**OMTWebService**

**[Interface Specification]**

**[Tony Liu]**

[OMNIMARKETTIDE]

OMT-WebService User Manual 3

1. Summary 3

2. System diagram 3

3. Web Pages 4

(1). Question List: http://query.omnimarkettide.com:15660/agm/index 4

(2). Approved List: http://query.omnimarkettide.com:15660/agm/approved 5

(3). Statistics of SharePrice: http://query.omnimarkettide.com:15660/stat/index 5

(4). Westpac announcements: http://query.omnimarkettide.com:15660/agm/news 5

(5). System Configuration: http://query.omnimarkettide.com:15660/cfg/index 5

(6). SMA history list: http://localhost:9090/omtwebservice/stat/smahistory 6

(7). SMA changing today: http://localhost:9090/omtwebservice/stat/smadiff 6

(8). All pages in one: http://query.omnimarkettide.com:15660/ 6

(9). Meeting management: Including company, meeting, contact, document management: 7

4. Web Service Interfaces 7

(1). Post Get Share Price Query 7

(2). Post Get Chart History 8

(3). Agm Post Create Question 8

(4). Agm Get Question History 9

(5). Get Top 10 for gains & declines 10

(6). Get Top 5 of gains 10

(7). Get Top 5 of declines 11

(8). Post Sending Emails 11

(9). Set User online status 12

(10). Get Single Chart History by Type 12

(11). Get Share Price by different markets 13

(12). Get Single Chart History by different markets 14

(13). Get Single Chart History by Market and Type 15

(14). Get SMA History by Type 16

(15). Get SMA Changing by Type 17

(16). Get Multiple Chart History List 18

(17). Get All Meeting list 19

(18). Get All Meeting Company List 22

(19). Get the meetings with CompanyID 23

(20). Validate the password of the meetings 23

(21). Get one day close price 24

5. Running mongo in the security mode 25

6. Tomcat SSL configuration 26

7. Testing in Locust using Python 27

8. Web Socket 28

9. Typical System Security Issues. 32

# OMT-WebService User Manual

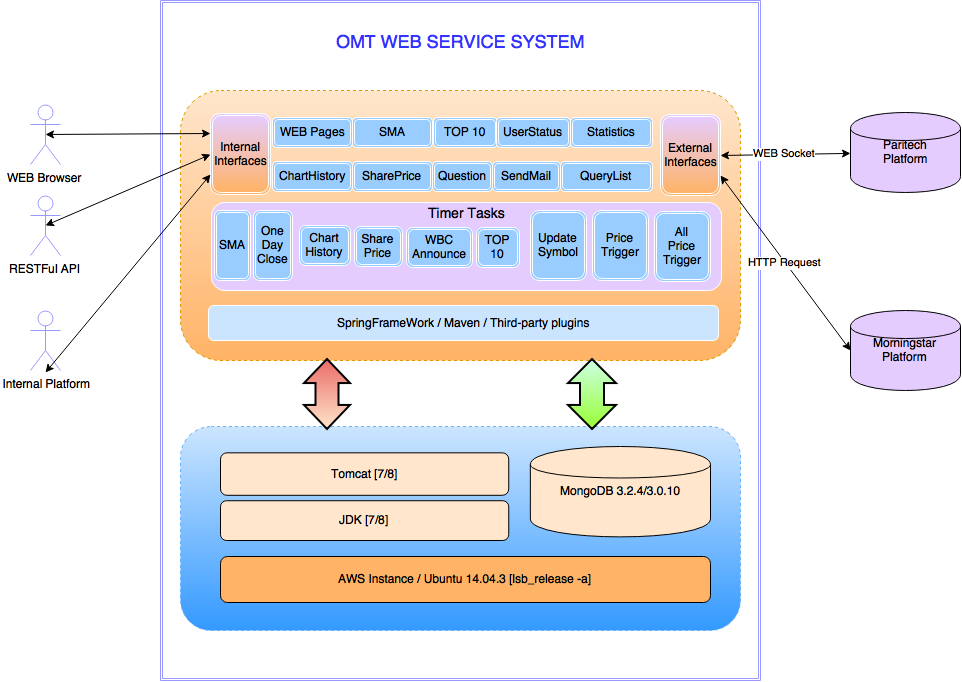
### Summary

This document can be used for querying for interface abilities and Web-UI features.

Including:

* + System Diagram
  + Web-UI Features
  + External Interfaces (postman)
  + Internal Interfaces
  + Testing load Balance method

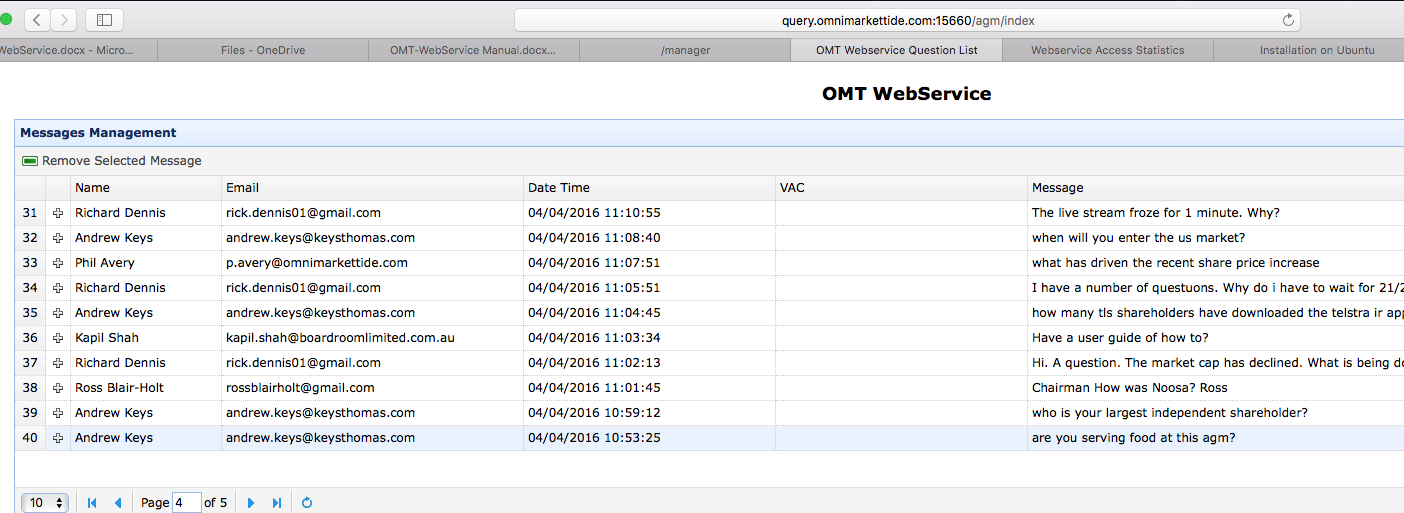
### System diagram



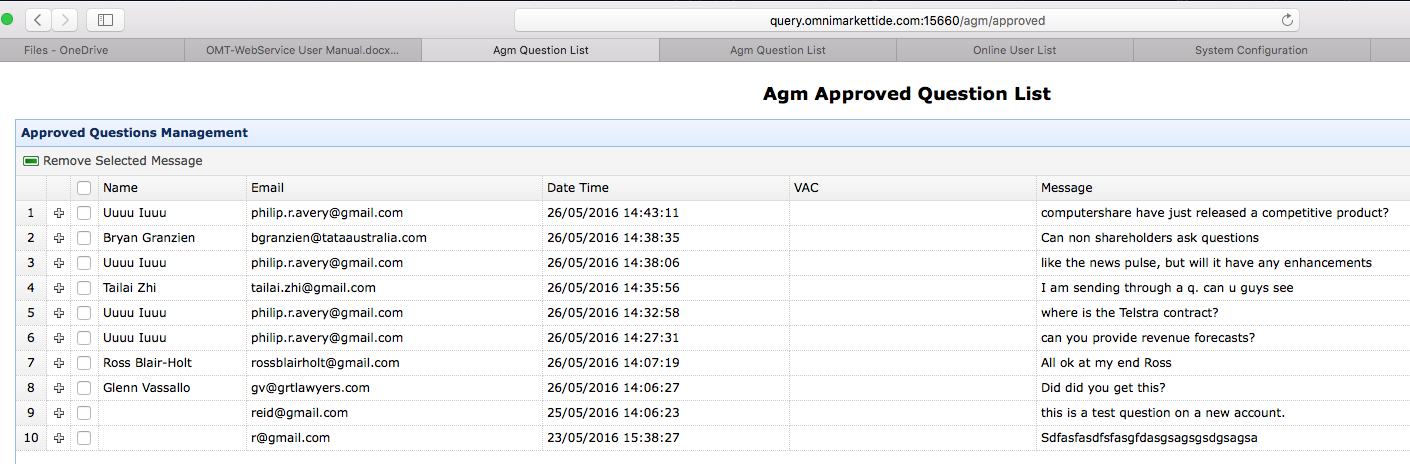
* + Two Data Sources from third-party platforms: Paritech Via WebSocket Connection and Morningstar Via HttpRequest
  + Three Clients outside: WEB Browser, RESTful API for mobiles, HTTP External interfaces like trigger for Price Change
  + WEB SERVICE running on AWS instance of UBUNTU 14.04.3
  + System Running Environment including: Tomcat, JRE, MongoDB
  + Based on Maven use a series of plugins like SpringFrameWork, JSON, WebSocket Client, Log4j
  + Modules including: Chart History, Share Price, Question, Send Mail, Top 10, Set User status, SMA, Statistics.
  + System including 8 timer tasks currently:
    - Share Price: running on a specific time interval (Configurable parameter) get data from MS
    - Chart History: every day morning clear cache and get new history data(5AM)
    - One Day close: every day morning get only last day close price for all (6AM)
    - WBC Announce: get WBC announcement running on a specific time interval(every 2 hours)
    - TOP 10: get top 5 gains and 5 declines running on a specific time interval (every 20 mins)
    - Update Symbol: every day night (8PM) update all symbol list from FTP server of MS
    - Price Trigger: send price trigger to another service url (configurable parameter)
    - All Price Trigger: send all price fields trigger to another endpoint (configurable parameter)

### Web Pages

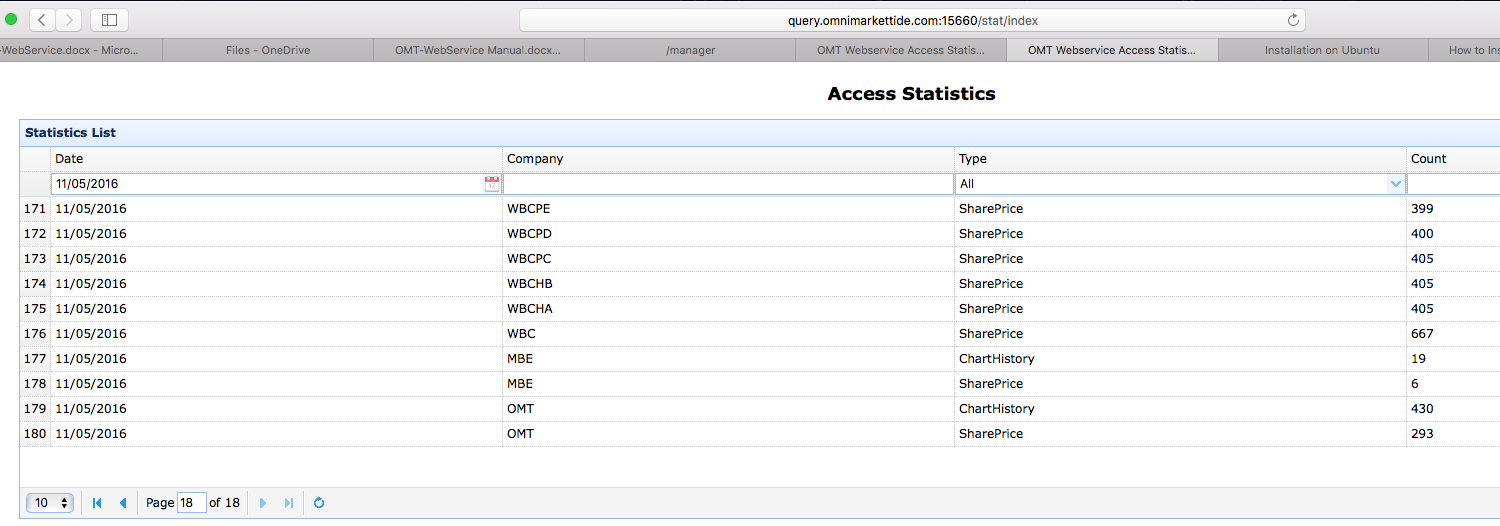
#### Question List: [http://query.omnimarkettide.com:15660/agm/index](http://query.omnimarkettide.com:15660/agm/%20%20index)



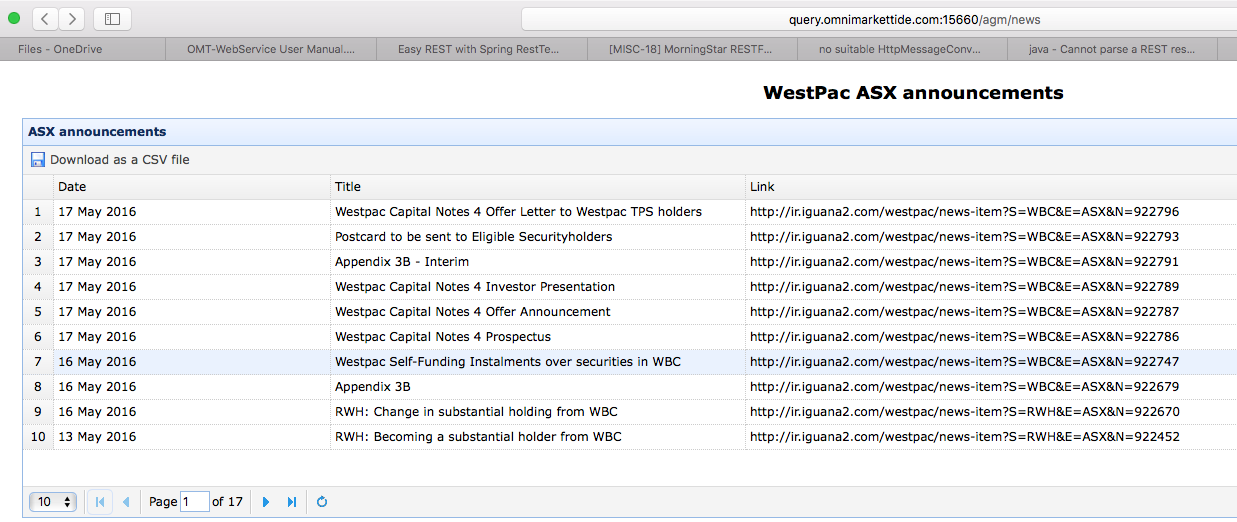
#### Approved List: [http://query.omnimarkettide.com:15660/agm/approved](http://query.omnimarkettide.com:15660/agm/%7Bcode%7D/approved)



#### Statistics of SharePrice: <http://query.omnimarkettide.com:15660/stat/index>

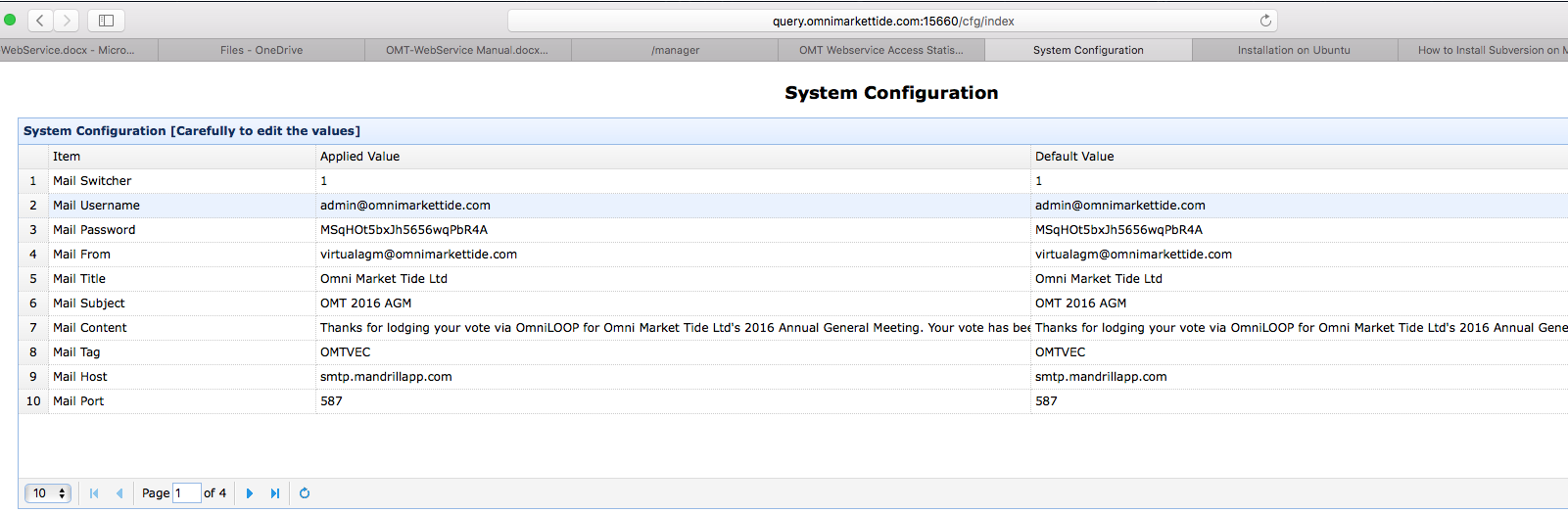


#### Westpac announcements: <http://query.omnimarkettide.com:15660/agm/news>

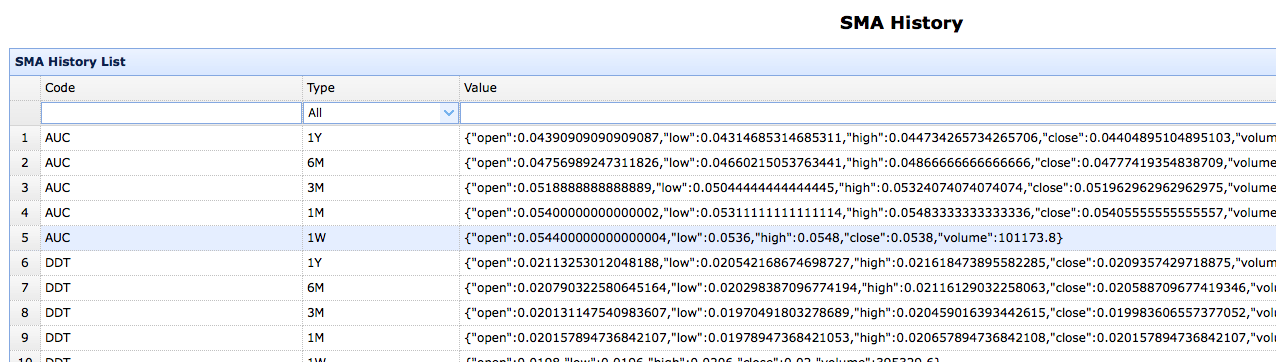


#### System Configuration: <http://query.omnimarkettide.com:15660/cfg/index>

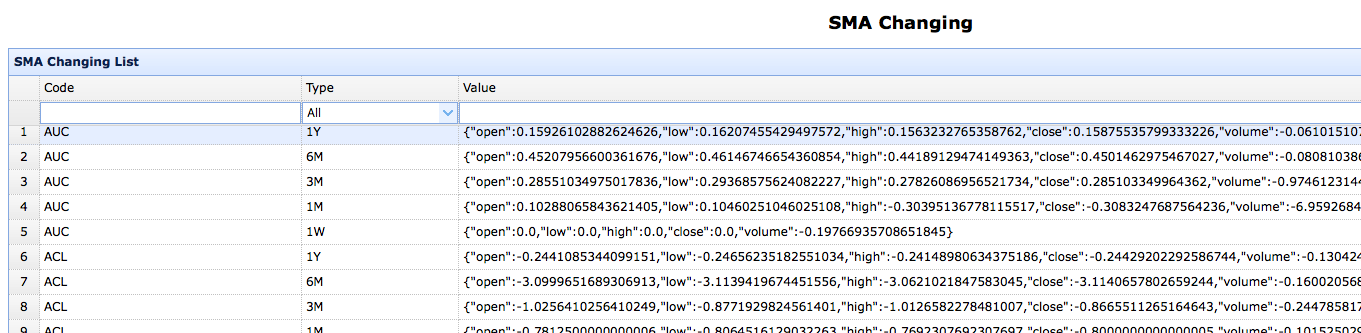
If you want to login and have a look ask username/password from admin



#### SMA history list: <http://localhost:9090/omtwebservice/stat/smahistory>

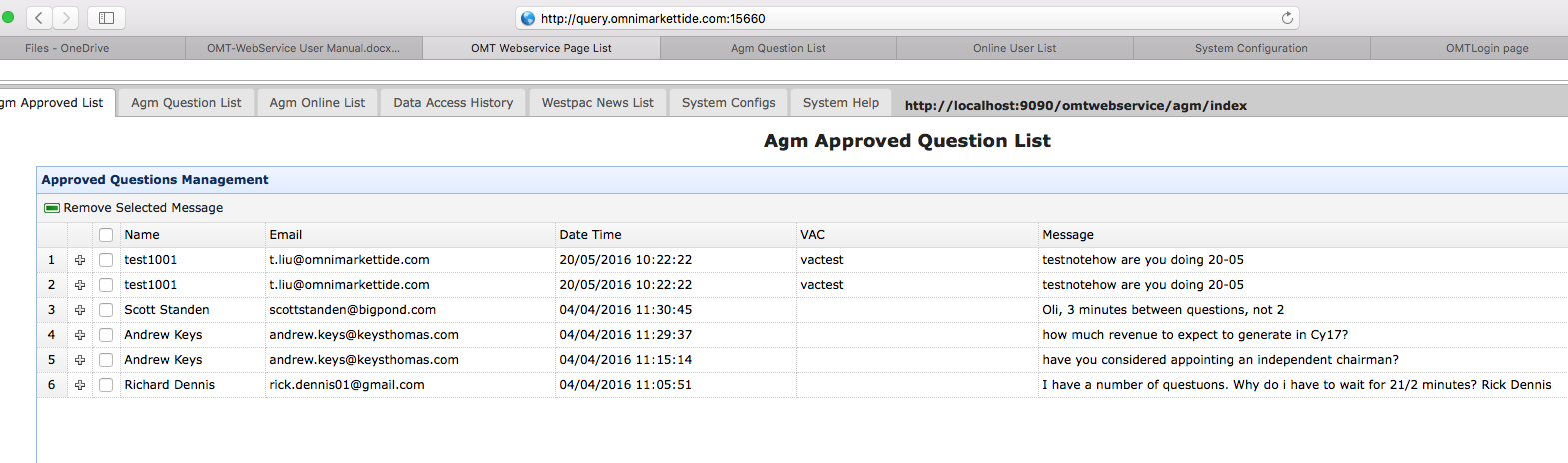


#### SMA changing today: <http://localhost:9090/omtwebservice/stat/smadiff>

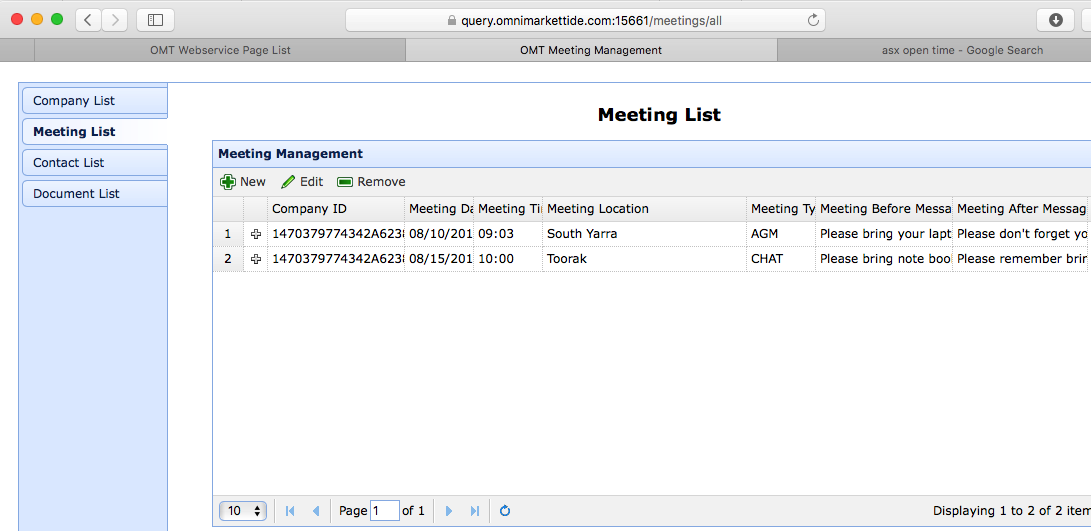


#### All pages in one: [http://query.omnimarkettide.com:15660/](http://query.omnimarkettide.com:15660/cfg/index)

please ask administrator for access username/password.



#### Meeting management: Including company, meeting, contact, document management:



### Web Service Interfaces

For HTTP Access: [**http://query.omnimarkettide.com:15660**](http://query.omnimarkettide.com:15660/wssocket/wssrequestcode)

For HTTPS Access: **https://query.omnimarkettide.com:156601**

#### Post Get Share Price Query

|  |  |
| --- | --- |
| REST URL | <http://query.omnimarkettide.com:15660/wssocket/wssrequestcode> |
| Method | Post |
| Request body  (Json Array) | [{"code":"OMT","timezone":"Australia/Victoria"}] |
| Content-Type | application/json |
| Result  (Json Array) | [  {  "Timestamp": "12/04/2016 10:44:39",  "Currency": "AUD",  "Index": null,  "Code": "OMT",  "Market": "ASX",  "BestBid": 0.031,  "BestAsk": 0.035,  "BidCount": 1,  "BidQuantity": 50000,  "BidUndisclosed": false,  "AskCount": 1,  "AskQuantity": 39462,  "AskUndisclosed": false,  "High": 0.035,  "Low": 0.035,  "Open": 0.035,  "Close": 0.035,  "Settlement": null,  "Last": 0.035,  "NumberOfTrades": 2,  "Volume": 13538,  "Trend": "None",  "Clazz": null,  "TradingState": "Normal",  "StrikePrice": null,  "CallOrPut": null,  "ContractSize": null,  "SubscriptionData": "All",  "AuctionPrice": 0.035,  "AuctionQuantity": 45033,  "AuctionRemainder": 13538,  "VWAP": 0.035,  "OpenInterest": null,  "ShareIssue": null,  "ValueChanged": 0,  "PercentChange": 0,  "Name": "OMNIMARKET FPO"  }  ] |

#### Post Get Chart History

|  |  |
| --- | --- |
| REST URL | <http://query.omnimarkettide.com:15660/wssocket/wssreqhistory> |
| Method | Post |
| Request body  (Json Object) | {"code":"sun","count":18} |
| Result  (Json Array) | [  {  "Low": 11.76,  "Open": 11.87,  "Close": 11.8,  "Date": "2016-03-15Z",  "Volume": 3684096,  "High": 11.91  },  ... |

#### Agm Post Create Question

Before starting create a question, you should follow the steps:

* Get a token via: POST <http://query.omnimarkettide.com:15660/rest/> with headers:

-H "X-Username: **{device\_id}**"

-H "X-Password: omt@mel123"

* On each following post request post together with the returned token like:

8LwW+4p57oLgIjjmWdOjIzZFUjy+ku5LA4jB+KdKbzA=

* Post to [http://query.omnimarkettide.com:15660/rest/creatusermsg](http://query.omnimarkettide.com:15660/rest/secure/creatusermsg) with

-H "X-Auth-Token: 8LwW+4p57oLgIjjmWdOjIzZFUjy+ku5LA4jB+KdKbzA="

* Any token error will result in an error in HTTPResponse Header: X-Token-Valid: true|false

|  |  |
| --- | --- |
| REST URL | [http://query.omnimarkettide.com:15660/rest/creatusermsg](http://query.omnimarkettide.com:15660/rest/%7Bcode%7D/creatusermsg) |
| Method | Post |
| Request Header | X-Auth-Token:lEdcPbsfvT4NkwxEgw8dM6gcCt3QVELX3Z/VjzrjKbU=  Content-Type:application/json  Accept-Code:OMT |
| Request body | [{"name":"test1001","email":"t.liu@omnimarkettide.com","notes":"testnotehow](mailto:%7B%22name%22:%22test1001%22,%22email%22:%22t.liu@omnimarkettide.com%22,%22notes%22:%22testnotehow) are you doing22222.","datetime":"29/03/2016 10:22:22"} |
|  |  |
| Result | {  "id": "570c4cb0e4b0af3f21d325d1",  "name": "test1001",  "email": "[t.liu@omnimarkettide.com](mailto:t.liu@omnimarkettide.com)",  "notes": "testnotehow are you doing22222.",  "datetime": "29/03/2016 10:22:22",  "status": 1  } |

#### Agm Get Question History

|  |  |
| --- | --- |
| REST URL | [http://query.omnimarkettide.com:15660/agm/getuserhistmsg](http://query.omnimarkettide.com:15660/agm/%7Bcode%7D/getuserhistmsg) |
| Method | Post |
| Request Header | Accept-Code:OMT |
| Request body | [rick.dennis01@gmail.com,r@gmail.com](mailto:rick.dennis01@gmail.com,r@gmail.com) |
| Content-type | text |
| Result  (Json Array) | [  {  "email": "[r@gmail.com](mailto:r@gmail.com)",  "messageList": [  {  "message": "This is long question that has lots of characters and more banteriusmatter in the content. This should show up in a single or multiple lines. Letssee how it goes. ",  "status": 1,  "datetime": "04/04/2016 10:45:00"  },  {  "message": "Robin's test message",  "status": 1,  "datetime": "11/04/2016 11:34:06"  }  ]  },  {  "email": "[r@gmail.com](mailto:r@gmail.com)",  "messageList": [  {  "message": "This is long question that has lots of characters and more banteriusmatter in the content. This should show up in a single or multiple lines. Letssee how it goes. ",  "status": 1,  "datetime": "04/04/2016 10:45:00"  },  {  "message": "Sda",  "status": 1,  "datetime": "11/04/2016 11:16:50"  },  {  "message": "Robin's test message",  "status": 1,  "datetime": "11/04/2016 11:34:06"  }  ]  }  ] |

#### Get Top 10 for gains & declines

|  |  |
| --- | --- |
| URL | [http://query.omnimarkettide.com:15660/asxdata/all](http://query.omnimarkettide.com:15660/agm/asxdata/all) |
| Method | Get |
| Request body | Nil |
| Result | Image |

#### Get Top 5 of gains

|  |  |
| --- | --- |
| URL | [http://query.omnimarkettide.com:15660/asxdata/gains](http://query.omnimarkettide.com:15660/agm/asxdata/gains) |
| Method | Get |
| Request body | Nil |
| Result | Image |

#### Get Top 5 of declines

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/asxdata/declines> |
| Method | Get |
| Request body | Nil |
| Result | Image |

#### Post Sending Emails

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/agm/sendmailcfm> |
| Method | Post |
| Request body  (Json Array) | [{"email":"t.liu@omnimarkettide.com","number":"TESTNUMBER123"}] |
| Result  (Json Array) |  |

#### Set User online status

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/status/setuserstatus> |
| Method | Post |
| Request Header | Accept-Code:OMT |
| Request body  (Json Array) | {"emailid":"[r@omt.com](mailto:r@omt.com)","status":1, "time":""} |
| Result  (Json Array) |  |

#### Get Single Chart History by Type

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/wssocket/charthistory> |
| Method | Post |
| Request body  (Json Array)  Count: 1W—1week 1M—1Month 3M—3Month 6M—6Month 1Y—1Year | {"code":"TLS","type":"1M"} |
| Result  (Json Array) |  |

#### Get Share Price by different markets

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/difmarket/wssrequestcode> |
| Method | Post |
| Request body  (Json Array) | {"code":"OMT", "market":"ASX", "count":260} |
| Result  (Json Array) |  |

#### Get Single Chart History by different markets

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/difmarket/wssreqhistory> |
| Method | Post |
| Request body  (Json Array) | {"code":"OMT", "market":"ASX", "count":260} |
| Result  (Json Array) |  |

#### Get Single Chart History by Market and Type

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/difmarket/charthistory> |
| Method | Post |
| Request body  (Json Array)  Count: 1W—1week 1M—1Month 3M—3Month 6M—6Month 1Y—1Year | {"code":"tls", "market":"ASX", "type":"1M"} |
| Result  (Json Array) |  |

#### Get SMA History by Type

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/wssocket/sma>  <http://query.omnimarkettide.com:15660/wssocket/smalist> |
| Method | Post |
| Request body  (Json Array)  Count: 1W—1week 1M—1Month 3M—3Month 6M—6Month 1Y—1Year | {"code":"TLS","type":"1W"}  [{"code":"TLS","type":"1W"}...] |
| Result  (Json Array) |  |

#### Get SMA Changing by Type

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/wssocket/smadiff>  <http://query.omnimarkettide.com:15660/wssocket/smadifflist> |
| Method | Post |
| Request body  (Json Array)  Count: 1W—1week 1M—1Month 3M—3Month 6M—6Month 1Y—1Year | {"code":"TLS","type":"1W"}  [{"code":"TLS","type":"1W"}...] |
| Result  (Json Array) |  |

#### Get Multiple Chart History List

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/wssocket/chartlist> |
| Method | Post |
| Request body  (Json Array)  Type: 1W—1week 1M—1Month 3M—3Month 6M—6Month 1Y—1Year | {"type":"1M", "codelist":[{"code":"OMT", "exchange":"ASX"},{"code":"WBC", "exchange":"ASX"},{"code":"TLS", "exchange":"ASX"}]} |
| Result  (Json Array) |  |

#### Get All Meeting list

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/meeting/all> |
| Method | GET |
| Request body | Nil |
| Result  (Json Array) | **EntyResponseData Example:**  [  {  "companyID": "1470379774342A623851223",  "companyName": "Omni Market Tide Ltd.",  "tickerCode": "OMT",  "companyLogo": "<http://www.omnimarkettide.com>",  "meetings": [  {  "meetingID": "1470380678114A1239966990",  "meetingDate": "08/10/2016",  "meetingTime": "09:03",  "meetingLocation": "South Yarra",  "meetingType": "AGM",  "meetingBeforeMsg": "Please bring your laptop",  "meetingAfterMsg": "Please don't forget your laptop",  "meetingContacts": {  "name": "Reid",  "email": "[reid@omt.com.au](mailto:reid@omt.com.au)",  "phone": "0432123321",  "weburl": "<http://www.omt.com.au>",  "contactID": "1470379927649A495319795"  },  "meetingDetails": {  "documents": [  {  "documentID": "1470379855695A1382752245",  "documentType": "Doc",  "documentName": "OmtLight",  "documentLink": "<http://www.google.com/>",  "documentAttributes": [  {  "attributeName": "length",  "attributeValue": "1:30"  },  {  "attributeName": "size",  "attributeValue": "2MB"  }  ]  },  {  "documentID": "1470379884368A856081389",  "documentType": "Video",  "documentName": "Reuqirement",  "documentLink": "<http://www.video.com.au>",  "documentAttributes": [  {  "attributeName": "length",  "attributeValue": "2:30"  },  {  "attributeName": "size",  "attributeValue": "200MB"  }  ]  }  ]  }  },  {  "meetingID": "1470381526834A694560670",  "meetingDate": "08/15/2016",  "meetingTime": "10:00",  "meetingLocation": "Toorak",  "meetingType": "CHAT",  "meetingBeforeMsg": "Please bring note book",  "meetingAfterMsg": "Please remember bring your stuff",  "meetingContacts": {  "name": "Tony",  "email": "[tony@omt.com.au](mailto:tony@omt.com.au)",  "phone": "0455834439489",  "weburl": "<http://www.omt.com.au>",  "contactID": "1470379948403A110078332"  },  "meetingDetails": {  "documents": [  {  "documentID": "1470379855695A1382752245",  "documentType": "Doc",  "documentName": "OmtLight",  "documentLink": "<http://www.google.com/>",  "documentAttributes": [  {  "attributeName": "length",  "attributeValue": "1:30"  },  {  "attributeName": "size",  "attributeValue": "2MB"  }  ]  }  ]  }  }  ]  }  ] |

#### Get All Meeting Company List

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/company/allcompany> |
| Method | GET |
| Request body |  |
| Result  (Json Array) |  |

#### Get the meetings with CompanyID

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/meeting/query> |
| Method | Post |
| Request body  (Json Array) | ["1470274922405A1684052826","1470358214818A1497020445"] |
| Result  (Json Array) |  |

#### Validate the password of the meetings

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/meeting/validatepwd> |
| Method | **Post** |
| Request body  (Json Array) | {"meetingID":"1470708040252","password":"123","valid":false} |
| Result  (Json Array) |  |

#### Get one day close price

|  |  |
| --- | --- |
| URL | <http://query.omnimarkettide.com:15660/wssocket/allclose> |
| Method | **GET** |
| Request body  (Json Array) |  |
| Result  (Json Array) |  |

### Running mongo in the security mode

* 1. Add users first:

with default roles:[dbOwner, ]

db.createUser(

{

user:"admin",

pwd:"admin",

roles:[{role:"dbOwner",db:"testdb"}]

}

)

* 1. Restart mongo

db.shutdownServer()

-- [need close all connections to mongo, like web application, cmd lines, and robomongo client] , otherwise you cannot start mongo with the same port:27017

* 1. sudo mongod --**auth**

>use testdb

>db.auth("admin","admin")

The above is mongodb command lines after login

* 1. **Programmly connect to auth mongodb**

Mongo client = new MongoClient("localhost");

new MongoTemplate(client, "testdb", null);

// mongod --auth [ for security reason ]

UserCredentials credential=new UserCredentials("admin", "admin");

MongoTemplate auth=new MongoTemplate(client, "testdb", credential);

### MongoDB Design

* 1. Related designing collection of

Each field which will be sorted should be indexed in group to increase the query speed.

* 1. MongoDB related configuration:

ulimit –n 1024  ------- to set the maximum file number[ulimit –a]

Netstat –an | grep 27017 -------- to make sure mongo can be shutdown

Sudo kill –9 and check /data/db/\*.lock files, normally mongod.lock

* 1. How to use commands in mongo to config the server:

MongoDB start stop on Ubuntu

Sudo service mongod stop

Sudo service mongod start

### MongoDB Basic Command

* 1. Backup and Restore
     1. Backup whole database:

$>mkdir mongobak

$>cd mongobak

$>mongodump

Resutl:

Bakpath: ./dump/[databasename]/[collectionname].bson

* + 1. Backup the specified database

$>mongodump -d omtdb

Result:

Bakpath: ./dump/[databasename]/[collectionname].bson

* + 1. Database restore

$>cd  mongobak

$>mongorestore --drop :// restore all data and drop the data in db.

$>mongorestore -d omtdb --drop //just restore the omtdb

$>mongorestore -d omtdb –c userMessage –drop ;// just restore collection

* + 1. Export data from Mongodb

$>mongoexport -d omtdb -c userMessage -q {} -f \_id,name,email,datetime,notes --type=csv > pages.csv

$>mongoexport -d omtdb -c historyPriceVO -q {} --type=json >historyPriceVO.json

* + 1. Import data to Mongodb

$>mongoimport -d omtdb -c page --type csv --headerline --drop < csvORtsvFile.csv ;

// type include: csv,json,tsv

$>mongoimport -d omtdb -c historyPriceVO -j 4 --type json --drop <historyPriceVO.json

* 1. Operation
     1. Basic command
        1. $>mongo //running mongo client
        2. >show dbs
        3. >use omtdb
        4. >show collections
        5. >db.instrument.find()
        6. >db.instrument.count()
        7. >db.instrument.drop()
        8. >db.instrument.remove({})
        9. >Object.keys(db.msChartHistory.findOne())[show collection fields]
     2. Batch update
     3. Query
        1. >db.userMessage.find({"notes":/.\*you.\*/})
        2. >db.msSharePrice.aggregate([{$sort:{"data.Last"**:-1**}},{$limit:1}]) **[Get the max value of data.Last]**
        3. > db.msSharePrice.find( {$and:[{"data.Last":{$gt:2}},{"data.Last":{$lt:3}}]})

### Tomcat SSL configuration

* 1. Get file from :smb://omtsvr/Development/Misc/SSLCertificate/omnimarkettide.com/Tomcat
  2. Reference to

<https://au.godaddy.com/help/tomcat-generate-csrs-and-install-certificates-5239>

$> openssl pkcs12 -export -in omt.crt -inkey certificate\_key.key -out **mycert.p12** -name tomcat -CAfile intermediate.crt -caname root -chain

$> keytool -importkeystore -srckeystore **mycert.p12** -destkeystore tomcat.keystore

$> keytool -list -keystore tomcat.keystore

Enter keystore password:

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 1 entry

tomcat, 01/08/2016, PrivateKeyEntry,

Certificate fingerprint (SHA1): C2:C6:07:30:6F:28:84:25:FF:CD:3C:98:41:0F:8D:7D:90:27:70:3C

In tomcat configuration file server.xml:

<Connector port="15661" protocol="org.apache.coyote.http11.Http11Protocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

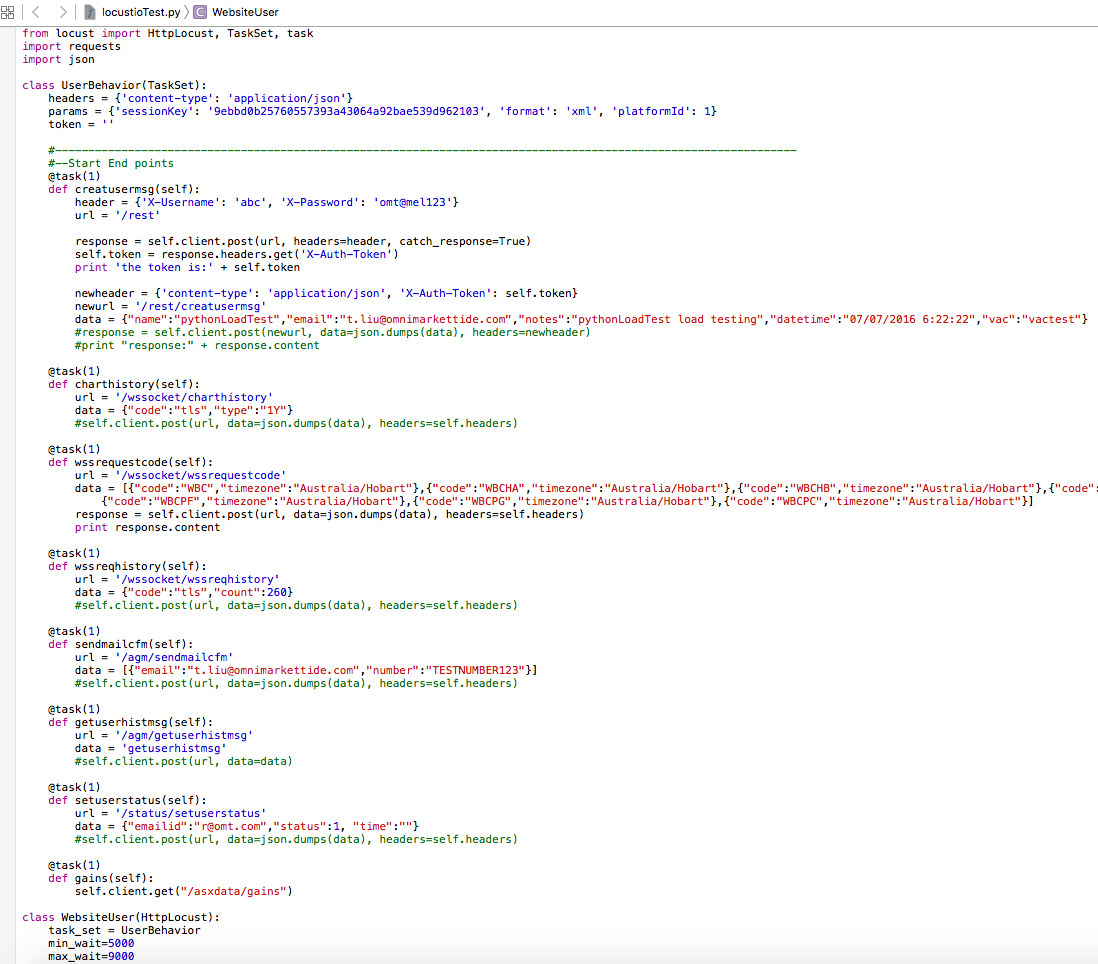
keystoreFile="/Users/tonyliu/tomcat/tomcat.keystore" keystorePass="omt@mel123" keyAlias="tomcat"

clientAuth="false" sslProtocol="TLS" />

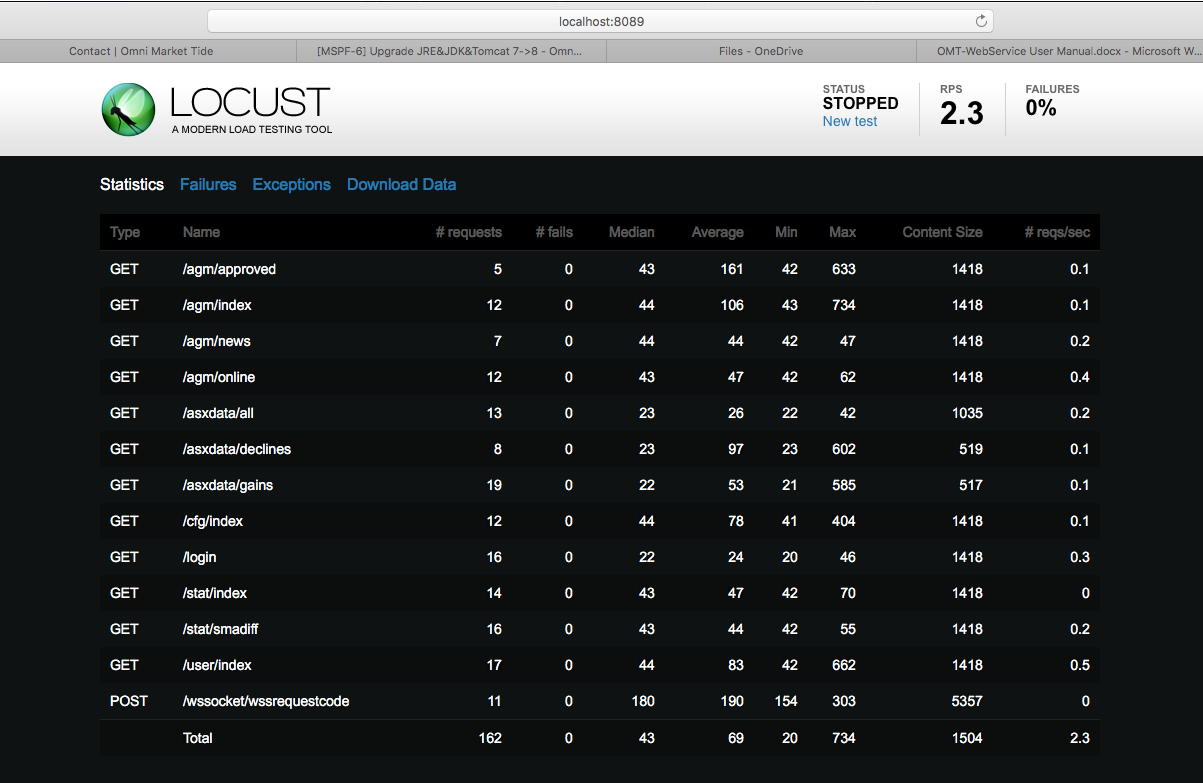
### Testing in Locust using Python

Ref: [<https://docs.python.org/2.7/>]

* 1. brew install python
  2. pip install locustio
  3. Mkdir testing; cd testing;
  4. Touch locustioTest.py
  5. Open locustioTest.py in Xcode or VIM

l

* 1. locust -f locustioTest.py --host="http://query.omnimarkettide.com:15660/"
  2. localhost:8089



### Web Socket

* 1. Spring Frame Work

🡪Use current container port

🡪Independent from container

1. Pom.xml, dependencies:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-websocket</artifactId>

<version>${spring.framework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-messaging</artifactId>

<version>${spring.framework.version}</version>

</dependency>

1. Initialize messageTemplate:

@Component

**public** **class** MessagingApplicationListener **implements** ApplicationListener<ContextRefreshedEvent> {

@Autowired

**public** MessagingApplicationListener(SimpMessagingTemplate messagingTemplate) {

WssController.*simpMessagingTemplate* = messagingTemplate;

}

@Override

**public** **void** onApplicationEvent(ContextRefreshedEvent arg0) {

// **TODO** Auto-generated method stub

}

}

1. Configurations

@Configuration

@EnableWebSocket

**public** **class** WebSocketConfig **extends** AbstractWebSocketMessageBrokerConfigurer {

@Override

**public** **void** configureMessageBroker(MessageBrokerRegistry config) {

config.enableSimpleBroker("/list"); // [Page Subscribe] End point

config.setApplicationDestinationPrefixes("/update"); // [Receive page requests] End point

config.setCacheLimit(100000);

}

@Override

**public** **void** registerStompEndpoints(StompEndpointRegistry registry) {

registry.addEndpoint("/voting").withSockJS(); // [Page SocketJS] end point

}

}

1. Controller to pages

@Controller

**public** **class** WssController {

**public** **static** SimpMessagingTemplate *simpMessagingTemplate*;

@MessageMapping("/voting")

@SendTo("/list/showResult")

**public** Result voting(CalcInput input) **throws** Exception {

Result message = **new** Result(input.getNum1()+"+"+input.getNum2()+"="+(input.getNum1()+input.getNum2()));

**return** message;

}

**public** **static** **void** broadcast(List<WebVotes> list){

*simpMessagingTemplate*.convertAndSend("/list/showResult", list);

}

}

1. Dispatch messages directly from server

*simpMessagingTemplate*.convertAndSend("/list/showResult", list);

1. Web pages
   1. Import spring web socket javascript support

<script type=*"text/javascript"* src=*"/js/sockjs-0.3.4.js"*></script>

<script type=*"text/javascript"* src=*"/js/stomp.js"*></script>

* 1. Implementation 1 [Receive messages from Server]:

**function** SpringStompUpdate(){

**var** socket = **new** SockJS('/omtwebservice/voting');

stompClient = Stomp.over(socket);

stompClient.connect({}, **function**(frame) {

console.log('Connected: ' + frame);

stompClient.subscribe('/list/showResult', **function**(result){

$('#dgVoting').datagrid({

data:JSON.parse(result.body)

});

});

});

}

* 1. Implementation 2 [Send message to server]:

**function** sendNum() {

**var** num1 = document.getElementById('num1').value;

**var** num2 = document.getElementById('num2').value;

stompClient.send("/update/voting", {}, JSON.stringify({ 'num1': num1, 'num2': num2 }));

}

* 1. Tomcat7/8

🡪Rely on Tomcat container

🡪Use the container port

🡪Use client side embedded web socket object

🡪Similar with Third-party plugin

* 1. Third-party Plugin

🡪Must listen on another port

🡪Independent from container

🡪Use client side embedded web socket object, but has its own client code for backend

1. Dependency

<dependency>

<groupId>org.java-websocket</groupId>

<artifactId>Java-WebSocket</artifactId>

<version>1.3.0</version>

</dependency>

1. Server side

**public** **class** WsServer **extends** WebSocketServer {

**private** **static** Logger *omtlogger* = Logger.*getLogger*(WsServer.**class**);

**private** **static** List<WebSocket> *connList* = **new** ArrayList<WebSocket>();

**public** WsServer(InetSocketAddress address) {

**super**(address);

}

**public** **static** **void** broadcast(String msg) {

**for** (WebSocket client : *connList*) {

**try** {

**synchronized** (client) {

*omtlogger*.info("broadcast message: "+ msg);

client.send(msg);

}

} **catch** (Exception e) {

*connList*.remove(client);

**try** {

client.close();

} **catch** (Exception e1) {

// Ignore

}

}

}

}

@Override

**public** **void** onOpen(WebSocket conn, ClientHandshake handshake) {

*omtlogger*.info("new connection to " + conn.getRemoteSocketAddress());

handshake.getResourceDescriptor();

handshake.getFieldValue("Username");

*connList*.add(conn);

}

@Override

**public** **void** onClose(WebSocket conn, **int** code, String reason, **boolean** remote) {

*omtlogger*.info("closed " + conn.getRemoteSocketAddress() + " exit code " + code + " info: " + reason);

*connList*.remove(conn);

}

@Override

**public** **void** onMessage(WebSocket conn, String message) {

*omtlogger*.info("received message from " + conn.getRemoteSocketAddress() + ": " + message);

}

@Override

**public** **void** onError(WebSocket conn, Exception ex) {

System.***err***.println("an error occured on connection " + conn.getRemoteSocketAddress() + ":" + ex);

}

**private** **static** WebSocketServer *server*;

**private** **static** String *host* = "localhost";

**private** **static** **int** *port* = 9091;

**public** **static** **void** getWssServerInstance(){

**if**(*server* == **null**){

*server* = **new** WsServer(**new** InetSocketAddress(*host*, *port*));

*server*.start();

}

}

**public** **static** **void** main(String[] args) {

WebSocketServer server = **new** WsServer(**new** InetSocketAddress(*host*, *port*));

server.run();

}

}

1. Client side 1

**public** **class** WsClient **extends** WebSocketClient {

**public** WsClient(URI serverUri, Draft draft) {

**super**(serverUri, draft);

}

**public** WsClient(URI serverURI) {

**super**(serverURI);

}

@Override

**public** **void** onOpen(ServerHandshake handshakedata) {

}

@Override

**public** **void** onClose(**int** code, String reason, **boolean** remote) {

}

@Override

**public** **void** onMessage(String message) {

System.***out***.println("client received message: " + message);

**while**(**true**){

**try** {

Thread.*sleep*(3000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**this**.getConnection().send("client send this message...");

}

}

@Override

**public** **void** onError(Exception ex) {

System.***err***.println("an error occurred:" + ex);

}

**public** **static** **void** main(String[] args) **throws** URISyntaxException {

// ------------------------------------------------------------------------------------------------------

WebSocketClient client = **new** WsClient(**new** URI("ws://localhost:8887"), **new** Draft\_10());

client.connect();

}

1. Client side 2

**function** openJavaConnect(){

**var** javaws = **new** WebSocket("ws://"+document.location.hostname+":8887");

javaws.onopen = **function**(){

};

javaws.onmessage = **function**(message){

document.getElementById("chatlog").textContent += message.data + "\n";

};

}

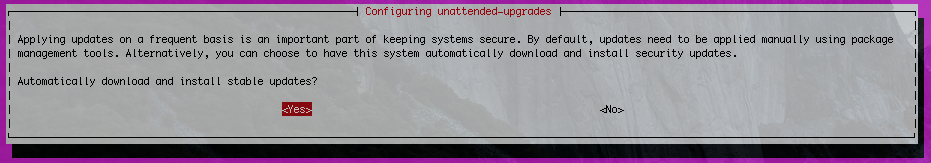
### Typical System Security Issues.

* 1. Ubuntu server security patches.

Issue: Ubuntu server missing critical and high security patches.

Solution:

* + 1. sudo apt-get install **unattended-upgrades**
    2. sudo unattended-upgrade
    3. sudo dpkg-reconfigure unattended-upgrades



* 1. Harden SSH on Ubuntu

Issue: Harden SSH access [weak algorithms, ciphers accepted]

* + 1. <http://www.cyberciti.biz/tips/linux-unix-bsd-openssh-server-best-practices.html>
    2. <http://askubuntu.com/questions/2271/how-to-harden-an-ssh-server>
    3. Typical AWS instance ssh configuration:

# Lifetime and size of ephemeral version 1 server key

KeyRegenerationInterval 3600

ServerKeyBits 1024

# Logging

SyslogFacility AUTH

LogLevel INFO

# Authentication:

LoginGraceTime 120

PermitRootLogin without-password

StrictModes yes

RSAAuthentication yes

PubkeyAuthentication yes

#AuthorizedKeysFile %h/.ssh/authorized\_keys

# Don't read the user's ~/.rhosts and ~/.shosts files

IgnoreRhosts yes

# For this to work you will also need host keys in /etc/ssh\_known\_hosts

RhostsRSAAuthentication no

# similar for protocol version 2

HostbasedAuthentication no

# Uncomment if you don't trust ~/.ssh/known\_hosts for RhostsRSAAuthentication

#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)

PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with

# some PAM modules and threads)

ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords

PasswordAuthentication no

# Kerberos options

#KerberosAuthentication no

#KerberosGetAFSToken no

#KerberosOrLocalPasswd yes

#KerberosTicketCleanup yes

# GSSAPI options

#GSSAPIAuthentication no

#GSSAPICleanupCredentials yes

X11Forwarding yes

X11DisplayOffset 10

PrintMotd no

PrintLastLog yes

TCPKeepAlive yes

#UseLogin no

#MaxStartups 10:30:60

#Banner /etc/issue.net

# Allow client to pass locale environment variables

AcceptEnv LANG LC\_\*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,

# and session processing. If this is enabled, PAM authentication will

# be allowed through the ChallengeResponseAuthentication and

# PasswordAuthentication. Depending on your PAM configuration,

# PAM authentication via ChallengeResponseAuthentication may bypass

# the setting of "PermitRootLogin without-password".

# If you just want the PAM account and session checks to run without

# PAM authentication, then enable this but set PasswordAuthentication

# and ChallengeResponseAuthentication to 'no'.

UsePAM yes

* + 1. Overall, disable password access, use auth access file.
  1. mDns Information Disclosure
     1. Issue: on UDP port 5353 mDNS is running which provides unauthenticated attackers with information about the server that shouldn’t be public.
     2. Solution:
        1. Stop the service:
        2. Build firewall to restrict the access to it.
  2. HTTPS redirection is not present at http site

For tomcat, just need to redirect all http request to https. This should be only a configuration thing.

[Reference to: No.6](#_Tomcat_SSL_configuration)

* 1. HTTPs wildcard certificate is in use

Use trusted and correct certificate.

[Reference to: No.6](#_Tomcat_SSL_configuration)

* 1. SSL issue—SSL has weak RC4 cipher enabled

ubuntu@ip-10-0-3-144:/var/lib/tomcat7/conf$ **openssl ciphers -v | awk '{print $2}' | sort | uniq**

**SSLv3**

**TLSv1.2**

<more cata or cat filename> | grep "@gmail.com" | cut –f<15> -d " " | sort | uniq

