

# Penguin, LLC Developer's Manual

February 2010, Pre-OSI

# TABLE OF CONTENTS

<b>Introduction .....</b>	<b>1</b>
How to Use This Book .....	2
Naming Conventions (and tips on searching this doc) .....	2
What's New in This Version .....	3
Related Links .....	3
Recommended Reading .....	3
 <b>Chapter 1 - History of Penguin .....</b>	 <b>4</b>
Quote Server .....	5
OrderRouter .....	5
Front end .....	5
Company Time Line .....	5
Future Goals .....	7
 <b>Chapter 2 - Business Overview .....</b>	 <b>8</b>
Business Rules .....	9
High Level Data Flows .....	10
 <b>Chapter 3 - How To Be Successful At Penguin .....</b>	 <b>11</b>
Bottom Line .....	11
Dan gets frustrated when... ..	12
Dan does not get frustrated when... ..	13
 <b>Chapter 4 - Technical Requirements .....</b>	 <b>14</b>
System Availability .....	14
Data Back-up .....	14
Hardware Requirements .....	15
Software Requirements .....	15
Business Recovery Requirements .....	15
 <b>Chapter 5 - Software Development Life Cycle (SDLC) ..</b>	 <b>16</b>
Developer's Workflow .....	17

pnProject .....	18
pnProduction .....	19
pnDeveloper .....	19
Communication .....	19
Coding Standards .....	20
Data Protocols .....	22
Testing .....	22
Replay Data Logging .....	22
Stock Quote Replay .....	22
Option Quote Replay .....	22
Exchange Simulators .....	22
Debugging .....	23
Nightly Build and OR Testing .....	24
Process .....	24
OrderRouter Testing .....	24
QA Server List .....	24
Software Installation and Management .....	25
pninstall (Windows clients) .....	25
Linux Deployment Kit .....	26

## **Chapter 6 - APIs .....29**

## **Chapter 7 - Quote Servers .....30**

Option Quote Servers .....	31
PIF (Person Interface) .....	31
PA .....	34
pnReplayData .....	35
nq .....	36
CBOE .....	36
ISE Spreadbook .....	36
Stock Quote Servers .....	37
Combo Server .....	38

## **Chapter 8 - OrderRouter .....39**

Option OrderRouter .....	40
Proxy .....	42
Global Position Server (GPS) .....	43
Drop Server .....	44

auth server .....	44
<b>Chapter 9 - Front End Applications .....</b>	<b>45</b>
Penguin Front End .....	46
TradeDesk .....	47
<b>Chapter 10 - Option Market Makers .....</b>	<b>48</b>
Option Market Maker Architecture .....	48
pnMM / pnMM Server .....	49
<b>Chapter 11 - Option Market Takers .....</b>	<b>56</b>
Architecture for Market Taker and Arbitrage Apps .....	57
<b>Chapter 12 - Option Arbitrage Applications .....</b>	<b>58</b>
<b>Chapter 13 - Stock Clients .....</b>	<b>59</b>
<b>Chapter 14 - important Files .....</b>	<b>60</b>
db_option.out, db_option_key.out .....	61
option_openinterest.txt .....	62
company_names.txt .....	62
stock_listed_exchange.txt .....	62
<b>Chapter 15 - Real Time Option Trade Statistics .....</b>	<b>63</b>
Option Trading Page .....	63
<b>Chapter 16 - Monitoring .....</b>	<b>65</b>
<b>Chapter 17 - Reporting .....</b>	<b>66</b>
<b>Appendix A - Process / Data Flows .....</b>	<b>67</b>
<b>Appendix B - Daily Time Line .....</b>	<b>68</b>

<b>Appendix C - Data Protocols .....</b>	<b>69</b>
<b>Glossary .....</b>	<b>70</b>
<b>Index .....</b>	<b>74</b>

# INTRODUCTION

Penguin is a private company that specializes in developing algorithms and technology infrastructure for high frequency trading of equities and derivatives.

***Penguin does not speculate.*** Penguin does not take a position and hope, pray, or wish that it moves in the company's favor. Penguin does not believe that historical data has any bearing on what will happen in the future. Instead, Penguin capitalizes on the intra-day fluctuations and temporary imbalances that exist in the market. In other words, ***Penguin profits come from market making and arbitrage in U.S. equities and equity options.***

The idea of a market maker is to simultaneously buy on the bid price and sell at the ask price. For example, if customer A wants to sell 200 shares of IBM at \$100.00, Penguin buys these shares and immediately sells them to customer B for \$100.01.

Arbitrage is simply capitalizing on the price differential for the same security traded on two different markets by buying the cheaper one and selling the more expensive one at the same time. Performed in great enough volume throughout the trading day, the sum of these spreads can add up to big profits.

Because profits are intimately linked to trade frequency and volume, speed and reliability are essential requirements for the system, which executes trades in an average of less than 100 milliseconds for options and less than one millisecond for stock. Therefore, machines and software must run for 6.5 hours every business day while the market is open (8:30 a.m. – 3:00 p.m. CST) without crashing or adding significant latency.

## How to Use This Book

---

If you're new to Penguin and the electronic trading business, please read the first twenty pages which provide an overview of Penguin, the online trading process, and the technical requirements for the trading environment. You should also understand the interrelationship among client applications for both options and stocks. The diagrams found in the appendix (starting with "Process / Data Flows" on page 67) provide a high level view of these data flows.

Because this is a developer's manual, the focus is on back end topics like source/target protocols, command line arguments, message definitions, configuration/deployment information, and testing notes. This is the general outline for most chapters.

A minimal level of front end functionality is provided for context only. For more granular front end information, you can refer to the context-sensitive help within each particular application or at <http://wiki.s/index.php/Applications> on the Penguin Portal.

As a developer, you should also peruse the chapter titled "Software Development Life Cycle (SDLC)" on page 5-16 which discusses the process of coding, implementing, testing, and releasing applications into production.

The remainder of this manual can be used as needed for your area(s) of expertise.

## Naming Conventions (and tips on searching this doc)

---

When searching this document for a particular application, remember to concatenate the words that make up the name. For example, you will not find any information on "OrderRouter" by inputting "Order Router" in the search field. Search terms are not case sensitive but they are space sensitive.

At some point you will be creating your own applications. When you do, remember to concatenate the words and try to keep the names of applications as short and meaningful as possible.

## What's New in This Version

---

- How to Be Successful at Penguin
- More stock and option applications
- Source/target data flow models in the appendix
- External links to data that change too frequently to remain relevant in print for a reasonable length of time
- An index to quickly locate specific items of interest
- A separate Front End and API section preceding the option and stock client application sections
- SDLC section

## Related Links

---

- Penguin Portal ([http://portal.p/index.php/Main\\_Page](http://portal.p/index.php/Main_Page)) -- contains information on a diverse range of topics from front end user manuals to what's for lunch today
- Internal Contact List (<http://portal.p/index.php/Directory>)
- Data Center Contacts ([http://portal.p/index.php/Data\\_Center\\_Contacts](http://portal.p/index.php/Data_Center_Contacts))
- PenguinProject Webpage (<http://www.epenguindata.com/PenguinProject/>)
- *Spread* (<http://portal.p/index.php/Newsletter>) is Penguin's monthly employee newsletter. It contains team features, individual profiles, HR topics, interviews with trading and technology experts, and various fun and informative articles.

## Recommended Reading

---

- *Trading and Exchanges: Market Microstructure for Practitioners* by Larry Harris
- *Option Volatility and Pricing: Advanced Trading Strategies and Techniques* by Richard D. Irwin
- *Dynamic Hedging: Managing Vanilla and Exotic Options* by Nassim Taleb



# CHAPTER 1

## HISTORY OF PENGUIN

As you read this, you're surrounded by cutting edge technology, a diverse team of professionals, and cool office space. But it wasn't always this way.

In 2005, Ben Black and Daniel White launched Penguin out of a dilapidated non-air-conditioned room with two computers, a ton of sweat, and whatever savings they had.

And for the first two or three weeks, the typical exchange between these two ex-CME colleagues went something like this:

“Dan, give me something I can trade!”

“Dude! I’m going as fast as I can.”

And at some point while Dan was coding and Ben was setting up Quickbooks, they came up with a name for the business they were so feverishly building.

*Penguin* — a cool climate bird and mascot of Linux, the operating system upon which Penguin’s critical routers and servers were built.

This chapter explores the history of Penguin’s architecture in more detail with a focus on the quote server, order server, and front end.

## Quote Server

---

[content removed]

## OrderRouter

---

[content removed]

## Front end

---

[content removed]

## Company Time Line

---

**Table 1.1** Technical evolution

Category	2005-06	2006-07	2007-08	2008-09
<b>Data Provider</b>	Penson	Arca/Inet	Penson Arca/Inet	Penson Arca/Inet Bats DirectEdge ISE Spreads CBOE Spreads
<b>Operating System</b>	Windows Linux Routers	Windows Linux Routers Linux Servers	Windows Linux Routers Linux Servers	Windows Linux Routers Linux Servers
<b>Programming Language</b>	C++/Perl/PHP	C++/Perl/PHP	C++/Perl/PHP	C++/Perl/PHP
<b>Connect</b>	Non-Tier 1 Internet	Tier 1 Internet DS3, OC3	Tier 1 Internet DS3, OC3	Tier 1 Internet DS3, OC3
<b>Front End</b>	IQ Thinkpipes	IQ Thinkpipes	IQ Thinkpipes	Penguin
<b>Office Location</b>	133 S. Main	456 W. Washington	456 W. Washington	456 W. Washington
<b>Home Grown Software</b>	PenguinA	Market Making System Stock System	Market Making System Stock System	Market Making System Stock System

**Table 1.1** Technical evolution

Category	2005-06	2006-07	2007-08	2008-09
<b>Networking</b>	Netgear switches, Linux routers 100 Mbps	Used Cisco	New Cisco Force10 Infiniband	Cisco Arista 10 Gbps Ethernet
<b>Workstation Hardware</b>	Pentium D	AMD	Core2Duo	Core i7, 4GB, 2-4 30" monitors
<b>Server Hardware</b>	Dual AMD	Dual AMD	Dual AMD	8 core, 16 GB, 10 Gbps
<b>Colocation</b>	Inet / Arca / Penson	Inet / Arca / Penson Equinix Chicago	Inet / Arca / Penson Equinix Chicago	Inet / Arca / Penson Equinix Chicago
<b># People</b>			8 tech, 30 traders	20 tech, 25 traders

## Future Goals

---

- Change network protocol from TCP to UDP throughout the trading environment
- Add option book feeds
- Replace fixed format `pn_protocol.h` with a more flexible, more generic but still high performance protocol
- Replace Instaquote protocol where possible with one that is faster and less verbose.

# CHAPTER 2

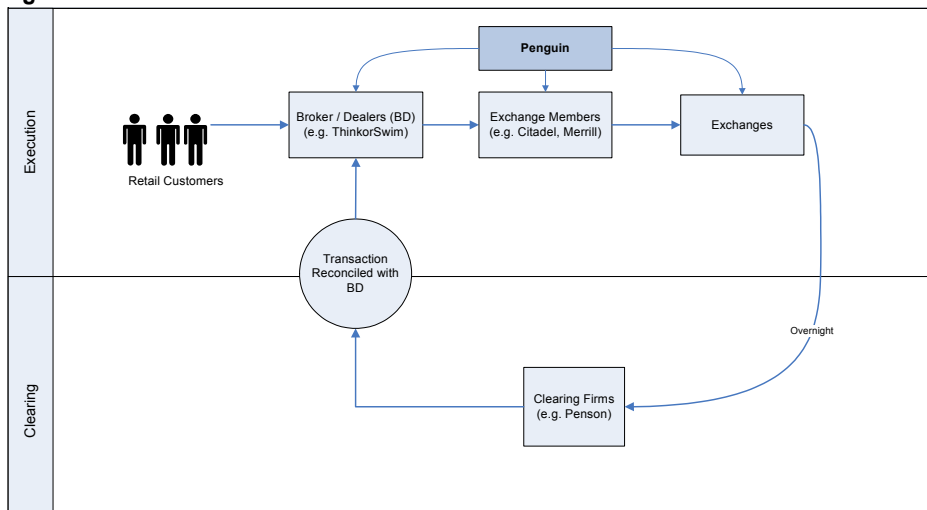
## **BUSINESS OVERVIEW**

Penguin buys and sells shares through a broker, through an exchange member, or directly through an exchange.

Membership in an exchange requires high monthly fixed costs. But because we are sponsored by an exchange member (Penson), we have advantages the typical retail customer does not such as sending orders directly to the exchange with very low latency.

At the end of the trading day, trades are sent to our clearing firm, Penson, which guarantees both sides of the transaction and debits/credits Penguin's account as appropriate.

Penguin pays all related execution and clearing fees.

**Figure 2.1** Trade Transaction Overview

## Business Rules

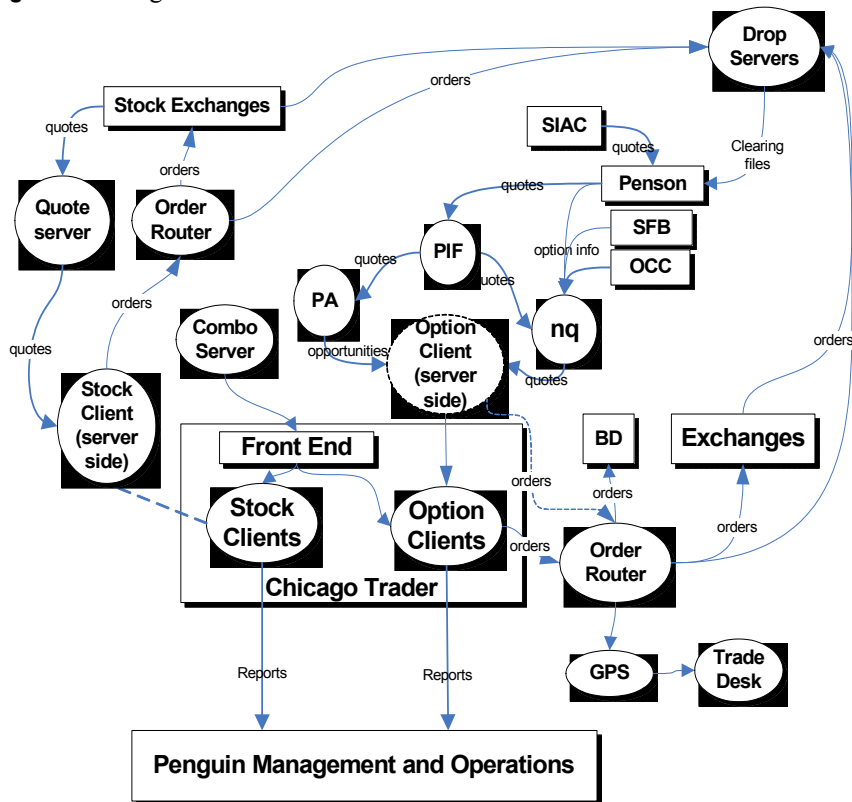
- Machines that support trading must run from 8:30 a.m. – 3:30 p.m. CST without crashing or adding latency.
- No system changes are performed during weeks in which options expire, usually the third Friday of every month
- No system maintenance is performed during the day from 7:00 a.m. – 3:30 p.m. CST except when absolutely necessary.

## High Level Data Flows

The following diagram shows data flows between users and systems within the Penguin Trading Environment (circles represent systems; squares represent users/groups)

The key data objects are **quotes** and **orders**

**Figure 2.2** High Level Data Flows



# CHAPTER 3

## HOW TO BE SUCCESSFUL AT PENGUIN

Penguin has a great team of people who are eager to answer questions and help each other out. We also have a variety of resources. If you need a book or something, just ask and we can order it. The most successful employees at Penguin are the ones who work with their team and are proactive about getting what they need to get the job done.

### Bottom Line

---

Penguin turns a profit when our strategies run on machines that don't crawl or crash. But reliability is a minimum expectation. The more efficiently we move trading strategies down the pipeline from concept to production, the more money we make.

In the next two pages, we'll go inside the mind of Dan White who has some useful tips on how to be proactive, efficient, and successful in your role.



## Dan gets frustrated when...

---

**We aren't moving forward**

Don't stay stuck! Do lots of little releases, abuse PenguinProduction so everyone knows it's moving forward.

**You shoot yourself in the foot**

Don't run too much on a given machine. Resources are not infinite.

**We have to revisit problems we thought were fixed**



**Software isn't tested at a "basic" level**

You should always make sure that the most important features still work.

**You don't ask for help**

If you're overwhelmed or you're going to miss deadlines or you need more hardware, please let your team lead know sooner rather than later.

**You aren't regularly on time to set up the applications you support throughout the trading day**

Remember that the market doesn't wait for you to open, so please be on time!

## Dan does not get frustrated when...

---

**You find a way to move forward**

**He gets short, positive updates even if overall it might not be good news.  
One sentence, two at most**

Bad example: "Dan, there is a bug, can't find it because code is a mess, could be a problem with developer xyz and their code, not sure what the problem is, can't reproduce it, might need to replace all code with Java."

Good example: "Dan, found a bug in situation xyz, will test it and get it released today."



**You build credibility**

**There are bugs, unless they clearly should have been tested**

I would love to have you pick up projects as they come your way and sit back and watch the Pnproduction emails come in as you roll it out. However, in order to get to that point you need to have a track record of being able to deliver quality software in a timely manner. I'm much more likely to agree to "the new python-based trading framework using a GPU and infiniband" if it comes from someone who has done an awesome job on the last eight projects they worked on over the last six months. You won't get challenging projects if you can't complete simple ones quickly and bug free.

The software we write is complex and we don't have perfect test environments. I expect there to be bugs when we roll out initially. If there aren't some minor bugs you probably aren't aggressive enough in rolling out code.

**You don't whine about  
The state of the source code.**

**You put in extra time when needed**

I know it sucks. I know it doesn't have any comments. I know it isn't const-correct. I know it is a mess. Rather than whining about it, you can either fix pieces as you go (which really doesn't seem like a reasonable use of your time) or you can learn to read code better.

Unfortunately, sometimes you need to put in extra time. You can have a laptop machine for use at home, you can VPN in, but you have to find the time to hit the deadlines you set, even when your time is getting soaked up with support or anything else you're doing.

**You're not unrealistically positive about schedules or anything else**

If I think it can be done in a day and you think it will take a week, you should stick to your estimate and then deliver.

# CHAPTER 4

## TECHNICAL REQUIREMENTS

### System Availability

---

#### **System Up-Time**

7:30 a.m. – 5:00 p.m. CST

#### **Routine Quiet Batch Window**

After normal business hours

#### **Routine Systems Maintenance Window**

After normal business hours, except during weeks when options expire (typically the week in which the third Friday of the month falls)

### Data Back-up

---

[Content removed]

## Hardware Requirements

---

[Content removed]

## Software Requirements

---

### Development Tools

Microsoft Visual Studio 2008, svn, gcc 4.1.2, Intel compiler

### Programming Language

C, C++, Perl, PHP, Python

### Database Management System

MySQL

### Middleware/ Messaging Software

Proprietary

## Business Recovery Requirements

---

[Content removed]

# CHAPTER 5

## **SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)**

This chapter discusses guidelines for developing, testing, and releasing new code into production. It also introduces the tools Penguin uses for testing applications and communicating progress at every stage of the SDLC.

Of the well known SDLC models, Penguin's most closely resembles "agile" or "scrum." We work in cross-functional teams in a single open office that promotes teamwork, collaboration, adaptability to frequent changes, and communication.

## Developer's Workflow

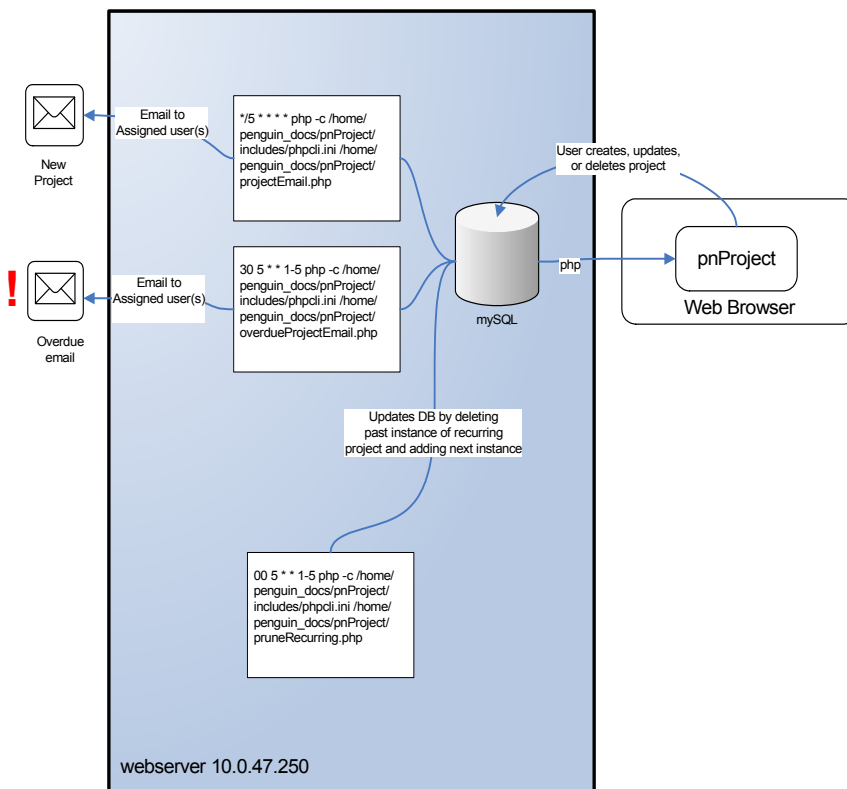
---

- 1** PnProject is created/assigned/updated.
- 2** Developer implements feature or fixes bug.
- 3** Developer tests software and makes sure memory, cpu, bandwidth, and disk usage is acceptable.
- 4** Developer checks code into svn.
- 5** Developer deploys limited release to Production and sends pnProduction email.
- 6** Trader tests release.
- 7** Developer deploys firm-wide release and sends pnProduction email.
- 8** Developer confirms with trader that software is functional and marks pnProject as closed.

## pnProject

pnProject is a workflow management tool used for tracking issues and work-in-progress. Every time a user adds or updates a project, an email is triggered and sent to the assigned user(s). The email contains the subject and the project ID. Similarly, whenever an item is more than one day overdue, a separate overdue email is sent to the assigned user(s) with an urgent status.

**Figure 5.1** pnProject Data Flows<sup>1</sup>



1. Data protocol is PN; network protocol is TCP

## pnProduction

---

Once a project flows out of the pipeline into production, it should be reported via pnProduction.

The purpose of pnProject is to track the progress of projects in development. pnProduction, on the other hand, is used for writing short, very high priority emails, letting everyone know what is going on. It is the end result of pnProject.

There is some overlap between the two, and pnProject could potentially replace pnProduction one day.

## pnDeveloper

---

pnDeveloper@googlegroups.com is a google group for developers to discuss and alert other developers to relevant changes in the development world.

# Communication

---

### Get it in writing!

As Penguin matures, becomes more complex, and projects cross group boundaries, effective communication becomes all the more important. In order to keep this process as smooth and productive as possible, make sure you **document important conversations**.

If you discuss a project verbally with someone, either follow up with an email or a pnProject that states what should happen and by when. **A specific project without a specific deadline is worthless if it is not written down somewhere.** That doesn't mean it will always get done by the specified date, but at least everyone knows what they are shooting for. This shouldn't be a long drawn out process; it only takes a couple minutes to send an email or do a pnProject.

Written communication (i.e. IM, email, and pnProject) provides three distinct advantages over verbal communication:

- 1 It tends to be clearer and more well thought out
- 2 It leaves a trail
- 3 It eliminates noise that can often interfere with the listening process (interruptions, distractions, etc.)



# Coding Standards

---

*"The code is more what you'd call guidelines than actual rules." - Captain Barbossa in Pirates of the Caribbean, Curse of the Black Pearl*

## First things first

The first "include" should be `include pn_pl.h` because it helps set up the standard environment. You should look through `L/kr/pn_pl.h` briefly. It includes standard types and functions used at Penguin.

## Conditional Compilation for Windows/Linux

```
#ifdef WIN32/#ifndef WIN32
```

## Tabbing/Spacing

Code should not include tabs.

Standard indent is 2 spaces as follows:

```
if ( )
{
}
else
{
}

for ( ;; )
{
}
```

## Standard function or other separator

```
//-----
//-----
```

## Standard class layout

```
class name
{
public:
```

```

        functions();
        variables;
protected:
        functions();
        variables;
private:
        functions();
        variables;
};

```

## Naming convention for functions, classes, structs

The first word should be lower case; the rest of the words — first letter upper case as in:

`getAppTime()` and `setBlocking()`.

Some classes begin with `pn_` as in `pn_baseSocket` but this is not required. Structs should be:

```

typedef struct
{
} goodNameForStruct;

```

and then should be referred to further as `goodNameForStruct`.

## Naming convention for variables

Loosely follow Hungarian Notation, as in:

```

string strAccount;
char *pszAccount;
int nShares;
hash_map< int, cstring > hmKeyValue;

```

Apply same capitalization rules per functions when naming variables. When in doubt, try to blend in with the de facto standard, and when that is in doubt, look at code in L/kr such as `pn_socket.h`, `pn_socket.cpp`.

## Data Protocols

---

[Content removed]

## Testing

---

**Testing is very important at Penguin.**

**The QA team** is dedicated to building all Penguin software nightly and performing OrderRouter testing. These processes are discussed in “Nightly Build and OR Testing” on page 5-24.

**You**, the developer, are responsible for testing your own applications and making sure that they're included in the nightly build. Here are some testing tools you can use.

### Replay Data Logging

---

[Content removed]

### Stock Quote Replay

---

[Content removed]

### Option Quote Replay

---

[Content Removed]

### Exchange Simulators

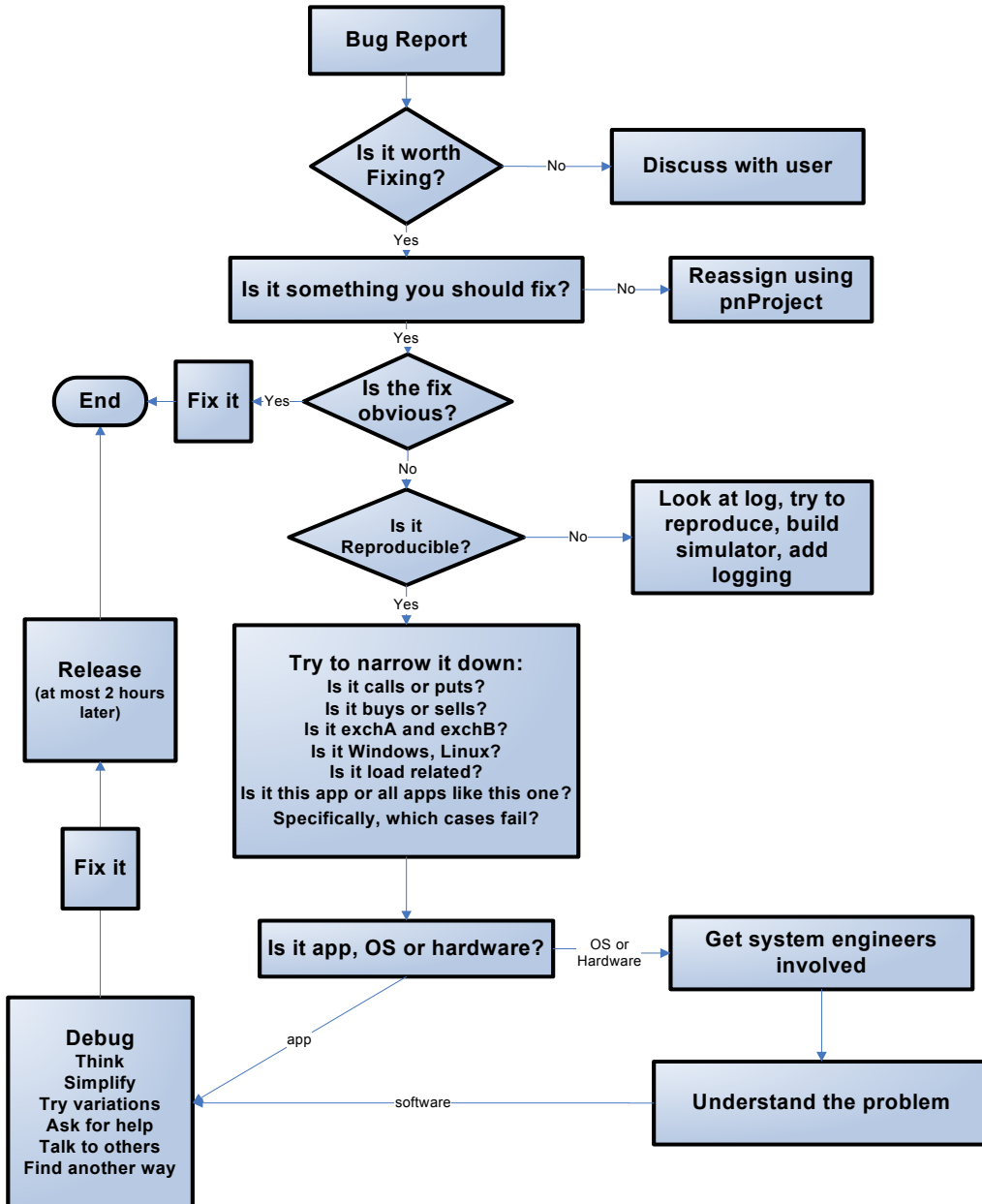
---

[Content removed]

# Debugging

Debugging is an ongoing process that can be mitigated using the following workflow

**Figure 5.2** Debugging workflow



## Nightly Build and OR Testing

---

Every night the QA team builds all Penguin software and performs OrderRouter testing. This build includes 110 windows applications on indiaqa4 and 67 Linux applications on indiaqa2.

### Process

---

- 1 Development checks in unit-tested code to the svn Source Control system.
- 2 At 7:00 p.m. a cronjob kicks off the Cruise Control\* build loop, which performs a nightly build of all Penguin software.
- 3 The nightly build finishes at approximately 6 a.m.
- 4 QA tests/re-tests code until it is bug-free, logging defects to pnProject as needed.
- 5 Cruise Control sends an automated report to the Development and QA group.
- 6 Developers then fix bugs as needed and close issues in pnProject before deploying code to production.

\* Cruise Control is an application that performs the nightly build and generates reports which show the build failures, if any, and the list of latest modifications done for that build. For a full list of Cruise Control reports go to 182.178.129.181:81 – Windows CC tab and Linux CC tab.

### OrderRouter Testing

---

[Content removed]

### QA Server List

---

[Content removed]

# Software Installation and Management

---

- Windows clients are installed via pninstall.
- Linux clients are installed via the linux deployment kit.

## pninstall (Windows clients)

---

[Content removed]

## Source Target Information

---

[Content removed]

## Message Definitions

---

All message traffic flows via http protocol from the web server.

## Deployment / Installation/ Configuration

---

pninstall is initially installed when a new machine is configured. Following the initial install, pninstall keeps itself up-to-date by downloading the newest version every time it runs. The script that launches pninstall (pninstall.bat) runs every day at 7:45 a.m. on the traders' machines.

pninstall is also used to update software and deployment to the trader machines. When pninstall runs, it kicks off many other scripts — scripts that check for various updates, changes, and scripts that start crucial programs such as pnPerfMon (*pnPerfMon monitors the health of the trader machines and multicasts it for pnMonitor*)

## Test Notes

---

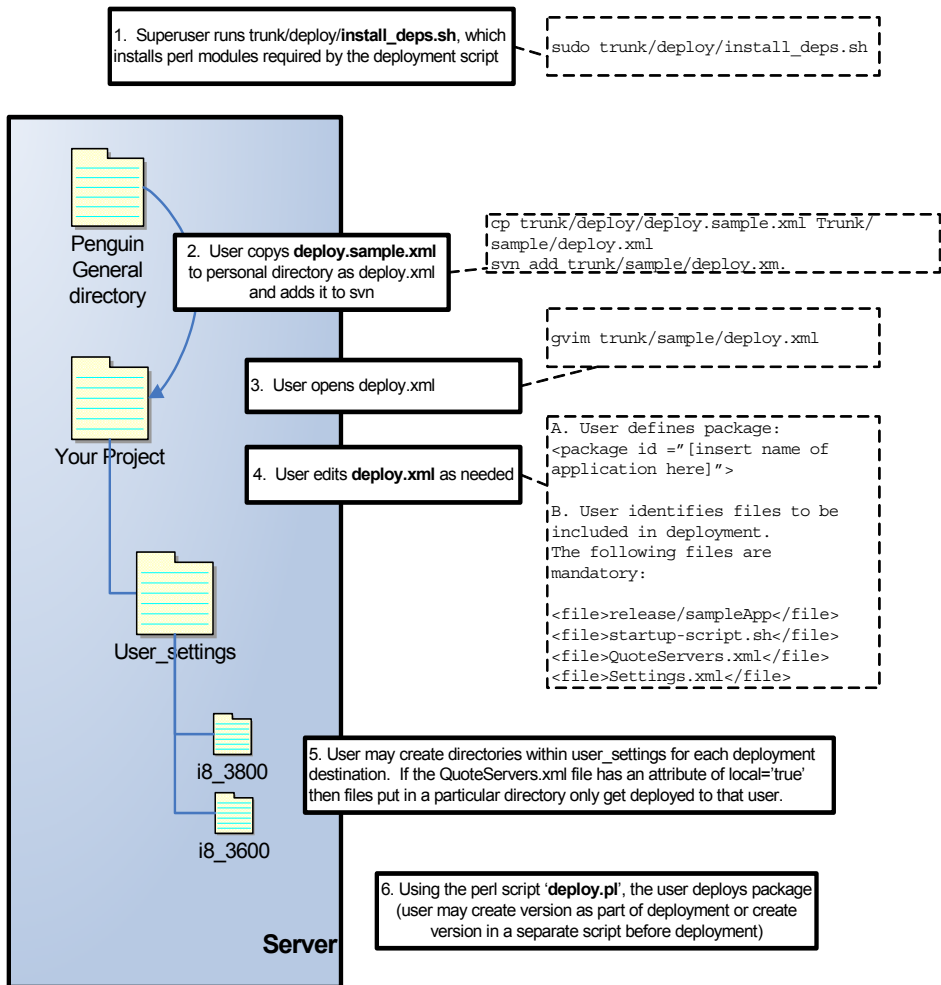
A version is released to a private “account” in production. This test account (an arbitrary alphanumeric name) can then be used to test the version without interfering with those in use.

## Linux Deployment Kit

The Linux Deployment Kit can be used to manage versioning and deployment of Linux applications. It is especially useful when deploying an application to many servers, a task which can be daunting when done manually.

The Linux Deployment Kit contains three files: *deploy.pl*, *deploy.sample.xml*, and *install\_deps.sh*. These files and the deployment process are discussed in more detail below.

**Figure 5.3** Linux Deployment Process



## Install\_deps.sh

*Install\_deps.sh* installs the modules required by the deployment script.

Note: Only a superuser can run this script.

## Deploy.xml

*Deploy.xml* contains three main sections:

- 1 **Repository** – this section specifies the location of the deployment repository. It also provides storage for multiple versions. You should not need to make any changes as there is only one repository. The host, user, and password are already filled in.
- 2 **Package** – within this section you will specify which files should be part of your deployment. Notice that package has an attribute called ‘id’. This should be set to the name of your application. Deployments with the same id will get grouped together and that is important when deploying to specific groups or ids.

- a Set this to the name of your application now.

```
<package id="sample">
```

- b Next, specify the files you would like to be included in your deployment. Use the file tag for this. Add every file that needs to be deployed.

```
<file>release/sampleApp</file>
```

```
<file>startup-script.sh</file>
```

```
<file>QuoteServers.xml</file>
```

```
<file>Settings.xml</file>
```

- c When deploying release/sample App, don't want deploy it into the 'release' folder. Keep it in the main directory like everything else. You can do this by adding the 'flatten' attribute.

```
<file flatten='true'> release/sampleApp</file>
```

- d For customized deployments, Use a separate QuoteServers.xml for each user receiving a customized deployment by setting the 'local' attribute for that file

```
<file local='true'>QuoteServers.xml</file>
```

- e If there is a file that you would like to deploy only once and never overwrite, you can specify this with the 'nooverwrite' attribute

```
<file nooverwrite='true'>Settings.xml</file>
```

- 3 **Destination** – a destination is a location for an actual deployment. You can have as many of these as you like. The elements of destination are **login**,



**password**, **machine**, and **directory**. It also contains the optional attributes **id** and **group**.

For most applications you will want to have a default destination (`id='default'`). It is optional, but its setting will get inherited by your other destinations and this can save you time.

A sample destination definition is shown in Figure 5.4. In this sample the user wants three destinations aside from the default.

**Figure 5.4** sample destination file



[Content removed]

# CHAPTER 6

## APIs

[Chapter removed]

# CHAPTER 7

## QUOTE SERVERS

Quote servers receive quotes from exchanges and pass these data to the appropriate client applications.

### **Option quote servers**

- PIF
- PA
- nq
- CBOE
- ISE Spreadbook

### **Stock quote servers**

- bs book servers
- combo server

# Option Quote Servers

[Content removed]

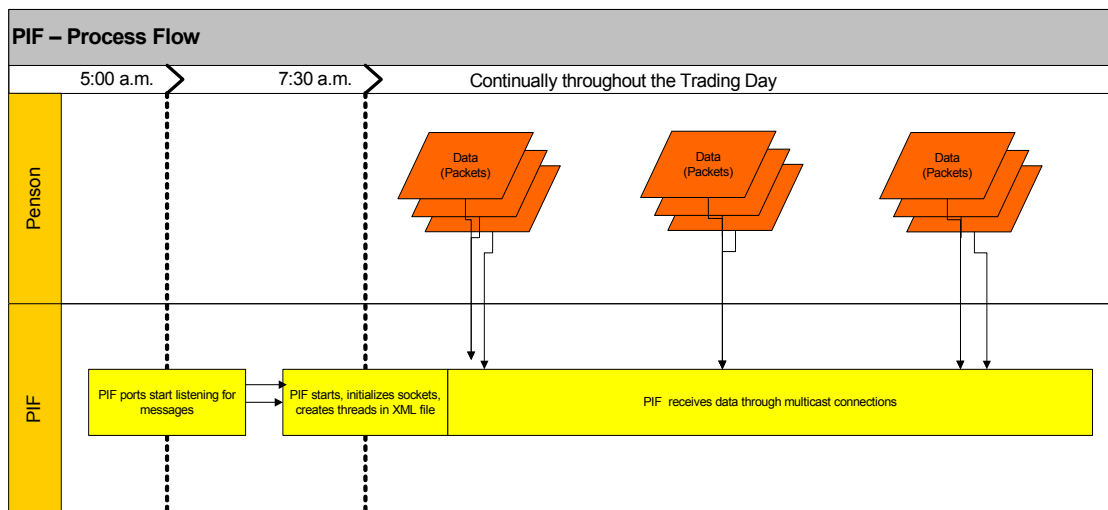
## PIF (Pencon Interface)

PIF is an application that parses and reformats data received through multiple multicast connections with Pencon. The goals of PIF are to:

- Read multicast streams from Pencon and write them into shared memory
- Reformat Pencon data into a more compact binary format. Pencon sends data down in ASCII format at a speed of roughly 400 Mbps (and growing rapidly). Significant savings in memory bandwidth and processing downstream are achieved by converting data to a binary format.

A multicast socket must be opened to join the group and receive Pencon data, which is transmitted in blocks. Each block can contain many events, and block lengths can be as large as 1,000 bytes. The blocks are designed so they can be conveyed the same regardless of the transmission method. The part of the code that reads the data from the socket must be free running. The code should read data, queue it, and immediately return to reading for new data.

**Figure 7.1** PIF Process Flow



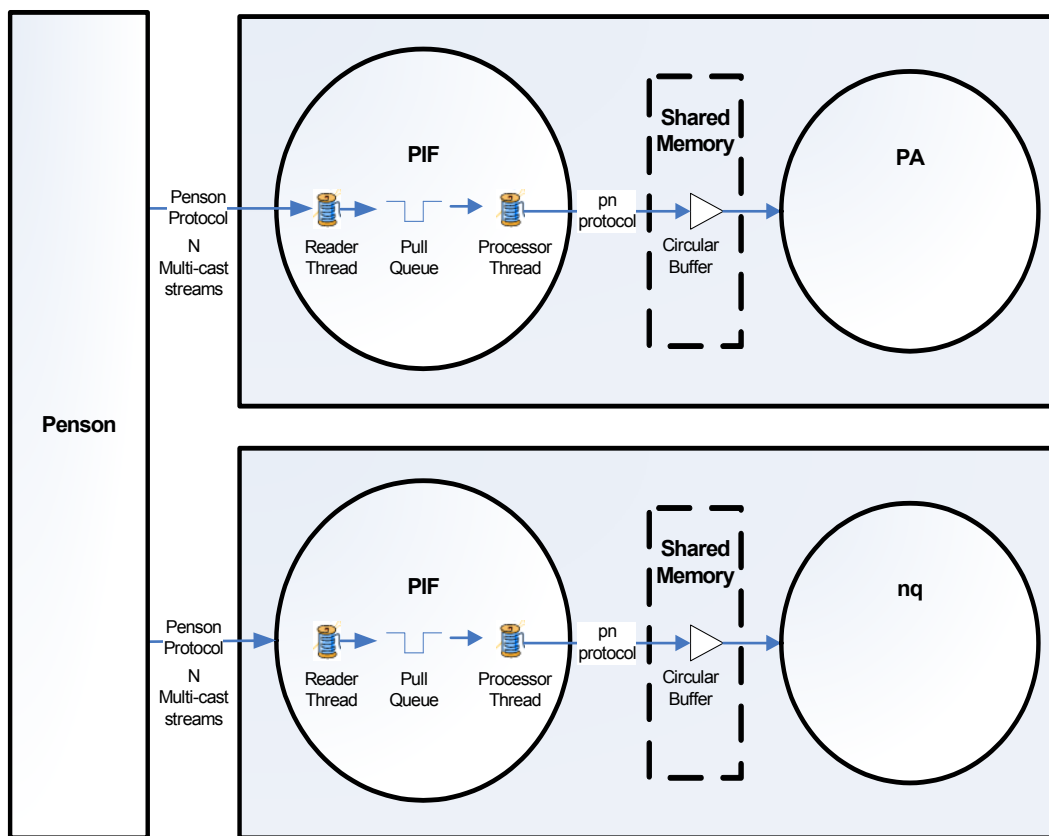
## Logging / Error Processing

Penson sends data only once. If the shared memory queue reaches capacity, the circular buffer will overwrite the oldest data in the shared memory queue, effectively dropping the overwritten quote data from the network. PIF detects drops in the network and writes these to the error log. Additionally, Nagios sends an email to notify the Technology Team.

Logs are written to a logs subdirectory in the PIF working directory. The high level log is *PIF\_<yyyymmdd>\_<pid>.log*. There are also per-thread logs written to this directory as well. The number of threads is defined in the xml configuration file.

## Source / Target Information

Figure 7.2 PIF Interface



Note PIF parses data from Penson, reformats them into pn protocol, and makes them available via shared memory to PA and nq applications.

## Message Definitions

---

**Table 7.1** Message Definitions<sup>1</sup>

Message Type	Description
MMQuote	(from docs\penson\TickStreamSpecification.html) gets converted to pn_stockTrade (L\kr\pn_protocol.h)
Quote message	(from docs\penson\TickStreamSpecification.html) gets converted to pn_stockQuote (L\kr\pn_protocol.h)
Trade message	(from docs\penson\TickStreamSpecification.html) gets converted to pn_mmQuote2 (L\kr\pn_protocol.h) if it is a stock. If it is an option it gets converted to pn_optionQuote (L\kr\pn_protocol.h) or pn_optionQuote2 (L\kr\pn_protocol.h). pn_optionQuote2 is used if the bid and ask prices are not in pennies.

1. The code for the conversions is in penson.cpp

## Configuration / Deployment / Installation

---

```
<PARAMS>
<PUBLISH_PORT>38000</PUBLISH_PORT>
<THREAD1>1</THREAD1>
<THREAD1_CALLBACK>1</THREAD1_CALLBACK>
<THREAD1_MCASTPORT>8901</THREAD1_MCASTPORT>
<THREAD1_MCASTIP>236.63.73.63</THREAD1_MCASTIP>
<THREAD1_BINDIP>10.65.141.6</THREAD1_BINDIP>
<THREAD2>2</THREAD2>
<THREAD2_CALLBACK>1</THREAD2_CALLBACK>
<THREAD2_MCASTPORT>8902</THREAD2_MCASTPORT>
<THREAD2_MCASTIP>236.63.73.63</THREAD2_MCASTIP>
<THREAD2_BINDIP>10.65.141.6</THREAD2_BINDIP>
<THREAD3>3</THREAD3>
<THREAD3_CALLBACK>1</THREAD3_CALLBACK>
<THREAD3_MCASTPORT>8903</THREAD3_MCASTPORT>
<THREAD3_MCASTIP>236.63.73.63</THREAD3_MCASTIP>
<THREAD3_BINDIP>10.65.141.6</THREAD3_BINDIP>
...
...
...
</PARAMS>
```

## Command Line Arguments

---

**Table 7.2** Command Line Arguments

Command Line Argument	Definition	Default
-daemonize	“1” means it runs in the background as real server; there is no client for it.	1
-udpSocketPort		38300
-symbol	!STAT	
-xmlconfig		

## PA

---

[Content removed]

## **pnReplayData**

---

[Content removed]

## **Command Line Arguments**

---

[Content removed]

## **Deployment / Installation / Configuration**

---

[Content removed]

## **Logging / Error Processing**

---

[Content removed]

## **Test Notes**

---

### **Steps**

- 1 Set `threadxExitAtEOF` to 1 in both the new `pnReplayData` and the `pnReplayData` to be tested against. This way, both instances will exit after reading the log file once.
- 2 Connect quoteprinters to each of the 2 instances of `pnReplayData`.
- 3 Once the application terminates, clean the quoteprinter log files using the command `perl cleanLogFile.pl`
- 4 Compare the clean log files using the command `fc file1 file2`.

### **Alternate Steps**

- 1 Terminate application by setting a limit on the number of messages it receives, automatically terminating `pnReplayData` instances when it reaches the specified number of messages.



## **nq**

---

[Content removed]

## **CBOE**

---

[Content removed]

## **ISE Spreadbook**

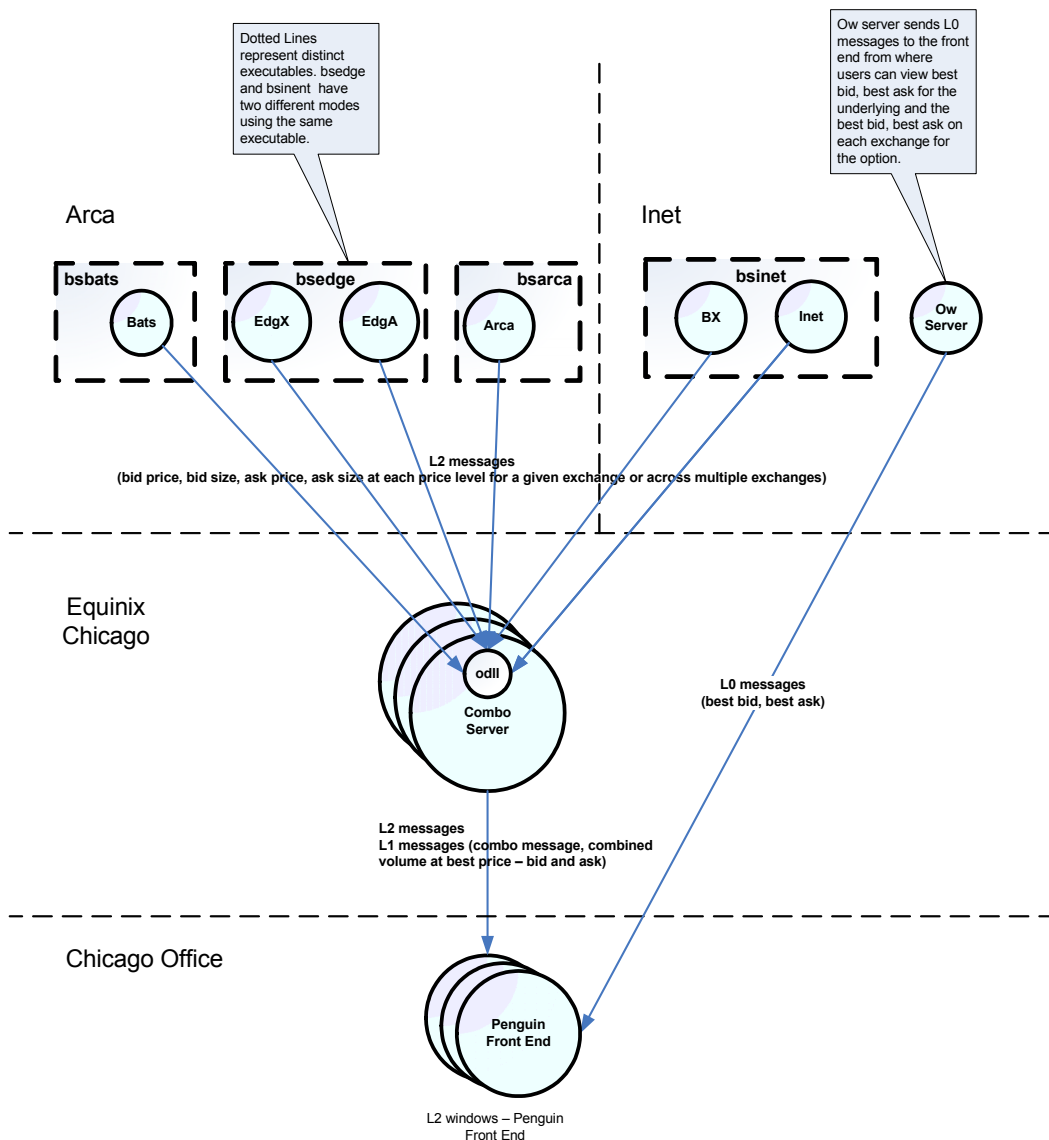
---

[Content removed]

## Stock Quote Servers

Stock quote servers include books servers (bsarca, bsinet, etc.) and the ouch server. They send quotes from the Arca and Inet colos to the Combo Server at Equinix in Chicago.

**Figure 7.3** Stock Quote Servers



Note: Trading apps (on the traders' machines) subscribe to the stock servers also.

## Combo Server

---

[Content removed]

# CHAPTER

# 8

## ORDERROUTER

The OrderRouter (OR) receives, cancels, and update orders. Client applications send order data to a Consolidated OR, which then routes the data to the appropriate Edge OR, vendor, or exchange.

Although the Consolidated OR can send data directly to vendors or exchanges, data is typically routed through an Edge OR first.

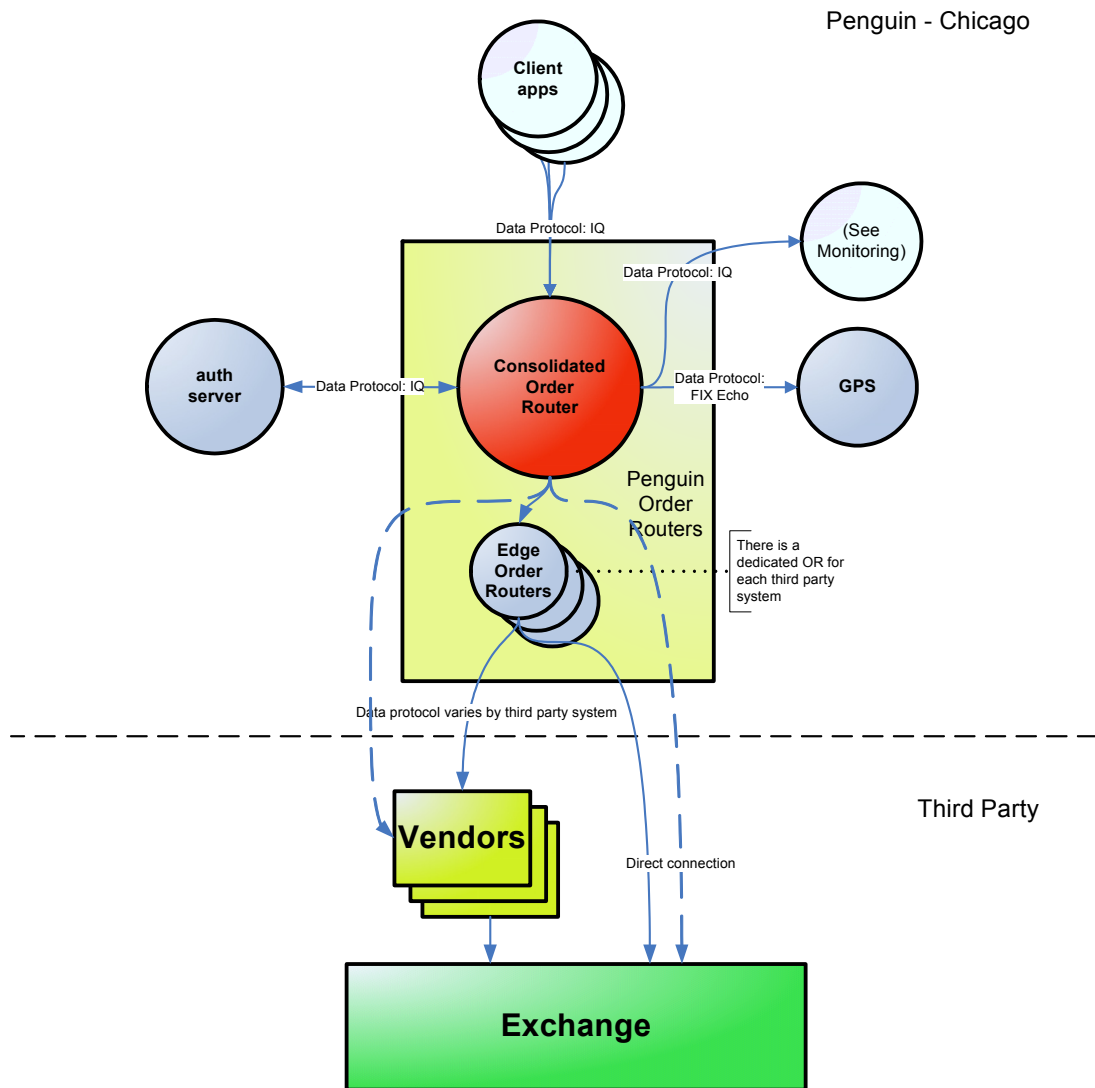
There is really only one OR but several different configurations. An edge OR is the same thing as an exchange OR. They both connect to an external order execution partner, which might be an exchange or a third party vendor like Merrill.

A high level view of the Penguin OR architecture is presented in Figure 8.1.

**Figure 8.1** Penguin OR Infrastructure  
[Content removed]

## Option OrderRouter

**Figure 8.2** Option OrderRouter Data Flows



## Message Definitions

---

Messages from the client to the OrderRouter are in Instaquote (IQ) format.

Messages from OrderRouter to the third party application is in whatever protocol the third party uses. Currently OrderRouter supports IQ, FIX, RASH, OUCH (2.1a and 4.0), OTTO, NOMAD and arcaDirect (2.0, 3.2, and 4.0).

[Content removed]

## Logging / Error Processing

---

Logs are written to a `\logs` subdirectory in the OrderRouter working directory. The general log contains diagnostic data or information not captured on the exchange or on the client application log.

General log:

- *orderrouter\_<yyyymmdd>\_<pid>.log* for Linux
- *orderrouter\_<r or d>.exe\_<yyyymmdd>\_<pid>.log* for Windows.

There are a number of other logs that are created based upon the configuration of the particular OrderRouter.

- *fixlog\_pid\_hhmmss\_#.log* contains all the messages OrderRouter has sent and received from the third party.
- *risk.log* (created one directory up, in the working directory of OrderRouter) contains the state information for the risk management component of the OrderRouter
- *drop.log* is used for persistence in the built-in OR drop feed. The drop feed is a separate port in OR that echoes order flow occurring within OR. Its format is: exchange order id, client order id, IQ status message

Tip For a full list and description of logs, go to the following URL on the Penguin Portal: [http://portal.p/index.php/General\\_Information\\_%26\\_Procedures](http://portal.p/index.php/General_Information_%26_Procedures)

[Content removed]

## Test Notes

---

[Content removed]

## Deployment / Installation / Configuration

---

- The configuration of OrderRouter is controlled by *OR\_Config.xml* ([http://portal.p/index.php/Configuration\\_File](http://portal.p/index.php/Configuration_File)), which must be present in the current working directory.
- OrderRouter may also be installed by hand on remote servers if necessary.

### Order of deployment:

- 1 Consolidated OrderRouter
- 2 Edge OrderRouter(s), if applicableStock OrderRouter

[Content removed]

## Proxy

---

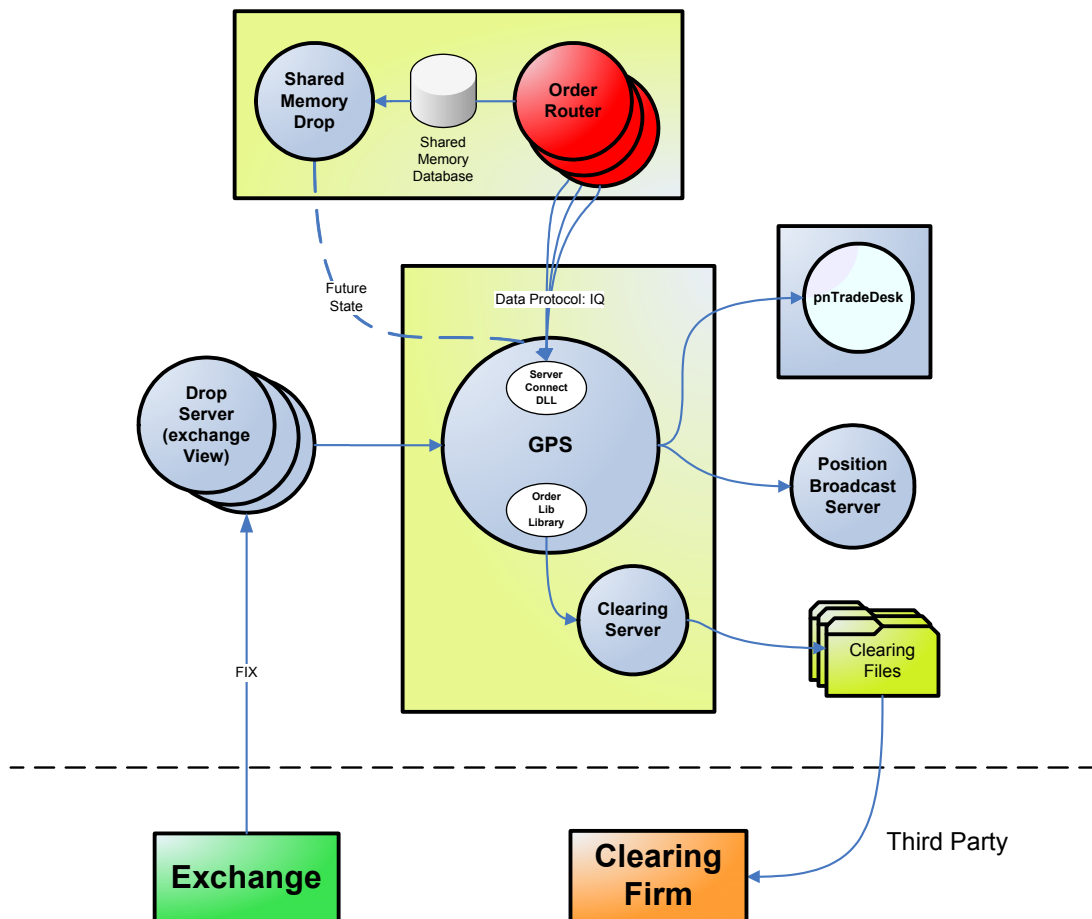
[Content Removed]

## Global Position Server (GPS)

Global Position Server (GPS) tracks the order status for all orders and provides a data feed of orders and order statuses to connected clients. It also compares order updates between the OrderRouter and drop server.

Discrepancies between the OrderRouter and drop server are reflected in TradeDesk.

**Figure 8.3** GPS Data Flows





[Content removed]

## Drop Server

---

[Content Removed]

## auth server

---

[Content Removed]

# CHAPTER 9

## FRONT END APPLICATIONS

The front end applications display information about an account's positions, orders, and option and stock quotes.

The **Penguin Front End** is used to monitor a position, view the market, and place orders.

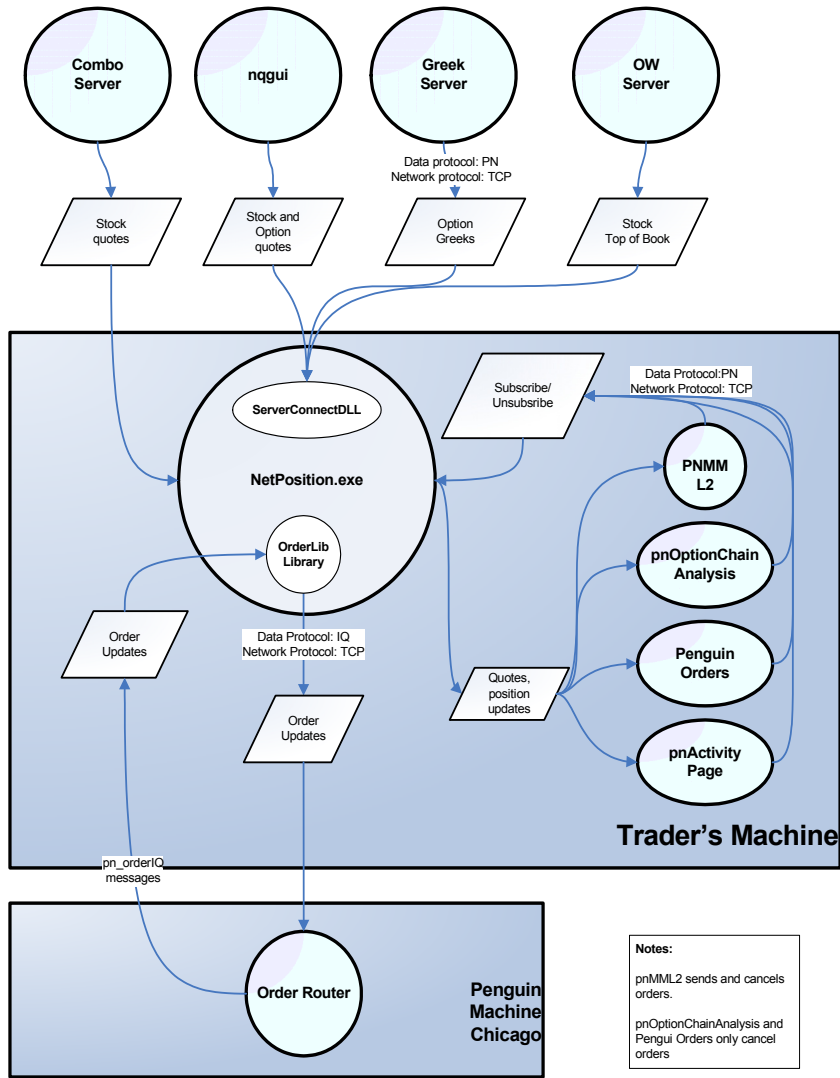
**TradeDesk** is used to watch order flow positions for *all* accounts and OrderRouters throughout the day.

# Penguin Front End

The Penguin Front End displays option quotes, stock quotes, and information about an account's positions/risks.

The Penguin Front End is comprised of several different applications (as shown in Figure 9.1) with NetPosition.exe serving as the main piece. A description of each application follows the diagram.

**Figure 9.1** Penguin Front End Architecture



# TradeDesk

TradeDesk allows users to see detailed order, position, and trade information for specified accounts and OrderRouters throughout the day (as opposed to the Penguin Front End which only displays one account at a time). It also keeps track of trading fees and profit and loss.

Although it cannot execute trades, TradeDesk serves as a back office risk management tool for troubleshooting, canceling, and monitoring orders.

**Figure 9.2** TradeDesk Front End

The screenshot displays the TradeDesk Front End interface. At the top, there is a navigation bar with tabs for 'Home', 'Save Layout', 'Status Bar', 'Error Orders', 'Live Orders', 'Positions', 'Exercised Options', 'Filled Orders', 'Order Panels', 'Position Panels', 'Exercise Options', and 'Filled Orders'. Below this is a table with columns for 'ALL ACCOUNTS', 'ALL USERS', 'ALL ROUTES', 'ALL SYMBOLS', 'ALL UNDERLYING', 'ALL INSTRUMENT', 'ALL ORS', and 'ALL ORDERS'. The table contains a large number of rows, each representing an order or position. The columns include 'OrderID', 'Time', 'Symbol', 'OrderStatus', 'Side', 'Type', 'Price', 'Shares', 'Filled', 'Last Price', 'Last Shares', 'Exchange', 'Subroute', 'Route', 'IntrMD', 'Orderrouter', 'Executing', 'Rejected', and 'Liquidity Flag'. The table is filtered to show orders for the account '19719466' and the symbol 'DOH1G1'. The orders are sorted by 'Time' and 'OrderStatus'. The table shows a variety of order types, including 'BUY', 'SELL', 'PLACED', 'PENDING CANCEL', and 'PENDING'. The 'OrderStatus' column shows various statuses such as 'PENDING CANCEL', 'PENDING', 'PLACED', 'EXECUTED', and 'REJECTED'. The 'Exchange' column shows various exchanges, including 'ARCAO', 'ARCAO', 'PSE', and 'ARCA\_DJR'. The 'Subroute' column shows various subroutes, including 'MER', 'MER', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'Route' column shows various routes, including 'MER', 'MER', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'IntrMD' column shows various intraday market data, including 'MER\_OR', 'MER\_OR', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'Orderrouter' column shows various order routers, including 'MER\_OR', 'MER\_OR', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'Executing' column shows various executing orders, including 'MER\_OR', 'MER\_OR', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'Rejected' column shows various rejected orders, including 'MER\_OR', 'MER\_OR', 'ASE', 'MER\_OR', and 'MER\_OR'. The 'Liquidity Flag' column shows various liquidity flags, including 'MER\_OR', 'MER\_OR', 'ASE', 'MER\_OR', and 'MER\_OR'.

Note For more on the front end, go to <http://wiki.s/index.php/pnTradeDesk>

[Content removed]

# CHAPTER 10

## OPTION MARKET MAKERS

Market making strategies place orders on the bid and offer and aim to make money on the bid/ask spread. These strategies add liquidity and rest on the market until they are filled.

### Option Market Maker Architecture

---

**Figure 10.1** Market Maker Data Flows

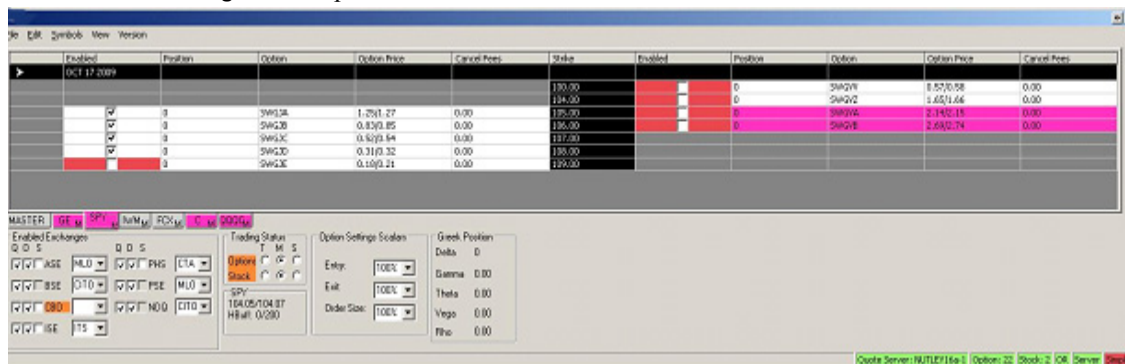
[Content removed]

## pnMM / pnMM Server

The pnMM client application is a market making program that consists of a front end (pnMM) and back end (pnMM Server).

pnMM is used to buy options at the bid price, sell at the ask price, and hedge any fills with the underlying stock. pnMM does this on many option symbols simultaneously by putting in orders and canceling them according to user-defined parameters.

**Figure 10.2** pnMM Front End



**Note** To see a larger view of the screenshots with a full description of front end functionality refer to <http://portal.p/index.php/pnMM>

pnMM has two different ways of trading options:

1. *Theo Symbol*, a traded based on option volume where the user sets entry and exit volumes

**OR**

2. *Vol Symbol*, a trade based on the option's volatility where the user sets the entry and exit bid/ask volatilities

The user can also set the order size and max levels for each exchange (per symbol). Each option symbol has a max position and min exchanges required. All of these conditions are set by the user through the edit screen in pnMM (shown in Figure 10.3 below).

**Figure 10.3** pnMM Edit Panel

**Edit**

☒ Theo Symbol ☐ Vol Symbol

Option Symbol Info

Symbol: SWGJC

Delta Hedge: 0

Exchanges Req: 5

Max Position: 250

Disable Shorts @ .05 ☒

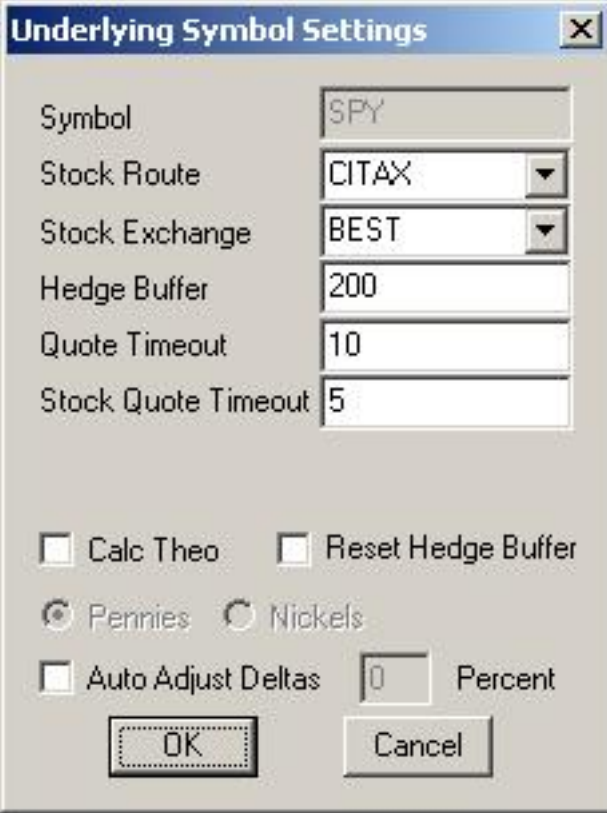
Reset to: Average Yester

Option Exchange Prefs

	Bid Ask	Base Order Size	Current Order Size	Max Levels	Base Entry Vol	Current Entry	Base Exit Vol	Current Exit
<input checked="" type="checkbox"/> ASE <input type="radio"/>	45	45	1	1800	1800	1050	1050	
<input checked="" type="checkbox"/> BSE <input checked="" type="checkbox"/>	30	30	1	2400	2400	1100	1100	
<input checked="" type="checkbox"/> CBO <input type="radio"/>	50	50	5	1800	1800	750	750	
<input checked="" type="checkbox"/> ISE <input type="radio"/>	55	55	5	2800	2800	1200	1200	
<input checked="" type="checkbox"/> PHS <input type="radio"/>	57	57	6	1900	1900	700	700	
<input checked="" type="checkbox"/> PSE <input checked="" type="checkbox"/>	25	25	1	2400	2400	1100	1100	
<input checked="" type="checkbox"/> NDQ <input checked="" type="checkbox"/>	25	25	2	2400	2400	1000	1000	

OK Cancel

The user can choose to hedge option orders automatically through pnMM with stock orders using the Delta Hedge in the Edit panel (Figure 10.3) and the Underlying Symbol Settings panel (Figure 10.4).

**Figure 10.4** Underlying Symbol Settings

The image shows a Windows-style dialog box titled "Underlying Symbol Settings". It contains several input fields and checkboxes. The "Symbol" field is a text box containing "SPY". The "Stock Route" and "Stock Exchange" fields are dropdown menus showing "CITAX" and "BEST" respectively. The "Hedge Buffer" field is a text box containing "200". The "Quote Timeout" field is a text box containing "10". The "Stock Quote Timeout" field is a text box containing "5". Below these fields are two checkboxes: "Calc Theo" and "Reset Hedge Buffer", both of which are unchecked. There are also two radio buttons: "Pennies" (which is selected) and "Nickels". At the bottom, there is a checkbox for "Auto Adjust Deltas" (unchecked) next to a text box containing "0" and the word "Percent". At the very bottom are "OK" and "Cancel" buttons.

Symbol	SPY
Stock Route	CITAX
Stock Exchange	BEST
Hedge Buffer	200
Quote Timeout	10
Stock Quote Timeout	5

☐ Calc Theo    ☐ Reset Hedge Buffer

☒ Pennies    ☐ Nickels

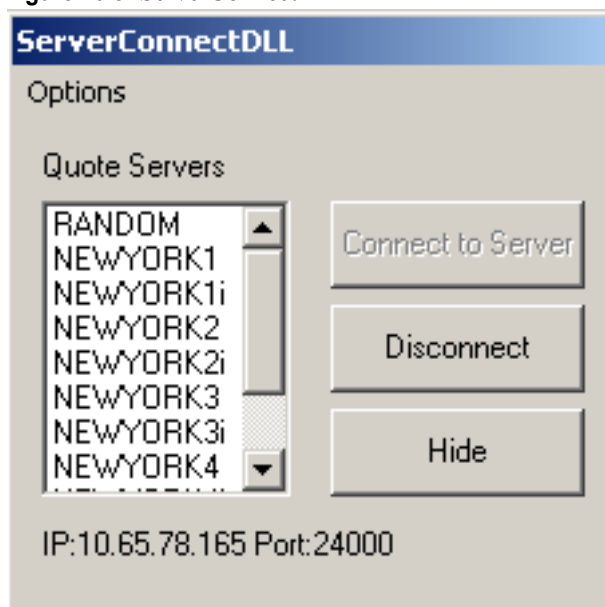
☐ Auto Adjust Deltas    0    Percent


OK    Cancel



The pnMM Server automatically assigns a server to which users connect, but users also have the ability to choose servers via the ServerConnectDLL window (see Figure 10.5).

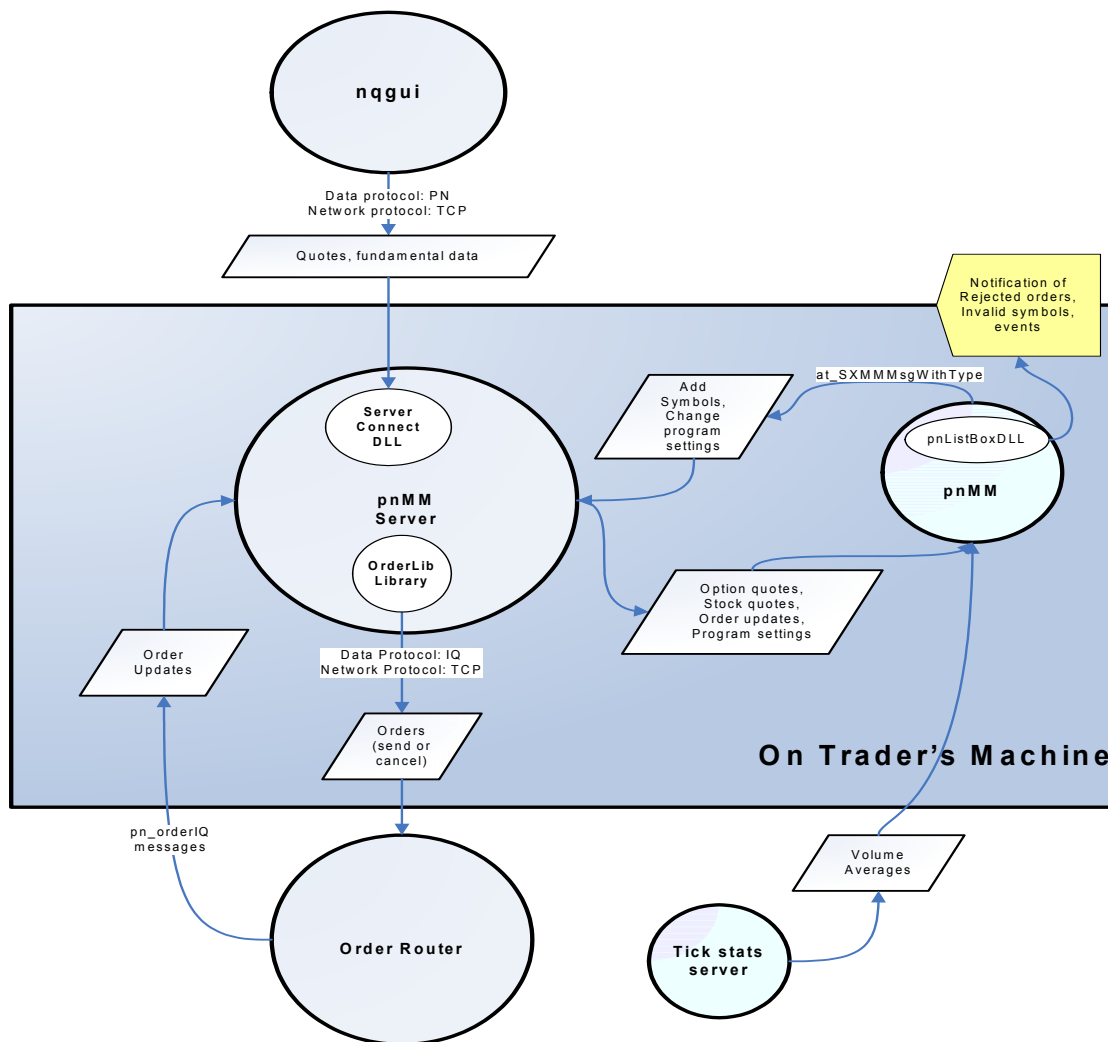
**Figure 10.5** ServerConnectDLL



The icon for pnMMServer:  is displayed in the lower right hand corner of the trader's screen. This icon changes based on the connection status of the server to the OrderRouter and the quote server. If pnMMServer is connected to the quote server and receiving quotes, this icon is green. If quotes are disconnected, the icon is red. If the OrderRouter is down, the icon changes to a blinking red.

## Source / Target Information

**Figure 10.6** pnMM Data Flows



## Logging / Error Processing

---

[Content removed]

## Configuration / Deployment / Installation

---

[Content removed]

**Table 10.1** pnMM Server Command Line Arguments

Command Line Argument	Definition	Default
-otServerIP	The IP address of the machine running OrderRouter	localhost
-otServerPort	The port the OrderRouter is running on	21900
-Account	Account number of the IQ/Liquid account pnMM is using	
-Domain	the domain of the IQ/Liquid account pnMM is using	
-user	The username of the IQ/Liquid account pnMM is using	
-clientPort	The port on which pnMM front end clients connect to pnMMServer. The default is 24024	24024
-logQuotes	Switch added to the command line for pnMMServer to write every quote it gets to the log file.	
-nqip	The IP address of the quote server pnMMServer connects to.	Q.s (localhost if UseLocalQuotes is present)
-nqport	The port of the quote server pnMMServer connects to.	24000 (37037 if UseLocalQuotes is present)
-ORUser	The username the app logs into the OR with.	USER_API
-DefaultOptionpnRoute	The pnRoute that is used in lieu of "DEFAULT".	TOS
-enableQSMon	This switch tells ServerConnect whether or not to connect to the QSMon (quote server monitor).	

**Table 10.1** pnMM Server Command Line Arguments

Command Line Argument	Definition	Default
-EnableNDQCancelReplace	This switch tells the app whether or not it can send cancel/replace orders on the NASDAQ (this only applies for pnMMs using the OTTO connection).	
-RemoteServerConnectPort	Port to listen on for ServerConnect remote clients.	25025
-UseLocalQuotes	If this switch is present, ServerConnect will connect to pnQuotes, rather than nq.	
-UseMaxLevelsAsMin	If this switch is present, pnMM will put orders out behind the market and keep us out on X price levels at all times. It uses the max levels parameter to determine how many price levels out we should be on a given exchange per symbol.	
-wxWebPubSubPort	Port of wxWeb client to connect to	46046

[Content removed]

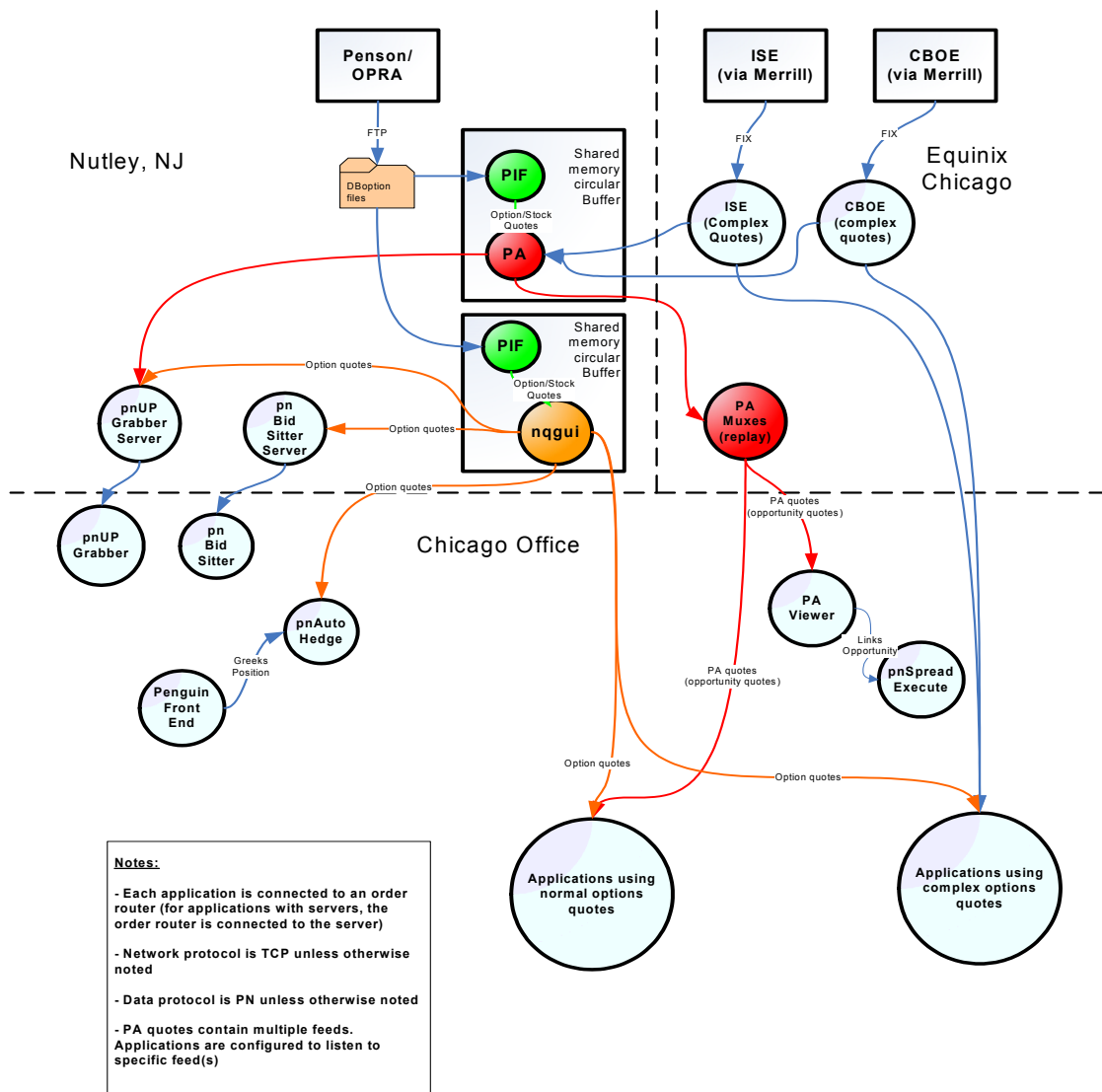
# CHAPTER 11

## OPTION MARKET TAKERS

Market taking strategies remove liquidity and do not rest on the market. They lock in edge by buying (or selling) mispriced options.

# Architecture for Market Taker and Arbitrage Apps

Figure 11.1 Market Taker and Arbitrage Data Flows



[Remaining contents of chapter removed]

# CHAPTER 12

## OPTION ARBITRAGE APPLICATIONS

[Chapter Removed]

# CHAPTER 13

## **STOCK CLIENTS**

[Chapter removed]



# CHAPTER 14

## IMPORTANT FILES

The following files comprise fundamental data used by quote servers, OrderRouters, and various web-based applications. If Penguin doesn't have these files, Penguin can't trade.

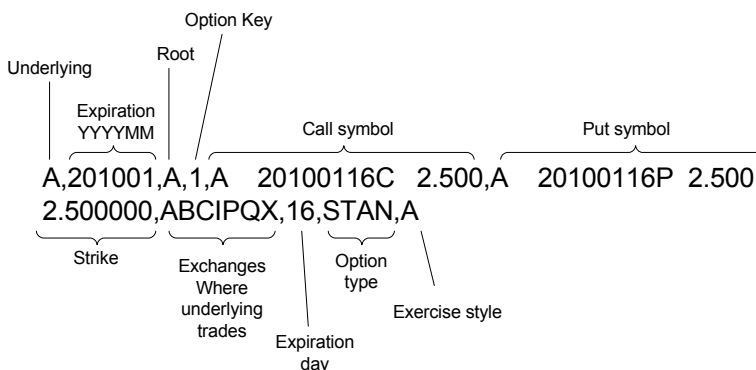
## db\_option.out, db\_option\_key.out

*db\_option.out* and *db\_option\_key.out* are used to figure out the underlying strike, expiration, put/call, exchange, and deliverables for a given option symbol.

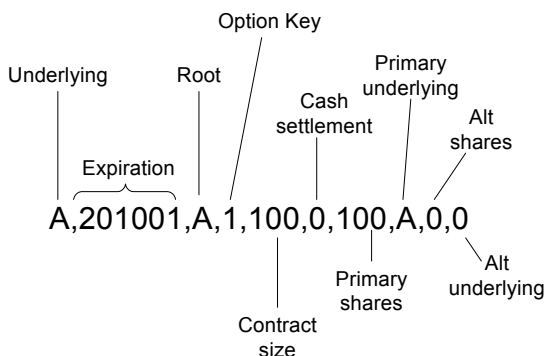
These .csv output files are generated each morning using input files received from Pension, the OCC, ThinkOrSwim, or SFB. Typically, the Pension input files are used as the source file, but occasionally there are problems for which Pension needs to be called. When this happens, Penguin uses the backup files generated from the OCC or ThinkOrSwim data.

Although the input files are in different formats, the CSV output formats for *db\_option.out* and *db\_option\_key.out* are as follows:

**Figure 14.1** dp\_option.out file format



**Figure 14.2** .db\_option\_key.out file format



## **option\_openinterest.txt**

---

[Content removed]

## **company\_names.txt**

---

[Content removed]

## **stock\_listed\_exchange.txt**

---

[Content removed]

# CHAPTER 15

## REAL TIME OPTION TRADE STATISTICS

Traders use the Option Trading Page to find specific patterns of trades, especially those occurring frequently or likely to occur.

### Option Trading Page

---

[Content removed]

### Job Sequence

---

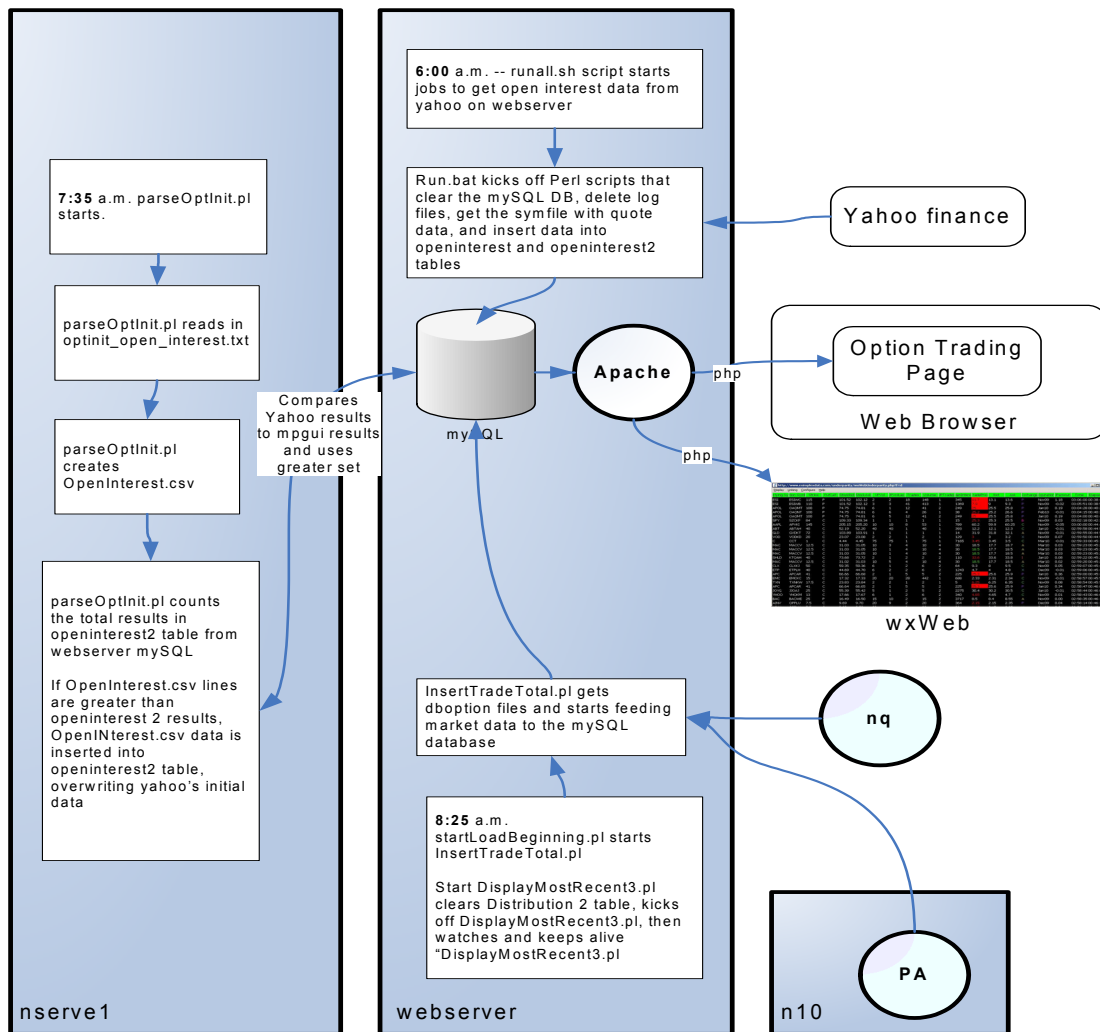
Table 15.1, “Job Sequence” lists the jobs that execute the process of populating the Options Trading page.

**Table 15.1** Job Sequence

Job #	Job Name / Script	Time	Description
1	[Content removed]	6:00 a.m.	[Content removed]
2	[Content removed]	6:00 a.m.	[Content removed]
3	[Content removed]	7:35 a.m.	[Content removed]
4	[Content removed]	8:15 a.m.	[Content removed]
5	[Content removed]	6:00 p.m.	[Content removed]

## Source / Target Data

Figure 15.1 Option Trading Load Perl



# CHAPTER 16

## **MONITORING**

[Chapter removed]

# CHAPTER 17

## REPORTING

[Chapter removed]



## PROCESS / DATA FLOWS

[Appendix Removed]



## DAILY TIME LINE

[Appendix removed]

## DATA PROTOCOLS

[Appendix removed]

# GLOSSARY

**Account** Unique account number tied to an amount of capital (an account number could represent one trader or a group of traders).

**Arbitrage** The practice of taking advantage of a price differential between two or more markets: striking a combination of matching deals that capitalize upon the imbalance, the profit being the difference between the market prices.

**Arbs** arbitrage opportunities.

**Ask Price** The price at which market participants are willing to sell.

**BBO** best bid and offer, the highest bid and the lowest offer for a given exchange.

**Bid price** The price at which market participants are willing to buy.

**Blackbox** An independent trader/connection that uses Penguin architecture to execute trades.

**Butterfly** Option position made up of three options on the same underlying stock, expiration date, and put/call where the strikes are equal distance apart. The ratio is 1:(2):1 for a “long” butterfly and (1):2:(1) for a “short” butterfly.

**Clearing files** The settlement of accounts or exchange of financial instruments especially between banks.

**Clearing firm** An organization which works with exchanges to handle confirmation delivery and settlement of transactions. Such corporations play a key role in ensuring that executed trades are settled within a specified period of time and in an efficient manner (also called clearing corporation or clearing house).

**Client** An application a trader uses to execute a trade.

**Colocation** (or Colo) Type of data center where multiple customers locate network,

server, and storage gear and interconnect to a variety of telecommunications and other network service provider(s) with a minimum of cost and complexity.

**dboption files** a term used at Penguin (and throughout this document) to mean all important files except *EasytoBorrow.txt*, *locate.txt*, and *unique-cp-YYYYMMDD.log*. Keep this in mind when you're looking at data flow diagrams in particular.

**Delta** Amount by which an option price changes given a small change in the underlying price.

**Demultiplexer** Takes a single data stream and breaks it out into multiple data streams. For example, pnLauncher is a demultiplexer. It takes a single stream and breaks out a data feed that has inversion and locked spreads and another feed that has under parity spreads.

**Edge** Amount on either side of the theoretical midpoint between the bid price and ask price.

**Equinix** Provider of network-neutral data center and interconnection services, offering colocation, traffic exchange, and outsourced IT infrastructure solutions.

**Equity option** An option in which the underlier is the common stock of a corporation, giving the holder the right to buy or sell its stock at a specified price by a specific date (also called stock option).

**Exchange** A center where securities or commodities are bought and sold.

**Exchange member** An individual or party who pays a high fixed cost for the privilege of buying and selling a particular stock or commodity on behalf of investors.

**Execution** The process of filling an order to buy or sell stock or options.

**Foreign exchange** The exchange of foreign currency. On the foreign exchange market, foreign currency is bought and sold for immediate (spot) or forward delivery.

**Free spreads** Spreads in which the sum of the current market prices of the legs is zero. This allows the free spread traders to execute spreads for zero up-front cost with very little downside risk. The spreads will also have a chance of increasing in value as the expiration draws nearer or as the underlying stock moves.

**Futures** Commodities or stocks bought or sold upon agreement of delivery in time to come.

**gcc** Compiler on Linux.

**Good fill** Buying (selling) below theoretical value.

**Greeks** The mathematical characteristics of the Black-Scholes model are named after the greek letters used to represent them in equations. These are known as the option greeks. The five option greeks measure the sensitivity of the price of stock options in relation to four different factors: changes in the underlying stock price, interest rate, volatility, and time decay.

**Hedge** A purchase or sale of a financial product, having as its purpose the elimination of loss arising from price fluctuations.

**Hedge fund** A flexible investment company for a small number of large investors (usually the minimum investment is \$1 million); can use high-risk techniques (not allowed for mutual funds) such as short-selling and heavy leveraging.

**Hidden orders** (or hidden liquidity) The bid and ask prices with a narrower spread than what is visible to the trader.

**Hit the bid/offer** Submit a buy (sell) order that matches the current market's best offer (bid).

**Independent traders** Traders who rely on their own relationships with brokers and clearing firms. Independent traders borrow money and technology to execute trades. They use space at Penguin's 550 W. Washington location.

**Instaquote** Third-party tool to execute sophisticated trading strategies, with an emphasis on fast access to quotes, detailed charting capabilities, and the flexibility to trade any security on any exchange.

**Inverted market** Inverted markets are when one exchange has a bid price that is higher than the ask price at another exchange. This represents an arbitrage opportunity by buying at the ask price and selling at the bid price for a profit. On a single exchange this is not possible and is an easy way to check the quality of quotes. If a single exchange is inverted then either the exchange is having problems or there is a problem in processing the data.

**Join the bid/offer** Put in a buy (sell) order that matches the current market's best bid (offer).

**Limit orders** Orders with trader-specified constraints (a specific quantity and/or price).

**Linux** Any Unix-like computer operating system; file names used in code are case sensitive.

**Liquid** See "Thinkpipes"

**Locked market** A short-term situation occurring within a market where both the bid and ask are identical, resulting in no bid-ask spread.

**Long call** Buying the right to purchase the stock (a call option) at a fixed price (exercise price) with the expectation that the stock's price will increase above the exercise price by more than the premium paid.

**Long put** Buying the right to sell a stock at a fixed price (a put option) with the expectation that the stock's price will decrease below the exercise price by more than the premium paid.

**Market Maker** (1) A firm who quotes both a buy and a sell price in a financial instrument or commodity hoping to make a profit on the turn or the bid/offer spread. (2) A Penguin client/application designed to achieve such an effect.

**NASDAQ** An American stock exchange; the largest electronic screen-based equity securities trading market in the United States. With approximately 3,200 companies, it lists more companies and has more trading volume per day than any other stock exchange in the world.

**NBBO** National Best Bid and Offer, the highest bid and the lowest offer aggregated across exchanges.

**Offer price** *See* Ask price.

**One lot** An order for one option contract.

**Open interest** The number of options contracts that have not been exercised.

**Option** A right granted by a corporation to officers or employees as a form of compensation that allows purchase of corporate stock at a fixed price usually within a specified period.

**Option chain** A display showing all the options of all the expiration dates and strike prices for an underlying security. This can be seen on the Penguin Front End via the Option Analysis window.

**Options** Financial instruments that convey the right, but not the obligation, to engage in a future transaction on some underlying security, or in a futures contract. In other words, the holder does not have to exercise this right, unlike a forward or future.

**OrderRouter** A piece of software, a gateway, that provides a connection between a client and a broker or exchange.

**Penston** Penguin's clearing firm.

**PHP** PHP is a widely-used general purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge.

**Picked off** Having orders in the market and getting executed but having the market move before the hedge or arbitrage orders can be executed.

**Puke order** Unwanted position that a trader wants to get rid of as soon as possible.

**quote** Listing of representative bid prices and ask prices for a specific stock traded on a specific exchange during a particular trading day. Stock prices are quoted in percentage points and in increments of an eighth of a point. In the U.S. each point equals one dollar, and each increment equals 12.5 cents.

**Rack** A cabinet that houses PCs.

**Radianz** Colocation provider in New Jersey where servers are housed.

**Removing liquidity** When a new order is executed against pre-existing orders, the new order is said to have removed liquidity.

**Run over** Having a buy (sell) order get filled and have the bid (ask) price drop down at least one tick.

**S&P 500** A stock market index containing the stocks of 500 large cap corporations, most of which are American. The index is the most notable of the many indexes owned and maintained by Standard & Poor's, a division of McGraw-Hill. S&P 500 is used in reference not only to the index but also the 500 companies that have their common stock included in the index.

**Savvis** A company that provides data centers and technology infrastructure for enterprise applications.

**Short call** Selling a call option (selling the stock short) with the expectation that the stock price will decrease. If the stock price decreases, the short call position will make a profit in the amount of the premium. If the stock price increases over the exercise price by more than the amount of the premium, the short will lose money.

**Short put** Selling a put option with the expectation that the stock price will increase. If the stock price at expiration is above the exercise price, the short put position will make a profit in the amount of the premium. If the stock price at expiration is below the exercise price by more than the amount of the premium, the trader will lose money.

**SIAC** Provides systems to support the current and future business needs of the NYSE, Amex, NSCC, and their subsidiaries.

**Spread** The difference between the price available for an immediate sale (bid) and an immediate purchase (ask).

**Spread orders** (or complex orders) orders that are grouped together.

**Stock** An instrument that signifies an ownership position (called equity) in a corporation, and represents a claim on its proportional share in the corporation's assets and profits.

**Subversion** The source control system on `svn://182.167.70.147/trunk`.

**pnProject** A communication tool used to manage work-in-progress and other development and testing issues.

**TCP/IP** Packet-based transmissions. Internet protocol suite (commonly TCP/IP) is the set of communications protocols that implement the protocol stack on which the internet and most commercial networks run. It is named for two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

**Theoretical Value** The quantifiable fair price for a security. Buying below or selling above theoretical value has positive edge.

**ThinkOrSwim (TOS)** Penguin's broker (ThinkOrSwim is owned by TD Ameritrade)

**Underlying** The price or rate of an asset or liability, but not the asset or liability itself.

**Under parity options** An option is under parity if you can buy the call (put) and sell (buy) the stock and exercise the option all for a profit.

# INDEX

## A

API 29  
auth server 44

## B

business recovery requirements 15  
business rules 9

## C

CBOE, *see* option quote servers  
combo server 38  
communication 19  
cruise control 24  
*see also* nightly build process

## D

data back-up 14  
data flows  
    arbitrage 57  
    market maker 48  
    penguin front end 46  
database management system, *see* software requirements  
debugging 23, 25  
deployment, linux 26  
developer's workflow 17  
development tools, *see* software requirements

## E

Exchange 22  
exchange simulators 22  
*see also* testing

## F

front end 5, 45–47  
TradeDesk 47

## G

GPS 43  
grepbook 24

## H

Hardware 15  
hardware requirements 15

## I

ifp 32  
*see also* option quote servers  
ISE spreadbook  
*see also* option quote servers

## L

linux deployment kit 26

## M

market maker  
    data flows 48  
    sxMM 49  
market taker  
    data flows 57  
middleware/messaging software, *see* software requirements  
mp  
*see also* option quote servers

## N

nagios 32

## O

option quote server

history 5

option quote servers

CBOE 36

IFP 31–32

ifp 31

ISE spreadbook 36

mp 32

option trading page 63–64

optiontest 22

*see also* testing

OR, *see* OrderRouter

order server 5

OrderRouter 5, 24, 41–42, 52, 54

OrderRouter testing 24

## P

penguin front end 46

penon 8

clearing files 61

programming language, *see* software requirements

proxy 42

## Q

QA server 24

quote servers 30

## R

real time option trade statistics 63

router recovery 14

## S

SDLC 16–18, 22–23, 25

simulators 22

Software 16

software development life cycle, *see* SDLC

software installation and management

linux deployment kit 26

sxinstall

software requirements 15

Source 25

stock quote replay 22

*see also* testing

stocktest 22

*see also* testing

svn 14, 17

*see also* developer's workflow

SXA 34

sxDeveloper 19

sxinstall 25

*see also* software installation and management

sxMM 49

sxPerfMon 25

sxProduction 17, 19

sxProject 18, 24

*see also* SDLC

sxReplayData 35

sxreplaydata 22, 35

system availability, *see* technical requirements

system up-time, *see* technical requirements

## T

technical requirements 14

testing 22

*see also* SDLC



## U

underlying 50

## W

wiki, *see* penguin portal  
wxWeb 55