



Nath Corp
Building Applications for Tomorrow, Today

2010

QUALITY ASSURANCE EXPERTISE IN THE FINANCIAL DOMAIN



Microsoft
GOLD CERTIFIED
Partner

SA
NATHCORP
1/16/2010

INDEX

- About us
- Our Expertise - Financial Domain
- Stock Exchanges with Options Trading
- Financial Protocols - FIX Protocol
 - Processes carried out on FIX files
- Binary Protocols - ARCA , ARCA DIRECT
- Windows Automation Tools
- Linux Automation using Perl
 - GUI Automation using Perl
- Languages like PERL ,PHP ,Expect , HTML, CSS
- CruiseControl
- Conclusion

ABOUT US

NathCorp is a boutique consulting company offering services in two areas:

1. Financial systems Quality Assurance, nightly clearing files services, monitoring of production systems.
2. Rapid Windows 7 64-bit application migration and deployment.

We currently provide services to Wall Street high-frequency trading firm, and to Fortune companies in the areas of Windows 7 application migration and deployment.

The QA team comprises of software professionals with a total of more than fifty man years of experience. We have a team of global standard associates who are equipped to test critical financial software used in stock exchange as well as Futures and Options trading. The team includes financial experts who ensure that the software passes rigorous regression and performance tests to ensure top quality results while in production.

The testing services include following types of applications / products:

- Windows Migration Applications
- Web based applications
- Linux based client-server applications

We specialize in the types of tests given below:

- Linux / Windows Usability testing
- Financial Domain Applications testing
- Automation Testing using **Mercury Quick Test Pro**
- Automation Testing using **Perl / Shell scripting**
- Stress Testing using **Mercury Load Runner**
- Performance Testing using Load Runner and Perl Scripts
- Windows Comparison testing – For various platforms
- Windows / Linux End-to-end testing

This document would further show in detail how we can effectively test critical financial software with the latest technology.

OUR EXPERTISE – FINANCIAL DOMAIN

NathCorp's QA team is equipped to handle the all kinds of Software testing but we specialize in 2 areas.

- Windows Application Migration Testing
- Financial Domains Applications Testing

This document would focus on our expertise in the financial domain.

"Finance is the science of funds management. The general areas of finance are business finance, personal finance, and public finance. Finance includes saving money and often includes lending money. The field of finance deals with the concepts of time, money and risk and how they are interrelated. It also deals with how money is spent and budgeted."

The above is how Wikipedia describes finance. In NathCorp however we define finance simply as "money". Any software which is expected to work coherently in the complex world of finance should be robust and error- free. This is because every single error costs money. And not only to the company who created the software but those who buy it and those who use it. So we essentially examine any financial software stripped down to it skeletal system testing each bit and piece of it.

We specialize in the trading areas of financial domain:

- Stock Exchanges along with Options trading
- Testing with the Financial protocol – FIX
- Testing with binary protocols like those used in ARCA 2.0, 3.2, OTTO
- Testing the connections between client and Exchanges

STOCK EXCHANGES WITH OPTIONS TRADING

A stock exchange is an entity which provides "trading" facilities for stock brokers and traders, to trade stocks and other securities. Stock exchanges also provide facilities for the issue and redemption of securities as well as other financial instruments and capital events including the payment of income and dividends. The securities traded on a stock exchange include: shares issued by companies, unit trusts, derivatives, pooled investment products and bonds.

Any software which wants to trade with the stock exchange needs to conform to many regulations. The direct Electronic feed sent by the stock exchanges needs to be read accurately and responded according to strict regulations. The way to "talk" to these exchanges is using their "language" or protocols. Most common protocol is FIX and there are several other protocols which are binary like those used in ARCA. Most of the trading

which happens has some fixed sequence and to better understand the processes listed below, a small glossary of terms is given below.

Sequence:

1. Trader views current market prices and arbitrage opportunities on the LIVE feed screen
2. Trader sends in the trade which gets bypassed through the client application
3. The Client application forwards trade to the Exchange for processing.

In this sequence, there are many messages which are exchanged between the Trader and the exchange. The basic types of messages are given below

- New Order to the Exchange
- Acknowledgement from the Exchange
- Cancel Order from the trader
- Execution Report from the Exchange
 - Filled Report – Trade got filled
 - Partial Filled Report – Trade got partially filled
 - Cancel Report – Trade got cancelled as requested
- Heartbeat to ensure connection is LIVE and valid
- Login/Logout messages

These messages are standard message types and vary slightly depending on different protocols.

The details for these protocols are provided below.

FINANCIAL PROTOCOLS

FIX – FINANCIAL INFORMATION EXCHANGE

The **Financial Information eXchange (FIX) Protocol** is a messaging standard developed specifically for the real-time electronic exchange of securities transactions. The Financial Information eXchange ("FIX") Protocol is a series of messaging specifications for the electronic communication of trade-related messages. It has been developed through the collaboration of banks, broker-dealers, exchanges, industry utilities and associations, institutional investors, and information technology providers from around the world. This is the major protocol used in most of the financial institutions. It is defined by a series of numbers followed by "=" and another number. There is a specific sequence of numbers and is defined in the FIXIMATE manuals.

NathCorp specializes in dealing with FIX protocols with many tools and utilities developed specifically to read the FIX feed and translate it. A sample of FIX feed is given below.

8=3209=23635=849=EDGE56=liquid34=452=20091106-
09:10:401=102320086=011=414=017=220=031=032=037=1000138=5039=040
=244=147=055=ABC58=NEW ORDER

As can be seen from the above it is difficult to simply read in all the FIX protocol files and interpret them. It takes expertise to understand what each message means and what information has been transmitted through these messages. NathCorp handles and processes many such files on a daily basis.

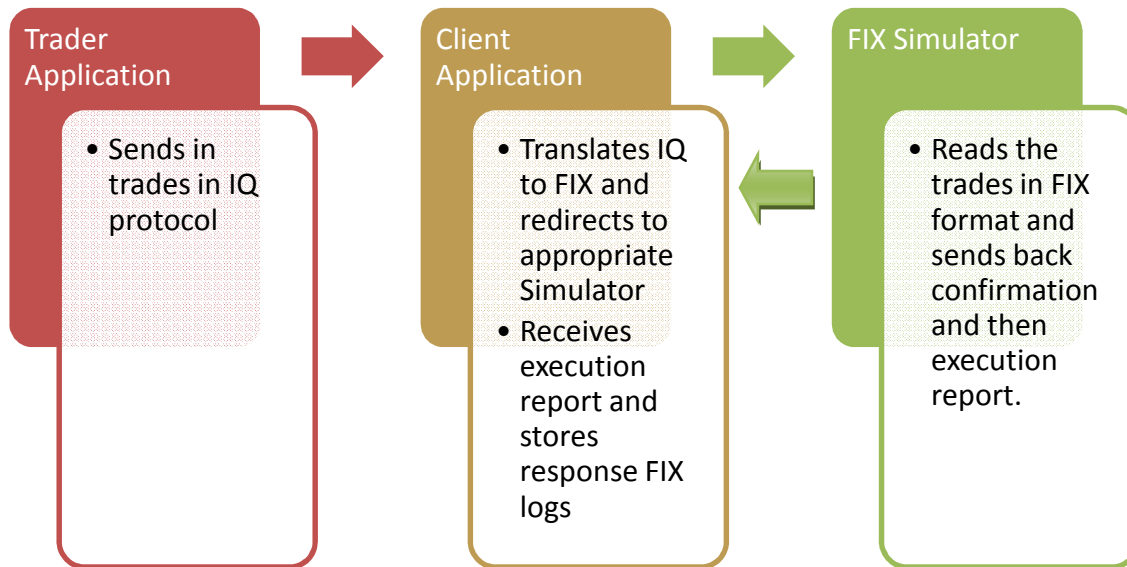
BASIC TYPES OF FIX MESSAGES

- Trade Messages
 - * Execution Report
 - * Order Cancel Reject
 - * Order Mass Status Request
 - * New Order Single
 - * Order Cancel Request
 - * Order Cancel Replace Request
 - * Order Status Request
 - * Dont Know Trade D K
 - * Order Mass Cancel Report
 - * Order Mass Cancel Request

TEST PROCESSES CARRIED OUT ON FIX PROTOCOLS

The basic processing that NathCorp needs to do on FIX protocol is to test the message validity and accuracy. We have to test a client application which receives feeds from ECN's which send FIX – type messages. In order to accurately carry out this testing, we need a FIX –generating application. So NathCorp has developed multiple “Simulators” which generate FIX messages whenever the client application connects to it. Since these simulators need to be feed specific we have different modes of FIX which can be generated from the same simulator like BATS, NEXA and EDGE. These modes are nothing but different types on ECN network which send out FIX feed from the exchanges with slight modifications.

Also, we need a valid trader application which can connect to the client application to send in trades which can be forwarded to the simulator for trading. A diagram describes this process.



In the above outlined process, we use many tools and utilities which are written in scripting languages to help process the files.

The Trader application has also been developed at NathCorp and creates the Trades in IQ protocol which is another popular protocol to transmit trade messages. The client application reads these messages and converts them to FIX and routes it to the right ECN (simulator) depending on the information in the trade. The Simulator then reads the FIX files, and executes them based on the conditions.

The basic input / output messages by the simulator are given below.

```
in: 8=FIX.4.29=9835=D49=liquid56=EDGE34=252=20100108-
06:56:321=112611=3121=138=2040=244=254=555=C77=E10=063
```

```
[1, [REDACTED], 93, 01:56:32
550547625543736.0]nUsed before read : 0
```

```
[1, d[REDACTED], 93, 01:56:32
550547628989780.0]out: 8=FIX.4.29=22635=849=EDGE56=liquid34=352=20100108-
06:56:321=11266=011=40H0003-14=017=220=031=032=037=1000138=2039=641=3147=054=255=C58=ORDER
PENDING60=076=0110=0126=0150=6198=0210=0375=0382=0437=0438=09882=010=216
```

```
[1, d[REDACTED], 93, 01:56:32
550547634161436.0]out: 8=FIX.4.29=22835=849=EDGE56=liquid34=452=20100108-
06:56:321=11266=011=40H0003-14=017=320=031=032=037=1000238=2039=441=3147=054=255=C58=ORDER
CANCELLED60=076=0110=0126=0150=4198=0210=0375=0382=0437=0438=09882=010=079
```

After these messages are generated, they are stored in a log and our Perl tools read them as input and provide an output stating if the FIX messages are valid or not and if they have right trade information or not. A sample output snapshot is given below.

```
***** END *****
3=FIX.4.209=67035=A049=EDGE056=1iquid034=1052=20100108-06:56:25098=00108=600141=Y010=0040
***** Direct Edge output *****
Header and Trailer          =Sequence OK
Message Type                =Logon
Header Tags                 =All Present
Body Tags                   =All Present
Trailer Tags                =All Present
Body Length Matched:       =67,67
Checksum                    =004,004
Checksum                    =OK,
```

This perl file breaks down each message to each field so the QA team can know for sure the whole test process worked out fine.

OTHER PROCESSES CARRIED OUT ON FIX PROTOCOLS

There are two basic processes used on these files.

1. Daily Loading of CSV / LOG files into DB to be used for Webpage querying
2. Conversion of XML to Test cases to be used to test Client execution servers using realistic data.

WEBPAGE QUERYING:

In this process, we pick up the files from the servers, run scripts on them to load them in to the DB and update the DB daily. This creates a history of trade execution information which is later used for querying using a webpage. The stored information is very helpful to trace past trades.

- The input files in this process is the CSV file and the drop_data*.log files.
- There is a SQL DB created on Linux boxes and shell scripts are used to load these CSV / LOG files into the DB separating each field and filling it under the various records in the DB. The script screen shot is given below:

```

#!/usr/bin/perl
use strict;
use DBI;
use DBD:mysql;

my $dbh = DBI->connect( "dbi:mysql:bspensondrop:127.0.0.1:3306", "root", "simplex" );

# Argument

my $today;
if( "$ARGV" =~ /--date/)
{
# Trimming

    sub rtrim($);
    my $year2= "$ARGV[1]\n";
    substr( $year2 ,4, 0 ) = "_";
    substr( $year2 ,7, 0 ) = "_";
    my $tdate="$year2";
}

```

```

my $tablecreate = "DROP TABLE IF EXISTS $today";
my $sth = $dbh->prepare($tablecreate);
#$sth->execute();

$tablecreate = "CREATE TABLE IF NOT EXISTS ` $today` (
`id` int(11) NOT NULL auto_increment,
`orderType` varchar(15) default NULL,
`orderQty` int(11) default NULL,
`price` float default NULL,
`ClOrdID` varchar(30) default NULL,
`timestamp` datetime default NULL,
`side` varchar(10) default NULL,
`OSymbol` varchar(10) default NULL,
`USymbol` varchar(10) default NULL,
`account` varchar(11) default NULL,
`execBroker` varchar(11) default NULL,
`orderID` varchar(50) default NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1";

$sth = $dbh->prepare($tablecreate);
$sth->execute();

my $hUSymbol;

sub load
{
    my $FIN = shift;

    my $status = ();

    while(<$FIN>)
    {
        chomp $_;
        my $line = $_;

        my $mact:

```

These Perl scripts are used to split the CSV / Log files field by field and put into the DB.

The DB schema is given below:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
orderType	varchar(15)	YES		NULL	
orderQty	int(11)	YES		NULL	
price	float	YES		NULL	
ClOrdID	varchar(30)	YES		NULL	
timestamp	datetime	YES		NULL	
side	varchar(10)	YES		NULL	
OSymbol	varchar(10)	YES		NULL	
USymbol	varchar(10)	YES		NULL	
account	varchar(11)	YES		NULL	
execBroker	varchar(11)	YES		NULL	
orderID	varchar(50)	YES		NULL	

This DB is updated daily using daily production files. The DB is then connected to Orders Webpage using PHP. This webpage can be queried anytime to get information on past trades.

The webpage snapshot is given below.

SEARCH STOCK TRADING

CONNECT TO DATABASE: <input type="text" value="BSPENSONDROP"/>	USYMBOL: <input type="text"/>	<div style="background-color: #800000; color: white; padding: 2px; text-align: center; font-weight: bold;">CHOOSE COLUMNS TO BE DISPLAYED IN SEARCH RESULT</div> <input type="checkbox"/> ORDER TYPE <input type="checkbox"/> ORDER QTY <input type="checkbox"/> PRICE <input type="checkbox"/> CLORDID <input type="checkbox"/> DATE AND TIME <input type="checkbox"/> USYMBOL <input type="checkbox"/> OSYMBOL <input type="checkbox"/> ACCOUNT <input type="checkbox"/> EXEC BROKER <input type="checkbox"/> SIDE <input type="checkbox"/> ORDER ID
ACCOUNT: <input type="text"/>	OSYMBOL: <input type="text"/>	
ORDER ID: <input type="text"/>	EXEC BROKER: <input type="text"/>	
ORDER TYPE: <input type="text" value="SELECT ORDER TYPE"/>	SIDE: <input type="text"/>	
ORDER QTY. BETWEEN <input type="text"/> AND <input type="text"/>	NO OF ROWS TO BE DISPLAYED: <input type="text" value="100"/>	
PRICE BETWEEN <input type="text"/> AND <input type="text"/>	START DATE & TIME: <input type="text" value="10-22-2009 00:01"/>	
SORT BY: <input type="text" value="DATE AND TIME"/>	END DATE & TIME: <input type="text" value="10-22-2009 23:59"/>	
SORT ORDER: <input type="text" value="ASCENDING ORDER"/>		
<input type="button" value="Search"/>		

This enables users and clients to query the DB with customized queries on the information.

PROCESSING OF FILES TO CONVERT TO XML

NathCorp also uses a small process to create XML from FIX / CSV files. The files loaded in the first process are also used to create “REAL” test scenarios for near close testing of the Order executing servers.

- The files are FIX / CSV
- Script screenshot given below is used to read these files and create 1 XML with both.

```

open(PROD, "<${filename}") || die("could not open input file.");
open(INTERCSV, ">inter.csv") || die("could not open csv file.");
while(<PROD>)
{
    if ($read > 500) { last;}
    my($line) = $_;
    printf INTERCSV $line;
    $read= $read+1;
}
open(INTERLOG, ">inter_log.log") || die("could not open log file.");
foreach $LogFile (@files)
{
    if ($LogFile =~ m/(drop_data.*)/ ){ $FN = $LogFile;}
    print "\nthe Drop Log file which was opened: $FN";
    open(LOGFILE, "<$FN") or die "$! error trying to open file";
    while(<LOGFILE>)
    {
        if ($readlog > 500) { last;}
        my($line) = $_;
        printf INTERLOG $line;
        $readlog= $readlog+1;
    }
    close(INTERLOG);
}
$status = $csv_obj->parse_doc("inter.csv", {headings => 1});
sleep 4;
$csv_obj->print_xml("out.xml", {parent_tag => "order", file_tag => "ORDER_LIST"});
sleep 4;
open(ORDER, ">orderInfo.xml") || die("could not open xml file.");
open(IN, "<out.xml") || die("could not open xml file.");
while (<IN>)
{
    my($line) = $_;
    chomp ($line);
    if ( $line =~ m/\<order/>/ )
    {
        $cnt= $cnt+ 1;
    }
}

```

The XML is then sent as orders for Test Scenarios.

DIFFERENT ECN'S WHICH USE FIX PROTOCOLS

The FIX protocol is used by major ECN which are listed below. NathCorp is adept at handling all these ECN messages even with the slight modifications which each of these bring to the FIX protocol.

- BATS – (Better Alternative Trading System) using FIX feeds
- NEXA - Brokerage ECN providing feeds from Penson
- DirectEdge – ECN with all quotes from the ISE (International Stock Exchange)
- TOS – (Think Or Swim) Online Brokerage company specializing in Options

BINARY PROTOCOLS - ARCA , ARCA DIRECT

Apart from the non-binary protocols like FIX with different ECN's like BATS,NEXA etc, we also handle binary protocols which are of the type used in ARCA or ARCA DIRECT. Binary protocols have the advantage of faster transmission and low error rates and compared to non-binary protocols.

The processing done on these protocols is similar to the processing done on the FIX protocol with the only difference being our capability to convert the binary feeds to ASCII feeds and then test the applications. We have also created a simulator for the binary feeds which generates the binary feeds which is converted to ASCII in the client application and then we apply our Perl tools to read the output feed and interpret them. A screen shot from the ASCII converted feed.

```
IN: Msg=D, var=1, Len=76, Seq=1, ClientOID=4640000, PCSLinkID=0, OrderQty=2, Price=1, ExDest=0, PriceScale=0
OUT: Msg=a, Variant=1, Len=32, Seq=1, TransTime=1260775044, ClientOID=4640000, ArcaOID=1000, MsgTerm=
IN: Msg=F, Variant=1, Len=72, Seq=3, OrderID=0, ClientOrderID=4640000, StrikePrice=0, UnderQty=0, ExDest=1
OUT: Msg=6, Variant=1, Len=32, Seq=2, TransTime=1260775044, ClientOID=4640000, OrderID=0, MsgTerm=
```

We also test the following Binary ECN's.

- ARCA Direct - ArcaDirect is a message-based interface to accept equity and option orders for the NYSE Arca Equities & Options Exchange and to the new NYSE AMEX Options Exchange.
- OUCH - Nasdaq used protocol for trading
- OTTO – (OUCH to Trade Terminal) OUCH like protocol
- RASH – (Routing and Special Handling), is a digital communications protocol that allows customers of the NASDAQ (National Association of Securities Dealers Automated Quotations) to conduct business in the options market.

We are fully equipped to handle any data supplied in any of the protocols mentioned above.

WINDOWS AUTOMATION TOOLS

NathCorp QA team also has considerable expertise in test automation. Test Automation is not just about using tools but having realistic expectations of tool capabilities, thereby ensuring maximum performance. NathCorp can accelerate testing effort and implement tools effectively thereby achieve optimum ROI.

A critical aspect of any deployment of tools is to develop a realistic plan for implementation, establishing a business case that sets some expectations and structure around the project. Our test automation focus is to ensure stability of the program code environment that is critical to success. Script writing is very similar to software development life cycle hence, we give importance to develop a well-structured, easily maintainable and reusable scripts. Our QA team has deep expertise in building high quality testing harnesses and frameworks to make your project a success.

We have a rich experience in **Automation of**

- Smoke Tests/ Build Verification Tests
- Functional / Regression Test
- Acceptance Testing
- Testing across Multiple Environments

Our expertise in using popular test automation tools include –

- Microsoft Visual Studio Team Test (VSTT)
- Mercury Quick Test Professional (QTP)
- Using Open Source tools like–
 - WatiN / WatiR
 - Selenium with Ruby

PERFORMANCE TESTING TOOLS

NathCorp provide performance testing services for Web and Windows applications/services to troubleshoot and fine-tune the applications for

- Optimum Performance (response time)
- Scalability
- Reliability

The various services include –

- Load Testing
- Volume Testing
- Response Time Testing
- Stress Testing

- Transaction Through-put
- Performance Assessments
- Support services

Our QA expertise in using popular performance tools include –

- Loadrunner
- Grinder (Open Source)

NathCorp **Performance Testing Framework** covers

- Complete automation from Scheduling to Reporting
- Adds notion of Load Test, Performance Test, Stress Test, and Capacity Test suites
- Report multiple runs of the same Test suite and compare results

LINUX AUTOMATION

Automation in Linux Terminal screen is more difficult as compared to Windows. As every server is character based, we specialize in automating such applications using Perl language and Shell scripts. Below given screenshots show some shell scripts and some perl scripts.

```

$kill proxyheartbeat
cd /root/ORLinuxSetup/ORTestSetup/ProxyControl/
rm -rf logs
./proxyheartbeat -ip 127.0.0.1 -port 44000 &
1,0-1 Top

while (<LOGFILE>)
{
    #print "$cnt\n";
    my($line) = $_;
    chomp ($line);
    if ( $line =~ m/(@=FIX\.4\.*)/ )
    {
        #print "\nLINE:$line";
        $log_line = substr($line,index($line, " ") +1);
        print OUT "$log_line\n";
        #print "$log_line\n";
    }
}
40,3 5%

```

GUI AUTOMATION USING PERL

We also have considerable expertise in automating the GUI screens using GUI modules of the Perl script. Below given screen shot shows the window which was automated along with the script which did it.

CRUISECONTROL

The QA team also specializes in handling build control and integration open source software called CruiseControl.

Introduction to CruiseControl software

CruiseControl is free, open-source software, distributed under a BSD-style license. It is both a continuous integration tool and an extensible framework for creating a custom continuous build process. It includes, but is not limited to, plugins for email notification, Ant, and various source control tools. A web interface is provided to view the details of the current and previous builds. It allows one to perform a continuous integration of any software development process.

There are 2 main components of CruiseControl.

- **Build loop**

The build loop is designed to run as a daemon process, which periodically checks the revision control system for changes to the codebase, builds if necessary, and publishes a notice regarding the status of the software build.

- **Build reporting**

CruiseControl provides two ways of reporting build status. The first (classic) reporting is the reporting JSP and the second is the dashboard.

- ▲ Results JSP

The build reporting is designed to present the results of the CruiseControl build loop. It's based on a HTML report managed by a JSP page. The left side of the page tells us about whether CruiseControl is currently building your project, and provides links to the details of previous builds. The right side of the page presents the results of the build -- including compilation errors, test results and details about what files have changed since the last build.

- ▲ Dashboard

The dashboard is a powerful tool to help visualizing the project build statuses. Previous project build result is color-coded so that one can get an instant snapshot of how the projects are doing at the moment. Users can hover their mouse over the various icons to see the name and some information about the project. The 'Builds' tab of the dashboard shows all projects (color-coded) sorted by name providing some more information.

Getting started with Cruise Control

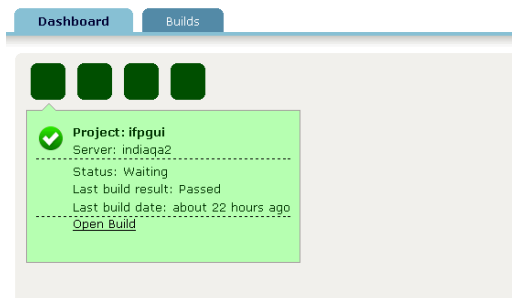
The process of continuous integration starts with SVN. Every time any developer checks in code to the SVN repository, the CruiseControl software will build the particular server in which change was detected. After checking in code in SVN, the developer can navigate to the following URL for web interface of CruiseControl.



This is called the CruiseControl Dashboard. The details for this page are given below.

On this page, previous project build result is color-coded so that the user can get an instant snapshot of how the projects are doing . The build results are overlaid with icons showing the current status of the project (for example paused, queued or building).

Dashboard Server : indiaqa2



When the user places the mouse over the project builds, represented by the colored squares, the user can see more information on that project, such as the project name, the server the project is running on, and when the last build occurred.

In general, the squares will either have green or red as the background color. A red background indicates that the last build failed, while a green background indicates that it passed. The longer the build has been in its current state, the darker the colour will be. The exception is projects that are building - these will be green if the project passed last time it built, and orange if the project failed last time.

CONCLUSION

This document shows in detail the comprehensive expertise of the team at NathCorp. We also have other services in place for other areas of expertise like Application migration but those are not detailed here. For further information pls send a mail to Info@NathCorp.Com or visit us at www.nathcorp.com