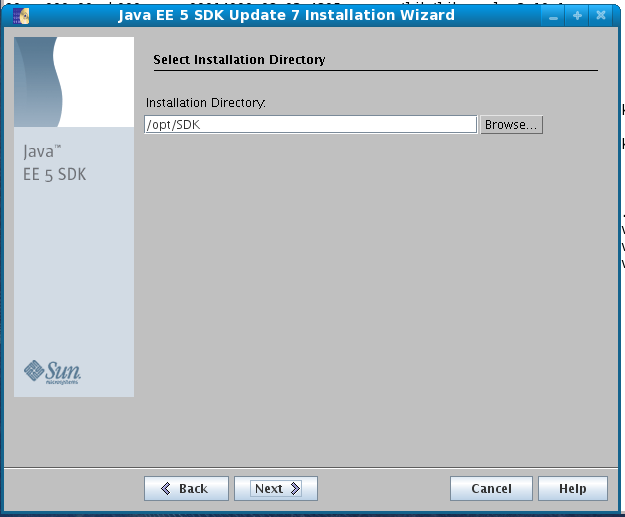
Hình 5‑33

Thực hiện việc cài đặt jdk theo hướng dẫn setup



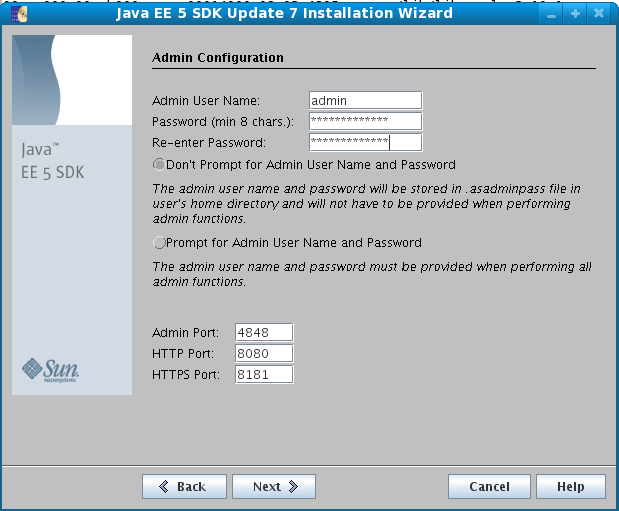
Hình 5‑34

Chọn đường dẫn lưu trữ jdk



Hình ‑

Cài đặt quyền admin



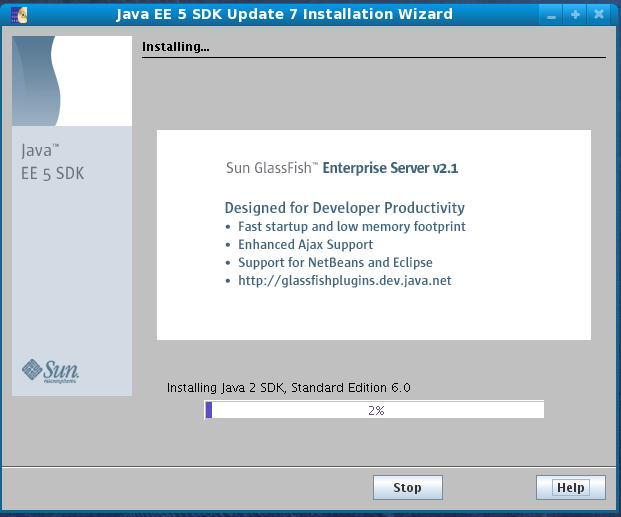
Hình ‑



Hình ‑

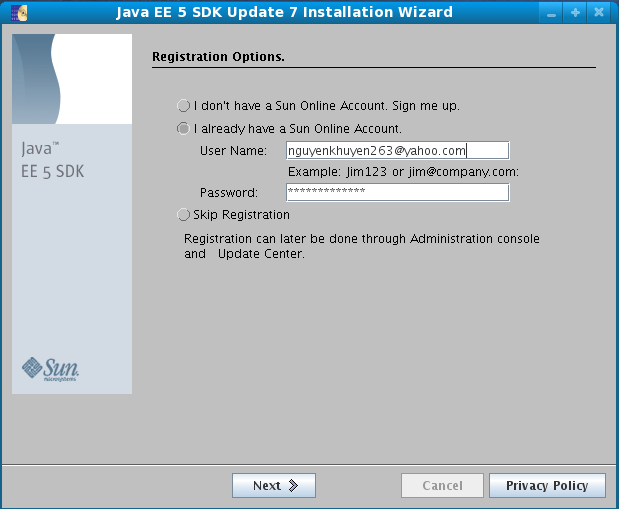


Hình ‑

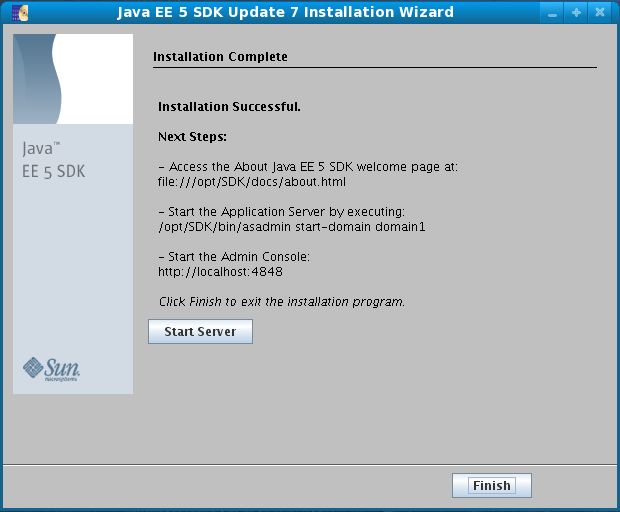


Hình 5‑39

Thực hiện đăng kí jdk với SUN (có thể bỏ qua (skip))



Hình ‑



Hình 5‑41

Kết thúc quá trình cài đặt JDK

* Cài đặt Apache Ant

Thực hiện cài đặt ant

Download gói ant về ta có được file nén sau :

apache-ant-1.7.1-bin.tar.gz

Tiến hành giải nén file này ra với câu lệnh

tar –xzvf ./apache-ant-1.7.1-bin.tar.gz

Copy thư mục apache-ant-1.7.1 vừa giải nén vào thư mục /user/local

cp –r ./apche-ant-1.7.1 /user/local

* Cài đặt SVN client

Cài đặt svn để download về với câu lệnh

yum install svn (đòi hỏi quyền root)

Thiết lập các biến môi trường

Tiến hành cài đặt cấu hình JAVA\_HOME và ANT\_HOME cho linux

Mở file /etc/profile với lệnh vi và sử dụng quyền của root

vi /etc/profile

copy đoạn code sau vào file /etc/profile và save lại

PATH=$PATH:/usr/local/apache-ant-1.7.1/bin

ANT\_HOME=/usr/local/apache-ant-1.7.1

JAVA\_HOME=/opt/SDK/jdk

export ANT\_HOME

export JAVA\_HOME

export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE

logout và login lại để xác lập những thay đổi trên file /etc/profile

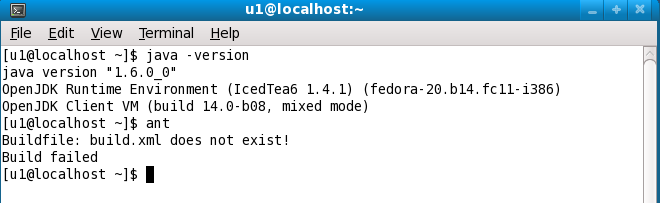
* Test các gói cài đặt

Kiểm tra xem java và ant có hoạt động hay không thông qua 2 câu lệnh

java –version

ant

nếu kết quả như sau là thành công



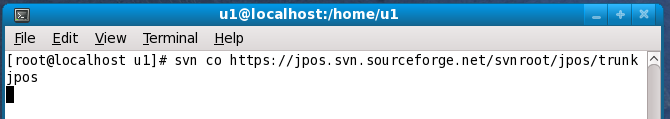
Hình ‑

* Download và biên dịch cái gói jpos

Tiến hành download jpos và jposee từ svn về từ các địa chỉ thùng chứa (trunk)

* Với jpos : <https://jpos.svn.sourceforge.net/svnroot/jpos/trunk>

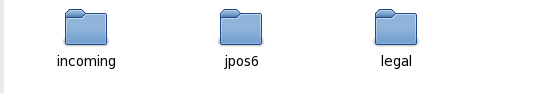
Thực hiện checkout



Hình 5‑43

svn co https://jpos.svn.sourceforge.net/svnroot/jpos/trunk jpos (download jpos về và chứa vào thư mục jpos

sau khi check out xong ta sẽ có các thư mục nằm trong thư mục jpos như sau



Hình 5‑44

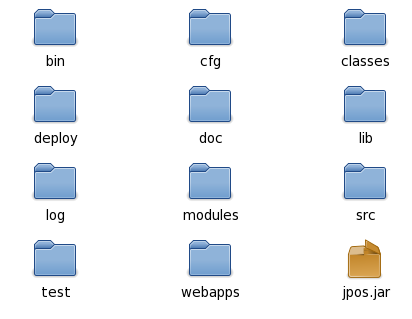
Tiến hành build jpos với ant

Vào thư mục jpos6 và gõ lệnh ant



Hình 5‑45

Sau khi build xong ta sẽ được thư mục build như sau



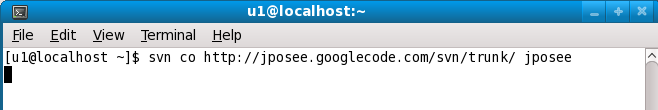
Hình 5‑46

Bây giờ ta có thể sử dụng file jpos.jar làm library hay các mục đích khác trong việc xây dựng jpos

* Với jposee : <http://jposee.googlecode.com/svn/trunk/>

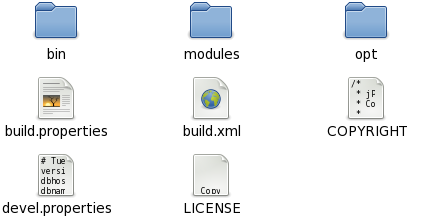
Thực hiện check out với lệnh :

**svn co http://jposee.googlecode.com/svn/trunk/ jposee** (download jposee về và chứa vào thư mục jposee)



Hình 5‑47

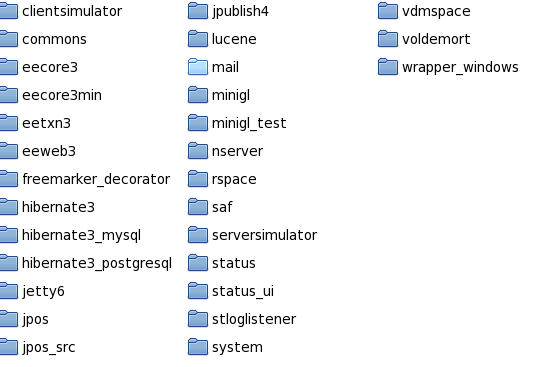
Kết quả sau khi checkout



Hình ‑

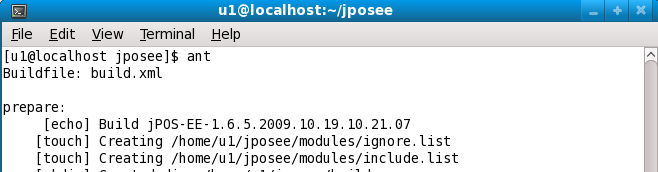
Tiến hành build jpossee với ant

Copy các thư mục trong thư mục opt vào thư mục modules



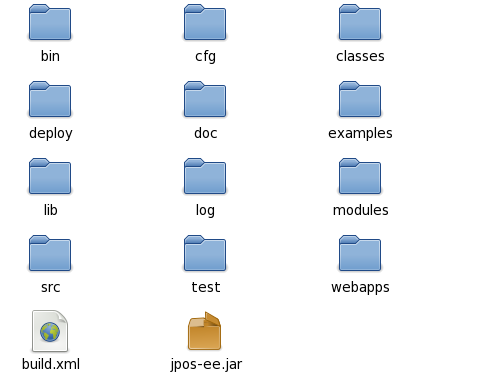
Hình 5‑49

Thực thi lệnh ant tại thư mục jpossee



Hình 5‑50

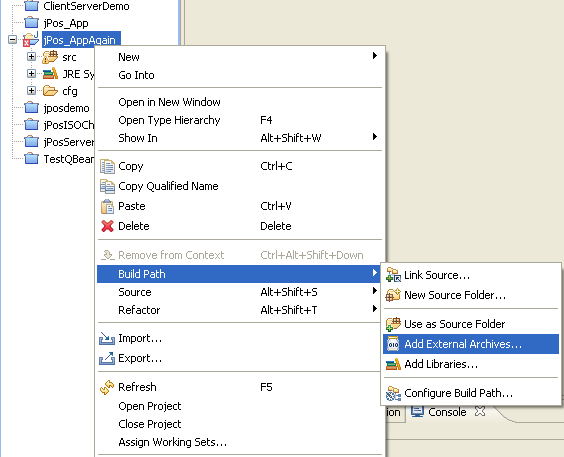
Kết quả ta sẽ có thư mục build với nội dung như sau



Hình ‑

## Sử dụng JPOS trên LINUX

Sử dụng Jpos với các IDE lập trình dưới dạng library.

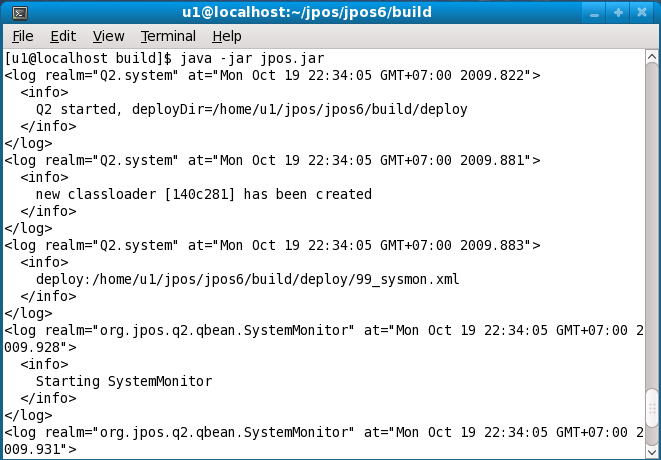


Hình 5‑52

Link tới file jpos.jar vừa biên dịch.

* Thực hiện test jpos

Thực hiện test thử gói jpos.jar



Nếu ra được kết quả như trên tức chúng ta đã cài đặt thành công tất cả, các thành phần khác khi thực hiện tương tự như trên window, các dòng lệnh không thay đổi.

Việc test thử các thành phần khác trong tương tự như trên window chỉ khác thao tác thực hiện là trên hệ điều hành LINUX.

## QSP & Q2

-Các ứng dụng của bạn cần một số thành phần được cấu hình trong lúc khởi động, một chuỗi các hành động đa kết nối cũng như kiểm định.

-Lấy điển hình hàm main của ứng dụng JPOS nhận nhiệm vụ đọc các file cấu hình và rồi tạo thể hiện của các thành phần chạy một vài Thread mới hay khởi tạo kết nối với database.v.v

-Vậy thì QSP là một cách chuẩn cho các nhiệm vụ đã được đề cập ở trên.Nó đọc tất cả các file cấu hình và trả về thể hiện của các đối tượng cho bạn sử dụng.

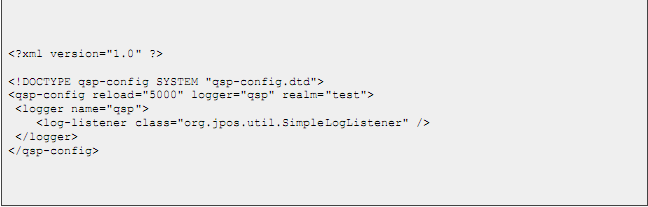
Từ phiên bản 1.4.6 QSP của Jpos đã được nâng cấp với một tên gọi đầy mới mẽ Q2.(QSP version 2). QSP vẫn được hỗ trợ một cách đầy đủ nhưng theo kinh nghiệm thì chúng ta nên sử dụng Q2 cho ứng dụng của mình bởi vì nó sẽ làm cho cuộc đời của bạn dễ dàng hơn và tốt đẹp hơn rất nhiều.Trong các trường hợp khác nhau về nguyên tắc QSP và Q2 có rất nhiều điểm tương đồng do đó để hiểu được Q2 chúng ta nên tìm hiều về QSP trước.

### QSP

Chúng ta sẽ không đi sâu vào mục đích cài đặt QSP. Trong phần này chúng ta cần nắm kỹ một số thành phần(component) được cung cấp trong Jpos sẽ được cấu hình trong file xml vì vậy yếu tố khiến chúng ta quan tâm đó là các thành phần đó sẽ được cấu hình như thế nào và chúng ta sẽ sử dụng cấu hình đó trong ứng dụng của mình như thế nào.

Bây giờ để có thể có ý niệm đầu tiên về cách thức mà QSP cấu hình chúng ta thực hiện một bài test sau:

Tao file test.xml (file này được đặt cùng thư mục với jpos.jar) có nội dung như sau:

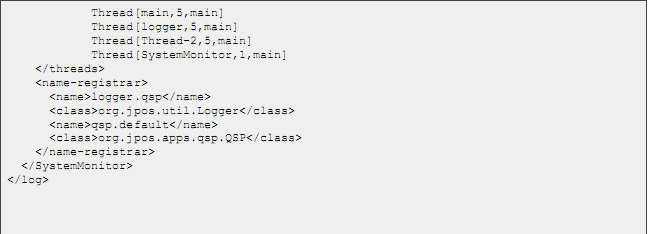
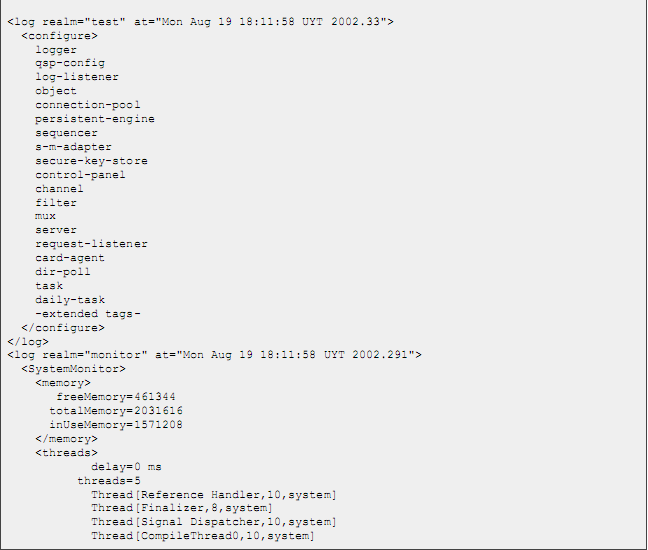


Hình 5‑53

Đến đây chúng ta chưa quan tâm vội đến nội dung của file test.xml này.Vào command line gõ lệnh sau:

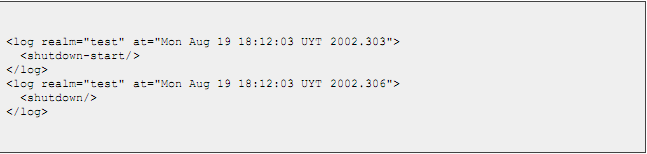
Java –jar jpos.jar test.xml

Bạn sẽ nhận được một thông điệp như sau:



Hình 5‑54

Bây giờ bạn di chuyển file test.xml đến một nơi khác trong lúc vẫn để cửa sỗ command- line hoạt động (QSP vẫn đang chạy) . Chúng ta sẽ nhận được thông điệp sau.



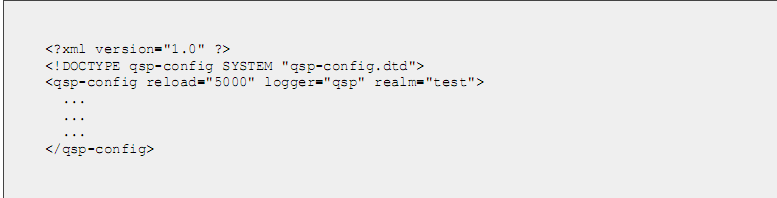
Hình 5‑55

Lập tức Q2 tắt .Bây giờ tôi nghĩ bạn đã phần nào suy tưởng ra được cách thức cấu hình cho một component bất kì của Jpos(không chỉ là QSP).

Việc cấu hình động (runtime configuration) giúp cho hệ thống hoạt động liên tục mà không cần phải khởi động lại hệ thống khi có một sự câp nhật xảy ra. Điều này quả là tuyệt vời.

### Cấu hình.

#### Cấu hình QSP



Hình 5‑56

-Thuộc tính reload chỉ ra thời gian mà QSP kiểm tra file cấu hình cho bất kì thay đổi nào.

-logger và realm là 2 thuộc tính tùy chọn đặt cung cấp việc đặt logger cho QSP.

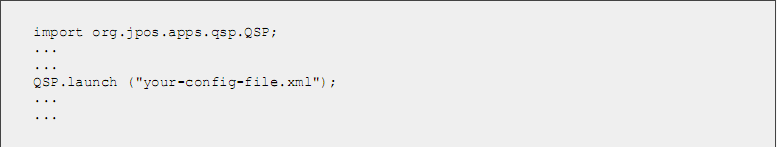
#### Object.

Có ít nhất 2 cách để sử dụng QSP:

+Biểu diễn QSP từ ứng dụng của bạn

+Biểu diễn ứng dụng của bạn bằng QSP

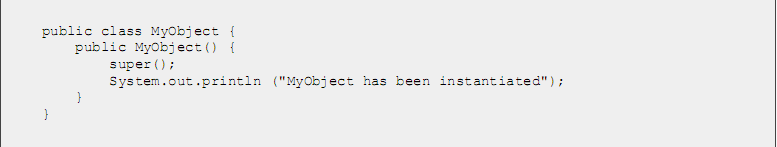
Để khởi động QSP từ ứng dụng của bạn chúng ta thực hiện như sau:



Hình 5‑57

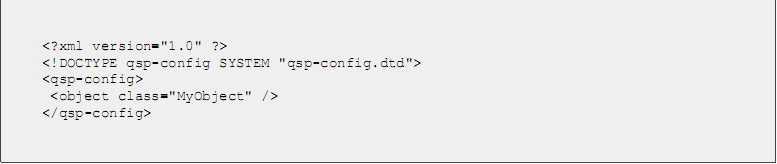
Nhưng trên thực tế thì người ta thường dùng cách thứ hai. Bạn không nên tạo quá một thể hiện của QSP trong JVM.

Để thực hiện gọi ứng dụng bằng QSP đơn giản bạn chỉ cần dùng thẻ <object> trong file cấu hình QSP. Giả sử bạn có một lớp tên là MyObject.java do bạn tự định nghĩa có nội dung như sau:



Hình 5‑58

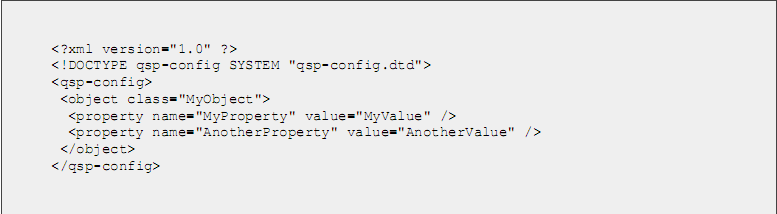
Và để sử dụng lớp này thông qua QSP chúng ta sử dụng thuộc tính class của thẻ <object> chỉ tới tên đối tương cần sử dụng.



Hình ‑

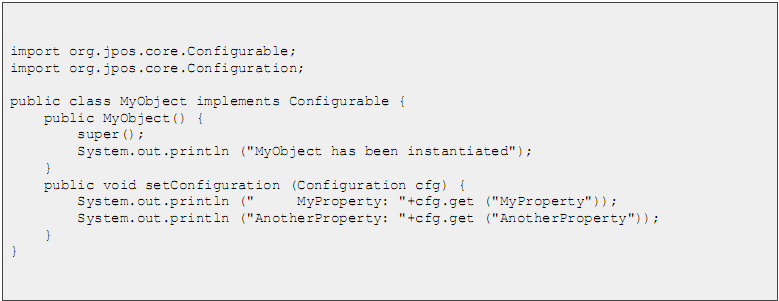
#### Configuration Object

Bạn có thêm các thẻ con <properties> bên trong thẻ <object> như sau:



Hình 5‑60

.Để đọc các giá trị của thẻ <properties> thì đơn giản bạn thừa kế lại interface Configuration và sữ dụng đối tượng Configuration của hàm setConfiguration().



Hình 5‑61

Và khi biên dịch lại file MyObject.java sau đó chạy lại file test.xml chúng ta sẽ được kết quả sau:



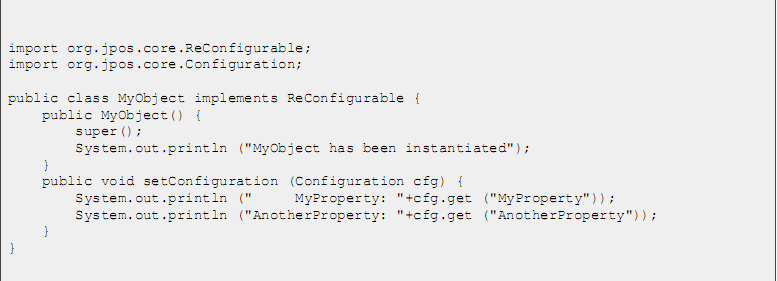
Hình ‑

#### Reconfiuration Object

Đối với các ứng dụng 24/7/365 việc khởi động lại hệ thống nếu có thay đổi xảy ra sẽ làm gián đoạn hệ thống.

Bây giờ chúng ta xem Jpos khắc phục điều này như thế nào.

Trong lớp MyObject.Java thay vì thừa kế từ Interface Configuration chúng ta sẽ thừa kế từ Interface Reconfiguration như sau:



Hình 5‑63

Chỉnh sửa lại file test.xml với thuộc tính reload.



Hình 5‑64

Bây giờ chúng ta sẽ biên dịch lại file MyObject.java

Và chạy file test.xml chúng ta sẽ thu được kết quả tương tự như phần Configuration ở trên.

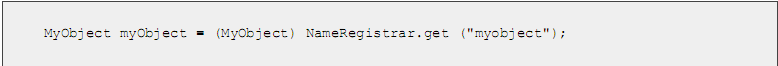
Bây giờ trong lúc Command line đang chạy thử thay đổi giá trị của <properties> từ MyValue sang MyNewValue. Và bạn sẽ thấy điều mới mẻ sau.



Hình ‑

#### Truy xuất đối tượng

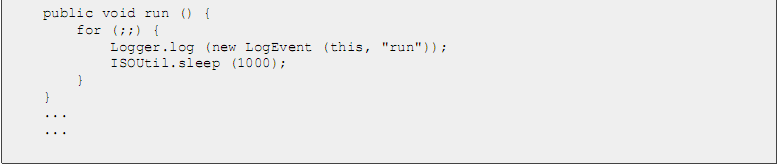
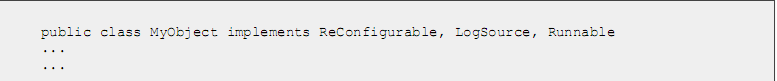
Nếu bạn sử dụng thuộc tính name cho thẻ <object> thì QSP sẽ sử dụng NameRegistrar đăng kí <object> của bạn. Và để truy xuất đối tượng đó đơn giản như sau.



Hình ‑

#### Runable Object

Đơn giản là bạn thực thi interface Runnable

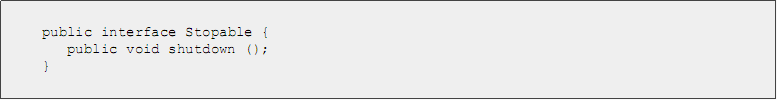


Hình 5‑67

Nếu QSP nhận ra rằng đối tượng của bạn thừa kế từ lớp **Runnable** thì đơn giản nó sẽ chạy đối tượng trong một **Thread mới**

Shutting Down

Nếu bạn muốn sử lý một số việc gì đó khi hệ thống tắt thì bạn có thể thực thi Interface Stopable sau.



Hình ‑

Cụ thể

Public class MyObject implements Stopable{

…

Public void shutdown(){

System.out.println(“Good Bye”);

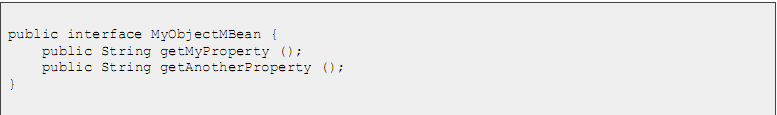
}

}

#### JMX support

Nếu bạn muốn kiểm soát hoặc thể hiện <object> của bạn thông qua JMX[[7]](#footnote-8) , đơn giản là tạo ra một đối tượng theo chuẩn MBean và QSP sẽ đăng kí nó với MBean server cho bạn.

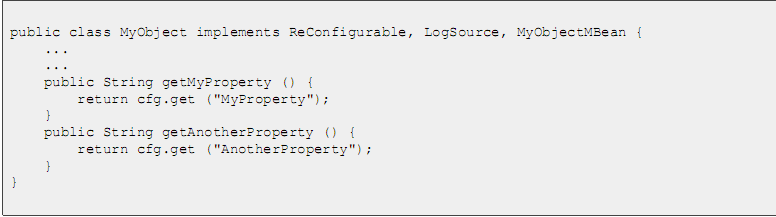
Cụ thể. Đầu tiên bạn tạo một interface theo chuẩn MBean.



Hình 5‑69

Và sau đó là thực thi đối tượng này.

Sau đó thực thi Interface bạn mới định nghĩa.



Hình 5‑70

Tuy Reconfiguration hỗ trợ bạn có thể chỉnh sửa file cấu hinh tuy nhiên việc gỡ bỏ và thêm mới file cấu hình phải khởi động lại QSP. Điều này có ảnh hưởng rất lớn đến các hệ thống 24/7.

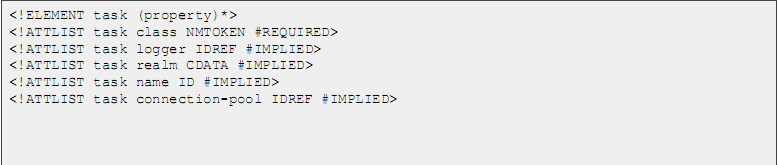
#### <Task>

<task> được kiểm soát gần giống với <object> với một số khác biệt nhỏ sau:

<task> là một loại component được khởi tạo bởi QSP.

<task> được đăng kí với NameRegistar với tiền tố “task.”

Xem xét file dtd mô tả cho thẻ <task> :



Hình 5‑71

Cũng giống như <object>, <task> có thể thực thi :

* LogSource
* [Re]Configurable
* Runable
* Stopable
* \*MBean
* <channel>

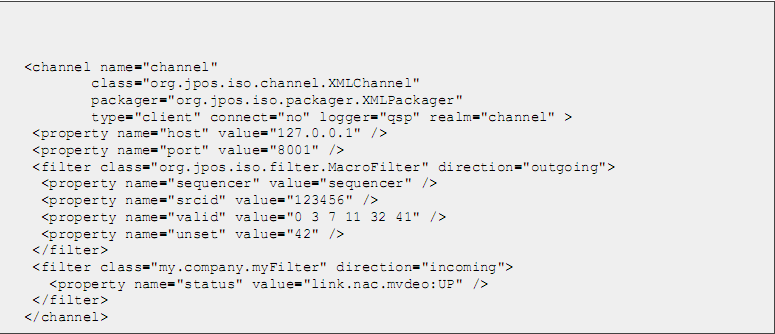
#### Channel

Channel được sử dụng ở hầu hết các ứng dụng của Jpos , QSP được thiết kế để cấu hình channel có thể tương tác được với các component khác như server,mux,



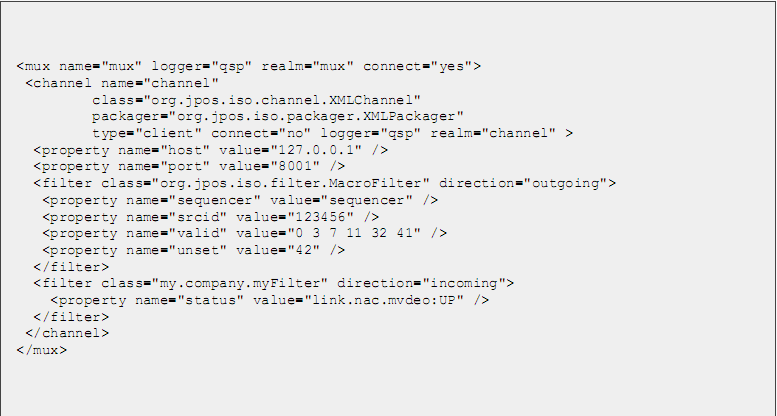
Hình 5‑72

Và thật đơn giản để thêm vào một bộ filter cho channel. Filter được thêm vào bằng thẻ con <filter>.



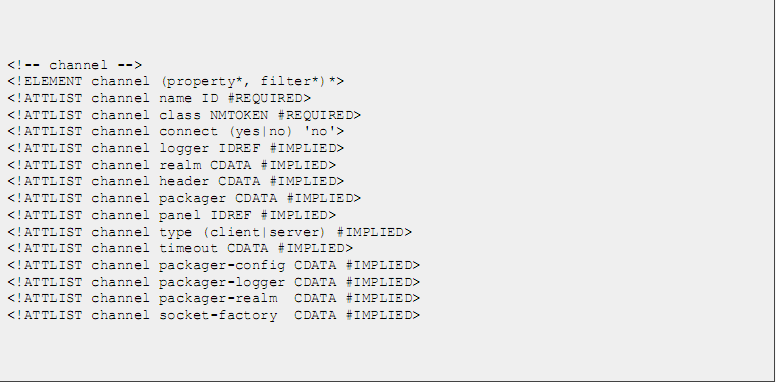
Hình 5‑73

Tương tự thật dễ dàng để cấu hình mux sử dụng channel này.



Hình ‑

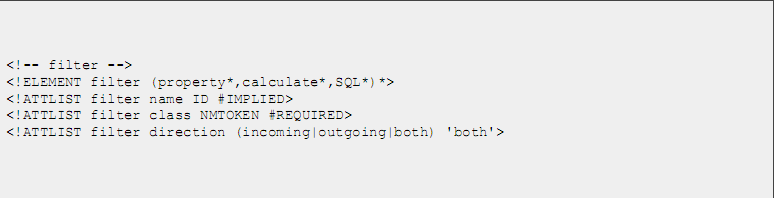
Xem xét file dtd mô tả cho channel :



Hình ‑

#### <filter>

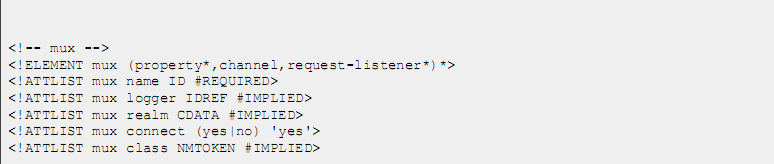
<filter> được QSP dùng để cấu hình cho ISOFilter. Xem xét file DTD của <filter>



Hình ‑

#### <mux>

<mux> của QSP dùng để cấu hình cho ISOMUX.



Hình 5‑77

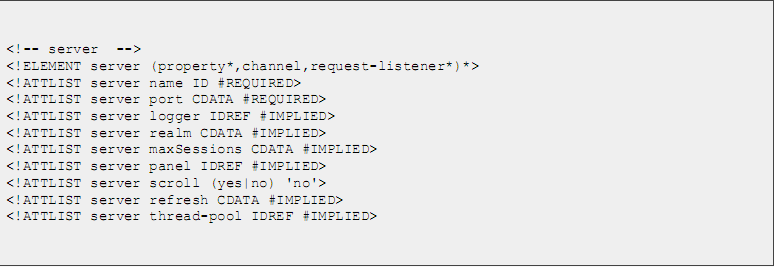
Cấu hình cần một thuộc tính di nhất name để đăng kí với NameRegistar. Một cặp logger/reaml để đặt logger cho <mux> .Thuộc tính connect dùng để cắt kết nối với <mux> bằng cách thiết lập conncet=”no” .Thuộc tính class được dùng để sử dụng các lớp ISOMUX do người dùng định nghĩa.(mặc định là org.jpos.iso.ISOMUX).



Hình ‑

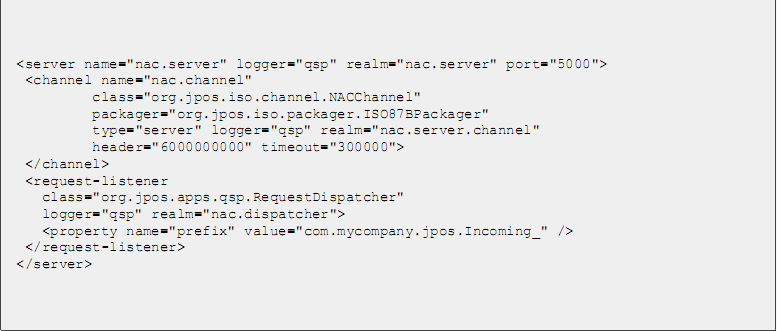
#### <server>

<server> dùng để cấu hình cho đối tượng ISOServer



Hình 5‑79

Nó đòi hỏi di nhất một thuộc tính *name* để đăng kí với NameRegistar.Thuộc tính *port* chỉ ra chỉ số *port* mà ISOServer sẽ lắng nghe.



Hình 5‑80

Bạn có thể tham chiếu đến đối tượng ISOSERVER bằng hàm static ISOSERVER.getServer(string name).

Đối tượng *maxSessions* quy định số *session* tối đa có thể kiểm soát động thời bởi thể hiện của <server> này.

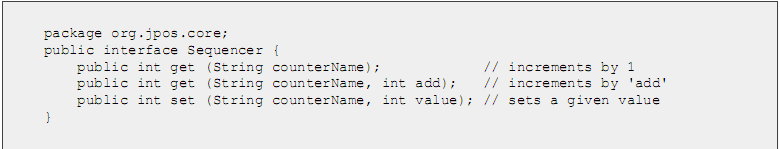
Chú ý: ISOSERVER sử dụng một ThreadPoll để kiểm soát các kết nối một cách đồng bộ.

Mội tiểu trình trong TheadPool sẽ kiểm soát các thông điệp đến sau đó chuyển chúng tới các ISORequestListener theo cách đồng bộ. Trách nhiệm của ISORequestListener là dùng một ThreadPool khác để kiểm soát một cách đồng thời. hay các giao dịch dài hơi.

#### <sequencer>

Các ứng dụng thường yêu cầu một chuỗi các số tự tăng chẳng hạn để định nghĩa cho thông điệp.(trace number hoặc retrival number)

Sau đây là intreface của Sequencer:



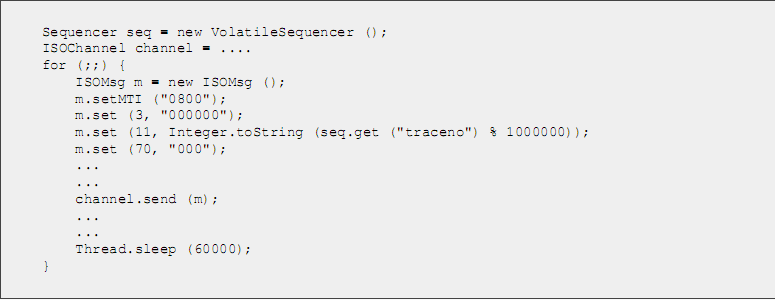
Hình 5‑81

Có 2 loại Sequencer:

VolatileSequencer : Mất đi khi tắt JVM do được lưu trên bộ nhớ.

ReliableSequenser : được lưu trên ổ cứng.

Ví dụ sử dụng:



Hình ‑

### Q2

Việc tích hợp tất cả các file cấu hình trong một file làm cho việc cấu hình trở nên cồng kềnh.Vì thế Jpos đã nghĩ đến chuyện phải tách nó ra.

Một số component hỗ trợ Reconfiguration nhưng cũng có một số không hỗ trợ kết quả là khi có một sự thay đổi trong hệ thống thì phải khở động lại QSP.

Khi chúng ta ngắt file cấu hình hoặc thêm vào một user library mới chúng ta phải khởi động lại hệ thống.

Vì vậy Jpos quyết định một file cấu hình trên một component và một vòng lặp đơn giản để thực thi các component đó.

Biên dịch Q2 [[8]](#footnote-9)

Khi chúng ta khởi chạy Q2 thì nó sẽ kiểm tra thư mục delog để phát hiện file cấu hình cho các component cũng như thư mục deloy/lib cho các file jars mới.

Bây giờ chúng ta tìm hiểu cách mà Qbean hoạt động.

Qbean đại diện bằng đối tượng QbeanSupport. QBeanSupport thực chất là một service và thực sự mang “bản chất” của một service.QBeanSupport cung cấp cho bạn 4 hàm và 4 hàm này có thể override lại ở các lớp con thừa kế.

initService()

startService()

stopService()

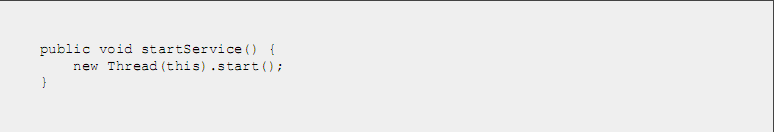
destroyService()

Những phương thức này cung cấp các kiểm soát tới file cấu hình xml (đặt trong thư mục deloy)

Thêm vào đó initService(),startService(),stopService(),destroyService() có thể bắt và log lại tất cả các ngoại lệ trong quá trình thực thi.

Bằng cách mở rộng QBeanSupport bạn không cần phải thực thi tất cả các hàm trong vòng đời của QBean.

Nếu bạn muốn khởi tạo lớp QBean là một đối tượng Runnable thì đơn giản bạn cho đối tượng của bạn chạy trên một Thread mới.



Hình 5‑83

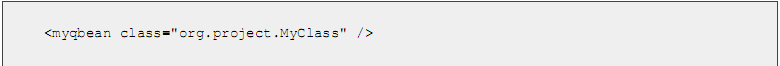
JMX yêu cầu mở rộng các interface \*MBean để các thuộc tính được quản lý bởi JMX và thông qua MBeanServer.

Các phương thức setPersist(Element) và Element getPersist() được cung cấp bởi QbeanSupport để QBean có thể nhận về file cấu hình xml mà nó sở hữu từ đó có thể duy trùy kết nối cũng như nhận biết các sự thay đổi trên file cấu hình đó.

#### QBean description

File xml của Qbean cực kì linh động.

Một cấu hình đơn giản nhất của qbean với tên thành phần và thuộc tính class chỉ tới lớp thực hiện.



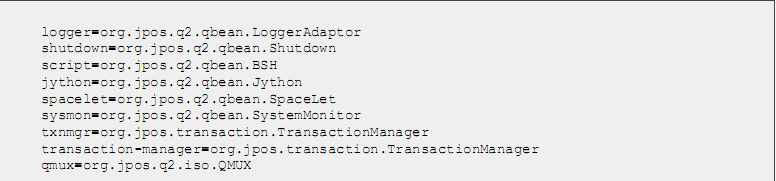
Hình 5‑84

Lớp ***org.project.MyClass*** của bạn được giả định rằng thực thi interface Qbean bằng cách dễ dàng thực hiện bằng cách mở rộng lớp ***org.jpos.q2.QBeanSupport***.

Q2 sẽ cố gắng khởi tạo thể hiện lớp của bạn và đăng ký chúng với MbeanServer thông qua JMX dưới tên “Q2: type=qbean name=mybean”

Điều quan trọng là nếu lớp của bạn có phuơng thức void setName(string name) thì Q2 sẽ đăng kí lớp của bạn với NameRegistrar

Nếu bạn không chỉ ra thuộc tính class, Q2 sẽ cố gắng lấy những lớp thích hợp trong file QFactory.properties tương thích với tên thẻ (ví dụ là mybean như trên).



Hình 5‑85

Vây để tắt Qbean thì đơn giản bạn chỉ cần thiết kế file cấu hình shutdow.xml như sau.



Hình 5‑86

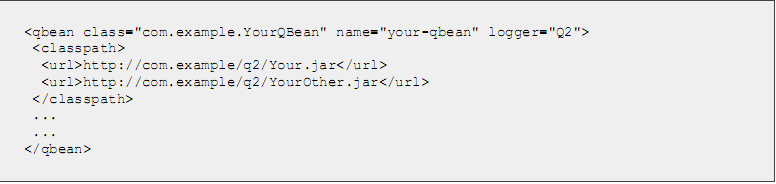
Điều này tương đương với <shutdown class=”org.jpos.q2.qbean.Shutdown />

Một điểm rất hay của Jpos là file cấu hình xml có thể có bất kỳ định dạng nào bằng cách thực thi org.jpos.core.XmlConfiguaration interface. (Q2 sẽ đưa JDom Element tới QBean để nó có thể truy xuất bất kì thành phần nào của file cấu hình xml)

#### Q2 Dynamic class loading

Nếu bạn phải thêm bất kì chức năng mới nào vào hệ thống đang chạy thì bạn hãy biên dịch nó trở thành file .jar và để ở thư mục deloy/lib và sau đó đặt cấu hình xml ở deloy nhầm giúp Q2 khởi động QBean của bạn.

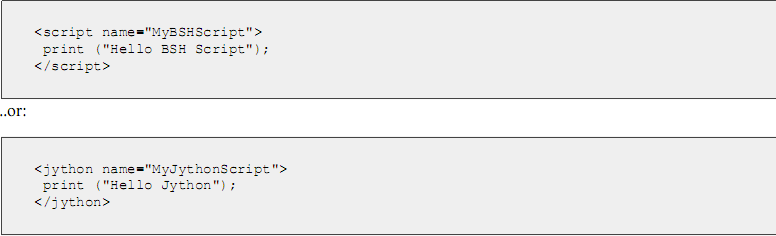
Ngoài ra không chỉ nạp các fle cố định có trên máy tính của bạn Q2 còn có thể nạp các file jar từ các host trên mạng.



Hình ‑

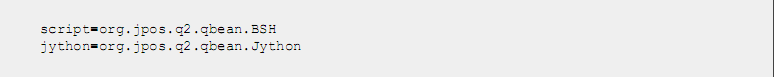
#### Q2 Script

Q2 cung cấp những cách dễ dàng để chạy BSH hay JyThon-Script



Hình 5‑88

Nếu bạn nhìn vào org.jpos.q2.Qfactory.properties chúng ta sẽ thấy rằng :



Hình 5‑89

script chỉ tới class org.jpos.q2.qbean.BSH và jython chỉ tới class org.jpos.q2.qbean.Jython

## Space

Spcae giống như là một đối tượng Map trong Java thuần túy trong đó các entry của nó là một tập các đối tượng và một điều quan trọng là các đối tượng được đưa vào hoặc lấy ra từ Space thì luôn luôn được đồng bộ hóa.

Sau đây là một số phương thức chính cùa Space :

* Void out(Object key,Object value)

Đặt một object vào trong Space nếu Object Value ứng với Object Key đã tồn tại thì Object value đó sẽ được xếp hàng tuần tự vào cuối Space.

* Object rd(Object key)

Đọc một Object từ Space với Object Key truyền vào.Block cho đến khi Object Key cần lấy được lấy.

* Object in(Object key)

Xóa một đối tượng ra khỏi Space. Nếu có nhiều đối tượng cùng khóa thì xóa đối tượng đầu tiên tình từ đầu Space. Block cho đến khi đối tượng được xóa trong Space.

* Void out(Object key,Object value,long timeout)

Đặt một đồi tượng vào trong Space trong một khoảng thời gian nhất định. Hết thời gian này. Đối tượng sẽ tự động được xóa.

* Object rd(Object key,long timeout)

Đọc một đối tượng trong Space. Nếu hết thời gian timeout vẫn chưa đọc được thì trả về null.

* Object rdp(Object key,long timeout)

Đọc một đối tượng trong Space nếu nó tồn tại.

* Object inp(Object key)

Xóa một đối tượng trong Space nếu nó tồn tại.

* Void push(Object key,Object value)

Giống với out tuy nhiên đối tượng được đặt ở đầu của danh sách.(GIống với Stack)

## Đặc tả Usercase

### Đặc tả usercase Subtract Point

#### Tóm tắt:

Usercase này nhầm trừ điểm vào thẻ của khách hàng khi khách hàng trả lại hàng ứng với số tiền mà họ đã mua.

#### Dòng dự kiện

##### Dòng sự kiện chính :

Sự kiện này bắt đầu khi người bán hàng dùng máy EDC quét mã thẻ của khách hàng đồng thời mã tác vụ thực hiện là trừ điểm.

* Hệ thống yêu cầu máy EDC cung cấp thông tin về : Mã thẻ, mã tác vụ,số tiền của khách hàng đã trả.
* Hệ thống tiến hành kiểm chứng các giá trị được chuyển lên.
* Hệ thống trừ điểm và tiến hành lưu lại vào log trên database.
* Thông báo giao dịch thành công và trả thông tin quảng cáo cho người dùng.

##### Các dòng sự kiện khác

* Không gởi đủ các trường của thông điệp. Khi EDC chuyển lên không đủ các trường mà hệ thống yêu cầu hệ thống sẽ trả về người dùng thông báo lỗi các trường của thông điệp bị lỗi.
* Mã thẻ đã hết hạn sử dụng : khi EDC chuyển lên mã thẻ đã hết hạn sử dụng. Hệ thống sẽ trả về cho người dùng lỗi mã thẻ đã hết hạn sử dụng.
* Mã thẻ không tồn tại trong hệ thống : EDC chuyển lên mã thẻ chưa được đăng kí. Hệ thống sẽ trả về mã lỗi thẻ không tồn tại trong hệ thống.
* Mã thẻ chưa được kích hoạt : EDC chuyển lên mã thẻ chưa được kích hoạt. Hệ thống sẽ trả về mã lỗi thẻ chưa được kích hoạt trong hệ thống.
* TID và MID chưa được đăng kí trên hệ thống : Nếu EDC chuyển lên mã TID và MID chưa được đăng kí trên hệ thống thì hệ thống sẽ báo TID và MID không tồn tại.

#### Các yếu cầu đặc biệt

Không có

#### Trạng thái hệ thống khi bắt đầu thực hiện usercase.

Hệ thống server đã được kích hoạt và lắng nghe trên port đã cấu hình.

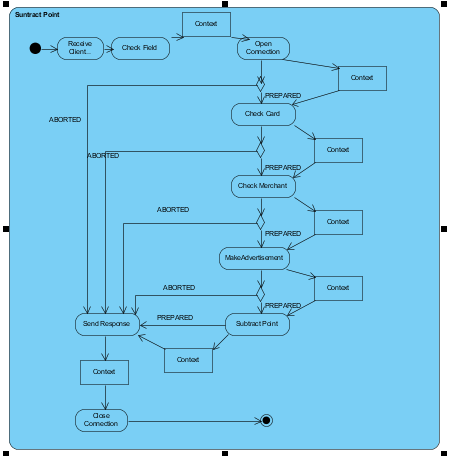
#### Trạng thái của hệ thống sau khi thực hiện usercase

Nếu usercase thành công thì giá trị điểm của thẻ khách hàng được cập nhật vào database. Ngược lại giá trị không thay đổi.

#### Điểm mở rộng :

Không có.

### Luồng Sự Kiện Cho usercase Subtract Point



Hình ‑

### Đặc tả usercase Redemption Point

#### Tóm tắt:

Usercase này nhầm đỗi điễm mà khách hàng đang có để đổi thành quà.

#### Dòng dự kiện

##### Dòng sự kiện chính :

Sự kiện này bắt đầu khi người bán hàng dùng máy EDC quét mã thẻ của khách hàng đồng thời mã tác vụ thực hiện là đổi quà.

* Hệ thống yêu cầu máy EDC cung cấp thông tin về : Mã thẻ, mã tác vụ, số điểm cần đổi.
* Hệ thống tiến hành kiểm chứng các giá trị được chuyển lên.
* Hệ thống trừ điểm của khách hàng và tiến hành lưu lại vào log trên database.
* Thông báo giao dịch thành công và trả thông tin món quà được nhận cho người dùng.

##### Các dòng sự kiện khác

* Không gởi đủ các trường của thông điệp. Khi EDC chuyển lên không đủ các trường mà hệ thống yêu cầu hệ thống sẽ trả về người dùng thông báo lỗi các trường của thông điệp bị lỗi.
* Mã thẻ đã hết hạn sử dụng : khi EDC chuyển lên mã thẻ đã hết hạn sử dụng. Hệ thống sẽ trả về cho người dùng lỗi mã thẻ đã hết hạn sử dụng.
* Mã thẻ không tồn tại trong hệ thống : EDC chuyển lên mã thẻ chưa được đăng kí. Hệ thống sẽ trả về mã lỗi thẻ không tồn tại trong hệ thống.
* Mã thẻ chưa được kích hoạt : EDC chuyển lên mã thẻ chưa được kích hoạt. Hệ thống sẽ trả về mã lỗi thẻ chưa được kích hoạt trong hệ thống.
* TID và MID chưa được đăng kí trên hệ thống : Nếu EDC chuển lên mã TID và MID chưa được đăng kí trên hệ thống thì hệ thống sẽ báo TID và MID không tồn tại.
* Số điểm của khách hàng không đủ để đổi quà. Hệ thống sẽ trả thông báo về cho khách hàng biết là không đủ điểm để đổi quà.
* Số điểm gởi lên không có trong danh mục bất cứ món quà nào trong databse. Hệ thống sẽ trả thông báo cho người dùng biết là số điểm của họ không hợp lệ.

#### Các yếu cầu đặc biệt

Không có

#### Trạng thái hệ thống khi bắt đầu thực hiện usercase.

Hệ thống server đã được kích hoạt và lắng nghe trên port đã cấu hình.

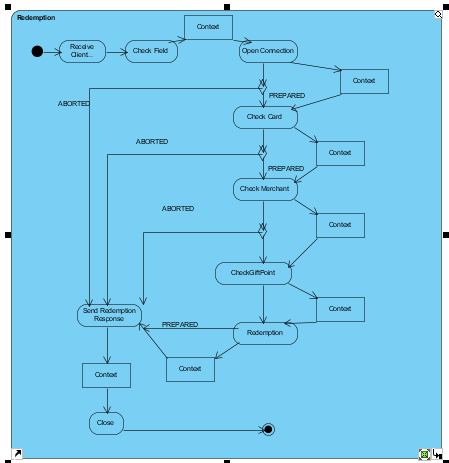
#### Trạng thái của hệ thống sau khi thực hiện usercase

Nếu usercase thành công thì giá trị điểm của thẻ khách hàng được cập nhật vào database. Ngược lại giá trị không thay đổi.

#### Điểm mở rộng :

Không có.

### Luồng Sự Kiện Cho Usercase Redemption Point



Hình ‑

### Đặc tả usercase Balance Inquiry

#### Tóm tắt:

Usercase này nhầm truy vấn xem trong thẻ có bao nhiêu điểm.

#### Dòng dự kiện

##### Dòng sự kiện chính :

Sự kiện này bắt đầu khi người bán hàng dùng máy EDC quét mã thẻ của khách hàng đồng thời mã tác vụ thực hiện là truy vấn điểm.

* Hệ thống yêu cầu máy EDC cung cấp thông tin về : Mã thẻ, mã tác vụ
* Hệ thống tiến hành kiểm chứng các giá trị được chuyển lên.
* Hệ thống truy vấn điểm của khách hàng
* Thông báo giao dịch thành công và trả thông tin điểm cho người dùng.

##### Các dòng sự kiện khác

* Không gởi đủ các trường của thông điệp. Khi EDC chuyển lên không đủ các trường mà hệ thống yêu cầu hệ thống sẽ trả về người dùng thông báo lỗi các trường của thông điệp bị lỗi.
* Mã thẻ đã hết hạn sử dụng : khi EDC chuyển lên mã thẻ đã hết hạn sử dụng. Hệ thống sẽ trả về cho người dùng lỗi mã thẻ đã hết hạn sử dụng.
* Mã thẻ không tồn tại trong hệ thống : EDC chuyển lên mã thẻ chưa được đăng kí. Hệ thống sẽ trả về mã lỗi thẻ không tồn tại trong hệ thống.
* Mã thẻ chưa được kích hoạt : EDC chuyển lên mã thẻ chưa được kích hoạt. Hệ thống sẽ trả về mã lỗi thẻ chưa được kích hoạt trong hệ thống.
* TID và MID chưa được đăng kí trên hệ thống : Nếu EDC chuển lên mã TID và MID chưa được đăng kí trên hệ thống thì hệ thống sẽ báo TID và MID không tồn tại.

#### Các yếu cầu đặc biệt

Không có

#### Trạng thái hệ thống khi bắt đầu thực hiện usercase.

Hệ thống server đã được kích hoạt và lắng nghe trên port đã cấu hình.

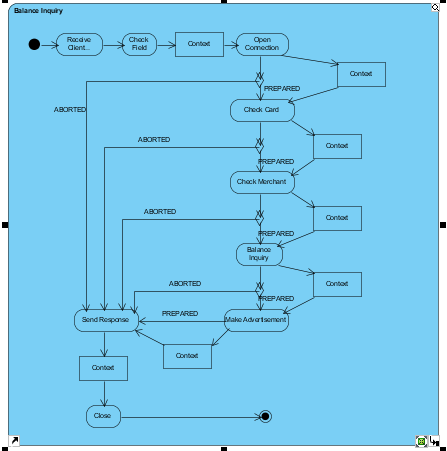
#### Trạng thái của hệ thống sau khi thực hiện usercase

Nếu usercase thành công thì khách hàng nhận được điểm ứng với mã thả của mình.

#### Điểm mở rộng :

Không có.

### Luồng sự kiện Balance Inquiry



Hình ‑

## Bảng đặc tả cho các thông điệp được thiết kế

### Subtract Point Transaction

#### Tổng Quát

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field No | Data Element Name | Attribute | Request | Respone | Note |
|  | MTI | n 4 | M | M | HEX |
|  | Bitmap | b 8 |  |  | HEX |
| 3 | Processing code | n 6 | M | M | HEX |
| 4 | Amount Transaction | n 12 | M |  | HEX |
| 35 | Track 2 Data | z..37 | M |  | HEX |
| 39 | Response Code | an 2 |  | M | ASCII |
| 41 | Card Acceptor Terminal ID | ans 8 | M | M | ASCII |
| 42 | Card Acceptor Identification Code | ans 15 | M | M | ASCII |
| 48 | Reserse Private | ans…999 |  | M | Length of field (Hex) and Content of field (ASCII) |
| 61 | Reverse Private | ans …999 |  | M | Length of field (Hex) and Content of field (ASCII). Use Tag Length value for each message |

#### Đặc tả chi tiết

##### MTI (Message Type Indicator) :

|  |  |  |
| --- | --- | --- |
| Request | Respone | Description |
| 0200 | 0210 | Subtract Point |

##### Field No : 3 Processing code

Processing code chia làm ba thành phần:

* Transaction Type code,AN2
* Account Type Code 1,AN2
* Account Type Code 2,AN2 (skip : default : 00)

**The transaction Type Code(TTC)** : Sử dụng kết hợp với MTI để chỉ ra tác vụ cần thực hiện.

**Request :**

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0200 | 41 | Subtract Point |

**Response :**

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0210 | 41 | Subtract Point |

* **Account Type Code (ATC)** : Mã chỉ định loại của tài khoản.

|  |  |
| --- | --- |
| ATC | Description |
| 70 | Loyalty Card |

##### Field 48

|  |  |  |
| --- | --- | --- |
| Tag | Length | Description |
| FF51 | 04 | Số điểm trừ |
| FF52 | 04 | Tổng điểm còn lại |

##### Message sample :

###### Message request:

**Header** : 6000030000 🡺 5 byte (hex)

**MTI** : 0200 🡺 2 byte (hex)

**Bitmap** : 3000000020C00000 🡺 8 byte (Hex)

**F3** : 417000 🡺 3 byte (Hex)

**F4** : 000000200000 🡺 6 byte (hex)

**F35** : 379604215000000046D140610114680300000000 🡺 20 byte ( hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

Total Message length : 5 + 2 + 8 + 3 + 6 + 20 + 8 + 15 = 67 🡺 2 byte (hex) 0043

0043600003000002003000000020C00000417000000000200000379604215000000046D1406101146803000000003232322020202020322020202020202020202020202020

###### Message Response :

**Header** : 6000000003 🡺 5 byte (hex)

**MTI** : 0210 🡺 2 byte (hex)

**Bitmap** : 2000000000C10008 🡺 8 byte (Hex)

**F3** : 417000 🡺 3 byte (Hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

**F48**: 0014FF510430303032FF520430303238 🡺 16 byte

**F61** :

**Message 1st:**

Tag : FF01 => 2 byte hex

Length 1st message : 20 => 0x14 => 1 byte

Message content 1st : 31302042414e204e48414e20414f205448554e20

(Value : 10 BAN NHAN AO THUN)

**Message 2nd :**

Tag : FF02 => 2 byte

Length 2nd message : 18 => hex 0x12 => 1 byte

Message 2nd content: 32302042414e204e48414e20584520444150

(Value : 20 BAN NHAN XE DAP)

Total F61 : 2 + 1 + 20 + 2 + 1 + 18 = 44 + 2 = 46 ( 2byte to send 0044)

0044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

Total Message length : 5 + 2 + 8 + 3 + 8 + 15 + 16 + 46 = 103 🡺 2 byte (hex) 0067

0067600000000302102000000000C1000841700032323220202020203220202020202020202020202020200014FF510430303032FF5204303032380044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

### Balance Inquiry Transaction

#### Tổng quát

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field No | Data Element Name | Attribute | Request | Respone | Note |
|  | MTI | n 4 | M | M | HEX |
|  | Bitmap | b 8 |  |  | HEX |
| 3 | Processing code | n 6 | M | M | HEX |
| 35 | Track 2 Data | z..37 | M |  | HEX |
| 39 | Response Code | an 2 |  | M | ASCII |
| 41 | Card Acceptor Terminal ID | ans 8 | M | M | ASCII |
| 42 | Card Acceptor Identification Code | ans 15 | M | M | ASCII |
| 48 | Reserse Private | ans…999 |  | M | Length of field (Hex) and Content of field (ASCII) |
| 61 | Reverse Private | Ans …999 |  | M | Length of field (Hex) and Content of field (ASCII). Use Tag Length value for each message |

#### Đặc tả chi tiết

##### MTI (Message Type Indicator) :

|  |  |  |
| --- | --- | --- |
| Request | Respone | Description |
| 0200 | 0210 | Balance Inquiry |

##### Field No : 3 Processing code

Processing code chia làm ba thành phần:

* Transaction Type code,AN2
* Account Type Code 1,AN2
* Account Type Code 2,AN2 (skip : default : 00)

**The transaction Type Code(TTC)** Sử dụng kết hợp với MTI để chỉ ra tác vụ cần thực hiện.

**Request**

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0200 | 42 | Balance Inquiry |

**Response**

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0210 | 42 | Balance Inquiry |

* **Account Type Code (ATC):** Loại tài khoản giao dịch.

|  |  |
| --- | --- |
| ATC | Description |
| 70 | Loyalty Card |

##### Field 48

|  |  |  |
| --- | --- | --- |
| Tag | Length | Description |
| FF51 | 04 | Total Points |

##### Message sample :

###### Message request:

**Header** : 6000030000 🡺 5 byte (hex)

**MTI** : 0200 🡺 2 byte (hex)

**Bitmap** : 2000000020C00000 🡺 8 byte (Hex)

**F3** : 427000 🡺 3 byte (Hex)

**F35** : 379604215000000046D140610114680300000000 🡺 20 byte ( hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

Total Message length : 5 + 2 + 8 + 3 + 20 + 8 + 15 = 61 🡺 2 byte (hex) 003d

003d600003000002002000000020C00000427000379604215000000046D1406101146803000000003232322020202020322020202020202020202020202020

###### Message Response :

**Header** : 6000000003 🡺 5 byte (hex)

**MTI** : 0210 🡺 2 byte (hex)

**Bitmap** : 2000000000C10008 🡺 8 byte (Hex)

**F3** : 427000 🡺 3 byte (Hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

**F48**: 0007FF510430303235 🡺 9 byte (hex)

**F61** :

**Message 1st:**

Tag : FF01 => 2 byte hex

Length 1st message : 20 => 0x14 => 1 byte

Message content 1st : 31302042414e204e48414e20414f205448554e20

(Value : 10 BAN NHAN AO THUN)

**Message 2nd :**

Tag : FF02 => 2 byte

Length 2nd message : 18 => hex 0x12 => 1 byte

Message 2nd content: 32302042414e204e48414e20584520444150

(Value : 20 BAN NHAN XE DAP)

Total F61 : 2 + 1 + 20 + 2 + 1 + 18 = 44 + 2 = 46 ( 2byte to send 0044)

0044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

Total Message length : 5 + 2 + 8 + 3 + 8 + 15 + 9 + 46 = 96 🡺 2 byte (hex) 0060

0060600000000302102000000000C1000842700032323220202020203220202020202020202020202020200007FF5104303032350044FF011431302042414e204e48414e20414f205448554e20FF021232302042414e204e48414e20584520444150

### Redemption Transaction

#### Tổng quát

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field No | Data Element Name | Attribute | Request | Respone | Note |
|  | MTI | n 4 | M | M | HEX |
|  | Bitmap | b 8 |  |  | HEX |
| 3 | Processing code | n 6 | M | M | HEX |
| 4 | Amount Transaction | n 12 | M |  | HEX(Point is contained here) |
| 35 | Track 2 Data | z..37 | M |  | HEX |
| 39 | Response Code | an 2 |  | M | ASCII |
| 41 | Card Acceptor Terminal ID | ans 8 | M | M | ASCII |
| 42 | Card Acceptor Identification Code | ans 15 | M | M | ASCII |
| 48 | Reserse Private | ans…999 |  | M | Length of field (Hex) and Content of field (ASCII) |
| 61 | Reverse Private | Ans …999 |  | M | Length of field (Hex) and Content of field (ASCII). Use Tag Length value for each message |

#### Đặc tả chi tiết

##### MTI (Message Type Indicator) :

|  |  |  |
| --- | --- | --- |
| Request | Respone | Description |
| 0200 | 0210 | Redemption |

##### Field No : 3 Processing code

Processing code chia làm ba thành phần:

* Transaction Type code,AN2
* Account Type Code 1,AN2
* Account Type Code 2,AN2 (skip : default : 00)
* **The transaction Type Code(TTC)** Sử dụng kết hợp với MTI chỉ ra mã tác vụ.

Request

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0200 | 44 | Redemption |

Response

|  |  |  |
| --- | --- | --- |
| MTI | TTC | Description |
| 0210 | 44 | Redemption |

* **Account Type Code (ATC) :** Loại tài khoản sử dụng.

|  |  |
| --- | --- |
| ATC | Description |
| 70 | Loyalty Card |

##### Field No : 4 Amout Transaction

Số điểm dùng để đổi quà.

##### Field 48

|  |  |  |
| --- | --- | --- |
| Tag | Length | Description |
| FF51 | 04 | Point reserve after transaction |

##### Message sample :

###### Message request:

**Header** : 6000030000 🡺 5 byte (hex)

**MTI** : 0200 🡺 2 byte (hex)

**Bitmap** : 3000000020C00000 🡺 8 byte (Hex)

**F3** : 447000 🡺 3 byte (Hex)

**F4** : 000000000010 🡺 6 byte (hex)

**F35** : 379604215000000046D140610114680300000000 🡺 20 byte ( hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

Total Message length : 5 + 2 + 8 + 3 + 6 + 20 + 8 + 15 = 67 🡺 2 byte (hex) 0043

0043600003000002003000000020C00000447000000000000010379604215000000046D1406101146803000000003232322020202020322020202020202020202020202020

###### Message Response :

**Header** : 6000000003 🡺 5 byte (hex)

**MTI** : 0210 🡺 2 byte (hex)

**Bitmap** : 2000000000C10008 🡺 8 byte (Hex)

**F3** : 447000 🡺 3 byte (Hex)

**F41** : 3232322020202020 🡺 8 byte(ASCII) 🡺 (222)

**F42**: 322020202020202020202020202020 🡺 15 byte (ASCII)

**F63** : 0007FF510430303138 🡺 9 byte (hex)

**F61** :

Tag : FF01 🡺 2 byte hex

Message content length : 25 🡺 0x19 🡺 1 byte hex

Message content : 42414e20565541204e48414e2044554f4320414f205448554e

(Value : BAN VUA NHAN DUOC AO THUN)

F61 total message length = 2 + 1 + 25 = 28 + 2 = 30

0030FF011942414e20565541204e48414e2044554f4320414f205448554e

Total Message length : 5 + 2 + 8 + 3 + 8 + 15 + 9 + 30 = 80 🡺 2 byte (hex) 0050

0050600000000302102000000000C1000844700032323220202020203220202020202020202020202020200007FF5104303031380030FF011942414e20565541204e48414e2044554f4320414f205448554e

### Đặc tả cho các trường thông điệp chung :

#### Field No : 35 Track 2 Data

Trường này chứa giá trị trong track thứ 2 của thẻ từ.Thường sử dụng để chứa mã thẻ.

|  |  |
| --- | --- |
| Name | Format |
| Primary Account Number | up to 16 digit |
| Field Separator | can be either “=” or “D” |
| Expiration Day | in YYMM format |
| Service Restriction Code | Three Digit |
| Discretionary Data | balance of avaiable digits |

#### Field No : 39 Response code

Trường này trả về mã thông điệp từ client. Giá trị của mã thông điệp được lưu trong trường 61.

|  |  |
| --- | --- |
| **Value** | **Description** |
| 00 | TRANSACTION SUCESSFULLY |
| 03 | MID OR TID NOT FOUND |
| 14 | INVALID CARD NUMBER |
| 54 | EXPIRE CARD |
| 12 | OTHER ERROR |
| 15 | INVALID FIELD |

#### Filed No : 41 Card Acceptor Terminal ID “TID”

Trường này chứa mã di nhất xác định nguồn gốc xuất xứ của thông điệp.Thường chứa mã của thiết bị đã gởi thông điệp.

#### Field No 42 Card Acceptor Identify Code “MID”

Trường này thường chứa mã của nơi chấp nhận sử dụng thẻ.

#### Field 61

Gởi thông báo về cho client.

#### Conditional Code

|  |  |
| --- | --- |
| Type | Comment |
| O | Option |
| M | Mandantory |

1. http://en.wikipedia.org/wiki/Extended\_Binary\_Coded\_Decimal\_Interchange\_Code [↑](#footnote-ref-2)
2. Trong biểu diễn người ta thường lắp đầy(padding) các vị trí dư ra bằng số 0 đối với số và khỏang trắng đối với kí tự [↑](#footnote-ref-3)
3. Trong JposEE cung cấp cho chúng ta một dãy các module được xây dựng sẵn.(Các module này chứa đựng trong thư mục opt ) [↑](#footnote-ref-4)
4. Nên xem file build.xml một cách cẩn thận để có hiểu quá trình sao chép file của Jpos vào các thư mục cũng như các target thể hiện trong ant -projecthelp [↑](#footnote-ref-5)
5. Ở đây là C:\Program File\Java\jdk1.6.0\_07 [↑](#footnote-ref-6)
6. Cố thể để bất kì nơi nào trên ổ cứng. Đường dẫn chỉ mang tính gợi ý. [↑](#footnote-ref-7)
7. <http://java.sun.com/developer/technicalArticles/J2SE/jmx.html> [↑](#footnote-ref-8)
8. Xem phần [↑](#footnote-ref-9)