System Design Specification

for

SGS eAuction

UPDATED BY

ROLE	NAME	SIGNATURE	DATE (DD-MM-YYYY)
Team Lead	DEEPAK CHAKRAWANI		

REVIEWED BY

ROLE	NAME	SIGNATURE	DATE (DD-MM-YYYY)
PM	SUBRAMANYAM CR(CRIMSONLOGIC)		

MDD

APPROVED BY

ROLE	NAME	SIGNATURE	DATE (DD-MM-YYYY)
User PM	YIP EE XIU (MDD)		
User PM	JEREMY HOR (MDD)		

Change History

Version	Changes made	Date
4.0	1. Modification for HA, i.e. WAS clustering 2. Modification for MAS Bills	11/09/2011 (ZG)
5.0	1. Modification for iBanking project	12/09/2011 (THL)
5.1	1. Include documentation on allotment limits (Appendix 7.6) 2. File format changes in Section 5.1 to cater for negative yield (secupdt file) 3. Set minimum coupon rate of 0.125% 4. Changes in the fields for corridor parameters	29/10/2012 (THL)
5.2	1. Insert the private key table	09/04/2013 (THL)
5.3	1. New Authorisation check for all functions and sub functions. 2. Modification for new business rule to allow Level 3 users to input Auction Parameters and Tenders, with a Maker-Checker role check. 3. Insert new objects to store Authorised Actions per authorisation level and their descriptions. 4. New function to Generate Letter to AGD. 5. Exception bids are reported in Decrypt bids report.	01/06/2013 (BL/THL)
5.4	Update in details the steps for auction run	28/06/2013 (THL)
5.5	Added footnote to highlight the validation of T-Series NRIC as well.	01/04/2014 (BL)
5.6	• Allow MDD to control the publication of auction result to website and PDs • Update of PDs' minimum underwriting obligation for eApps	Nov 2014 (THL/Francis)
5.7	Changes for includes SRS changes. - UI changes to include new application type and CPF/SRS account no. - Auction Results file for MEPS+	Mar 2018
5.8	Section 3.2.49 New MSABRT05/1 csv format. Section 7.2 CPF validation change	27 July 2018 (CL)

	Section 3.6.14 and Section 3.7 Preventing Concurrent Login	24 Dec 2019 (CL)
5.12	Changes for CR - CuR-MAS-SBA-16 Section 3.2.21 and 3.2.27 To accept Negative Yield for Bid submission	04 Jun 2020 (CL)
5.13	Changes for CR - CuR-MAS-SBA-15 Section 7.9 Changes for CR - CuR-MAS-SBA-17 Section 2, 3.1, 3.1.2, 3.1.46, 3.1.47, 3.1.54 to 3.1.59, 3.2.1, 3.2.2, 3.2.6, 3.2.7, 3.2.13 to 3.2.18, 3.6.3, 4.15, 4.16, 4.23 to 4.28, 6.1.3 to 6.1.5, 7.8, 7.10 User Maintenance for eApps Change in Sign off for the Letter to AGD for SGS/Tbill/SSB Generated by eAuction Section 7.7, 7.7.1	30 Jul 2020 (CL)
5.14	Changes for CR - CuR-MAS-SBA-20 Section 7.11 Changes for CR - CuR-MAS-SBA-24 Section 7.12 Changes for TFS-98597 - eAuction bug on User creation/maintenance Section 3.2.2, 3.2.12	13 Jun 2022 (CL)
5.15	Changes for CR - CuR-MAS-SBA-25 – To enhance the existing eAuction system to support high volume of bids and Tripwire to prevent concurrent auction runs Section 2 – A.4, 3.2.51, 3.6.2, 3.6.11, 3.6.15, 3.6.16 and 7.6	12 Oct 2022 (CL)
5.16	Changes for CR - CuR-MAS-SBA-26 – eApps 1.2 Release 1 and Release 2 Section 3.2.3, 3.2.31, 3.2.51, 3.2.54, 3.2.56, 3.6.17, 3.6.18, 3.6.19, 3.6.20, 3.6.21, 3.6.22, 3.6.23, 4.53	04 Jan 2024 (CL)
5.17	Changes for CR - CuR-MAS-SBA-26 – eApps 1.2 Release 3 Section 3.1.6, 3.1.22, 3.1.23, 3.1.24, 3.1.41, 3.1.42, 3.1.43, 3.2.1, 3.2.28, 3.2.29, 3.2.32, 3.2.43, 3.2.44, 3.2.45, 3.2.46, 3.5, 3.6.24, 4.49, 4.50, 4.51, 4.52	20 Feb 2024 (CL)

Content

1 INTRODUCTION 1

1.1 OVERVIEW 1
1.2 AUDIENCE 1
1.3 SDS STRUCTURE 1

2 PROCESS 2

3 SYSTEM DESIGN 10

3.1 WEB FRONT-END INTERFACE DESIGN: 10
JSP PAGES 16
BROWSE HISTORICAL SECURITY ISSUES 17
BROWSE ONGOING SECURITY ISSUES 18
BROWSE SECURITY DEALERS 18
SGS TENDER 19
SGS AUCTION 25
USER MAINTENANCE 26
MISCELLANEOUS 30
3.2 WEB BACK-END DESIGN 31
PACKAGE - MODEL 31
3.2.1 JAVA BEANS 31
PACKAGE - FORM 31
3.2.2 JAVA FORM BEANS 31
PACKAGE - ACTION 32
3.2.3 MAINMENU.JAVA 32
3.2.4 LOGOFF.JAVA 33
3.2.5 TIMEOUT.JAVA 33
PACKAGE - ACTION.USER 33
3.2.6 USERMAINTENANCEMENU.JAVA 33
3.2.7 USERMAINTENANCE.JAVA 33
3.2.8 ADDNEWUSER.JAVA 34
3.2.9 BROWSEUSERS.JAVA 34
3.2.10 SELECTUSER.JAVA 35
3.2.11 SUBUSER.JAVA 35
3.2.12 UPDATEUSER.JAVA 36
3.2.13 EAPPSUSERMAINTENANCE.JAVA 36
3.2.14 ADDNEWEAPPSUSER.JAVA 37
3.2.15 SUBEAPPSADDUSER.JAVA 37
3.2.16 BROWSEEAPPSUSERS.JAVA 38
3.2.17 UPDATEEAPPSUSER.JAVA 39

3.2.18 SUBEAPPSUPDATEUSER.JAVA 39
PACKAGE - ACTION.BROWSE 40
3.2.19 SELECTSECISSUE.JAVA 40
3.2.20 VIEWHISTSEC.JAVA 40
3.2.21 VIEWONGOINGSEC.JAVA 41
3.2.22 VIEWSECDETAILS.JAVA 41
3.2.23 SELECTSECDEALER.JAVA 42
3.2.24 VIEWSECDEALERS.JAVA 42
PACKAGE - ACTION.TENDER 43
3.2.25 SGSTENDERMENU.JAVA 43
3.2.26 ADDNEWTENDERMENU.JAVA 43
3.2.27 ADDNEWTENDERMETHODSELECT.JAVA 43
3.2.28 UPLOADTENDERCONFIRM.JAVA 44
3.2.29 UPLOADTENDERRESULT.JAVA 45
3.2.30 SELECTTENDER.JAVA 45
3.2.31 ADDNEWTENDER.JAVA 46
3.2.32 BROWSEALLTENDERS.JAVA 46
3.2.33 UPDATETENDER.JAVA 47
3.2.34 SUBTENDER.JAVA 48
3.2.35 VIEWUNRELTENDERS.JAVA 50
3.2.36 VERIFYTENDER.JAVA 50
3.2.37 MATCHTENDER.JAVA 51
3.2.38 TENDERRELEASED.JAVA 52
3.2.39 SELECTALLOTRESULT.JAVA (REMOVED JUN 2013) 53
3.2.40 VIEWALLOTRESULT.JAVA (REMOVED JUN 2013) 53
3.2.41 REACTRELTENDER.JAVA 53
3.2.42 TENDERUNRELEASED.JAVA 54
PACKAGE - ACTION.AUCTION 56
3.2.43 SGSAUCTIONMENU.JAVA 56
3.2.44 SGSAUCTIONSECLIST.JAVA 57
3.2.45 ~~SGSAUCTIONSECPARALIST.JAVA~~ (REMOVED JUN 2013) 57
3.2.46 SGSAUCTIONSECPARAINPUT.JAVA (ADDED JUN 2013) 58
3.2.47 SGSAUCTIONSECPARAVERIFY.JAVA (ADDED JUN 2013) 58
3.2.48 SGSAUCTIONSAVEPARA.JAVA 59
3.2.49 SGSAUCTIONVERIFYPARA.JAVA 59
3.2.50 SGSAUCTIONVALIDATEBIDS.JAVA 59
3.2.51 SGSAUCTIONRUNAUCTION.JAVA 60
3.2.52 SGSAUCTIONVIEWREPORTS.JAVA 63
3.2.53 SGSAUCTIONGENERATELETTER.JAVA 63
3.2.54 SGSAUCTIONRELEASERESULT.JAVA 64
PACKAGE – UTILS 65
3.2.55 DATABASE ACCESS 65
PACKAGE – SGSBES.AUCTION 66
3.2.56 AUCTION PROGRAM 66
3.3 INTEGRATION WITH ACTIVE DIRECTORY SERVER 66
3.4 ERROR HANDLING AND LOGGING 67

3.5 SOME MISCELLANEOUS JAVA CLASSES 67
3.6 ORACLE DATABASE DESIGN 68
3.6.1 "ABA0001_LIMIT" 68
3.6.2 "AQA0002_TRANSACTION" 69
3.6.3 "AQA0003_USER" 70
3.6.4 "AQA0004_LEVEL_ACTION" 70
3.6.5 "AQA0005_REPORT_COUNTER" 71
3.6.6 "AQA0006_EAPPS_TRANSACTION" 71
3.6.7 "AQA0007_COPY_SEC_MASTER_IND" 72
3.6.8 "AQA0008_SEC_AUCTION_PARA" 73
3.6.9 "AQA0010_ACTION_REF" 73
3.6.10 "ABA0001_SECURITY_MASTER" 73
3.6.11 "ABA0006_AUCTION_RESULT" 75
3.6.12 "AQA0103_PRIVATE_KEY" 75
3.6.13 ABA0214_SB_CD_FILE_TYPE 75
3.6.14 AQA0011_USER_SESSION 76
3.6.15 ABA0007_DETAIL_AUCTION_RESULT 76
3.6.16 AQA0017_AUCTION_IND 77
3.6.17 AQA0019_AUCTION_REPORT_IND 78
3.6.18 AQA0009_SEC_AUCTION_PARA_STG_CTG 78
3.6.19 AQA0020_AUCTION_RESULTS_REPORT 79
3.6.20 AQA0021_AUCTION_RESULTS_REPORT_CTG_STG 80
3.6.21 ABA0035_SECURITY_MASTER _CTG_STG 80
3.6.22 ABA0009_EAUCTION_CONFIG 81
3.6.23 AQA0018_RETAIL_BID_PROC_IND 82
3.7 PREVENTING CONCURRENT LOGIN 83

4 USER INTERFACE 84

4.1 SGSLOGIN.JSP 84
4.2 LOGINFAILED.JSP 84
4.3 LOGOFF.JSP 85
4.4 TIMEOUT.JSP 85
4.5 SGSMAINMENU.JSP 85
4.6 TRANSACTIONFAILED.JSP 86
4.7 UNAUTHORISEDACCESS.JSP 86
4.8 PAGENOTFOUND.JSP 86
4.9 SELECTSECISSUE.JSP 87
4.10 VIEWHISTSEC.JSP 87
4.11 VIEWONGOINGSEC.JSP 88
4.12 VIEWSECDETAILS.JSP 88
4.13 SELECTSECDEALER.JSP 89
4.14 VIEWSECDEALERS.JSP 89
4.15 USERMAINTENANCEMENU.JSP 89
4.16 USERMAINTENANCE.JSP 90

4.17 SELECTUSER.JSP 90
4.18 ADDNEWUSER.JSP 91
4.19 BROWSEUSERS.JSP 91
4.20 UPDATEUSER.JSP 92
4.21 USERADDED.JSP 92
4.22 USERUPDATED.JSP 92
4.23 EAPPSUSERMAINTENANCE.JSP 92
4.24 ADDNEWEAPPSUSER.JSP 93
4.25 BROWSEEAPPSUSERS.JSP 93
4.26 UPDATEEAPPSUSER.JSP 94
4.27 EAPPSUSERADDED.JSP 94
4.28 EAPPSUSERUPDATED.JSP 95
4.29 SGSTENDERMENU.JSP 96
4.30 SELECTTENDERADD.JSP 97
4.31 ADDNEWTENDER.JSP 97
4.32 SELECTTENDERUPDATE.JSP 98
4.33 UPDATETENDER.JSP 98
4.34 TENDERADDED.JSP 99
4.35 TENDERUPDATED.JSP 99
4.36 TENDERDELETED.JSP 100
4.37 TENDERRESUBMIT.JSP 100
4.38 SELECTVERIFYTENDER.JSP 101
4.39 VERIFYTENDER.JSP 101
4.40 REVERIFYTENDER.JSP 102
4.41 RELEASETENDER.JSP 102
4.42 TENDERRELEASED.JSP 104
4.43 BROWSEALLTENDERS.JSP 104
4.44 VIEWUNRELTENDERS.JSP 105
4.45 ~~SELECTALLOTRESULT.JSP~~ (REMOVED JUN 2013) 105
4.46 ~~VIEWALLOTRESULT.JSP~~ (REMOVED JUN 2013) 107
4.47 SELECTTENDERREACTIVATE.JSP 107
4.48 REACTRELTENDER.JSP 108
4.49 TENDERUNRELEASED.JSP 108
4.50 AUCTIONMENU.JSP 110
4.51 AUCTIONSECPARALIST.JSP (INPUT PARAMETERS) 110
4.52 AUCTIONSECPARALIST.JSP (VERIFY PARAMETERS) 111
4.53 AUCTIONSECLIST.JSP 111
4.54 AUCTIONVIEWREPORTMENU.JSP 113
4.55 AUCTIONREPORTLIST.JSP 113
4.56 USER NOT AUTHORISED MESSAGES 114
4.57 HELP.PDF 115

5 DATA FORMAT 116

5.1 SECURITY UPDATE FILE FOR MEPS+ AND FMBS 116

5.2 AUCTION RESULT FOR MEPS+ 119
5.3 BANK AUCTION RESULT FOR FMBS 120

6 DEPLOYMENT 121

6.1 COPYING OF WEB PAGES AND SERVLET FILES TO WAS 121
6.1.1 DIRECTORY STRUCTURE ON WAS 121
6.1.2 DIRECTORY STRUCTURE ON HTTP SERVER 121
6.1.3 COPYING OF STATIC WEB PAGES TO HTTP SERVER 122
6.1.4 COPYING OF STATIC WEB PAGES TO WAS 122
6.1.5 COPYING OF JAVA CLASSES TO WAS 125
6.1.6 CONFIGURING INTERFACE WITH ACTIVE DIRECTORY (AD) SERVER 129
6.2 CREATING ORACLE TABLES 129

7 APPENDIX 130

7.1 NRIC NO VALIDATION ALGORITHM 130
7.2 CPF NO. VALIDATION ALGORITHM 133
7.3 BID CHECKSUM ALGORITHM 138
7.4 FORMULAE USED IN SGS DAILY PRICE CALCULATION 139
7.4.1 FORMULAE FOR T-BILL 139
7.4.2 FORMULAE FOR BOND 140
7.5 SGS AUCTION SAFEGUARD MECHANISM 144
7.6 ALLOCATION LIMIT FOR BIDS 146
7.7 SAMPLE LETTER TO AGD 148
7.7.1 STRUCTURE OF LETTER 148
7.8 AUTHORISED ACTIONS PER USER LEVEL 149
7.9 FRN - TENDER SUBMISSION AND AUCTION 153
7.10 EAPPS USER MAINTENANCE 153
7.10.1 HOUSEKEEPING 153
7.10.2 STEPS TO FOLLOW WHEN PRIMARY DEALER BANK CODE IS CHANGED 153
7.11 CMTB AND SGS (INFRA) - TENDER SUBMISSION AND AGD LETTER/INTERFACE FILE 154
7.12 SGS SYNDICATION 154

1 Introduction

1.1 Overview

The System Design Specifications (SDS) document was prepared as an output of the SGS eAuction System Design. This document includes the detail description of some of the major processes of the SGS eAuction System.

1.2 Audience

This document is intended for use by any person involved in the development, approval and maintenance of the SGS eAuction System Project. It is also targeted for the user who requires a high level view of the application.

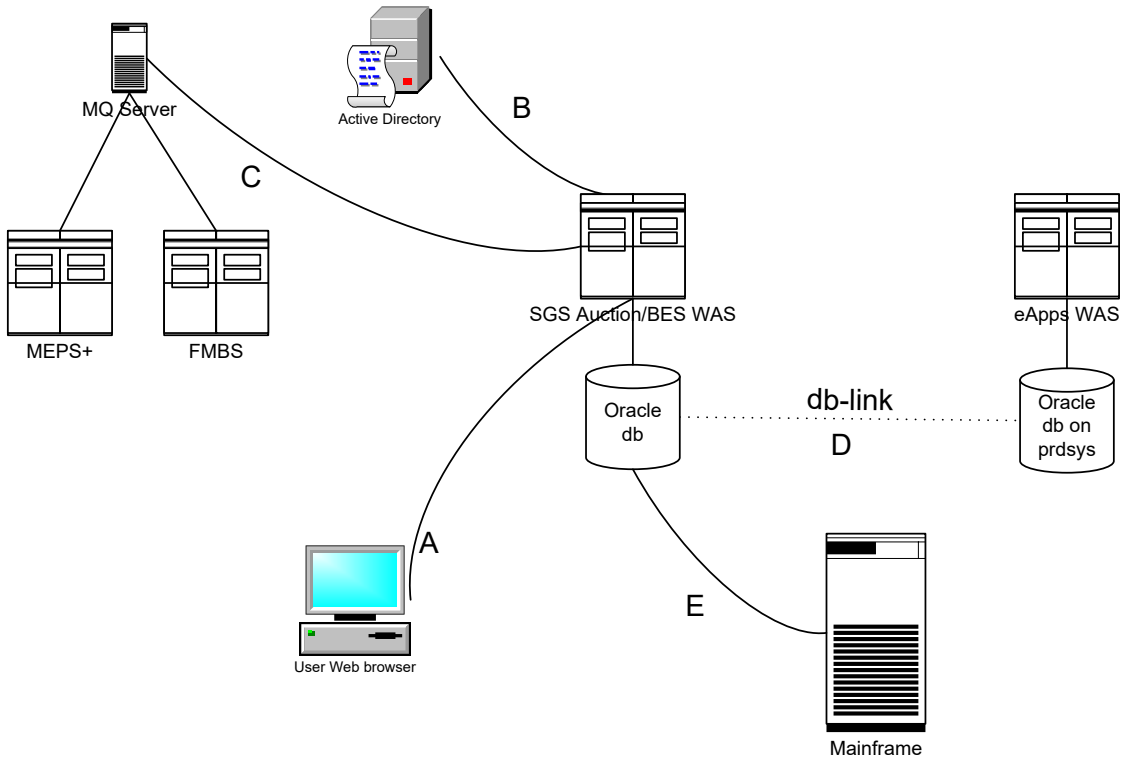
1.3 SDS Structure

This document contains:

Section 2	Processes This section focuses on the application architecture and overall system flow of the SGS eAuction System.
Section 3	System design Programs of each component of the SGS eAuction System.
Section 4	User interface Screen format of the SGS eAuction System.
Section 5	Data format Table structures, and other data formats.
Section 6	Deployment System setup and program installation.

2 Process

Overall Process of SGS eAuction System



Legend:

*A 1. Logon 2. Main menu 3. Browse SGS book entry system database 4. SGS Tender – Add a tender input 5. SGS Tender – Upload application file 6. SGS Tender – Browse all tender inputs 7. SGS Tender – Enquire/Update a tender input 8. SGS Tender – Browse all unreleased tender inputs 9. SGS Tender – Verify/Release a tender input ~~10. SGS Tender – Browse allotment result by issue code~~ 11. SGS Tender – Activate released inputs 12. SGS Auction – Input yield corridor parameters 13. SGS Auction – Verify yield corridor parameters 14. SGS Auction – Validate bids 15. SGS Auction – Run auction 16. SGS Auction – View result reports/Generate File for AGD 17. SGS Auction – Release results 18. User Maintenance	*B 1. Authenticate with Active Directory server *C 1. Assemble auction result files 2. Send auction results file via MQ *D 1. SGS Auction system uses DB-link to query Oracle database in SGS eApps system. *E 1. Mainframe uses FTD to get auction result files for 3 local banks 2. Mainframe sends auction result files to 3 local banks via MEFT

A.1. Logon

1. To access the application, User has to first use his user id and password to login via Active Directory (AD).
2. If the authentication by AD is successfully, the User will be able to see the SGS eAuction Main Menu.

Business Rules:
√ All Users will be required to logon successfully in order to execute the functions in SGS eAuction System application.
√ User will be assigned a user id and password for a specific user group.
√ Maintenance of user id and password will be on the AD end.
√ There are 7 types of user access levels defined in the application:
- Level 0: Users who are not authorized to use the application
- ~~Level 1: Users who are allowed to view record only~~ (Removed Jun 2013)
- Level 2: Users who are allowed to add/update tender record only
- Level 3: Users who are allowed to verify/release tender only, and run auction and view auction reports
- ~~Level 4: Users who are allowed to add/update/verify/release tender record only.~~ (Removed Jun 2013)
- Level 5: Users who are allowed to view auction reports.
- ~~Level 6: Users who are allowed to run auction and view auction reports.~~ (Removed Jun 2013)
- Level 9: Users who are allowed to define user privilege only.
- Level 8: Users who are allowed to View Audit Log Report
- Level 10: Users who are allowed to add/update CorpPass User details for eApps application

√ Each access level is assigned access to specific functions within the System.

A.2. Main Menu

1. The Main Menu consists of 3 groups, namely Browse Security Issues and Security Dealers List, SGS Tender, and SGS Auction.

A.3. Browse Security Issues and Security Dealers Database

1. The SGS eAuction database contains records of historical security issue (e.g. those without ISIN codes), records of ongoing security issue for tender and a list of banks/security dealers, which is updated by MEPS+ daily.
2. An User is allowed to browse the following information in the SGS eAuction database:
   - Historical Security Issues -When User selects this function, a list of historical Security Issues will be displayed.
   - Ongoing Security Issues - When User selects this function, a list of ongoing Singapore Government Securities will be displayed.
   - Security Dealers – When User selects this function, a List of Security Dealers will be displayed.

Business Rules:
√ All Users can browse the historical and ongoing Security Issues.
√ All Users can browse the Security Dealers.
√ All Users cannot add/update/delete the SGS Auction database.

A.4. SGSTender - Add a New Tender

1. Only Level 2 and Level 3 group users can access this function.
2. Users will have the choice of submitting bid using web form or using file upload.
3. When User selects using web form, he will be shown a tender form for him to key in the required information.

Business Rules:

√ Tender Number
- ~~If entered, the format TXXXX where XXXX is numeric fields.~~
- If entered, the format XXXXXX where XXXXXX is numeric fields (updated Aug 2022)
- Length must be 6 characters long and less than 500,000.

√ Issue Code
- It must be an existing issue code found in the Master Issue List.
- The issue must not be allotted.

√ Tender Date
- The format is DDMMYYYY

√ Amount Applied
- It must be >0
- It must be in multiple of 1000 with a minimum amount of 1000 for Bonds and T-Bills.
- If Amount Applied is empty, Competitive Code and Bid Yield must be empty.

√ Competitive Code
- It must be 'C' for competitive bid or 'N' for non-competitive bid.

√ Bid Yield
- It must be in range -99.99 to 99.99 for competitive bid. (0 is a valid bid yield)
- It must be =0 for non-competitive bid.
- It must be numeric up to 2 decimal places.

√ Custodian Bank
- It must be a valid trader account number found in the Bank List.

√ Custody Code
- If Applicant's type is PD, FB, RB, OB or MAS, it must be 'B'.
- Otherwise, it must be 'C' for customer.

√ Dealer Code
- It must be a valid dealer account number found in the Primary Dealer List.

√ Applicant's Name
- It must be empty when custody code = 'B'
- It must not be empty when custody code = 'C'

√ Applicant's Type
- It must be PD (Primary Dealer), FB (Full Bank), RB (Restricted Bank), OB (Offshore Bank), NB (Other Bank), MB (Merchant Bank), IA (Investment Advisor), FC (Finance Company, INS (Insurance Company), SB (Statutory Board, Government-Linked Company), IND (Individual), SGX (Singapore Exchange), MAS (Monetary Authority of Singapore, OTH (Others).

√ If Applicant's Type is 'IND', Nationality and Applicant's NRIC/Passport Number must not be empty.
- If Applicant's Type is not 'IND', Nationality, Applicant's NRIC/Passport Number and CPF Account Number must be empty.

√ Nationality
- It must be S or F or empty.
- If Applicant's Name is empty, Nationality must be empty.

√ Applicant's NRIC/Passport Number[1]
- It is not compulsory.

[1] Validates both S-series and T-series NRIC numbers.

- Singapore NRIC number's checksum must be correct.

√ Application Type
- It must not be empty for applicant type = "IND".
- Cash, CPF or SRS

√ Applicant's CPF/SRS Account Number
- It is not compulsory unless the Application Type is SRS or CPF.
  - If Application Type is CPF, to validate according to Appendix 7.2 CPF No. Validation
  - Below validation logic needs to be applied to validate the CPF number in eApps and eAuction
  - For Singapore Citizen/PR, CPF No should match the same as NRIC number (Length – 9 characters)
  - For Foreigner, CPF No starts with S or T, followed by the letter 'A', 'B', 'C', 'D,' 'F', or 'Y' (Length – 9 characters). Refer Section 7.2 for validation logic
  - If the CPF Acct No. is empty, then display the error "The CPF Account No. should not be empty."

√ Override NRIC/Passport Number
- Override must be 'Y' if NRIC/Passport Number is invalid.
- Override must be 'N' if NRIC/Passport Number is valid.
- Override must be 'N' if NRIC/Passport Number is blank.

√ Override CPF Account Number
- Override must be 'Y' if CPF Account Number is invalid.
- Override must be 'N' if CPF Account Number is valid.
- Override must be 'N' if CFP Account Number is blank.

√ Reject Code
- The default is blank.

A.5. SGSTender – Upload bid file

1. Only Level 2 and Level 3 group users can access this function.
2. When User selects using file upload, he will be shown a file upload form for him to upload file.
3. After user uploads the file, system will decrypt the file using the private key before performing the validation. The user will be shown the upload confirmation page, on which a table shows all accepted bids, and another table shows all rejected bids.
4. The user confirm the upload, he/she will be shown the upload result page. Again, on the page, a table shows all accepted bids, and another table shows all rejected bids.

Business Rules:

√ The Nominal Amount must be in multiple of one thousand with a minimum Nominal Amount of 1,000.

√ Bid yield must be numeric up to 2 decimal places.

√ Bid yield range accepted differs based on the "mnet.acceptNegativeYield" property value defined in sgsnet1.properties file
  - √ If the property value is true, then Bid yield range accepted is -99.99 to 99.99 (0 is a valid bid yield)
  - √ If the property value is empty or false, then Bid yield range accepted is 0.01 to 99.99 (0 is not a valid bid yield)

√ If Competitive Bid selected, then Bid yield must not be empty.

√ If Bid yield entered, the Competitive must be 'Y'.

√ Nationality must be S/F.

√ For Singapore citizen and PR (Nationality 'S'), the NRIC no. must be valid.

A.6. SGSTender – Browse All Tender Inputs

1. When User selects this function, he will be shown a list of all tender Inputs. From the same display, User can select the tender that he wants to update/delete.

Business Rules:
√ Only Level 2 and, Level 3 ~~and Level 4~~ group users can access this function.

A.7. SGSTender – Update a Tender Input

1. When User enters an existing tender number and selects this function, he will be shown details of the tender. At this point, the User can choose to update or delete the tender.

Business Rules:
√ Only Level 2 and Level 3 group users can access this function.
√ The tender number, if entered, must be existing record.
√ User can only update unreleased tender inputs.

A.8. SGSTender – Browse All Unreleased Tender Inputs
1. When Users selects this function, they will be shown a list of unreleased tenders. Users can select the tender number that they want to verify and release.

Business Rules:
√ Only Level 3 ~~and Level 4~~ group users can access this function.

A.9. SGSTender – Verify/Release a Tender Input

1. When Users enters an existing tender number and selects this function, they will be shown a tender form whereby they will need to enter all the information to verify the tender input. A message error is returned if no such tender exists.
2. Upon completion, Users can select the Verify button. Any inconsistencies in the data will be flagged out. Users must either check the data entered, or the content of the tender input to determine the error.
3. Users can re-enter the correct information and click the Verify button again. The application will respond with a tender verified message. The Users can then release the tender input by clicking the Release button. The System will responds with a 'tender successfully released' message.

Business Rules:
√ Only Level 3 ~~and Level 4~~ group users can access this function.
√ The tender number, if entered, must be existing record.
√ The application will check against user id that the same user cannot add/update and verify/release the same tender input.
√ Tender input that has been released cannot be released again.

~~A.10. SGSTender – Browse Allotment Result by Issue Code~~ (Removed Jun 2013)

1. When User specifies the issue code or tender number and selects this function, the application will return all tender records that have been successfully allotted. The User can click the Adjust button to adjust the allotment amount.

Business Rules:

√ Only Level 2 and Level 3 ~~and Level 4~~ group users can access this function.
√ The tender number or issue code, if entered, must be existing record.
√ Only Level 3 and Level 4 group users can use the Adjust button.

A.11. SGSTender – Activate Released Tender Input

1. This function allows the User to unrelease selected released tender input.

Business Rules:
√ Only Level 3 ~~and Level 4~~ group Users can access this function.
√ Only released tender inputs can be unreleased.
√ Users must enter a valid tender number existing in the record.

A.12. SGSAuction – Validate bids

1. This function allows the Users to validate the tender bids from SGS eApps.

Business Rules:
√ Only Level 3 ~~and Level 6~~ group users can access this function.

A.13. SGSAuction – Input yield corridor parameters

1. This function allows Users to input yield corridor parameters for bonds.

Business Rules:
√ Only Level 2 and Level 3 Users can access this function.

A.14. SGSAuction – Verify yield corridor parameters

1. This function allows Users to verify yield corridor parameters for bonds.

Business Rules:
√ Only Level 3 Users can access this function.

A.15. SGSAuction – Run Auction

1. Allows Users to execute the auction module to process the tender transactions.

Business Rules:
√ Only Level 3 ~~and Level 6~~ Users can access this function.
√ After auction, a report table will be displayed.

A.16. SGSAuction – View result reports/Generate File for AGD

1. Allows Users to view auction reports.
2. Allows specific users to generate a letter containing information on the issuance/redemption/interests payment of SGS Bonds and T-Bills to AGD, based on the issue/redemption/interests payment date entered.

Business Rules:
√ Level 3 and Level 5 ~~and Level 6~~ Users can view result reports.
√ Only Level 3 Users can generate file on SGS Bonds and T-Bills for AGD.

A.17. SGSAuction – Release results

1. Allows Users to release results to eApps.
2. Allows Users to release results to MEPS+ and FMBS.

Business Rules:
√ Only Level 3 ~~and Level 6~~ group users can access this function.
√ Files "auction.txt" and "secupdt.txt" will be sent to MEPS+.
√ Files "auction_fmbs.txt" and "secupdt_fmbs.txt" will be sent to FMBS.

A.18. User Maintenance

1. SGS Auction User Maintenance
   Only Level 9 group Users (eg. System Administrator) can access SGS Auction User Maintenance functions. The SGS Auction User Maintenance has the following functions:
   a. Add User
   b. Browse Users
   c. Update User
2. eApps User Maintenance
   Only Level 10 group Users (e.g. ITD-USC team) can access eApps User Maintenance. The eApps User Maintenance has the following functions:
   a. Add User
   b. Browse User

Business Rules:
√ Allows the System Administrator to determine the access level or user group for each user.
√ Allows the ITD-USC team to add/update the eApps CorpPass details for each user.

B.1. Authenticate with Active Directory server
1. After servlet on SGS eAuction Web site receives user's logon information, it will authenticate with the Active Directory server.
2. If the authentication is failed, the servlet will give Users a logon failure web page.

Business Rules:
√ Servlet only uses each User's logon id and password for authentication.

C.1. Assemble text file

1. After an auction is successfully finished, result files will be generated, which are auction results for banks and general result for security update. However, they are not called "auction.txt", "secupdt.txt", "auction_fmbs.txt" and "secupdt_fmbs.txt" at this stage.

Business Rules:
√ Only result for the current day will be retrieved.

C.2. Send result files via MQ

1. After user triggers "Release Result", auction system will decrypt the files generated by "RunBankResult" and rename respective result files to "auction.txt", "secupdt.txt", "auction_fmbs.txt" and "secupdt_fmbs.txt" and put them under pre-defined data directory for MQ program.
2. MQ program will then send those files to MEPS+ and FMBS.

Business Rules:
√ “auction.txt” contains auction results for all banks.
√ “auction_fmbs.txt” only contains auction results for MAS.
√ “secupdt.txt” and “secupdt_fmbs.txt” are same.

D.1. SGS eAuction system uses DB-link to use Oracle tables in SGS eApps system.

1. SGS eAuction system uses DB-link to read/write Oracle database in SGS eApps system.

Business Rules:
√ SGS eAuction system needs to have an ID and password that can access tables in SGS eApps Oracle database via DB-link.

E.1. Mainframe uses FTD to get auction result files for 3 local banks
1. Mainframe uses FTD to get auction result files for 3 local banks

Business Rules:
√ Mainframe needs to have an ID and password that can access PCSSYS90.
√ The FTP job is controlled by AutoSys

E.2. Mainframe sends auction result files to 3 local banks via MEFT
1. Mainframe sends auction result files to 3 local banks via MEFT

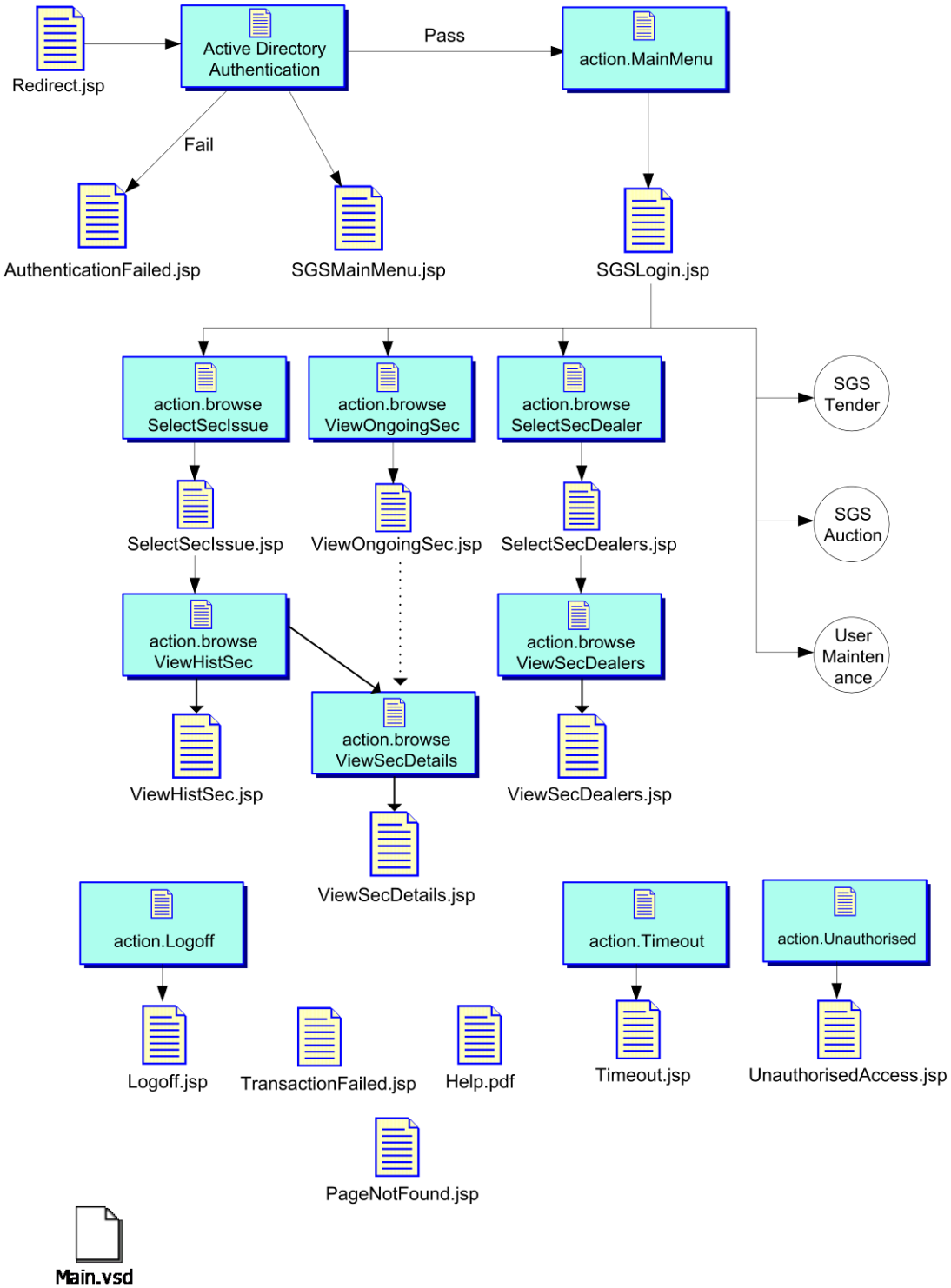
Business Rules:
√ Mainframe sends result files to DBS and UOB using NJE.
√ Mainframe sends result files to OCBC using ConnectDirect.

3 System design

The system architecture is illustrated as follows:

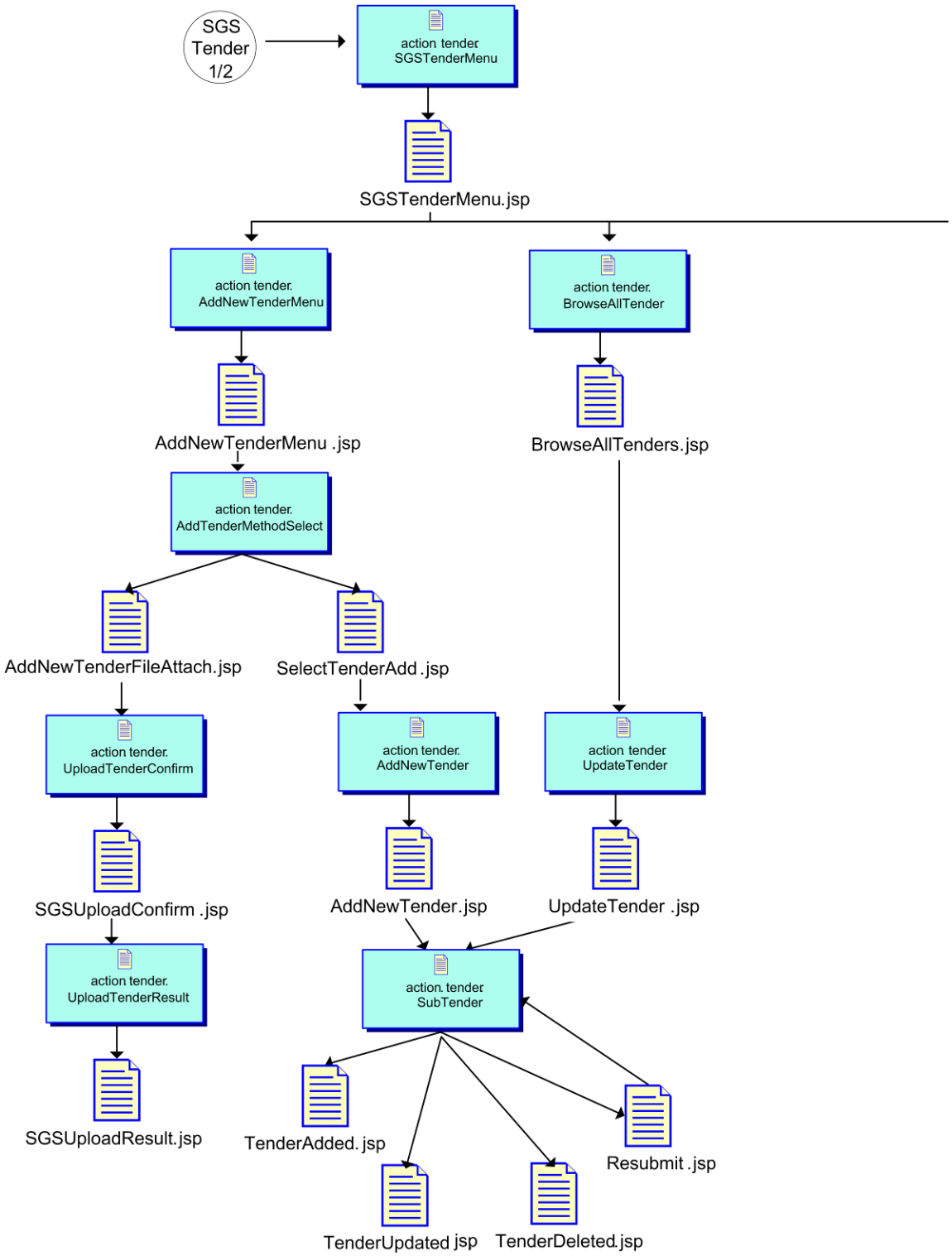
3.1 Web front-end interface design:

The structure of web pages is organized in the following chart:



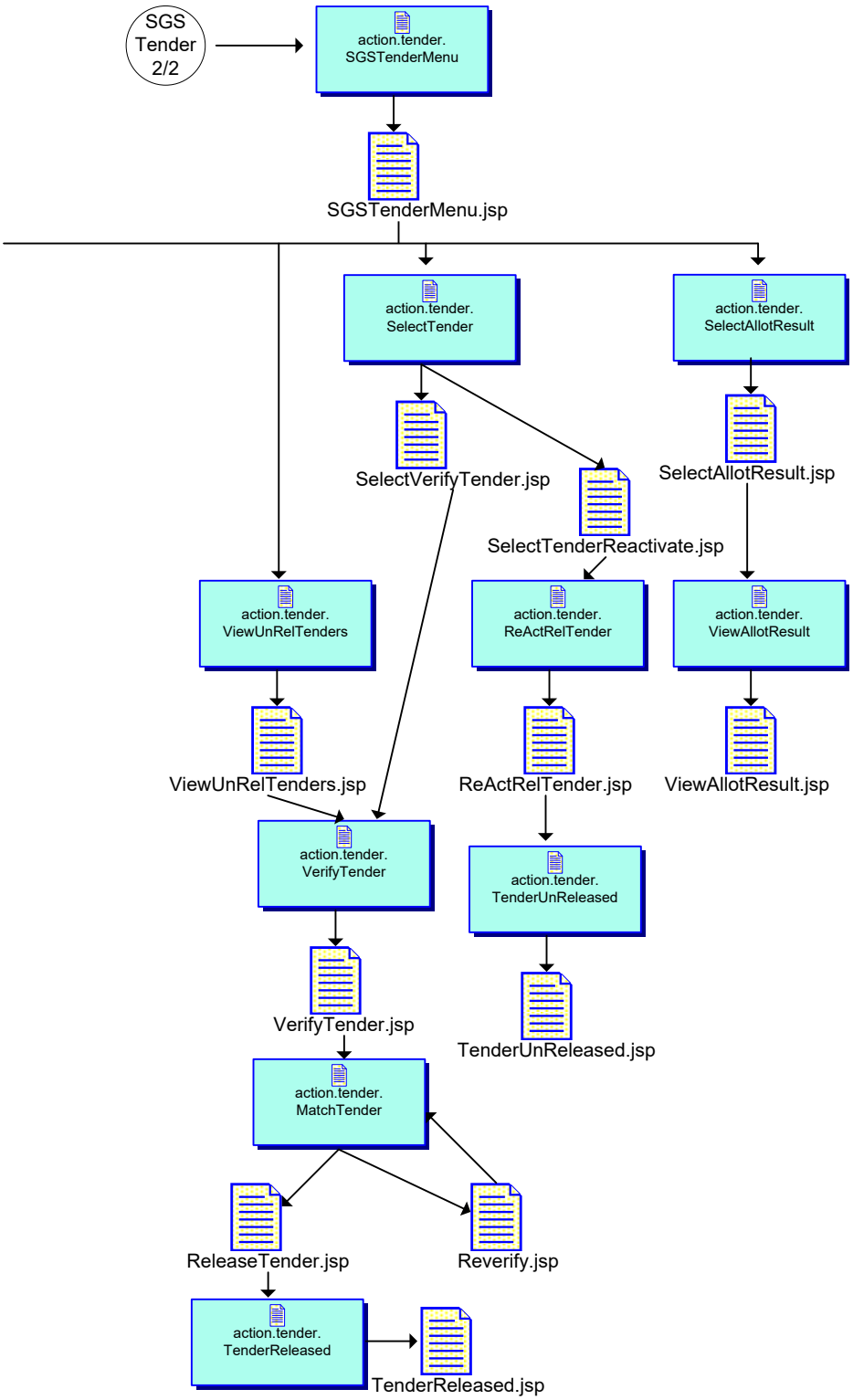
(Visio File for Main diagram)

SGS Tender (Updated May 2013):

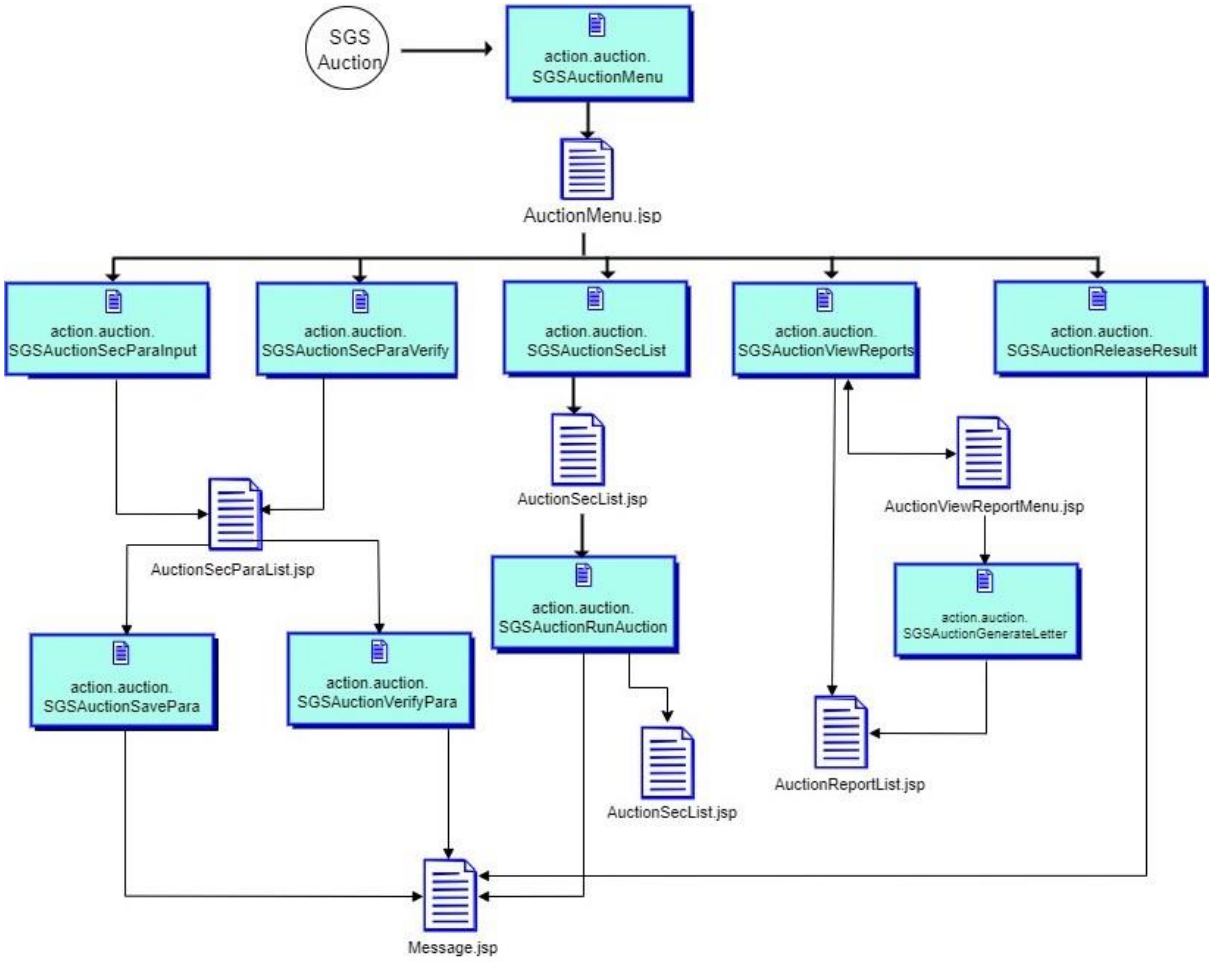


SGSTender-1.vsd

(Visio File for SGSTender-1 diagram)


SGS Tender 2/2
action.tender. SGSTenderMenu
SGSTenderMenu.jsp
action.tender. SelectTender
action.tender. SelectAllotResult
SelectVerifyTender.jsp
SelectTenderReactivate.jsp
SelectAllotResult.jsp
action.tender. ViewUnRelTenders
action.tender. ReActRelTender
action.tender. ViewAllotResult
ViewUnRelTenders.jsp
ReActRelTender.jsp
ViewAllotResult.jsp
action.tender. VerifyTender
action.tender. TenderUnReleased
VerifyTender.jsp
TenderUnReleased.jsp
action.tender. MatchTender
ReleaseTender.jsp
Reverify.jsp
action.tender. TenderReleased
TenderReleased.jsp

SGS Auction (Updated Jan 2024):

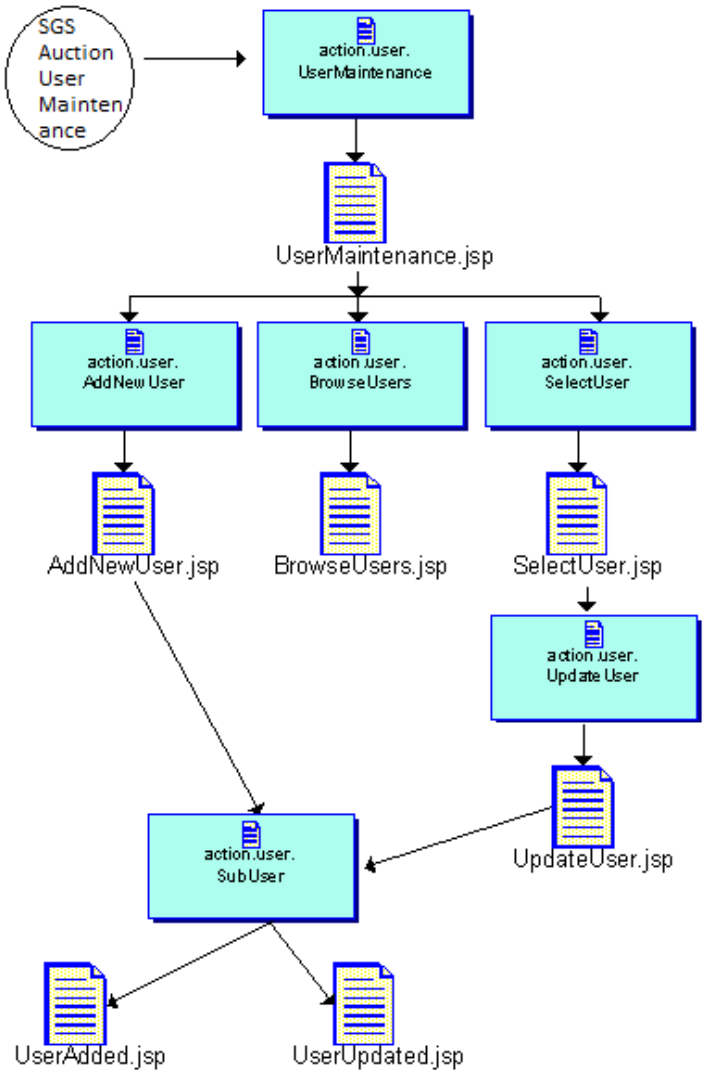


SGSAuction.vsdx

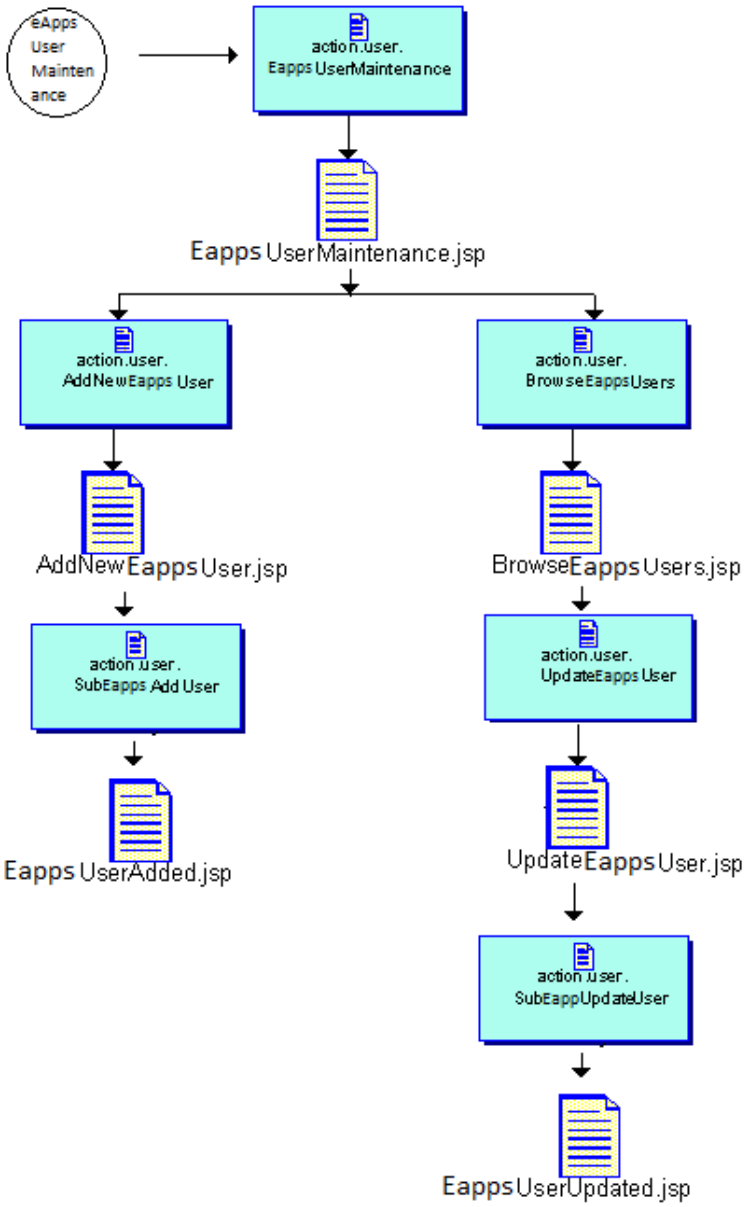
(Visio File for SGSAuction Diagram)

User Maintenance:

SGS Auction User Maintenance



eApps User Maintenance


eApps User Mainten ance
action.user. Eapps UserMaintenance
Eapps UserMaintenance.jsp
action.user. AddNewEapps User
action.user. BrowseEappsUsers
AddNewEappsUser.jsp
BrowseEappsUsers.jsp
action.user. SubEapps Add User
action.user. UpdateEapps User
Eapps UserAdded.jsp
UpdateEappsUser.jsp
action.user. SubEappUpdateUser
EappsUserUpdated.jsp

The web pages are mainly dynamic JSP pages that are generated by back-end servlets. Each web page is generally made up of a header and a body. The content layout of each web page is defined in a xml file called tiles-defs.xml.

3.1.1 JSP Layout Templates

Two JSP templates are used to provide a standard layout for the JSP pages. They are:

- defaultLayout.jsp - This template is made up of four area segments. There are the header, title, body and error message.

- defaultLayoutNoTitle.jsp - This template consists of the header, a body and error message.

3.1.2 Header

There are 7 types of headers used in JSP pages.

- header.jsp - This header has a menu bar that allows User to select the main menu functions.
- header_nomenu.jsp - This header has no menu bar.
- header_tender.jsp - This header has a menu bar that allows User to select the tender functions.
- header_usermaintenance.jsp - This header has a menu bar that allows User to select user maintenance functions.
- header_auction.jsp – This header has a menu bar that allows User to select the auction functions.
- header_user.jsp – This header has a menu bar that allows Users to select SGS Auction User Maintenance functions.
- header_eappsuser.jsp – This header has a menu bar that allows Users to select eApps User Maintenance functions.

JSP Pages

3.1.3 SGSLogin.jsp

The User will see this login page when they have not been authenticated by Active Directory server to access the SGS eAuction application. The User will submit the user id and password in the login form. If a user logons successfully, the SGS eAuction Main Menu will be displayed. A login failure page will be displayed if the user has entered an incorrect user id or password. The “Reset” button will clear the textbox fields if the User clicks on it.

3.1.4 Redirect.jsp

After the User has login-ed successfully, this page will be shown. After 2 seconds, this page will automatically re-direct the User to SGS eAuction Main Menu.

3.1.5 SGSMainMenu.jsp

This is the main menu page of the SGS eAuction application. The page will display the SGS eAuction functions that the User is authorized to access to. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The seven menu items on the menu bar are:

- Historical Securities
- Ongoing Securities
- Security Dealers
- SGSTender
- SGSAuction
- User Maintenance
- Help
- Exit

3.1.6 Logoff.jsp

When the User clicks on the "Exit" button on the menu bar, a "Logoff.jsp" will be displayed.

On the page "Logoff.jsp", the User can click on the "Logout" button to logout successfully. This will bring the User to the login page.

3.1.7 Timeout.jsp

If a User logons to SGS eAuction application and there is no activity by the User for 15 minutes, the User's session will be automatically timeout. The page "Timeout.jsp" will be shown.

On the page "Timeout.jsp", the User can click on the "Login" button to go to the login page.

3.1.8 UnauthorisedAccess.jsp

If a user is not authorized to access the function, the page "UnauthorisedAccess.jsp" will be shown.

The user can click on the "Back" button to go to the last accessed page.

3.1.9 PageNotFound.jsp

If a user has entered an invalid URL, the page "PageNoteFound.jsp" will be shown.

Browse Historical Security Issues

3.1.10 SelectSecIssue.jsp

This page allows the User to select historical security issues to view. The User can enter an issue code, a partial issue code or blank for the search. An issue code is made up of 8 alphanumerical characters. If a partial issue code is provided, the search program will return a list of security issues that match the search text input. The search program will return all security issues when no issue code is provided.

Clicking the "Cancel" button will return the User back to main menu page.

3.1.11 ViewHistSec.jsp

This page displays a list of historical security issue in table form. The page will display a “No security issue record found” message when there is no search result.

There are 5 columns in the table. They are Sequence Number, Issue Code, Coupon Rate, Issue Description and Due Date.

The servlet “action.browse.ViewHistSec” passes information of securities to the JSP file “ViewHistSec.jsp”. The code on the JSP page will automatically format those issues in the table format.

The User can view the details of a security issue by clicking its specific issue code in the table.

3.1.12 ViewSecDetails.jsp

This page displays the details of the security issue that is selected by User from ViewHistSec.jsp or ViewOngoingSec.jsp.

The servlet “action.browse.ViewSecDetails” passes information of security issue to the JSP file “ViewSecDetails.jsp”. The code on the JSP page will automatically format the details in table format.

The User can access other functions by clicking on the items on the horizontal menu bar.

Browse Ongoing Security Issues

3.1.13 ViewOngoingSec.jsp

This page displays a list of Ongoing Security Issues in table form. The page will display a “No ongoing security is available” message when there is no security issue available for tender.

There are 5 columns in the table. They are Issue Code, Issue Description, Issue Date, Due Date and Status.

The servlet “action.browse.ViewOngoingSec” passes information of ongoing securities to the JSP file “ViewOngoingSec.jsp”. The code on the JSP page will automatically format those issues in the table format.

The User can view the details of a security issue by clicking its specific issue code in the table.

Browse Security Dealers

3.1.14 SelectSecDealer.jsp

This page allows the User to select Security Dealers to view. The User can enter a 4-digit bank code, partial bank code or blank for the search. If a partial dealer code is provided, the search program will return a list of security dealers that match the text input. The search program will return all security dealers when no bank code is provided.

Clicking the “Cancel” button will return the User back to main menu page.

3.1.15 ViewSecDealers.jsp

This page displays a list of security dealers in table form. The page will display a "No Security Dealer Record Found" message when there is no search result.

There are 5 columns in the table – Sequence Number, Account Number, Account Name and Autodebit Status.

The servlet "action.browse.ViewSecDealers" passes information of securities dealers to the JSP file "ViewSecDealers.jsp". The code on the JSP page will automatically format those security dealers in the table format.

The User can continue by clicking on the items on the horizontal menu bar.

SGS Tender

3.1.16 SGSTenderMenu.jsp

This is the SGSTender menu page of the SGS eAuction application. The page will display the SGSTender functions that the User is authorized to access to. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The nine menu items on the menu bar are:

- Main
- Add New Tender Input
- Browse All Tender Inputs
- Update Tender Input
- Browse UnReleased Tender Inputs
- Verify/Release Tender Inputs
- ~~View Allotment Result By Issue Code~~ (Removed Jun 2013)
- Reactivate Released Tender Inputs
- Trigger Retail Bid
- Exit

3.1.17 AddNewTenderMenu.jsp

This page allows the User to select methods of submitting bids – via web form, or via file upload.

3.1.18 SelectTenderAdd.jsp

This page allows the User to create a new tender by providing the security issue code and tender number. The application will validate the security issue code and check for duplicate tender number.

3.1.19 AddNewTender.jsp

On this page, a form is displayed for User to input tender details.

The following fields are static fields, of which the values are generated by the Java code on the page: tender number, issue code and tender date.

The following fields are editable and require input from User: amount applied, competitive code, bid yield, applicant's name, nationality, applicant's NRIC/Passport number, application type, Applicant's CPF/SRS account number, applicant type, custodian bank, custodian code, dealer code, reject code.

3.1.20 AddNewTenderFileAttach.jsp

On this page, a file upload form is displayed for User to browse a file and upload.

3.1.21 TenderAdded.jsp

If a tender from the User has been accepted, a tender added acknowledgement page will be shown.

On this page, the user id that submits the bid application, the issue code, the tender date and the tender details are shown. User may print this page for future reference.

The information are provided by the servlet "action.tender.SubTender". The information is automatically formatted to HTML format by the Java code on the "TenderAdded.jsp".

3.1.22 SGSUploadConfirm.jsp

~~After user uploads a bid application file, backend program will decrypt the file using the private key and parses the file and returns the validation result.~~
After user uploads a bid application file, backend program will validate the file name, file size, perform antivirus scan, check if file already exists and returns the validation result.

If there is any fatal error, the whole file will be rejected.

~~If there is no, or minor, error, this page displays two tables – one for all accepted bids, and the other for all rejected bids.~~

This page will display message to confirm file submission to be saved.

If file is already existing, this page will display alert that by confirming the action, existing file will be replaced with newly uploaded file.

User can click on button "Confirm" to confirm bid submission.

The information are provided by the servlet "action.tender.UploadTenderConfirm". The information is automatically formatted to HTML format by the Java code on the "SGSUploadConfirm.jsp".

3.1.23 SGSUploadResult.jsp

After user confirms a bid application file upload, backend program returns the result.

~~Again this page displays two tables – one for all accepted bids, and the other for all rejected bids.~~

This page displays details of the userid who uploaded the bid application file, the file name, and the processing date time.

User can click on button "Print" to print hardcopy for reference.

The information are provided by the servlet “action.tender.UploadTenderResult”. The information is automatically formatted to HTML format by the Java code on the “SGSUploadResult.jsp”.

3.1.24 BrowseAllTenders.jsp

This page displays a list of Tender Inputs in a table. The page will display a “No tender input record found” message when there is no tender submission record.

In the table, there are 9 columns – Sequence No., Tender No., Issue Code, Dealer Code, Cust Bank, Cust Code, Bid Yield, Amount Applied, Release Status. User can click on the tender number to update the tender details.

At the bottom of this page, user can click on the button “>>” to navigate through the next page. User can also click on the button “<<” to go back to the previous page.

The servlet “action.tender.BrowseAllTenders” passes information of all tenders to the JSP file “BrowseAllTenders.jsp”. The code on the JSP page will automatically format those issues in the table format.

Clicking the “Return” button will bring the User back to SGSTenderMenu.jsp page.

3.1.25 SelectTenderUpdate.jsp

This page allows the User to enter the security issue code and tender number of a tender to update. The application will check for valid security issue code and tender number. The application will also check if the Users are attempting to update/delete the tender reactivated/input by themselves.

3.1.26 UpdateTender.jsp

On this page, a form is displayed for User to update the tender details. The form displays the original tender information details for a specific tender.

The following fields are static fields, of which the values are generated by the Java code on the page: tender number.

The following fields are editable by User: tender date, Issue code, amount applied, competitive code, bid yield, applicant’s name, nationality, applicant’s NRIC/Passport No., application type, Applicant’s CPF/SRS account no., applicant type, custodian bank, custodian code, dealer code, reject code.

User can either click “Update” button to update the tender details or “Delete” button to delete the tender if the tender was not reactivated or created by them.

3.1.27 TenderUpdated.jsp

If a tender from the User has been accepted, this tender updated acknowledgment page will be shown.

On this page, the user id that submits the bid application, the issue code, the tender date and tender details are shown. User may print this page for future reference.

The information of user id, security code, transaction date and time, and tender details are provided by the servlet "action.tender.SubTender". The information is automatically formatted to HTML format by the Java code on the "TenderUpdated.jsp".

3.1.28 TenderDeleted.jsp

If a tender from the User has been deleted, this page will be shown.

On this page, the user id that deletes the bid application, the issue code, the tender date and time, and tender number are shown. User may print this page for future reference.

The information of user id, security code, transaction date and tender number are provided by the servlet "action.tender.SubTender". The information is automatically formatted to HTML format by the Java code on the "TenderDeleted.jsp".

3.1.29 ReSubmit.jsp

The static fields and editable fields on this page are the same as those of "UpdateTender.jsp".

After a User submits an application (either via AddNewTender.jsp, UpdateTender.jsp or TenderResubmit.jsp itself), the back-end program will do some validation using the business rule spelled out in Section 2. If the validation fails, this page will be shown. Java code on the page automatically formats the error information to HTML format, i.e. to display the error message and to highlight some fields in red colour.

This page starts by showing the User the error message. All the fields, which caused validation failures, are highlighted in red colour.

3.1.30 ViewUnRelTenders.jsp

This page displays a list of Unreleased Tenders in a table. The page will display a "No unreleased tender input record found" message when there is no unreleased tender record available.

In the table, there are 8 columns – Sequence No., Tender No., Issue Code, Dealer Code, Cust Bank, Cust Code, Bid Yield, Amount Applied. User can click on the tender number to verify a specific tender.

The servlet "action.tender.ViewUnRelTenders" passes information of unreleased tender inputs to the JSP file "ViewUnRelTenders.jsp". The code on the JSP page will automatically format those tenders in the table format.

Clicking the "Return" button will bring the User back to SGSTenderMenu.jsp page.

3.1.31 SelectVerifyTender.jsp

This page allows the User to select a tender to verify by providing the security issue code and tender number. The application will check for valid security issue code and tender number.

3.1.32 VerifyTender.jsp

This is the page for User to verify the tender application.

On this page, an empty form with the specific tender number for verification is displayed for User to input the tender details. The static fields and editable fields on this page are the same as those of "UpdateTender.jsp".

On the page there is a "Verify" button. User can click on the button to verify the tender application after he has completed the form.

The servlet "action.tender.VerifyTender" passes information of specified tender to the JSP file "VerifyTender.jsp". The code on the JSP page will automatically format the tender details in the form format.

3.1.33 ReleaseTender.jsp

This is the page for User to release the tender application when verification is successful.

On this page, a form with the specific tender number for release is displayed for User. The fields n this page are the same as those of "UpdateTender.jsp". All the fields are static and not editable.

On the page there is a "Release" button. User can click on the button to release the tender application.

The servlet "action.tender.MatchTender" conducts the verification process and passes information of specified tender to the JSP file "ReleaseTender.jsp". The code on the JSP page will automatically format the tender details in the form format.

3.1.34 TenderReleased.jsp

If a tender from the User has been successfully released, this page will be shown.

On this page, the user id that released the bid application, the issue code, the tender date and time, and tender number are shown.

There is also a table showing the details of the tender application. User may print this page for future reference.

The information of user id, security code, transaction date and time, and tender number are provided by the servlet "action.tender.TenderReleased". The information is automatically formatted to HTML format by the Java code on the "TenderReleased.jsp".

3.1.35 ReVerify.jsp

The static fields and editable fields on this page are the same as those of "VerifyTender.jsp".

After a User verifies a tender (either via VerifyTender.jsp or TenderReVerify.jsp itself), the back-end program will do some matching between the tender input details and the unreleased tender version. If the matching fails, this page will be shown. Java code on the page automatically formats the error information to HTML format, i.e. to display the error message and to highlight some fields in red colour.

This page starts by showing the User the error message. All the fields, which caused matching failures, are highlighted in red colour.

3.1.36 ~~SelectAllotResult.jsp~~ (Removed Jun 2013)

This page allows the User to the allotment result of a selected security issue. The application will validate the security issue code.

3.1.37 ~~ViewAllotResult.jsp~~ (Removed Jun 2013)

This page displays a list of tender allotment in a table. The page will display a "No allotment result record found" message when there is no allotment result available.

In the table, there are 8 columns – Sequence No., Tender No., Issue Code, Dealer Code, Cust Bank, Cust Code, Bid Yield, Amount Applied.

The servlet "action.tender.ViewAllotResult" passes information of allotment result to the JSP file "ViewAllotResult.jsp". The code on the JSP page will automatically format those tender allotments in the table format.

Clicking the "Return" button will bring the User back to SGSTenderMenu.jsp page.

3.1.38 SelectTenderReactivate.jsp

This page allows the User to select a released tender by entering the security issue code and tender number. The application will validate the security issue code and tender number.

3.1.39 ReActRelTender.jsp

The static fields, editable fields and validation on this page are the same as those of "UpdateTender.jsp".

User can clicks the "Re-activate" button to unrelease a released tender.

The servlet "action.tender.ReActRelTender" passes information of released tender to the JSP file "ReActRelTender.jsp". The code on the JSP page will automatically format the released tender details in the form format.

3.1.40 TenderUnReleased.jsp

If the User has been successfully unreleased a tender, this page will be shown.

On this page, the user id that unreleased the bid application, the issue code, the tender date and time, and tender number are shown.

There is also a table showing the details of the tender application. User may print this page for future reference.

The information of user id, security code, transaction date and time, and tender number are provided by the servlet "action.tender.TenderUnReleased". The information is automatically formatted to HTML format by the Java code on the "TenderUnReleased.jsp".

3.1.41 UploadTenderList.jsp

After user selects “Trigger Retail Bid”, this page will display a table showing a list of all uploaded retail bid files.

A button “Delete” will be shown for each file. User can click on the button to delete specific file. Also, a button “Trigger” will also be shown for user to trigger retail bid process by creating a signal file that will be retrieve later by file watcher.

If there is no file uploaded for today, this page will display a message of “No files uploaded.”.

The list and information of uploaded files are provided by the servlet “action.tender.UploadTenderList”.

3.1.42 AcknowledgementFileDownloadMenu.jsp

After user selects “Download Retail Bid Acknowledgement Submitted via eAuction”, this page will display a form “Download acknowledgement file for date xx/xx/xxx (dd/mm/yyyy).”

“xx/xx/xxxx” is a textbox for user to input the processing date.

Clicking the “Submit” button will bring user to “AcknowledgementFileList.jsp”.

3.1.43 AcknowledgementFileList.jsp

This page will display a table showing list of acknowledgement files.

A button “Download” will be shown for each acknowledgement file. User can click on the button to download specific file and will be handle by the servlet servlet “action.tender.AcknowledgementFileDownload”.

If acknowledgement files do not exist for today, this page will display a message of “No file available.”.

The list and information of acknowledgement files are provided by the servlet “action.tender.AcknowledgementFileDownloadMenu”.

SGS Auction

3.1.44 AuctionMenu.jsp

This is the SGSAuction menu page of the SGS eAuction application. The page will display the auction functions that the User is authorized to use. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The seven menu items on the menu bar are:

- Main
- Input Auction Para (i.e. yield corridor parameters)
- Verify Auction Para (i.e. yield corridor parameters)
- Run Auction

- View Reports
- Release Result
- Exit

3.1.45 AuctionSecList.jsp

After user selects “Run Auction”, this page will display all the issues for auction today, which Auction System reads from eApps database (MNET). A button “Run Auction” will also be shown.

If there is no ongoing security for today, this page will display a message of “No ongoing issues”.

3.1.46 AuctionSecParaList.jsp

After user selects “Input Auction Para” or “Verify Auction Para”, this page will display all the issues for auction today, which Auction System reads from eApps database (MNET). For bond issue, the text fields “upper bound”, “lower bound”, “reference point”, and “MAS max amount” are shown. A button “Submit” will also be shown, for user to save the values into database.

If there is no ongoing security for today, this page will display a message of “No ongoing issues”.

3.1.47 AuctionViewReportMenu.jsp

After user selects “View Reports”, this page will display will display a table, which display 4 menu items – “View today’s reports” and “View xx/xx/xxx’s reports” to view Auction reports; “Generate today’s T-Bills and Bonds” and “Generate xx/xx/xxx’s T-Bills and Bonds” to generate a letter for AGD containing information on the Issuance, Interests Payment and redemption of SGS Bonds and T-Bills. “xx/xx/xxxx” is the textbox for user to input. The date entered for the letter should either be the issue date, interest payment date or maturity date

3.1.48 AuctionReportList.jsp

This page will display either a list of reports for the auction date or the letter for AGD, based on the option selected. The reports/letter will be based on the date selected by the user. The auction reports are “MSABXT00”, “MSABRT05/1”, “MSABRT05/2”, “MSABRT05/2”, “MSABRT08/1”, “MSABRT08/2”, “MSABRT06” and “MSABRT11”. The letter to AGD is “Letter to AGD”.

If there is no AGD letter for the selected date, a message “No report/letter for this date” will be displayed.

User Maintenance

3.1.49 UserMaintenanceMenu.jsp

This is the User Maintenance Menu page of the SGS eAuction application. The page will display the user maintenance functions that the User is authorized to use. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The four menu items of the menu bar are:

- Main

- SGS Auction User Maintenance
- eApps User Maintenance
- Exit

3.1.50 UserMaintenance.jsp

This is the SGS Auction User Maintenance menu page of the SGS eAuction application. The page will display the SGS Auction user maintenance functions that the User is authorized to use. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The five menu items on the menu bar are:

- Main
- Add New User
- Browse Users
- Update User
- Exit

3.1.51 AddNewUser.jsp

On this page, a form is displayed for User to add new user and customise user details. If the new user id added is not unique, the application will return the same page with error message.

A drop down list is provided to select the user level for the new User.

The seven user roles are:

- Access Level 0 – Unauthorised User
- ~~Access Level 1 – Browse Only~~ (Removed Jun 2013)
- Access Level 2 – Add and Update Tender
- Access Level 3 – Verify and Release Tender, run auction, view auction reports
- ~~Access Level 4 – Add/Update/Verify/Release Tender~~ (Removed Jun 2013)
- Access Level 5 – View Auction Reports
- ~~Access Level 6 – Run Auction Process, view auction reports~~ (Removed Jun 2013)Access Level 9 – System Administrator
- Access Level 8 – To access View Audit Log report in SGS Auction User Maintenance
- Access Level 10 – ITD-USC team to access eApps User Maintenance

JavaScript code on the page will automatically convert User's input to upper case.

3.1.52 UserAdded.jsp

If a User has been successfully created, this acknowledgment page will be shown.

On this page, the new user id and the access level are shown.

The information of user id and access level are provided by the servlet "action.user.UserAdded". The information is automatically formatted to HTML format by the Java code on the "UserAdded.jsp".

3.1.53 BrowseUsers.jsp

This page displays a list of users and authorization level for each function in a table. The page will display a "No User Record Found" message if the user id does not exist.

In the table, there are 5 columns – User Id, Browse, SGSTender, SGSAuction, View Report and User Maintenance.

The servlet "sgs.SGSBrowseUsers" passes information of users to the JSP file "BrowseUser.jsp". The code on the JSP page will automatically format the user details in the table format.

Clicking the "Return" button will bring the User back to "UserMaintenance.jsp" page.

3.1.54 SelectUser.jsp

In this page, the User is allowed to update a selected user id. The User must enter a valid user id. If the user id is not valid, the application will return the same page with specific error message.

After the User id is entered, the User can either click the button "Search" or hit Enter to start the search process to get the specific security dealer. Clicking the "Return" button will bring the User back to main menu page.

3.1.55 UpdateUser.jsp

On this page, a form is displayed for User to update another user's details. The form displays the original details for that user.

User can select the new access level for that user and click "Update" button to update that user details.

The User is not allowed to edit his own user profile.

3.1.56 UserUpdated.jsp

If a tender from the User has been successfully updated, this page will be shown.

On this page, the created user id and the different levels of access are shown.

The information of user id and access levels are provided by the servlet "sgsbes.UserUpdated". The information is automatically formatted to HTML format by the Java code on the "UserUpdated.jsp".

3.1.57 EappsUserMaintenance.jsp

This is the eApps User Maintenance menu page of the SGS eAuction application. The page will display the eApps user maintenance functions that the User is authorized to use. A brief description of each function is provided. User can click on the buttons on the menu bar for the desired functions.

The four menu items on the menu bar are:

- Main
- Add User
- Browse Users
- Exit

3.1.58 AddNewEappsUser.jsp

On this page, a form is displayed for User to add new/update inactive to active eApps user. If the Bank Code/UEN/NRIC, FIN or Foreign ID added is not unique, the application will return the same page with error message.

JavaScript code on the page will automatically convert User's input to upper case.

3.1.59 EappsUserAdded.jsp

If an eApps User has been successfully created/updated to active, this acknowledgment page will be shown.

On this page, system generated token user id along with UEN/NRIC, FIN or Foreign ID are shown.

The information of system generated token user id and UEN/NRIC, FIN or Foreign ID are provided by the Struts action class "action.user.SubEappsAddUser". The information is automatically formatted to HTML format by the Java code on the "EappsUserAdded.jsp".

3.1.60 BrowseEappsUsers.jsp

This page displays

- Active/Inactive drop down
  - Active – Users who can able to login to access eApps application
  - Inactive – Users whose access has been suspended or deleted and will not be able to access eApps application
- Bank Code drop down
  - Unique bank code found in both tables
    - Primary Dealer table
    - CorpPass Mapping table
- List of eApps users in a table for all Active eApps users for all Bank Code. The page will display a "No User Found" message if eApps user does not exist.
- Link on the Token User Id is provided to Update/Delete only for Active Users

In the table, there are 7 columns – Token User Id, Bank Code, UEN, NRlC,FIN or Foreign ID, Last Login Date for eApps, Last Modified Date and Updated By.

The Struts Action class "action.user.BrowseEappsUsers" passes information of eApps users to the JSP file "BrowseEappsUsers.jsp". The code on the JSP page will automatically format the eApps user details in the table format.

3.1.61 UpdateEappsUser.jsp

In this page, the User is allowed to update/delete/inactivate a selected Token user id from BrowseEappsUsers page. On this page, a form is displayed with eApps user details for the selected Token User ID.

If the selected Token User ID has not logged into eApps application

- User can update the eApps user details by clicking "Update" button
- User can delete the eApps user details by clicking "Delete" button

If the selected Token User ID has already logged into eApps application

- User can update the eApps user to Inactive by clicking “Set Inactive” button

3.1.62 EappsUserUpdated.jsp

If the selected Token User ID has been successfully updated/deleted/inactivated, this page will be shown.

On this page, system generated token user id along with UEN/NRIC, FIN or Foreign ID are shown.

The information of System generated token id, UEN/NRIC, FIN or Foreign ID are provided by the Struts action class “action.user.SubEappsUpdateUser”. The information is automatically formatted to HTML format by the Java code on the “EappsUserUpdated.jsp”.

Miscellaneous

3.1.63 help.pdf

This page displays the SGS eAuction user help manual.

3.1.64 TransactionFailed.jsp

If there is any exception, which occurs in the back-end processing, an error page will be generated from the servlet and displayed to the User. On the page, an error message which specifies the specific servlet that fails is shown. The User is recommended to contact the System Administrator.

Only WAS run-time errors are shown on the error page. Other errors such as “File not found”, “Internal server error” are handled by the HTTP server itself.

3.1.65 errormsg.jsp

If there are any validation errors, which occur in the back-end processing, the relevant error message extracted from the “application.properties” properties file and displayed to the User. The User is recommended to follow the instructions given.

3.2 Web back-end design

The back-end design consists of Java Beans, Java Form Beans, Java servlets and JSP logic design, database access, integration with Active Directory server, error logging and some miscellaneous Java classes. The back-end java classes are grouped in four packages, namely model, form, action and utils.

Package - model

3.2.1 Java Beans

Four Java Beans are used in the application:

- model.BankMaster.java – The Java class "model.BankMaster" is used to represent a security dealer. There are get/set methods to read or write a security dealer's attributes.

- model.OneEappsTransaction.java – The Java class "model.OneEAppsTransaction" is used to represent a tender form inputted via SGS eApps, which may consists of 5 tender bids. There are get/set methods to read or write a tender bid's attributes.

- model.OneTransaction.java – The Java class "model.OneTransaction" is used to represent a tender bid. There are get/set methods to read or write a tender bid's attributes.

- model.TransLimit.java – The Java class "model.TransLimit" is used to represent a transaction limit that a PD or a non-PD has. There are get/set methods to read or write a tender bid's attributes.

- model.User.java - The Java class "model.User" is used to represent a user. There are get/set methods to read or write a user's attributes.

- model.AuthorisedAction.java – The java class “model.AuthorisedAction” is used to represent an Authorized Action and its descriptions. There are get/set methods to read or write the Authorised Actions.

- model.EappsUser.java – The Java class “model.EappsUser.java” is used to represent an eApps User. There are get/set methods to read or write eApps User’s details.

- model.RetailBidInd.java – The java class “model.RetailBidInd” is used to represent a retail bid process indicator. There are get/set methods to read or write a retail bid indicator attributes.

Package - form

3.2.2 Java Form Beans

Four form beans are used in the application:

- form.LogonForm.java – The Java class "form.LogonForm" is used to represent a form for User to logon to eAuction System. There are get/set methods to read or write a form's attribute.

- form.SearchForm.java – The Java class "form.SearchForm" is used to represent a form for User to input issue code or bank code. There are get/set methods to read or write a form's attribute.

- form.SearchTenderForm.java – The Java class "form.SearchTenderForm" is used to represent a form for User to input issue code and tender number. There are get/set methods to read or write a form's attribute.

- form.TenderForm.java – The Java class "form.TenderForm" is used to represent a form for User to input tender details. There are get/set methods to read or write a form's attribute.

- form.UserForm.java – The Java class "form.UserForm" is used to represent a form for User to input user id. There are get/set methods to read or write a form's attribute. This class performs below validation of the input user id

    - User id is mandatory
    - User id should not contain whitespace in-between
    - If there is space before and/or after the User id, the whitespaces are trimmed, and the User id is accepted

- form.EappsUserForm.java – The Java class “form.EappsUserForm” is used to represent a form for User to input Bank Code/UEN/NRIC, FIN or Foreign Id. There are get/set methods to read or write a form’s attribute. This class performs below validation of the input fields

    - Bank Code is mandatory
    - UEN is mandatory
        - Length should be between 1 and 10
        - Should not contain whitespace in-between
    - NRIC, FIN, or Foreign Id is mandatory
        - Length should be between 1 and 20
        - Should not contain whitespace in-between
    - If there is space before and/or after the input fields, the whitespaces are trimmed, and the input field values are accepted

- form.BrowseEappsUserForm.java – The Java class “form.BrowseEappsUserForm” is used to represent a form for User to select the Bank Code/Active Indicator. There are get/set methods to read or write a form’s attribute.

Package - action

3.2.3 MainMenu.java

This servlet is used to determine the access level of the User and display the Main Menu page.

Processing Logic

- If session variable “session.user” is null,
    - Get system’s date
    - Obtain the user id from Active Directory server
    - Get User object from database using the user id.
    - Update User’s last login date using the system date.
    - Set User object to session variable “session.user”
- Get User access level from the User object.
- Get the authorized actions allowed for the User, based on the User’s access level.
- Get the relevant function names and descriptions based on the User’s access level.
- Set function names to session variable “session.title”.
- Set function descriptions to session variable “session.desc”.

- Set user id to session variable “session.currentuser”.
- Bind session variable with eAuction configuration.
- Forward control to SGSMainMenu.jsp.

Validation Checks
- When the User ID cannot be obtained from Active Directory server, the servlet will display LoginFailed.jsp.
- When User object is not found in the database, the servlet will display LoginFailed.jsp.
- When the servlet encounters exception, the servlet will display TransactionFailed.jsp.

3.2.4 Logoff.java

This servlet is used to handle user’s log off.

Processing Logic
- Invalidate all session variables.
- Forward control to Logoff.jsp

3.2.5 Timeout.java

This servlet is used to handle timeout. There will be timeout when there is no User’s activity on the application for 15 min.

Process Logic
- Invalidate all session variables.
- Forward control to Timeout.jsp

Package - action.user

3.2.6 UserMaintenanceMenu.java

This Struts action class is used to generate the User Maintenance Menu page

Processing Logic
- Get User object from session variable “session.user”.
- Check if the User is authorized to access this sub-function.
- Set user function names to session variable “session.title”.
- Set user function descriptions to session variable “session.desc”.
- Forward control to UserMaintenanceMenu.jsp

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.7 UserMaintenance.java

This Struts action class is used to generate the SGS Auction User Maintenance Menu page.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Set user function names to session variable "session.title".
- Set user function descriptions to session variable "session.desc".
- Set access descriptions to session variable "session.access".
- Forward control to UserMaintenance.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.8 AddNewUser.java

This servlet is used to generate a User form to add new user.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to AddNewUser.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp
- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.9 BrowseUsers.java

This servlet is used to generate a list of users for browsing.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get all users from database
- Set the array of users to session variable "session.users".
- Forward control to BrowseUsers.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.

- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.10 SelectUser.java

This servlet is used to generate a form for the system administrator to select a specific user to update.

Processing Logic
- Get User object from session.user.
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to SelectUser.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.11 SubUser.java

This servlet is used for adding new user or updating existing user to the database. The servlet is called by AddNewUser.jsp or UpdateUser.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Obtain the selected access level from user form in AddNewUser.jsp or UpdateUser.jsp.
- Get system date
- If servlet is called from AddNewUser.jsp
  - ❖ Get the new user id from user form in AddNewUser.jsp or UpdateUser.jsp.
  - ❖ Check for duplicate user id in the User database.
  - ❖ Create a new User object with the new user id and access level
  - ❖ Insert new User object to User database
- If servlet is called from UpdateUser.jsp
  - ❖ Get user id to update from session variable "session.update_user".
  - ❖ Remove session variables "session.update_user" and "session.update_level".
  - ❖ Get User object from User database using the user id.
  - ❖ Set access change date of the User object using system date.
  - ❖ Set access level of the User object using the selected access level
  - ❖ Update User object to User database
- Set user id to session variable "session.subuser".
- Set access level to session variable "session.subacess".
- Forward control to UserAdded.jsp or UserUpdated.jsp.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When duplicate user id is found, the servlet will display AddNewUser.jsp with the relevant error message.

3.2.12 UpdateUser.java

This servlet is used to retrieve the User object using the specific user id obtained from SelectUser,jsp or BrowseUsers.jsp. The User object is then displayed in UpdateUser.jsp for enquiry or update.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 9.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- If servlet is called from SelectUser.jsp, get user id to update from the user search form.
- If servlet is called from BrowseUsers.jsp, get user id to update from query string.
- Trim the input user id before fetching the User details from database
- Gets the User object from User database using the specific user id.
- Get the access level description that maps to the access level obtained from the User object.
- Set the specific user id to session variable "session.update_user".
- Set the access level description to session variable "session.update_level".
- Forward control to UpdateUser.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is not 9, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When the user is updating his own user profile, the servlet will display UserMaintenance.jsp with the relevant error message.
- When User object for update is not found, the servlet will display SelectUser.jsp with the relevant error message.

3.2.13 EappsUserMaintenance.java

This Struts action class is used to generate the eApps User Maintenance Menu page.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Set user function names to session variable "session.title".
- Set user function descriptions to session variable "session.desc".
- Set the Bank code found in the Primary Dealer DB table to session variable "session.bankcode".
- Forward control to EappsUserMaintenance.jsp.

Validation Checks

- When User object is not found, the action class will forward control to LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.14 AddNewEappsUser.java

This Struts action class is used to generate an EappsUser form to add new eApps user.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Forward control to AddNewEappsUser.jsp.

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.15 SubEappsAddUser.java

This Struts action class is used for adding new / update inactive to active eApps user to the database. This is called by AddNewEappsUser.jsp.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Check if the bank code selected is found in the List of Bank Code stored in the session variable "session.bankcode".
- Get the new user details from EappsUser Form in AddNewEappsUser.jsp.
- Check if the user details found in eApps User database.
    - ❖ If User details submitted is a new eApps user
        - Create a new EappsUser object with the Bank Code, UEN, **Base64 encoded NRIC, FIN or Foreign ID**, System generated Token User ID, Active Indicator set to "Y"
        - **System generated Token user Id** to start with the pattern 'CxxxxT01' and once reaches 'CxxxxT99', the next Token User ID generated would be 'DxxxxT01'. Similar pattern would be followed until the Token User ID reaches 'ZxxxxT99'. Where **xxxx** is the Bank Code selected.
            - Warning message as below would be shown once the Token User ID reaches 'ZxxxxT01' until 'ZxxxxT99' in the Success message page **'WARNING: System is reaching the maximum limit for generating the Token User ID. Please reach out to eApps Application team to housekeep before all the Tokens are used (ZxxxxT99).'**
            - Error message as below would be shown when trying to add user after reaching 'ZxxxxT99'
              'System has reached maximum limit to generate Token User ID. Please reach out to eApps Application team to housekeep in order to add new user.'

- **Note**: If any of the Token User ID is missing in between in the above pattern i.e. if 'CxxxxT55' is deleted, then during the next user creation those missing Token User ID would be assigned first.
  - Insert new EappsUser object to eApps User database table ABA0030_CORP_PASS_MAPPING@MNET
- If User details submitted is an existing Inactive eApps user
  - Create an EappsUser object with existing system generated Token User ID, Active Indicator set to "Y"
  - Update the EappsUser object to eApps User database table ABA0030_CORP_PASS_MAPPING@MNET
- If User details submitted is an existing Active eApps user
  - Error message as below would be shown in AddNewEappsUser.jsp **'Unable to add user: UEN/NRIC, FIN or Foreign ID is already in use.'**

- Set Token user id to session variable "session.subEappsTokenUserId".
- Set whether the user is updated from Inactive to Active in session variable "session.isEappsUserUpdated".
- Set UEN to session variable "session.subEappsAddCpEntityId".
- Set NRIC, FIN or Foreign ID to session variable "session.subEappsAddCpUid".
- Set whether to display warning message or not in session variable "session.displayWarningMsg".
- Forward control to EappsUserAdded.jsp.

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.16 BrowseEappsUsers.java

This Struts action class is used to generate a list of eApps users for browsing.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Get the Bank Code details from both Primary Dealer and Corp Pass Mapping table and store in session variable "session.bankCodeBrowse"
  - If any of the PD's bank code is changed, eApps User Maintenance page need to show the old Bank codes User details which have been made inactive during the bank code change
- Store "Active", "Inactive" list in the session variable "session.activeList" for Active/Inactive drop-down values
- Get eApps users details from database and set the array of users to session variable "session.eappsusers"
  - All active eApps Users details from database will be fetched for initial load of Browse Users page
  - Based on the input drop down values of Active/Inactive and the selected Bank Code, fetch the corresponding details from database
  - Mask the **NRIC, FIN or Foreign ID** except the last 4 characters
- Forward control to BrowseEappsUsers.jsp.

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp.

- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.17 UpdateEappsUser.java

This Struts action class is used to retrieve the EappsUser object using the specific TokenID obtained from BrowseEappsUsers.jsp. The EappsUser object is then displayed in UpdateEappsUser.jsp to update/delete/inactivate eApps user.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Get TokenId from query string and store in session variable "session.eappsUpdateTokenId".
- Gets the EappsUser object from eApps User database for the specific TokenId and store in session variable "session.eappsUpdateUser".
- If eApps User exists and active and
    - ❖ If eApps User has not logged into eApps application (Last Login Date to eApps is null), then display **NRIC, FIN or Foreign ID** in clear format after Base64 decoding in UI
    - ❖ If eApps User has logged into eApps application (Last Login Date to eApps is not null), then mask the **NRIC, FIN or Foreign ID** except the last 4 characters after Base64 decoding and display in UI
- Forward control to UpdateEappsUser.jsp.

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When EappsUser object is not found or inactive, the action class will forward control to UpdateEappsUser.jsp with the relevant error message

3.2.18 SubEappsUpdateUser.java

This Struts action class is used for update/delete/inactivating eApps user in the database. This is called by UpdateEappsUser.jsp.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Check if the TokenId matches with the TokenId stored in session variable "session.eappsUpdateTokenId"
- Check if the bank code selected is found in the List of Bank Code stored in the session variable "session.bankcode".
- Get the form action and eApps User details from the session variable "session.eappsUpdateUser" to validate and perform update/delete/inactivate user
- Inactivate user
    - ❖ If Form action is "Set Inactive" and Last Login Date to eApps is not null, then update the eApps User details for the specific TokenId with Active Indicator set to "N" to database table ABA0030_CORP_PASS_MAPPING@MNET
    - ❖ Set UEN to session variable "session.subEappsUpdateCpEntityId".
    - ❖ Set masked NRIC, FIN or Foreign ID to session variable except last 4 characters to "session.subEappsUpdateDisplayCpUid".

- Delete user
  - If Form action is "Delete" and Last Login Date to eApps is null, then delete the eApps User details for the specific TokenId from database table ABA0030_CORP_PASS_MAPPING@MNET
  - Set UEN to session variable "session.subEappsUpdateCpEntityId".
  - Set Base64 decoded NRIC, FIN or Foreign ID to session variable "session.subEappsUpdateDisplayCpUid"
- Update user
  - If Form action is "Update" and Last Login Date to eApps is null, then
    - Get the new user details from EappsUser Form in UpdateEappsUser.jsp
    - If new user details already exist, forward control to UpdateEappsUser.jsp with relevant error message
    - Else update the new eApps User details for the specific TokenId to database table ABA0030_CORP_PASS_MAPPING@MNET
  - Set UEN to session variable "session.subEappsUpdateCpEntityId".
  - Set Base64 decoded NRIC, FIN or Foreign ID to session variable "session.subEappsUpdateDisplayCpUid"
- Set Token user id to session variable "session.subEappsTokenUserId".
- Set whether the user is updated from Active to Inactive in session variable "session.isEappsUserUpdated".
- Set whether the user is deleted or not in session variable "session.isEappsUserDeleted".
- Forward control to EappsUserUpdated.jsp.

Validation Checks
- When User object is not found, the action class will forward control to LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

Package - action.browse

3.2.19 SelectSecIssue.java

This servlet is used to generate a search form for user to select a security issue code to view.

Processing Logic
- Get User object from "session variable" session.user.
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to SelectSecIssue.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.20 ViewHistSec.java

This servlet is used to generate the historical securities for viewing. The servlet is called from SelectSecIssue.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get issue code from search form in SelectSecIssue.jsp.
- Get the security issues from SecurityMaster database using the issue code.
- Set the array of security issues to session variable "session.issues".
- Forward control to ViewHistSec.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.21 **ViewOngoingSec.java**

This servlet is used to generate the historical securities for viewing.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get ongoing security issues from SecurityMaster database.
- Set the array of ongoing security issues to session variable "session.ongoingissues".
- Forward control to ViewOngoingSec.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.22 **ViewSecDetails.java**

This servlet is used to generate security issue details for viewing. The servlet is called from ViewHistSec.jsp or ViewOngoingSec.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get issue code from query string in ViewHistSec.jsp or ViewOngoingSec.jsp.
- Get the security issue details of the specific issue code from SecurityMaster database.
- Convert getInt_date from DDMMYYYY to DDMM.

- Set the security issue details to session variable "session.secdetails".
- Forward control to ViewSecDetails.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9 or 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.23 **SelectSecDealer.java**

This servlet is used to generate a search form for the User to select the bank code for a security dealer.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to SelectSecDealer.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.24 **ViewSecDealers.java**

This servlet is used to generate security dealers' detail for viewing. The servlet is called from SelectSecDealer.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is not 0.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get bank code from search form in SelectSecDealer.jsp.
- Get the bank objects from BankMaster database using the bank code.
- Set the array of bank objects to session variable "session.dealers".
- Forward control to ViewSecDealers.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 0, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

Package - action.tender

3.2.25 SGSTenderMenu.java

This servlet is used to generate the SGSTender Menu page.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Set tender function names to session variable "session.title".
- Set tender function descriptions to session variable "session.desc".
- Forward control to SGSTenderMenu.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.26 AddNewTenderMenu.java

This servlet is used to generate the add new tender Menu page.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to AddNewTenderMenu.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.27 AddNewTenderMethodSelect.java

This servlet is let user select submission method – via web form, or via file upload.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- If user select via web form, forward control to AddNewTender.jsp
- If user select via file upload, forward control to AddNewTenderFileAttach.jsp.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.28 UploadTenderConfirm.java

This servlet is used to validate the uploaded file.

Processing Logic

- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get file from AddNewTenderFileAttach.jsp.
- ~~Decrypt and parse the file~~
- ~~Get header, details header, details, and trailer.~~
- ~~Validate each bid.~~
- ~~Bid yield range accepted differs based on the "mnet.acceptNegativeYield" property value defined in sgsnet1.properties file~~
    - ~~If the property value is true, then Bid yield range accepted is -99.99 to 99.99 (0 is a valid bid yield)~~
    - ~~If the property value is empty or false, then Bid yield range accepted is 0.01 to 99.99 (0 is not a valid bid yield)~~
- ~~If the bid passes validation, store it in Vector AcceptedV.~~
- ~~If the bid fails in validation, store it in Vector RejectedV.~~
- ~~Store AcceptedV and RejectedV in session variables.~~
- Check if file name matches valid names in eApps config table.
- Check the uploaded file size with the max file size from session variable "session.maxfile.sizemb".
- Perform antivirus scan if session variable "session.antivirusScan.enabled" is true.
- Check whether there are ongoing securities.
- Set processing date as today's date.
- Check if file already exists.
    - If file exists, set relevant message to request attribute "message" and set true to request attribute "isFileExist".
    - If file doesn't exist, set relevant message to request attribute "message" and set false to request attribute "isFileExist".
- Set retail bid file object to session variable "session.retailbidfile".
- Forward to SGSUploadConfirm.jsp.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- User will be redirected to message.jsp with relevant error message if uploaded file name is not match with valid names in eApps config table.

- User will be redirected to message.jsp with relevant error message if uploaded file size is greater than maximum file size defined from session variable "session.maxfile.sizemb".
- User will be redirected to message.jsp with relevant error message if antivirus scan is failed.

3.2.29 UploadTenderResult.java

~~This servlet saves uploaded bids in database.~~
This servlet saves uploaded file in upload directory.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- ~~Get AcceptedV and RejectedV from session variables~~
- ~~Store AcceptedV and RejectedV in session variables.~~
- Get bidFile object from session variables "session.retailbidfile".
- Delete file from upload directory.
- Save file from bidFile object to upload directory.
- Set file name to request attribute "uploadedfilename".
- Set processing date to request attribute "fileprocessingdate".
- Set date time to request attribute "datetime".
- Set success message to request attribute "message".
- Forward to SGSUploadResult.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.30 SelectTender.java

The servlet is used to generate the relevant select form for user to specify the issue code and form number for adding new tender, updating existing tender, verifying unreleased tender and reactivating released tender.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Depending on the query string that call this servlet, forward control to one of these JSP pages:
  - ❖ SelectTenderAdd.jsp,
  - ❖ SelectTenderUpdate.jsp,
  - ❖ SelectTenderVerify.jsp or
  - ❖ SelectTenderReactivate.jsp

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is not 2, 3 or 4, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.31 AddNewTender.java

This servlet is used to validate the issue code and the new tender number obtained from SelectTenderAdd.jsp. If validation checks are successful, the servlet will forward control to AddNewTender.jsp to display the tender form.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get issue code and form number from search tender form in SelectTenderAdd.jsp.
- Check for valid tender number for restriction up to 499,999.
- Get security issue object from SecurityMaster database using issue code.
- Check duplicate tender number for the same issue code from Transaction database.
- Generate system date
- Compare the tender date of security issue object with system date
- If no duplicate tender is found, set the security issue object and form number to session variables "session.addtsec" and "session.addtform" respectively.
- Forward control to AddNewTender.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 3, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When security issue's tender date occurs before system date, the servlet will display SelectTenderAdd.jsp with the relevant error message.
- When duplicate tender transaction is found, the servlet will display SelectTenderAdd.jsp with the relevant error message.
- When invalid issue code is found, the servlet will display SelectTenderAdd.jsp with the relevant error message.

3.2.32 BrowseAllTenders.java

This servlet is used to generate all tenders in Transaction database to display in BrowseAllTenders.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)

- Check if the User is authorized to access this sub-function.
- ~~Get all tender objects with record type 'TE' from Transaction database.~~
- ~~Set the array of tender objects to session variable "session.alltenders".~~
- Parse and fetch valid numeric page number.
- Calculate pagination index and page count.
- Get all tender objects with record type 'TE' from Transaction table.
- Set the vector of tender objects to session variable "session.alltenders".
- Set page count to request attribute "pageCount"
- Forward control to BrowseAllTenders.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.33 UpdateTender.java

This servlet is used to generate a tender form with details of a specific tender for user to do update.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- If servlet is called from SelectTenderUpdate.jsp, issue code and form number are obtained tender search form in SelectTenderUpdate.jsp.
- If servlet is called from BrowseAllTenders.jsp, issue code and form number are obtained from query string defined in BrowseAllTenders.jsp.
- Gets the security issue object from SecurityMaster database using issue code.
- Gets the tender object from Transaction database using issue code and form number with record type 'TE'.
- Check if the Tender was reactivated/added by the same user and redirect to BrowseAllTenders.jsp with error message if the tender was reactivated/added by the user.
- Set security issue object to session variable "session.updatetsec".
- Set form number to session variable "session.updatetform".
- Set tender object to session variable "session.updatettrans".
- Forward control to UpdateTender.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 3, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If the tender is added/reactivated by the same user, the servlet will display SelectTenderUpdate.jsp with the relevant error message.
- When the tender has already been verified, the servlet will display SelectTenderUpdate.jsp or BrowseAllTenders.jsp with the relevant error message.

- When tender object is not found using the issue code and form number, the servlet will display SelectTenderUpdate.jsp or BrowseAllTenders.jsp with the relevant error message.
- When security issue object is not found using the issue code, the servlet will display SelectTenderUpdate.jsp or BrowseAllTenders.jsp with the relevant error message.

3.2.34 SubTender.java

This servlet is used to validate the tender transaction and update tender details to the Transaction database.

Processing Logic

- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get tender form details from AddNewTender.jsp or UpdateTender.jsp.
- If servlet is called from AddNewTender.jsp
  - ❖ Get security issue object from session variable "session.addtsec".
  - ❖ Get form number from session variable "session.addtform".
  - ❖ Remove session variable "session.addtsec".
  - ❖ Remove session variable "session.addtform".
  - ❖ Set indicator type to "Add".
- If the request to the servlet is to Update the tender (From UpdateTender.jsp)
  - ❖ Get security issue object from session variable "session.updatetsec".
  - ❖ Get form number from session variable "session.updatetform".
  - ❖ Get tender object from session variable "session.updatettrans"Remove session variable "session.updatetsec".
  - ❖ Remove session variable "session.updatetform".
  - ❖ Remove session variable "session.updatettrans".
  - ❖ Set indicator type to "Update".
- If the request to the servlet is to Delete the tender (From UpdateTender.jsp)
  - ❖ Get security issue object from session variable "session.updatetsec".
  - ❖ Get form number from session variable "session.updatetform".
  - ❖ Get tender object from session variable "session.updatettrans".Remove session variable "session.updatetsec".
  - ❖ Remove session variable "session.updatetform".
  - ❖ Remove session variable "session.updatettrans".
  - ❖ Set indicator type to "Delete".
- If servlet is called from TenderResubmit.jsp
  - ❖ Get security issue object from session variable "session.resubmittsec".
  - ❖ Get form number from session variable "session.resubmittform".
  - ❖ Get indicator type from session variable "session.resubmitind".
  - ❖ Remove session variable "session.resubmittsec".
  - ❖ Remove session variable "session.resubmittform".
  - ❖ Remove session variable "session.resubmitind".
- Generate System date.
- If the indicator type is "Delete"
  - ❖ Set record type to 'DE' and record update details in the transaction object.
  - ❖ Update the tender object in Transaction database.
  - ❖ Set form number in session variable "session.deletetform".
  - ❖ Set issue code to session variable "session.deletetsec".
  - ❖ Forward control to TenderDeleted.jsp
- If the indicator type is "Add" or "Update", the following rules are applied to validate the tender:

- ❖ Applied amount must not be empty in the bid section.
- ❖ Applied amount should be entered for a bid in the bid section.
- ❖ 'Y' or 'N' should be selected for competitive or non-competitive bid respectively in the bid section.
- ❖ Applied amount must be in digits.
- ❖ Application for Treasury Bills or Government bonds must be made in multiples of 1,000 and at the minimum nominal amount of 1,000.
- ❖ Yield must be entered for competitive bid.
- ❖ Yield must be empty for non-competitive bid.
- ❖ Yield should be numeric and the value range is -99.99 to 99.99. (0 is a valid yield)
    - All backend processes are able to process both positive and negative values.
- ❖ Applicant Type must not be NIL.
- ❖ Applicant Type must be 'MAS' when Dealer Code is 2500.
- ❖ Custody Bank must be 2500 when Dealer Code is 2500.
- ❖ Applicant Type must not be 'MAS' when Dealer Code is not 2500.
- ❖ Custody Bank must not be 2500 when Dealer Code is not 2500.
- ❖ Nationality must be entered for an individual.
- ❖ NRIC or Passport number must be entered for an individual.
- ❖ NRIC/Passport number override must be 'Y' when NRIC or Passport number is invalid.
- ❖ NRIC/Passport number override must be 'N' when NRIC or Passport number is valid.
- ❖ NRIC/Passport number override must be 'N' when NRIC or Passport number is for foreigner.
- ❖ If the Application Type is CPF, validate CPF number for Singaporean and Foreigner:
    - CPF account number override must be 'Y' when CPF account number is invalid.
    - CPF account number override must be 'N' when CPF account number is valid.
- ❖ Nationality must be empty when Applicant Type is not 'IND'.
- ❖ NRIC/Passport number must be empty when Applicant Type is not 'IND'.
- ❖ CPF/SRS account number must be empty when Applicant Type is not 'IND'.
- ❖ CPF/SRS account number must be entered if the Application Type is SRS or CPF.
- ❖ CPF account number override must be 'N' when CPF account number is empty.
- ❖ Custody Bank must be entered.
- ❖ Custody Bank must be the same as Dealer Code when Applicant Type is PD (Primary Dealer).
- ❖ Applicant Type must be PD when Dealer Code, not equal to 2500, is the same as Custody Bank and Custody Code is 'Bank'.
- ❖ Custody Code must be 'Bank' when Applicant Type is PD, RB, FB, OB or MAS.
- ❖ Applicant Name must be empty when Applicant Type is PD, RB, FB, OB or MAS.
- ❖ Custody Code must be 'Bank's customer' when Applicant Type is NB, MB, IA, FC, INS, SB, IND, SGX or OTH.
- ❖ Applicant Name must be 'Bank's customer' when Applicant Type is NB, MB, IA, FC, INS, SB, IND, SGX or OTH.
- ❖ Custody Bank should be 4 digits.
- ❖ Custody Bank and Dealer Code are 2500 or Custody Bank should be equal or more than 7000.
- ❖ Custody Bank should be found in BankMaster database.
- ❖ Autodebit Indicator for Custody Bank should be 'Y'.
- ❖ Dealer Code must not be empty.
- ❖ Dealer Code should be 4 digits.
- ❖ Dealer Code should be found in BankMaster database.

- If validation errors are found:
    - ❖ Set security issue object to session variable "session.resubmittsec".
    - ❖ Set form number to session.variable "session.resubmittform".
    - ❖ Set error status indicator to session variable "session.bl".
    - ❖ Set error messages to session variable "session.errMsg".
    - ❖ Set tender form object to session variable "session.tender".
    - ❖ Set the indicator to session variable "session.resubmitind".
    - ❖ Forward control to TenderReSubmit.jsp
- If no validation error is found:

- ❖ Create and set values to new transaction object.
- ❖ Set tender object to session.variable "session.subtrans".
- ❖ Set security issue object to session variable "session.subsec".
- ❖ Set current user id to session variable "session.currentuser".
- ❖ Insert or update tender object to Transaction database depending on the indicator type.
- ❖ Forward control to TenderAdded.jsp or TenderUpdated.jsp depending on the indicator type.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.When tender form object is not found, the servlet will display TransactionFailed.jsp.
- When the insert or update to database fails, the servlet will display TransactionFailed.jsp.

3.2.35 ViewUnRelTenders.java

This servlet is used to generate tenders, which need to be verified and released, for display in ViewUnRelTenders.jsp.

Processing Logic

- Get User object from session variable "session.user".
- ~~Check User's access level is 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get all unreleased tender objects from Transaction database.
- Set the array of unreleased tender objects to session variable "session.unreleasedtenders".
- Forward control to ViewUnRelTenders.jsp.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.36 VerifyTender.java

This servlet is used to generate a tender form of a specific tender for user to do verification.

Processing Logic

- Get User object from session variable "session.user".
- ~~Check User's access level is 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- If servlet is called from SelectVerifyTender.jsp, the issue code and form number are obtained search tender form in SelectVerifyTender.jsp.
- If servlet is called from ViewUnRelTenders.jsp, issue code and form number are obtained from query string defined in ViewUnRelTender.jsp.
- Get security issue object from SecurityMaster database using issue code.

- Get tender object from Transaction database using issue code and form number with record status not equal to 'V'.
- Set security issue object to session variable "session.verifytsec".
- Set form number to session variable "session.verifytform".
- Set tender object to session variable "session.verifyttrans".
- Forward control to VerifyTender.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0,1,5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When the user is trying to verify his own submitted tender, the servlet will display SelectTenderVerify.jsp or ViewUnRelTenders.jsp with the relevant error message.
- When the tender object is not found using the issue code and form number, the servlet will display SelectTenderVerify.jsp or ViewUnRelTenders.jsp with the relevant error message.
- When the security issue object is not found using the issue code, the servlet will display SelectTenderVerify.jsp or ViewUnRelTenders.jsp with the relevant error message.

3.2.37 MatchTender.java

This servlet is called from VerifyTender.jsp or ReVerifyTender.jsp. The servlet is then used to do detail matching for a specific tender. If matching or verification is successful, the servlet will forward control to ReleaseTender.jsp. Otherwise, the servlet will forward control to ReVerifyTender.jsp with the relevant matching error messages.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get tender form details from VerifyTender.jsp or ReVerifyTender.jsp
- Generate system date in YYYYMMDD format.
- If servlet is called from VerifyTender.jsp
  - ❖ Get security issue object from session variable "session.verifytsec".
  - ❖ Get form number from session variable "session.verifyform".
  - ❖ Get tender object from session variable "session.verifytrans".
  - ❖ Remove session variable "session.verifytsec".
  - ❖ Remove session variable "session.verifytform".
- If servlet is called from ReVerifyTender.jsp
  - ❖ Get security issue object from session variable "session.reverifytsec".
  - ❖ Get form number from session variable "session.reverifyform".
  - ❖ Get tender object from session variable "session.reverifytrans".
  - ❖ Remove session variable "session.reverifytsec".
  - ❖ Remove session variable "session.reverifytform".
- Do matching for Quantity Applied.
- Do matching for Bid Yield.
- Do matching for Custody Bank Code
- Do matching for Custody Code
- Do matching for Dealer Code

- If there is matching error and re-verification is required
  - ❖ Set security issue object to session variable "session.reverifytsec".
  - ❖ Set form number to session variable "session.reverifytfrom".
  - ❖ Set error status indicator to session variable "session.reverifyerrMsg".
  - ❖ Set error message to session variable "session.reverifyerrMsg".
  - ❖ Set tender form details to session variable "session.reverifytender".
  - ❖ Forward control to ReVerifyTender.jsp
- If matching of tenders is successful
  - ❖ Set security issue object to session variable "session.releasesec".
  - ❖ Set tender object to session variable "session.releasetrans".
  - ❖ Remove session variable "session.verifyttrans".
  - ❖ Forward control to ReleaseTender.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.38 TenderReleased.java

This servlet is used to release tender by updating the tender record status to 'V'.

Processing Logic
- Get User object from session variable "session.user".
- Check User's access level is 3 or 4. (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get security issue object from session variable "session.releasesec".
- Get tender object from session variable "session.releasetrans".
- Remove session variable "session.releasesec".
- Remove session variable "session.releasetrans".
- Generate system date.
- Set record status to 'V' and update the details in tender object.
- Update the tender object to Transaction database.
- Set issue code to session variable "session.secreleased".
- Set form number to session variable "session.formreleased".
- Forward control to TenderReleased.jsp

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When the update to database fails, the servlet will display TransactionFailed.jsp.

3.2.39 SelectAllotResult.java (Removed Jun 2013)

This servlet is used to generate a search form for the user to specify an issue code to view the allotment result.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Forward control to SelectAllotResult.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.40 ViewAllotResult.java (Removed Jun 2013)

This servlet is used to generate a list of unreleased tenders for display in ViewAllotResult.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2, 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get issue code from the search form in SelectAllotResult.jsp.
- Get all allotted tender objects using issue code.
- Set the array of allotted tender objects to session variable "session.alltenders".
- Forward control to ViewAllotResult.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.41 ReActRelTender.java

This servlet is used to generate the details of a released tender for display in ReActRelTender.jsp.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get security issue object from SecurityMaster database using issue code.
- Get tender object from Transaction database using issue code and form number.
- Set security issue object to session variable "session.reactivatetsec".

- Set form number to session variable “session.reactivatetform”.
- Set tender object to session variable “session.reactivatettrans”.
- Forward control to ReActRelTender.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When tender object is not found, the servlet will display SelectTenderReactivate.jsp with the relevant error message.
- When security issue object is not found, the servlet will display SelectTenderReactivate.jsp with the relevant error message.

3.2.42 TenderUnReleased.java

This servlet is used to reactivate a released tender.

Processing Logic
- Get User object from session variable “session.user”.
- ~~Check User’s access level is 3 or 4.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get security issue object from session variable “session.reactivatetsec”.
- Get form number from session variable “session.reactivatetform”.
- Get tender object from session variable “session.reactivatettrans”.
- Remove session variable “session.reactivatetsec”.
- Remove session variable “session.reactivatetform”.
- Remove session variable “session.reactivatettrans”.
- Set issue code to session variable “session.unreleasedtsec”.
- Set form number to session variable “session.unreleasedtform”.
- Set current user id to session variable “session.currentuser”.
- Generate system date.
- Set record status to ‘A’ and update details in the tender object.
- Update tender object to Transaction database.
- Forward control to TenderUnReleased.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 5 or 6, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- ~~When User access level is 2, the servlet will display SGSTenderMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- When the update to database fails, the servlet will display TransactionFailed.jsp.

3.2.43 UploadTenderList.java

This servlet is used to display list of uploaded retail bid files.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Get retail bid indicator for today's date.
- Set retail bid indicator to session variable "session.retailbidstatus".
- Get the list for all the uploaded retail bid files.
- Set the list to session variable "session.listOfFiles".
- Forward control to UploadTenderList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.44 TriggerRetailBid.java

This servlet is used to delete specific retail bid file and to trigger retail bid process.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Check which button selected.
    - If delete button selected, get retail bid file name from parameter "toDeleteFileName" and call "deleteFile" method to delete file from file system in the upload directory.
    - Else if trigger button selected, generate signal file in the location /was_share/SGSAppSrv/SGSAuction/signal/ adhoc_eauction_retail_bids.dat. File watcher will look for the signal file and start retail bid process once the file generated.
    - Set successful message to request attribute "message".
- Forward control to UploadTenderList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.45 AcknowledgementFileDownloadMenu.java

This servlet is used to display a table of acknowledgement files.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Get parameter "processing_date" from web form.
- Get parameter "submit" from web form.
- Get a map of list of acknowledgement files.
- Set processing date to session variable "session.processingdate".
- Set the map of list of acknowledgement files to session variable "session.listOfAcknowledgementFiles".

- Forward control to AcknowledgementFileList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.46 AcknowledgementFileDownload.java

This servlet is used to download acknowledgement file.

Processing Logic
- Get User object from session variable "session.user".
- Check if the User is authorized to access this sub-function.
- Get parameter "path" from web form.
- Get parameter "ackFileName" from web form.
- Call method "outputFile" to download acknowledgement file into file system.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If path and ackFileName is blank, the servlet will display TransactionFailed.jsp.

Package - action.auction

3.2.47 SGSAuctionMenu.java

This servlet is used to generate the SGSAuction Menu page.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3, 5 or 6.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Set tender function names to session variable "session.title" (for Level 5 user, this variable is different from that of Level 3 and 6 user).
- Set tender function descriptions to session variable "session.desc" (for Level 5 user, this variable is different from that of Level 3 and 6 user).
- Forward control to AuctionMenu.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, or 4, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.48 SGSAuctionSecList.java

This servlet is used to display today's securities for auction. This servlet should be called with a query string.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3, or 6.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get value of query string.
- Set query string to session variable "session.curtask".
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Get a list of today's securities for auction.
- Set the list of security to session variable "session.todayrunsecurities".
- Get Auction Indicator for today's
- Set today's auction indicator status to session variable "session.auctionstatus".
- Forward control to AuctionSecList.jsp

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If the value of query string is null, the servlet will display SGSMainMenu.jsp with the relevant error message.

3.2.49 ~~SGSAuctionSecParaList.java~~ (Removed Jun 2013)

This servlet is used to display today's securities for auction, with text fields for users to input yield corridor parameters. This servlet should be called with query string.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 3.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get value of query string.
- Set query string to session variable "session.curtask".
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Get a list of today's securities for auction.
- Set the list of security to session variable "session.todayrunsecurities".
- Forward control to AuctionSecParaList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- When User access level is 9, 0, 1, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.
- If the value of query string is null, the servlet will display SGSMainMenu.jsp with the relevant error message.

3.2.50 SGSAuctionSecParaInput.java (Added Jun 2013)

This servlet is used to display today's securities for auction, with text fields for users to input yield corridor parameters. This servlet should be called with query string.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 3.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Set session variable "session.curtask" to "input"
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Get a list of today's securities for auction.
- Set the list of security to session variable "session.todayrunsecurities".
- Forward control to AuctionSecParaList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

If the value of query string is null, the servlet will display SGSMainMenu.jsp with the relevant error message.

3.2.51 SGSAuctionSecParaVerify.java (Added Jun 2013)

This servlet is used to display today's securities for auction, with text fields for users to input yield corridor parameters. This servlet should be called with query string.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2 or 3.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Set session variable "session.curtask" to "verify".
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Get a list of today's securities for auction, which are not input by the same user.
- Set the list of security to session variable "session.getAuctionableSecuritiesToVerify".
- Forward control to AuctionSecParaList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If the value of query string is null, the servlet will display SGSMainMenu.jsp with the relevant error message.

3.2.52 SGSAuctionSavePara.java

This servlet is used to save yield corridor parameters in database.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 2.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get yield corridors from SGSAuctionSecParaList.jsp.
- Save parameters in database, but set the status as "A", i.e. draft.
- Forward control to message.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 3, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.53 SGSAuctionVerifyPara.java

This servlet is used to verify yield corridor parameters in database.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get yield corridors from SGSAuctionSecParaList.jsp.
- If the parameters are same as those in database, set the status as "V", i.e. verified.
- Forward control to message.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.54 SGSAuctionValidateBids.java

This servlet is used to validate today's bids data and show the result. This function is no longer used.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3, or 6.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Update audit log file.

- Create auction sgsbes.auction.Transaction object.
- Call MSABXT00, which validates checksum.
- Get the return message from MSABXT00.
- Set the return message to session variable “session.message”.
- Forward control to message.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, 4, or 5, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.55 SGSAuctionRunAuction.java

This servlet is used to run auction and show the result.

Processing Logic

1) Check authorisation
   - Get User object from session variable “session.user”.
   - Check if the User is authorized to access this sub-function.

2) Check Auction is currently running or not (To avoid multiple Auction getting triggered when user clicks on "Run Auction")
   - Check in table AQA0017_AUCTION_IND, for the AQA0017_AUCTION_RUN indicator for today’s date
     - If the AQA0017_AUCTION_RUN != R or null, then Auction is not running and proceed to the next steps
     - If the AQA0017_AUCTION_RUN = R, then the Auction is already running and page displayed with message “**Auction is currently running. Please come back later.**” and Run Auction Button is disabled.

3) Signal file will be generated in the location /was_share/SGSAppSrv/SGSAuction/signal/adhoc_auction.dat. File watcher job MSABADSGSAFW will look for the signal file (adhoc_auction.dat) and start auction process once the file generated.

4) Redirect to Auction Securities List page

- Display message based on the "session.auctionstatus" flag
  if flag value is **null** display
  "Click on the button to run auction:"
  if flag value is **F** display
  "Auction for today is already completed with failure.
  Click on the button to run auction:"
  if flag value is **R** display
  "Auction is currently running. Please come back later."
  if flag value is **S** display
  "Auction for today is already completed. Please check the reports."
- Run Auction button will be disabled if the flag values is **R**

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

3.2.55.1 Description of Unique fields involved for the Sorting in different Auction steps

Sorting Fields	Description
SECURITY_CODE	Ongoing Security Code
CUS_BANK	Customer Bank Code eApps Online Bid Submission – “Securities account no. with MAS” field value eAuction Tender Submission – “Custodian Bank” field value Retail Bid file Process – 3 local PDs (DBS, OCBC, UOB)
COMP_NC_C	N – For Non-competitive C – For Competitive
BID_YL	“Yield” value submitted for the Competitive bids submitted via eApps Online Bid Submission, eAuction Tender Submission and Retail bid file process
FORM_NO	System generated Form No i.e., Application/Tender Number seen in the Auction reports
S_NAME	System generated S_NAME If CUS_CD = “B”, then S_NAME = CUS_BANK Else If APP_TY = “IND”, then S_NAME = NRIC_PPNO Else S_NAME = APP_NAME
ALLOT_IND	System generated ALLOT_IND R – Rejected L – Limit " " – Within Allotment Limit or Partial Allotment N – Not Allotted A – Allotted MSABAT01: – Sets ALLOT_IND value to “ ” or “R” or “L” (based on Allotment Limit) MSABAT02: – Sets ALLOT_IND value to “ ” or “L” (Non-comp allotment) MSABUT02: – Sets ALLOT_IND value to “ ” or “N” (Competitive allotment based on Cut-off yield) MSABUT03: – Sets ALLOT_IND value to “A” for all Non-comp bids and bids with Bid yield less than Cut-off yield where ALLOT_IND is not “L” or “R” – Sets ALLOT_IND value to “A” or “N” for bids with Bid yield equal to Cut-off yield – After this step, the ALLOT_IND value will be either “A” or “N” or “R” or “L” only MSABRT08: – Sets ALLOT_IND value from “A” to “Z” before sorting RunBankResult: – Sets ALLOT_IND value from “Z” to “A” before sorting
QTY_APP	“Nominal amount” field in bids submitted via eApps Online Bid Submission, eAuction Tender Submission and Retail bid file process
CUS_CD	Custody Code B – Bank

	C – Bank's Customer eApps Online Bid Submission – "The allotted securities will be transferred to the <Bank/Bank's Customer> account" field value eAuction Tender Submission – "Custody Code" field value Retail Bid file Process – Should be "C" – Bank's Customer
NRIC_PPNO	NRIC or Passport Number submitted for the Retail Bids i.e., APP_TY = "IND" Bids submitted via eApps Online Bid Submission, eAuction Tender Submission and Retail bid file process
EOVA_CPF_ACNO	N – No Y – Yes This field "Eovr CPF/SRS Account no" is available only in eAuction Tender submission. For bid submission done via eApps Online and Retail bid file process – the value will be defaulted to "" (empty) always.
APP_TY	"Type of Applicant" field in eApps Online Bid Submission and eAuction Tender Submission • Primary Dealer (PD) • Full Bank (FB) • Restricted Bank (RB) • Offshore Bank (OB) • Other Bank – customer of another bank (NB) • Merchant Bank (MB) • Investment Advisor (IA) • Finance Company (FC) • Insurance Company (INS) • Statutory Board, Government-Linked Company (SB) • Individual (IND) • Singapore Exchange (SGX) • Monetary Authority of Singapore (MAS) • Others (OTH)
DEALER_CODE	eApps Online Bid Submission – 13 PDs Bank Code and MAS (2500) who can login to eApps Application eAuction Tender Submission – "Dealer Code" field value – It can be one of the bank codes received from MEPS+ bank.txt file Retail Bid file Process – 3 local PDs (DBS, OCBC, UOB)
WS_IND	System generated WS_IND If CUS_CD = "B" (BANKS), then WS_IND = 1 Else If SUBMISSION_METHOD = 'A' (INDIVIDUAL BY ATM SUBMISSION), then WS_IND = 4 Else If SUBMISSION_METHOD = 'B' (INDIVIDUAL BY IBANKING SUBMISSION), then WS_IND = 6 Else If SUBMISSION_METHOD = 'C' (INDIVIDUAL BY MOBILE SUBMISSION), then WS_IND = 5 Else If APP_TY = "IND" (INDIVIDUAL BY EAPPS OR MANUAL SUBMISSION), then WS_IND = 3 Else (CORPORATIONS) WS_IND = 2
APP_NAME	"Full name of applicant" field in bids submitted via eApps Online Bid Submission, eAuction Tender Submission and Retail bid file process

	When Type of applicant is “PD”, “FB”, “RB”, “OB”, or “MAS”, the Full name of applicant is not required. Otherwise, it is required.

3.2.56 SGSAuctionViewReports.java

This servlet is used to display the table of auction reports.

Processing Logic
- Get User object from session variable “session.user”.
- ~~Check User’s access level is 3, 5, or 6.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get absolute file path.
- Get parameter “historical_report_date1”, which is from web form.
- Get parameter “date_type_ind”, which is from web form.
- Set report date, be it today or a historical date, to session variable “session.reportDate”.
- Get report counter value from database.
- Set report names (get from static values defined in MSABReport) to session variable “session.reportName”.
- Set report extensions (get from static values defined in MSABReport) to session variable “session.reportExt”.
- Set report descriptions (get from static values defined in MSABReport) to session variable “session.reportDesc”.
- Remove unused session variables for AGD letter.
- Set counter values to session variable “session.reportCounter”.
- Set report file path to session variable “session.reportPath”.
- Forward control to AuctionReportList.jsp.

Validation Checks
- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, or 4, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If there are no Reports for the specified date, a message “There are no report/letter” will be displayed in AuctionReportList.jsp.

3.2.57 SGSAuctionGenerateLetter.java

This servlet is used to generate a “letter” that contains information on the SGS Bonds and T-Bills for FD to print and mail to AGD. The information within the letter informs AGD of the new issuance/interests payment/redemption of SGS Bonds and T-Bills, and their details (e.g. ISIN Code, Nominal Amount, Discount/Premium, Net Proceeds, Interests paid, etc). The information is extracted according to the issuance/redemption/interests payment date entered by the users.

Please see Section 7.7 for a sample of letters generated for AGD and explanation on the structure.

Processing Logic
- Get User object from session variable “session.user”.
- Check if the User is authorized to access this sub-function.
- Get parameter “date_type_ind”, which is from the web form.

- Get parameter "historical_report_date2", which is from the web form, if a historical date was entered.
- Call DBUtils to synchronize aba0001_security_master table in PRI1 with its counterparty in MNET, and set the indicator that today's synchronization of security master table has taken place.
- Call GenerateLetter
    - To extract all transactional information for new issuances, reopening of Bonds, interest payments and redemptions of T-Bills and SGS Bonds via functions in SGSEDT.jar.
    - For each transaction, format the description of the security.
    - Format the output for MS Word by padding spaces to figures generated.
    - Compose the paragraphs in the letter, according to the type of transactions extracted for the date entered/selected.
- Remove unused session variables for Auction Reports.
- Set report date to session variable "session.reportDate".
- Set file name to session variable "session.AGDFileName".
- Forward control to AuctionReportList.jsp.

Validation Checks
- When the User object is not found, the servlet will display LoginFailed.jsp.
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.
- If there is no information available for the specified date, the user will be shown a message "There are no report/letter" in AuctionReportList.jsp.

3.2.58 SGSAuctionReleaseResult.java

This servlet is used to update table aba0001_security_master and aqa0020_auction_results_report in eApps database, and also create interface files for MEPS+ and FMBS.

Processing Logic
- Get User object from session variable "session.user".
- ~~Check User's access level is 3, or 6.~~ (Removed Jun 2013)
- Check if the User is authorized to access this sub-function.
- Get absolute file path.
- Get today's date.
- Create ReleaseAuctionResult object.
- Set file path for ReleaseAuctionResult object.
- Set report date for ReleaseAuctionResult object.
- Set transaction date for ReleaseAuctionResult object.
- Copy result files. This step creates auction.txt, auction_fmbs.txt, secupdt.txt and secupdt_fmbs.txt.
- To decrypt before sending to MEPS+ MQ.
- Update aba0001_security_master table in eApps database.
- ~~Update aba0006_auction_result table in eApps database.~~
- Update aqa0020_auction_results_report in eApps database
- Update aba0007_detail_auction_result in eApps database
- Update aqa0008_sec_auction_para in eApps database.
- If copying of files is successful, set message to "successful", otherwise set it to "failed".
- Set the message to session variable "session.message".
- Forward control to message.jsp.

Validation Checks

- When User object is not found, the servlet will display LoginFailed.jsp.
- ~~When User access level is 9, 0, 1, 2, or 4, the servlet will display SGSMainMenu.jsp with the relevant error message.~~ (Removed Jun 2013)
- User will be redirected to UnauthorisedAccess.jsp if they are not authorized to access the function.

Package – utils

3.2.59 Database access

SGS eAuction uses JDBC 2.0 Optional Package API to access the Oracle database. Using the JDBC 2.0 Optional Package API, the SGS eAuction access the relational database as follows:

1) Retrieve a DataSource object from the JNDI naming service.
2) Obtain a Connection object from the datasource.
3) Send SQL queries or updates to the database management system.
4) Process the results.

The connection obtained from the datasource is a pooled connection. The datasource is created and bound into JNDI in the WebSphere Application Server. The pool of connections is managed by WebSphere Application Server. After a DataSource object is obtained, the application code calls getConnection() on the datasource to get a Connection object. With the Connection object, SGS eAuction can query the Oracle database.

SGS eAuction uses the following java classes to interface with the Oracle databases:

- utils.sqlmanager.DBUtils.java
  This java class contains method to read/write to both eAuction and eApps Oracle databases.

- utils.sqlmanager.SQLManager.java
  This java class contains method to obtain connection with the Oracle database.

- utils.sqlmanager.SQLSecurityMaster.java
  This java class contains methods to query the SecurityMaster table in Oracle database instance name “mnet”.

- utils.sqlmanager.SQLTransaction.java
  This java class contains methods to query, insert and update the Transaction table in Oracle database instance name “tri1”. It also contains a method to extract transaction data use sgsedt.jar.

- utils.sqlmanager.SQLBankMaster.java
  This java class contains methods to query the BankMaster table in Oracle database instance name “mnet”.

- utils.sqlmanager.SQLUser.java
  This java class contains methods to query the User table in Oracle database instance name “tri1”.

- utils.sqlmanager.SQLAuthorisedAction.java
  This java class contains methods to retrieve the authorized actions for the user that has signed in.

The SGS eAuction uses two Oracle database instance names. The two instance names are “tri1” and “mnet”. The owner of these two instances is "ms9aba". Tables are created under owner instance name “ms9aba”. In the SQL statements implanted in Java classes, all the table names should be prefixed by "ms9aba".

The editor role of these two instances is “ms7aba”. A DB link between the two Oracle databases is configured using the user id “ms7aba”. Therefore, the database instance owner “ms9aba” should grant the access right of the created tables to “ms7aba”, which can use the DB link.

In addition, the database instance owner should grant the access right to the Oracle database user id, which is used by the Web application.

Package – sgsbes.auction

3.2.60 Auction program

This is the package which encapsulates the auction business rules. It has following java classes:

- sgsbes.auction.AuctionSystemLog.java
  This is the auction system log class.

- sgsbes.auction.GenerateLetter.java
  To generate a letter for AGD. This contains transaction information on new issuances, reopening of Bonds, interest payments and redemptions of T-Bills and SGS Bonds.

- sgsbes.auction.ReleaseAuctionResult.java
  To copy the interface files to pre-defined directories in Unix file system.

- sgsbes.auction.ReportAndLog.java
  To represent a generic auction report.

- sgsbes.auction.SgsbesException.java
  To represent an auction exception.

- sgsbes.auction.TransactionDescriptor.java
  To store some static variables.

- sgsbes.auction.Transactions.java
  Contain some methods to manage bid applications.

- sgsbes.auction.EauctionConfigUtil.java
  To retrieve eAuction configuration.

3.3 Integration with Active Directory Server

SGS eAuction uses form-based authentication with Active Directory Server.

All SGS eAuction Users should be authenticated by Active Directory Server. A customized login page will be displayed for User to enter the User Id and password. This information will be passed on to Active Directory Server for authentication. Upon successful authentication, the User will be allowed to use the SGS eAuction.

SGS eAuction uses form-logout to logout the User successfully.

3.4 Error Handling and Logging

When there is any exception thrown by the servlet, the method “sgsnet.common.SvtMnetError" will be called. It in turns to call "sgsnet.common.MnetLogger" to write error messages to the log file, SGSAuctionError.log. The file name of the log file is specified in the configuration file, which is represented by an instance of class "sgsnet.common.MnetConfig".

Each error message stored in the log file has an associated unique error identification code with a time stamp. Programmer, system administrator or helpdesk technician can open the log file, locate the error using the error id, and find the details of the error.

At the same time, the servlet will display an error page (TransactionFailed.jsp) to inform the User.

3.5 Some miscellaneous Java classes

There are several miscellaneous Java classes and resource file: "sgseapps.Utils", "sgseapps.PriceUtils", "sgseapps.SGSeAppsVariables", “sqseapps.AuthorisedActionUtil”, “sgseapps.AntivirusScanException”, “sgseapps.AntivirusScanUtil”, “sgseapps.EappsConfigUtil”, “sgseapps.RetailBidFileUtil”, and “application.properties”.

The Java class "sgseapps.Utils" provides some static methods, such as date/time formatting, string formatting, etc.

The Java class "sgseapps.PriceUtils" provides some static methods for price/yield, accrued interest calculations.

The Java class "sgseapps.SGSeAppsVariables" stores some static variables, such as dbOwner which allows the Auction System easily switches db instance, defaultFilePath which was added for HA clustering. The variable defaultFilePath is set as “/was_share/SGSAppSrv/SGSAuction/sgsbes/”.

The Java class “sgseapps.AuthorisedActionUtil” provides the method to validate the user’s request to access the function/sub function. This is done, by comparing the function/sub function requested by users with their authorised functions stored in the session variable “authorisedActions”.

The Java class “sgseapps.AntivirusScanException” provides some methods for antivirus scan error exception.

The Java class “sgseapps.AntivirusScanUtil” provides some static methods for antivirus scanning.

The Java class “sgseapps.EappsConfigUtil” provides some static methods for retrieving eApps configuration.

The Java class “sgseapps.RetailBidFileUtil” provides some static methods for saving, deleting, and fetching retail bid files.

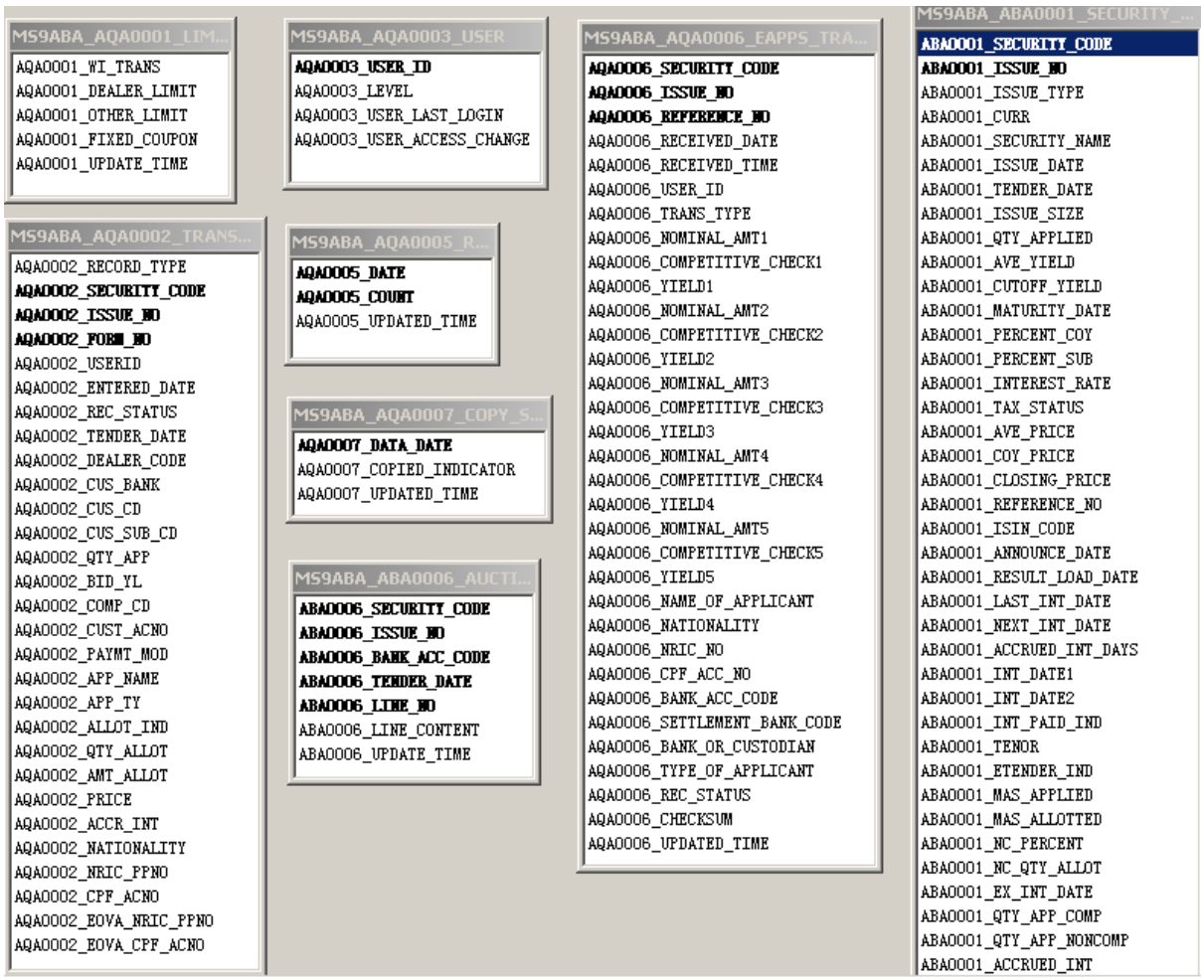
The resource file “application.properties” provides customised messages that are used by the JSP web pages.

3.6 Oracle database design

Oracle database is used to store security reference information, tender application data and user's logon information. For SGS eAuction system, it uses two Oracle database instances, namely "mnet" with table space "netapp" and “tri1” with table space “abaapp”. SGS eAuction system is accessing the “mnet” database instance via DB link.

Twelve tables are created for SGS eAuction system - "aqa0001_limit", "aqa0002_transaction", "aqa0003_user", “aqa0004_action_level”, "aqa0005_report_counter", "aqa0006_eapps_transaction", "aqa0007_copy_sec_master_ind", "aqa0008_sec_auction_para", "aba0001_security_master" and "aba0006_auction_result", “aqa0010_action_ref”, “aqa0103_private_key”. After the creation of these tables, the access rights of “insert”, “update” and “delete” should be granted to the WAS user.

The two tables, namely “aba0001_security_master” and “aba0009_bank_master”, which have already been created for SGS eApps system, are accessed by SGS eAuction system via DB link. The access rights of “insert”, “update” and “delete” should be granted to the WAS user.



3.6.1 "aba0001_limit"

This table stores transaction limit for PD and non-PD. The default values are 30 and 15. The table creation statement is as below:

```
CREATE TABLE aqa0001_limit
(
```

```
	aqa0001_wi_trans		VARCHAR(5),
	aqa0001_dealer_limit		NUMBER(5)	NOT NULL,
	aqa0001_other_limit		NUMBER(5)	NOT NULL,
	aqa0001_fixed_coupon		CHAR(1)		DEFAULT 'Y',
	aqa0001_update_time		DATE		DEFAULT SYSDATE
)
	TABLESPACE abaapp
	STORAGE
	(
		INITIAL		10K
		NEXT		10K
		MINEXTENTS	  1
		MAXEXTENTS	100
		PCTINCREASE	  0
	);
```

3.6.2 "aqa0002_transaction"

This table is to store user's tender applications. The table creation statement is as below:

```
CREATE TABLE aqa0002_transaction
(
	aqa0002_record_type		CHAR(2)		NOT NULL,
	aqa0002_security_code		CHAR(8)		NOT NULL,
	aqa0002_issue_no		CHAR(1)		NOT NULL,
	aqa0002_form_no		NUMBER(6)	NOT NULL,
	aqa0002_user_id		CHAR(10),
	aqa0002_entered_date		DATE,
	aqa0002_rec_status		CHAR(1),
	aqa0002_tender_date		DATE,
	aqa0002_dealer_code		NUMBER(4),
	aqa0002_cus_bank		NUMBER(4),
	aqa0002_cus_cd		CHAR(1),
	aqa0002_cus_sub_cd		CHAR(3),
	aqa0002_qty_app		NUMBER(11),
	aqa0002_bid_yl		NUMBER(5,2),
	aqa0002_comp_cd		CHAR(1),
	aqa0002_cust_acno		VARCHAR(20),
	aqa0002_paymt_mod		CHAR(1),
	aqa0002_app_name		VARCHAR(30),
	aqa0002_app_ty		CHAR(3),
	aqa0002_allot_ind		CHAR(1),
	aqa0002_qty_allot		NUMBER(11),
	aqa0002_amt_allot		NUMBER(13,2),
	aqa0002_price		NUMBER(7,4),
	aqa0002_accr_int		NUMBER(11,2),
	aqa0002_nationality		CHAR(11),
	aqa0002_nric_ppno		CHAR(14),
	aqa0002_cpf_acno		CHAR(16),
	aqa0002_eova_nric_ppno		CHAR(1),
	aqa0002_eova_cpf_acno		CHAR(1),
	aqa0002_file_type		varchar(3),
	CONSTRAINT pk_aqa0002_transaction PRIMARY KEY
	(
		aqa0002_security_code,
		aqa0002_issue_no,
		aqa0002_form_no
	)
)
```

```
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       50K
            NEXT          50K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

Columns "aqa0002_security_code", "aqa0002_issue_no" and “aqa0002_form_no” are primary keys of this table.

SGS eAuction system will insert a new record into this table after a tender application is successfully received.

3.6.3 "aqa0003_user"

This table is used to store user logon information. The table creation statement is as below:

```
CREATE TABLE aqa0003_user
(
      aqa0003_user_id               VARCHAR2(10)      NOT NULL,
      aqa0003_level                 NUMBER(3)   DEFAULT 0,
      aqa0003_user_last_login       DATE        DEFAULT SYSDATE,
      aqa0003_user_access_change    DATE        DEFAULT SYSDATE,
      CONSTRAINT pk_aqa0003_user PRIMARY KEY
      (
            aqa0003_user_id
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       10K
            NEXT          10K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

Columns "aqa0003_user_id" is the primary key of this table.

3.6.4 “aqa0004_level_action”

This table is used to store authorized actions for each level of authorization. The script to create the table is below:

```
CREATE TABLE MS9ABA.AQA0004_LEVEL_ACTION (
      AQA0004_LEVEL           NUMBER (3,0)      NOT NULL,
      AQA0004_MODULE          VARCHAR2 (100)    NOT NULL,
      AQA0004_ACTION          VARCHAR2 (50)     NOT NULL,
      CONSTRAINT PK_AQA0004_LEVEL_ACTION PRIMARY KEY
      (
            AQA0004_LEVEL,
            AQA0004_MODULE,
            AQA0004_ACTION
      )
);
```

Columns "AQA0004_LEVEL, AQA0004_MODULE, AQA0004_ACTION" are the composite primary key of this table

3.6.5 "aqa0005_report_counter"

This table is used to store the report counter. During one day, the counter will increment 1 when user runs auction. The counter is used to get the latest auction result files when user views auction result.

```
CREATE TABLE aqa0005_report_counter
(
      aqa0005_date                  DATE        NOT NULL,
      aqa0005_count                 NUMBER(3)   NOT NULL,
      aqa0005_updated_time          DATE        DEFAULT SYSDATE,
      CONSTRAINT pk_aqa0005_report_counter PRIMARY KEY
      (
            aqa0005_date, aqa0005_count
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       10K
            NEXT          10K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

The column "aqa0005_date" and "aqa0005_count" are the primary keys of this table.

3.6.6 "aqa0006_eapps_transaction"

This table is used to store eTender forms inputted via eApps. When the decrypted bids file is FTPed to PCSSYS90, a UNIX job will load the data into this table.

```
CREATE TABLE aqa0006_eapps_transaction
(     aqa0006_security_code         CHAR(8),
      aqa0006_issue_no              CHAR(1)     DEFAULT 0,
      aqa0006_reference_no          NUMBER(5),
      aqa0006_received_date         DATE        DEFAULT SYSDATE,
      aqa0006_received_time         DATE        DEFAULT SYSDATE,
      aqa0006_user_id               VARCHAR2(8),
      aqa0006_trans_type            CHAR(3),
      aqa0006_nominal_amt1          NUMBER(11)  NOT NULL,
      aqa0006_competitive_check1    CHAR(1)     NOT NULL,
      aqa0006_yield1                NUMBER(5,2),
      aqa0006_nominal_amt2          NUMBER(11),
      aqa0006_competitive_check2    CHAR(1),
      aqa0006_yield2                NUMBER(5,2),
      aqa0006_nominal_amt3          NUMBER(11),
      aqa0006_competitive_check3    CHAR(1),
      aqa0006_yield3                NUMBER(5,2),
      aqa0006_nominal_amt4          NUMBER(11),
      aqa0006_competitive_check4    CHAR(1),
      aqa0006_yield4                NUMBER(5,2),
      aqa0006_nominal_amt5          NUMBER(11),
      aqa0006_competitive_check5    CHAR(1),
```

```
      aqa0006_yield5                NUMBER(5,2),
      aqa0006_name_of_applicant     VARCHAR2(30),
      aqa0006_nationality           CHAR(1),
      aqa0006_nric_no               VARCHAR2(14),
      aqa0006_cpf_acc_no            CHAR(16),
      aqa0006_bank_acc_code         NUMBER(4),
      aqa0006_settlement_bank_code  NUMBER(4),
      aqa0006_bank_or_custodian     CHAR(1),
      aqa0006_type_of_applicant     VARCHAR2(3) NOT NULL,
      aqa0006_rec_status            CHAR(1),
      aqa0006_checksum              NUMBER(15),
      aqa0006_updated_time          DATE        DEFAULT SYSDATE,
      aqa0006_ file_type                 varchar(3),
      CONSTRAINT pk_aqa0006_eapps_transaction PRIMARY KEY
      (
            aqa0006_security_code,
            aqa0006_issue_no,
            aqa0006_reference_no
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       50K
            NEXT          50K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

The column “aqa0006_security_code”, “aqa0006_issue_no” and “aqa0006_reference_no” are the primary keys of this table.

3.6.7 “aqa0007_copy_sec_master_ind”

This table is used to store indicator whether the two security master tables have been synchronized.

```
CREATE TABLE aqa0007_copy_sec_master_ind
(
      aqa0007_data_date             DATE        NOT NULL,
      aqa0007_copied_indicator      CHAR(1)     NOT NULL,
      aqa0007_updated_time          DATE        DEFAULT SYSDATE,
      CONSTRAINT pk_aqa0007_copy_sec_master_ind PRIMARY KEY
      (
            aqa0007_data_date
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       10K
            NEXT          10K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

The column “aqa0007_data_date” is the primary key of this table.

3.6.8 "aqa0008_sec_auction_para"

This table is used to store parameters for yield corridor.

```
CREATE TABLE aqa0008_sec_auction_para
(
      aqa0008_security_code         CHAR(8),
      aqa0008_issue_no              CHAR(1),
      aqa0008_midpoint_yield        NUMBER(6,3),
      aqa0008_corridor              NUMBER(5,3),
      aqa0008_mas_max_amt           NUMBER(13),
      aqa0008_updated_time          DATE        DEFAULT SYSDATE,
      aqa0008_status                CHAR(1),
      aqa0008_record_status         CHAR(1),
      AQA0008_REF_POINT             NUMBER(6,3),
      CONSTRAINT  pk_aqa0008_sec_auction_para PRIMARY KEY
      (
            aqa0008_security_code,aqa0008_issue_no
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       100K
            NEXT          100K
            MINEXTENTS      1
            MAXEXTENTS    100
            PCTINCREASE     0
      );
```

3.6.9 "aqa0010_action_ref"

This table is used to store the description for the various actions available in the System. The create table statement is below:

```
CREATE TABLE MS9ABA.AQA0010_ACTION_REF (
      AQA0010_MODULE VARCHAR2 (100) NOT NULL,
      AQA0010_ACTION VARCHAR2 (50) NOT NULL,
      AQA0010_DESC VARCHAR2 (300),
      CONSTRAINT PK_AQA0010_ACTION_REF PRIMARY KEY
      (
            AQA0010_MODULE, AQA0010_ACTION
      )
);
```

The columns "aqa0010_module" and "aqa0010_action" form the primary composite key of this table.

3.6.10 "aba0001_security_master"

This table structure is same as aba0001_security_master in eApps database. It is used to temporally store the security summary result after auction, before user "Release Auction Result".

```
CREATE TABLE aba0001_security_master
(
```

```
        aba0001_security_code         CHAR(8),
        aba0001_issue_no              CHAR(1),
        aba0001_issue_type            CHAR(1),
        aba0001_curr                  CHAR(3),
        aba0001_security_name         VARCHAR2(30)      NOT NULL,
        aba0001_issue_date            DATE              NOT NULL,
        aba0001_tender_date           DATE              NOT NULL,
        aba0001_issue_size            NUMBER(13)        NOT NULL,
        aba0001_qty_applied           NUMBER(13),
        aba0001_ave_yield             NUMBER(5,2),
        aba0001_cutoff_yield          NUMBER(5,2),
        aba0001_maturity_date         DATE              NOT NULL,
        aba0001_percent_coy           NUMBER(5,2),
        aba0001_percent_sub           NUMBER(5,2),
        aba0001_interest_rate         NUMBER(7,4),
        aba0001_tax_status            CHAR(1),
        aba0001_ave_price             NUMBER(7,4),
        aba0001_coy_price             NUMBER(7,4),
        aba0001_closing_price         NUMBER(7,4),
        aba0001_reference_no          NUMBER(5),
        aba0001_isin_code             CHAR(12),
        aba0001_announce_date         DATE,
        aba0001_result_load_date      DATE,
        aba0001_last_int_date         DATE,
        aba0001_next_int_date         DATE,
        aba0001_accrued_int_days      NUMBER(3),
        aba0001_int_date1             DATE,
        aba0001_int_date2             DATE,
        aba0001_int_paid_ind          CHAR(1)           DEFAULT 'N',
        aba0001_tenor                 NUMBER(3),
        aba0001_etender_ind           CHAR(1)           DEFAULT 'Y',
        aba0001_mas_applied           NUMBER(13),
        aba0001_mas_allotted          NUMBER(13),
        aba0001_nc_percent            NUMBER(5,2),
        aba0001_nc_qty_allot          NUMBER(13),
        aba0001_ex_int_date           DATE,
        aba0001_qty_app_comp          NUMBER(13),
        aba0001_qty_app_noncomp       NUMBER(13),
        aba0001_accrued_int           NUMBER(7,4)       DEFAULT 0,
        aba0001_announce_indicator    CHAR(1)           DEFAULT 'N',
        aba0001_median_yield          NUMBER(5,2),
        aba0001_median_price          NUMBER(7,4),
        CONSTRAINT pk_aba0001_security_master PRIMARY KEY
        (
              aba0001_security_code,aba0001_issue_no
        )
)
        TABLESPACE abaapp
        STORAGE
        (
              INITIAL       20K
              NEXT          20K
              MINEXTENTS      1
              MAXEXTENTS    100
              PCTINCREASE     0
        );
```

The column “aba0001_security_code” and “aba0001_issue_no” are the primary keys of this table.

3.6.11 "aba0006_auction_result"

This table structure is same as aba0006_auction_result in eApps database. It is used to temporally store the auction result for banks, before user "Release Auction Result".

```
CREATE TABLE aba0006_auction_result
(
      aba0006_security_code         CHAR(8)           NOT NULL,
      aba0006_issue_no              CHAR(1)           NOT NULL,
      aba0006_bank_acc_code         NUMBER(4)         NOT NULL,
      aba0006_tender_date           DATE              NOT NULL,
      aba0006_line_no               NUMBER(9)         NOT NULL,
      aba0006_line_content          VARCHAR2(200),
      aba0006_update_time           DATE              DEFAULT SYSDATE,
      CONSTRAINT pk_aba0006_auction_result PRIMARY KEY
      (
            aba0006_security_code,
            aba0006_issue_no,
            aba0006_bank_acc_code,
            aba0006_tender_date,
            aba0006_line_no
      )
)
      TABLESPACE abaapp
      STORAGE
      (
            INITIAL       500K
            NEXT          500K
            MINEXTENTS      1
            MAXEXTENTS  1000
            PCTINCREASE     0
      );
```

The column "aba0006_security_code", "aba0006_issue_no", "aba0006_bank_acc_code", "aba0006_tender_date" and "aba0006_line_no" are the primary keys of this table.

3.6.12 "aqa0103_PRIVATE_KEY"

```
CREATE TABLE "MS9ABA"."AQA0103_PRIVATE_KEY"
  (
    "AQA0103_PRIVATE_KEY" VARCHAR2(4000 BYTE) NOT NULL ENABLE,
    "AQA0103_UPDATED_DT" DATE,
    "AQA0103_CURRENT_KEY" VARCHAR2(1 BYTE),
    CONSTRAINT "PK_AQA0103_PRIVATE_KEY" PRIMARY KEY ("AQA0103_PRIVATE_KEY")
)
```

3.6.13 ABA0214_SB_CD_FILE_TYPE

Enhance the existing file type table to include fields for eApps & eAuction usage.

```
Update TABLE ABA0214_SB_CD_FILE_TYPE add
(
      ABA0214_SB_TYPE_DISPLAY       VARCHAR(100) NOT NULL,
      ABA0214_SB_OBS_IND            Boolean      NOT NULL DEFAULT N,
      ABA0214_SB_TYPE_CATEGORY      VARCHAR(25)
)
```

Data Info

ABA0214_SB_CD_FILE_TYPE	ABA0214_SB_DESCRIPTION	ABA0214_SB_TYPE_DISPLAY	ABA0214_SB_OBS_IND	ABA0214_SB_TYPE_CATEGORY
AP1	Application For Cash	CASH	N	Subscription
AP2	Application For CPFIS	CPFIS	Y	Subscription
AP3	Application For SRS	SRS	N	Subscription
RA1	Auction Result for CASH	CASH	N	AuctionRst
RA2	Auction Result for CPFIS	CPFIS	Y	AuctionRst
RA3	Auction Result for SRS	SRS	N	AuctionRst

- In-Use Indicator denotes the state of File Type. Is it active or inactive.
- UI interface program will read this field to display in the list box.

3.6.14 AQA0011_USER_SESSION

This table stores the User Session details for Concurrent Login.

```
CREATE TABLE MS9ABA.AQA0011_USER_SESSION
(
  AQA0011_USER_ID                VARCHAR2(20 BYTE),
  AQA0011_SESSION_ID             VARCHAR2(200 BYTE),
  AQA0011_LAST_LOGIN_DATE        TIMESTAMP (6) DEFAULT CURRENT_TIMESTAMP
) TABLESPACE "ABAAPP";

grant select,update,insert,delete on "MS9ABA"."AQA0011_USER_SESSION" to MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0011_USER_SESSION" to
MS_ABA_EDITOR_ROLE;
```

3.6.15 ABA0007_DETAIL_AUCTION_RESULT

This table stores the Detail auction results during the Auction process

```
 CREATE TABLE "MS9ABA"."ABA0007_DETAIL_AUCTION_RESULT"
  (       "ABA0007_SECURITY_CODE" VARCHAR2(8 BYTE) NOT NULL ENABLE,
          "ABA0007_ISSUE_NO" VARCHAR2(1 BYTE) NOT NULL ENABLE,
          "ABA0007_REF_NO" NUMBER(5,0),
          "ABA0007_BANK_REF_NO" VARCHAR2(12 BYTE),
          "ABA0007_FORM_NO" NUMBER(6,0) NOT NULL ENABLE,
          "ABA0007_TENDER_DATE" DATE NOT NULL ENABLE,
          "ABA0007_PRI_DLR_CODE" NUMBER(4,0) NOT NULL ENABLE,
          "ABA0007_NAME_OF_APPLN" VARCHAR2(30 BYTE),
          "ABA0007_NATIONALITY" VARCHAR2(1 BYTE),
```

```
"ABA0007_IC_PASSPORT" VARCHAR2(14 BYTE),
"ABA0007_COMP_NONCOMP" VARCHAR2(1 BYTE),
"ABA0007_BID_YIELD" NUMBER(5,2),
"ABA0007_APPLIED_AMT" NUMBER(11,0),
"ABA0007_ALLOTED_AMT" NUMBER(11,0),
"ABA0007_SETTLEMENT_AMT" NUMBER(15,2),
"ABA0007_ACCRUED_INT" NUMBER(11,2),
"ABA0007_CUST_BANK_CODE" NUMBER(4,0),
"ABA0007_CUST_BANK_BC" VARCHAR2(1 BYTE),
"ABA0007_TYPE_OF_APPLN" VARCHAR2(3 BYTE),
"ABA0007_CDP_ACC_NO" VARCHAR2(16 BYTE),
"ABA0007_SUB_MTD" CHAR(1 BYTE),
"ABA0007_UPDATED_DATETIME" DATE DEFAULT SYSDATE,
"ABA0007_CPF_ACNO" CHAR(16 BYTE),
"ABA0007_FILE_TYPE" VARCHAR2(3 BYTE) DEFAULT 'AP1' NOT NULL ENABLE,
 CONSTRAINT "PK_ABA0007_IND_AUCT_RESULT" PRIMARY KEY
("ABA0007_SECURITY_CODE", "ABA0007_ISSUE_NO", "ABA0007_FORM_NO")
 USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
 STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP"  ENABLE
  ) SEGMENT CREATION IMMEDIATE
 PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
 STORAGE(INITIAL 516096 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP" ;
```

3.6.16 AQA0017_AUCTION_IND

This table stores whether the auction is currently running or not for a specific date

```
CREATE TABLE "MS9ABA"."AQA0017_AUCTION_IND"
  (
  "AQA0017_AUCTION_DATE" VARCHAR2(10 BYTE),
  "AQA0017_AUCTION_RUN" VARCHAR2(1 BYTE),
  "AQA0017_DECRYPT_BIDS" VARCHAR2(1 BYTE) DEFAULT NULL,
  "AQA0017_AUCTION_ENGINE" VARCHAR2(1 BYTE) DEFAULT NULL,
  CONSTRAINT "PK_AQA0017_AUCTION_DATE" PRIMARY KEY
("AQA0017_AUCTION_DATE")
 USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
 STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP"  ENABLE
  ) SEGMENT CREATION IMMEDIATE
 PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
 STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP" ;
```

```
grant select,update,insert,delete on "MS9ABA"."AQA0017_AUCTION_IND" to MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0017_AUCTION_IND" to
MS_ABA_EDITOR_ROLE;
```

3.6.17 AQA0019_AUCTION_REPORT_IND

This table stores auction results report status for a specific date

```
CREATE TABLE "MS9ABA"."AQA0019_AUCTION_REPORT_IND"
  (
  "AQA0019_REPORT_NAME" VARCHAR2(50 BYTE) ,
  "AQA0019_REPORT_RUN_DATE" VARCHAR2(10 BYTE),
  "AQA0019_REPORT_IND" VARCHAR2(1 BYTE),
  CONSTRAINT "PK_AQA0019_AUCTION_REPORT_IND" PRIMARY KEY
("AQA0019_REPORT_NAME","AQA0019_REPORT_RUN_DATE")
 USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
 STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP"  ENABLE
  ) SEGMENT CREATION IMMEDIATE
 PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
 STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"."AQA0019_AUCTION_REPORT_IND" to
MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0019_AUCTION_REPORT_IND" to
MS_ABA_EDITOR_ROLE;
```

3.6.18 AQA0009_SEC_AUCTION_PARA_STG_CTG

This staging table stores auction params received from MDD for contingency of specific date

```
CREATE TABLE "MS9ABA"."AQA0009_SEC_AUCTION_PARA_CTG_STG"
  (
   "AQA0009_SECURITY_CODE" CHAR(8 BYTE),
         "AQA0009_ISSUE_NO" CHAR(1 BYTE),
         "AQA0009_REF_POINT" NUMBER(6,3),
         "AQA0009_UPPER_BOUND" NUMBER(6,3),
         "AQA0009_LOWER_BOUND" NUMBER(6,3),
         "AQA0009_MAS_MAX_AMT" NUMBER(13,0),
         CONSTRAINT PK_AQA0009_SEC_AUCTION_PARA_CTG_STG PRIMARY KEY
(AQA0009_SECURITY_CODE,AQA0009_ISSUE_NO)
 USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
 STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "ABAAPP"  ENABLE
  ) SEGMENT CREATION IMMEDIATE
 PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
```

```
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"."AQA0009_SEC_AUCTION_PARA_CTG_STG" to
MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0009_SEC_AUCTION_PARA_CTG_STG " to
MS_ABA_EDITOR_ROLE;
```

3.6.19 AQA0020_AUCTION_RESULTS_REPORT

This to table stores transaction details required to generate reports

```
CREATE TABLE "MS9ABA"."AQA0020_AUCTION_RESULTS_REPORT"(
        "AQA0020_RECORD_TYPE" CHAR(2 BYTE) ,
        "AQA0020_SECURITY_CODE" CHAR(8 BYTE) ,
        "AQA0020_ISSUE_NO" CHAR(1 BYTE) ,
        "AQA0020_FORM_NO" NUMBER(6,0),
        "AQA0020_REC_STATUS" CHAR(1 BYTE) ,
        "AQA0020_TENDER_DATE" DATE,
        "AQA0020_DEALER_CODE" NUMBER(4,0),
        "AQA0020_CUS_BANK" NUMBER(4,0),
        "AQA0020_CUS_CD" CHAR(1 BYTE) ,
        "AQA0020_CUS_SUB_CD" CHAR(3 BYTE) ,
        "AQA0020_QTY_APP" NUMBER(11,0),
        "AQA0020_BID_YL" NUMBER(5,2),
        "AQA0020_COMP_CD" CHAR(1 BYTE) ,
        "AQA0020_CUST_ACNO" VARCHAR2(20 BYTE) ,
        "AQA0020_APP_NAME" VARCHAR2(40 BYTE) ,
        "AQA0020_APP_TY" CHAR(3 BYTE) ,
        "AQA0020_QTY_ALLOT" NUMBER(11,0),
        "AQA0020_AMT_ALLOT" NUMBER(13,2),
        "AQA0020_PRICE" NUMBER(7,4),
        "AQA0020_NATIONALITY" CHAR(11 BYTE) ,
        "AQA0020_NRIC_PPNO" CHAR(14 BYTE) ,
        "AQA0020_CPF_ACNO" CHAR(16 BYTE) ,
        "AQA0020_BANK_REF_NO" VARCHAR2(12 BYTE) ,
        "AQA0020_CDP_ACNO" VARCHAR2(16 BYTE) ,
        "AQA0020_SUB_MTD" CHAR(1 BYTE) ,
        "AQA0020_FILE_TYPE" VARCHAR2(3 BYTE)  NOT NULL ENABLE,
        "AQA0020_ALLOT_IND" CHAR(1 BYTE) ,
        "AQA0020_EOVA_NRIC_PPNO" CHAR(1 BYTE) ,
        "AQA0020_ACCR_INT" NUMBER(11,2),
        "AQA0020_EOVA_CPF_ACNO"         CHAR(1 BYTE),
        "AQA0020_USERID" VARCHAR2(20 BYTE),
        "AQA0020_PAYMT_MOD" CHAR(1 BYTE),   CONSTRAINT "PK_
AQA0020_AUCTION_RESULTS_REPORT" PRIMARY KEY   ("AQA0020_SECURITY_CODE",
"AQA0020_ISSUE_NO", "AQA0020_FORM_NO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP"  ENABLE
```

```
  ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"."AQA0020_AUCTION_RESULTS_REPORT" to
MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0020_AUCTION_RESULTS_REPORT" to
MS_ABA_EDITOR_ROLE;
```

3.6.20 AQA0021_AUCTION_RESULTS_REPORT_CTG_STG

This staging table stores auction results reports details received from MDD as part of contingency plan of specific date

```
CREATE TABLE "MS9ABA"."AQA0021_AUCTION_RESULTS_REPORT_CTG_STG"(
        "AQA0021_SECURITY_CODE" CHAR(8 BYTE) ,
        "AQA0021_ISSUE_NO" CHAR(1 BYTE) ,
        "AQA0021_FORM_NO" NUMBER(6,0),
        "AQA0021_QTY_APP" NUMBER(11,0),
        "AQA0021_APP_NAME" VARCHAR2(40 BYTE) ,
        "AQA0021_QTY_ALLOT" NUMBER(11,0),
        "AQA0021_BANK_REF_NO" VARCHAR2(12 BYTE) ,
        "AQA0021_ALLOT_IND" CHAR(1 BYTE) ,
        "AQA0021_ACCR_INT" NUMBER(11,2) ,  CONSTRAINT "PK_
AQA0021_AUCTION_RESULTS_REPORT_CTG_STG" PRIMARY KEY
("AQA0021_SECURITY_CODE", "AQA0021_ISSUE_NO", "AQA0021_FORM_NO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP"  ENABLE
  ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"."AQA0021_AUCTION_RESULTS_REPORT
_CTG_STG" to MS7ABA;
grant select,update,insert,delete on "MS9ABA"." AQA0021_AUCTION_RESULTS_REPORT
_CTG_STG" to MS_ABA_EDITOR_ROLE;
```

3.6.21 ABA0035_SECURITY_MASTER _CTG_STG

This staging table stores security master details received from MDD as part of contingency plan of specific date

```
CREATE TABLE "MS9ABA"."ABA0035_SECURITY_MASTER _CTG_STG"
  (
```

```
  "ABA0035_SECURITY_CODE" CHAR(8 BYTE),
        "ABA0035_ISSUE_NO" CHAR(1 BYTE),
        "ABA0035_QTY_APPLIED" NUMBER(13,0),
        "ABA0035_AVE_YIELD" NUMBER(5,2),
        "ABA0035_CUTOFF_YIELD" NUMBER(5,2),
        "ABA0035_PERCENT_COY" NUMBER(5,2),
        "ABA0035_PERCENT_SUB" NUMBER(5,2),
        "ABA0035_INTEREST_RATE" NUMBER(7,4),
        "ABA0035_AVE_PRICE" NUMBER(7,4),
        "ABA0035_COY_PRICE" NUMBER(7,4),
        "ABA0035_CLOSING_PRICE" NUMBER(7,4),
        "ABA0035_MAS_ALLOTTED" NUMBER(13,0),
        "ABA0035_NC_PERCENT" NUMBER(5,2),
        "ABA0035_NC_QTY_ALLOT" NUMBER(13,0),
        "ABA0035_QTY_APP_COMP" NUMBER(13,0),
        "ABA0035_QTY_APP_NONCOMP" NUMBER(13,0),
        "ABA0035_ACCRUED_INT" NUMBER(7,4) DEFAULT 0,
        "ABA0035_MEDIAN_YIELD" NUMBER(5,2),
        "ABA0035_MEDIAN_PRICE" NUMBER(7,4),
  CONSTRAINT "PK_ABA0035_SEC_MASTER_CTG_STG" PRIMARY KEY
("ABA0035_SECURITY_CODE", "ABA0035_ISSUE_NO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  STORAGE(INITIAL 32768 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP"  ENABLE
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 57344 NEXT 32768 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"." ABA0035_SECURITY_MASTER_CTG_STG" to
MS7ABA;
grant select,update,insert,delete on "MS9ABA"." ABA0035_SECURITY_MASTER_CTG_STG" to
MS_ABA_EDITOR_ROLE;
```

3.6.22 ABA0009_EAUCTION_CONFIG

This table is to store the config details in DB

```
CREATE TABLE "MS9ABA"."ABA0009_EAUCTION_CONFIG" (
        "ABA0009_KEY" VARCHAR2(50 BYTE) NOT NULL ENABLE,
        "ABA0009_VALUE" VARCHAR2(100 BYTE) NOT NULL ENABLE,
        CONSTRAINT "PK_ABA0009_EAUCTION_CONFIG" PRIMARY KEY
("ABA0009_KEY")
        USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
        STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
        PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
        BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
        TABLESPACE "ABAAPP"  ENABLE )  SEGMENT CREATION IMMEDIATE
        PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
```

```
	NOCOMPRESS LOGGING
	STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
	PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
	BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE
DEFAULT)
	TABLESPACE "ABAAPP" ;

GRANT SELECT, UPDATE, INSERT, DELETE ON "ABA0009_EAUCTION_CONFIG" TO
MS_ABA_EDITOR_ROLE;
GRANT SELECT, UPDATE, INSERT, DELETE ON "ABA0009_EAUCTION_CONFIG" TO
MS7ABA;
```

3.6.23 AQA0018_RETAIL_BID_PROC_IND

This table is to store the indicators for Retail Bid Process Batch Job

```
CREATE TABLE "MS9ABA"."AQA0018_RETAIL_BID_PROC_IND"(
  "AQA0018_RETAIL_BID_PROC_DATE" VARCHAR2(10 BYTE),
  "AQA0018_RETAIL_BID_IND" VARCHAR2(1 BYTE),
  CONSTRAINT "AQA0018_RETAIL_BID_PROC_DATE" PRIMARY KEY
("AQA0018_RETAIL_BID_PROC_DATE")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP"  ENABLE )  SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
  NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1
MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "ABAAPP" ;

grant select,update,insert,delete on "MS9ABA"."AQA0018_RETAIL_BID_PROC_IND" to
MS7ABA;
grant select,update,insert,delete on "MS9ABA"."AQA0018_RETAIL_BID_PROC_IND" to
MS_ABA_EDITOR_ROLE;
```

3.6.24 ABA0036_RETAILBID_FILE

This table stores the acknowledgement file details only for the files uploaded in eAuction Online

```
CREATE TABLE "MS9ABA"."ABA0036_RETAILBID_FILE"
  (
	"ABA0036_FILE_NAME" VARCHAR2(50 BYTE) NOT NULL ENABLE,
	"ABA0036_PROCESSING_DATE" DATE NOT NULL ENABLE,
	"ABA0036_PROCESSING_TIME" TIMESTAMP (6),
	"ABA0036_ACKNOWLEDGEMENT_FILE_PATH"  VARCHAR2(300 BYTE),
	"ABA0036_ACKNOWLEDGEMENT_FILE_NAME" VARCHAR2(150 BYTE),
	CONSTRAINT "UK_ABA0036" UNIQUE ("ABA0036_FILE_NAME",
"ABA0036_PROCESSING_DATE", "ABA0036_ACKNOWLEDGEMENT_FILE_NAME")
 ) TABLESPACE "ABAAPP" ;

GRANT SELECT, UPDATE, INSERT, DELETE ON MS9ABA.ABA0036_RETAILBID_FILE TO
MS_ABA_EDITOR_ROLE;
```

GRANT SELECT, UPDATE, INSERT, DELETE ON MS9ABA.ABA0036_RETAILBID_FILE TO MS7ABA;

3.7 Preventing Concurrent Login

The prevention of concurrent login is to disable multiple login of the same login user id from different system/browsers.
This prevention will only allow one active session at a time per login user id.
The prevention process will be of first in, first out.

During user id login to the eAuction application, if an existing session of the same login user id was found, the existing session will be "Kick out".

Design:

1. Once authentication is successful (MainMenu.java), delete any record in the AQA0011_USER_SESSION table with the login user id and insert a new record into the table with Login User ID and JSESSION ID

2. In Login Filter - to check against the new table AQA0011_USER_SESSION whether a record exists for the login user id and JSESSION ID

    - If record exist, allow the user to continue
    - If record does not exist, log the action to audit log as "Concurrent login", invalidate the session and redirect to login page

3. While user clicks Logout, delete the record for the login user id from the DB table AQA0011_USER_SESSION

Example:

1. User id: AdminUser3 login to the eAuction application.
    a) System Verify: there is an existing login of AdminUser3 found in another session (a different system/browser)
    b) System action: Replace the existing login session details with new login session details in Database.

2. User id: AdminUser3 perform action on the session in point 1a).
    - System Verify: Existing login session details does not match with the new login session details in Database
    - System action: "Kick out" - redirect to the login page.

3. User id: AdminUser3 continue perform action on the session in point 1 accessing to eAuction application

4 User interface

General

4.1 SGSLogin.jsp

4.2 LoginFailed.jsp

4.3 Logoff.jsp

4.4 Timeout.jsp

4.5 SGSMainMenu.jsp

4.6 TransactionFailed.jsp

4.7 UnauthorisedAccess.jsp

4.8 PageNotFound.jsp

Browse

4.9 SelectSecIssue.jsp

4.10 ViewHistSec.jsp

4.11 ViewOngoingSec.jsp

Issue Code	Issue Description	Tender Date	Issue Date	Due Date
BS13202G	6 mths Tbill	12/04/2013	14/04/2013	13/10/2013
BS13202H	6 mths Tbill	14/04/2013	16/04/2013	15/10/2013
ML13201G	12 wk Mbill	12/04/2013	14/04/2013	07/07/2013
ML13201H	12 wk Mbill	14/04/2013	16/04/2013	09/07/2013
N213201G	2-yr Bond	12/04/2013	14/04/2013	14/04/2015
N213201H	2-yr Bond	14/04/2013	16/04/2013	16/04/2015

4.12 ViewSecDetails.jsp

No.	Issue Date	Amount Offered	Ave Yield
1	27/04/2013	1,000,000,000	0.0

4.13 SelectSecDealer.jsp

4.14 ViewSecDealers.jsp

User

4.15 UserMaintenanceMenu.jsp

4.16 UserMaintenance.jsp

4.17 SelectUser.jsp

4.18 AddNewUser.jsp

4.19 BrowseUsers.jsp

User Id	Access Level	Last Login Date	Access Change Date
ABC	Access level 0 : Non-Authorised User	06/09/2012	06/09/2012
ADMINUSER1	Access level 1 : Authorised User can browse database only	22/08/2012	31/05/2004
ADMINUSER2	Access level 2 : Authorised User can add/update tenders	12/04/2013	30/07/2004
ADMINUSER3	Access level 3 : Authorised User can verify/release tenders and run auction/view reports	12/04/2013	18/06/2004
ADMINUSER4	Access level 4 : Authorised User can add/update/verify/release tenders	03/01/2012	03/06/2004
ADMINUSER5	Access level 5 : Authorised User can view auction reports	20/02/2013	07/12/2011
ADMINUSER6	Access level 6 : Authorised User can run auction/view reports	24/10/2012	18/06/2004
ADMINUSER7	Access level 9 : System Administrator and user maintenance	12/04/2013	07/12/2011

4.20 UpdateUser.jsp

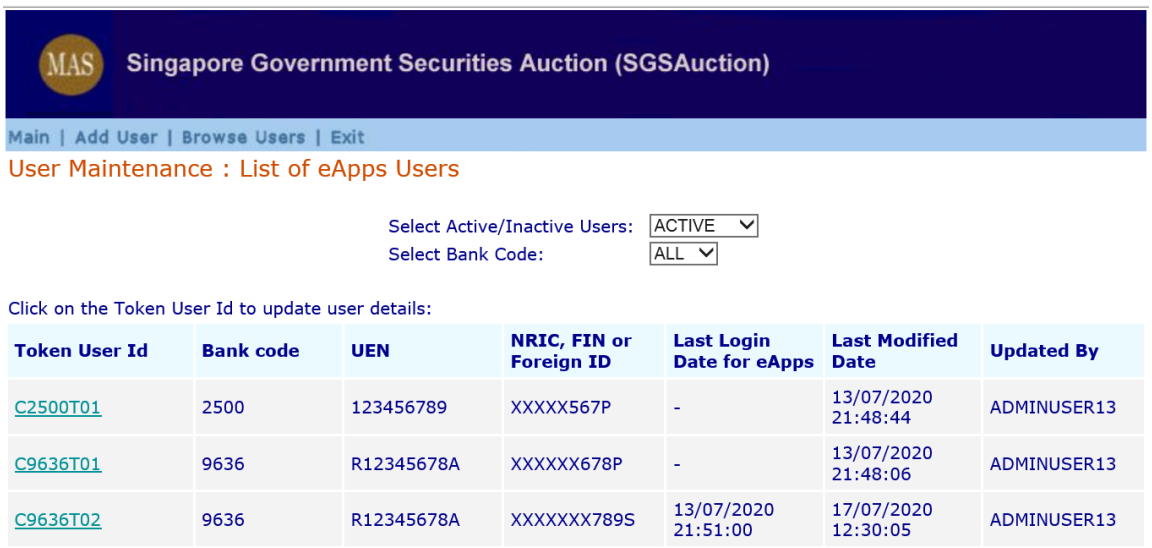
4.21 UserAdded.jsp

4.22 UserUpdated.jsp

4.23 EappsUserMaintenance.jsp

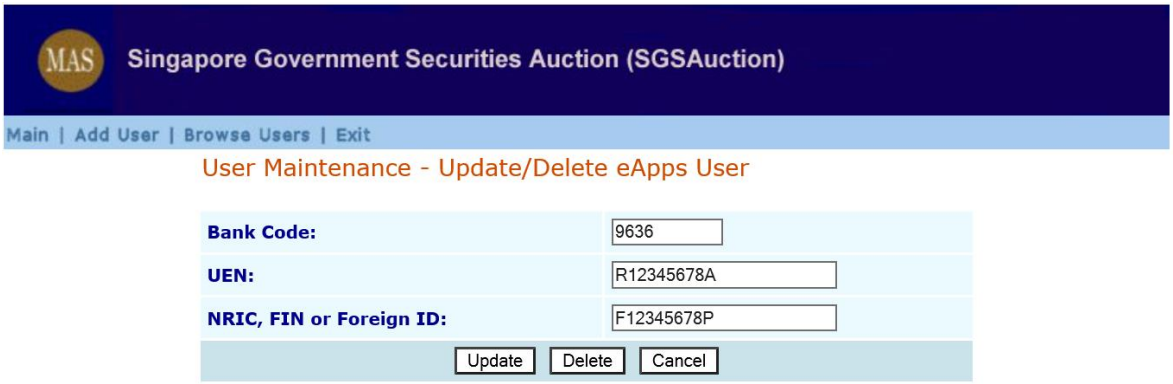
4.24 AddNewEappsUser.jsp

4.25 BrowseEappsUsers.jsp

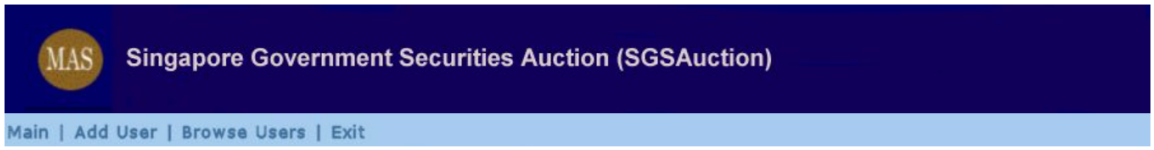


Token User Id	Bank code	UEN	NRIC, FIN or Foreign ID	Last Login Date for eApps	Last Modified Date	Updated By
C2500T01	2500	123456789	XXXXX567P	-	13/07/2020 21:48:44	ADMINUSER13
C9636T01	9636	R12345678A	XXXXXX678P	-	13/07/2020 21:48:06	ADMINUSER13
C9636T02	9636	R12345678A	XXXXXXX789S	13/07/2020 21:51:00	17/07/2020 12:30:05	ADMINUSER13

4.26 UpdateEappsUser.jsp



4.27 EappsUserAdded.jsp



User Maintenance - eApps User Added

- System generated Token User Id **C9636T01** has been added to eApps user table with below User details.
  - UEN: R12345678A
  - NRIC, FIN or Foreign ID: F12345678P

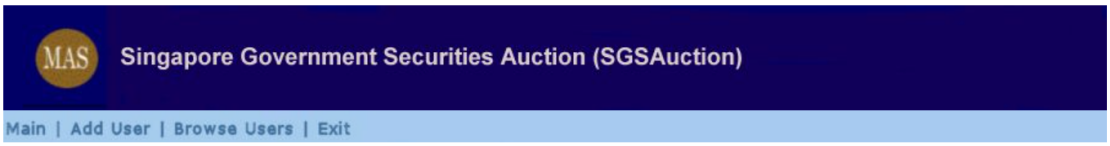
User Maintenance - eApps User Added

- System generated Token User Id **C9636T02** has been updated to Active in eApps user table with below User details.
  - UEN: R12345678A
  - NRIC, FIN or Foreign ID: F123456789S

4.28 EappsUserUpdated.jsp

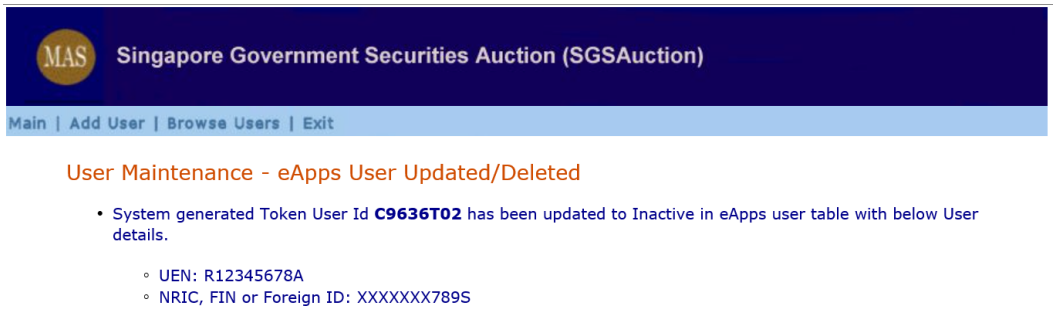
User Maintenance - eApps User Updated/Deleted

- System generated Token User Id **C9636T01** has been updated in eApps user table with below User details.
  - UEN: R12345678A
  - NRIC, FIN or Foreign ID: F22345678P



User Maintenance - eApps User Updated/Deleted

- System generated Token User Id **C9636T01** has been deleted in eApps user table with below User details.
  - UEN: R12345678A
  - NRIC, FIN or Foreign ID: F22345678P



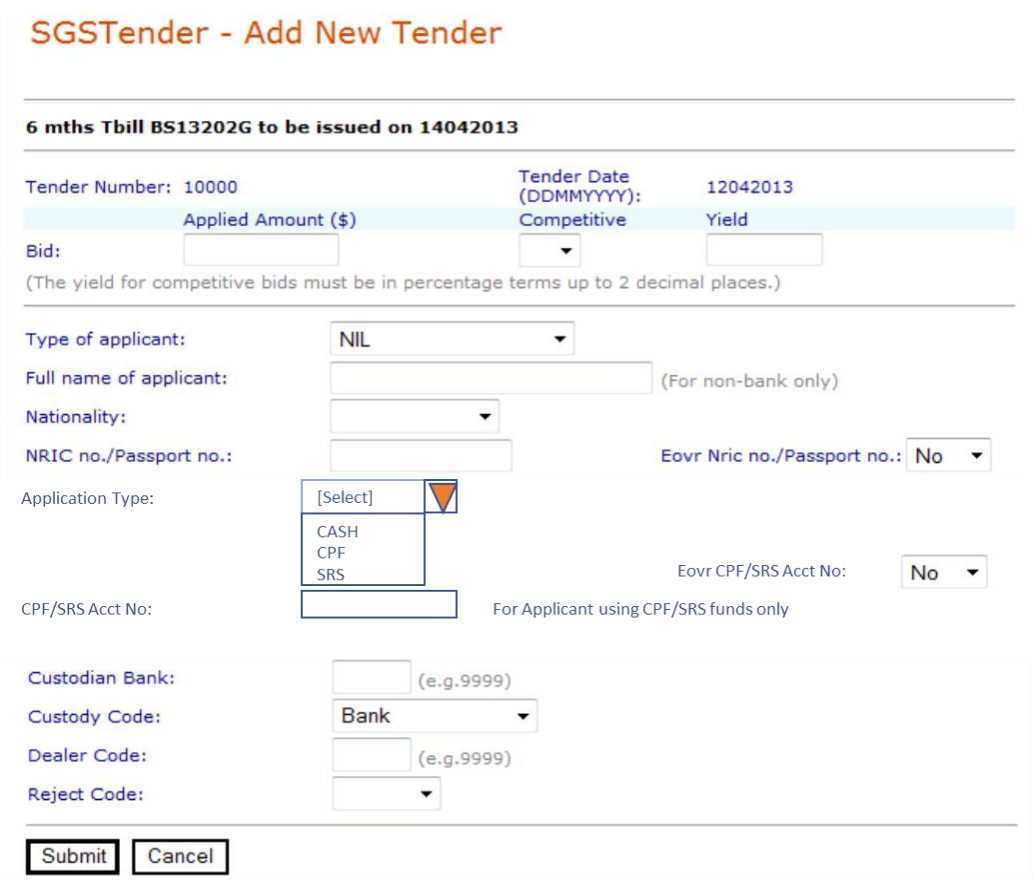
Tender

4.29 SGSTenderMenu.jsp

Add tender

4.30 SelectTenderAdd.jsp

4.31 AddNewTender.jsp



Update tender

4.32 SelectTenderUpdate.jsp

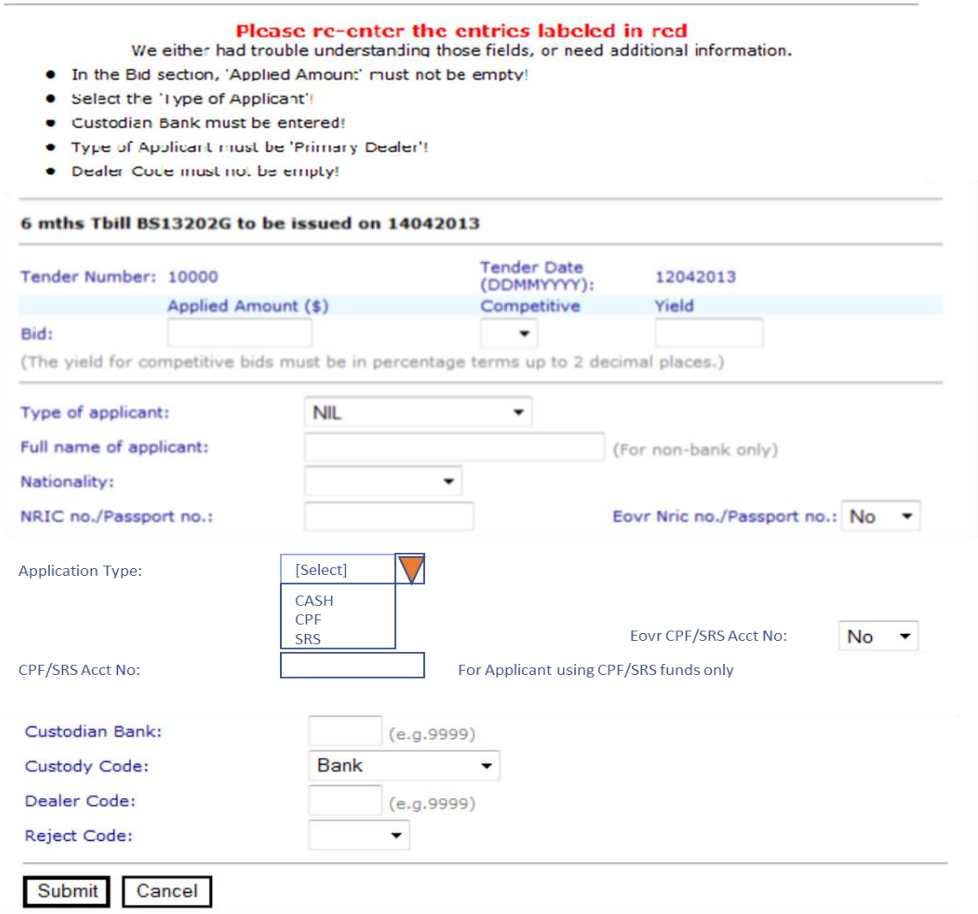
4.33 UpdateTender.jsp

4.34 TenderAdded.jsp

4.35 TenderUpdated.jsp

4.36 TenderDeleted.jsp

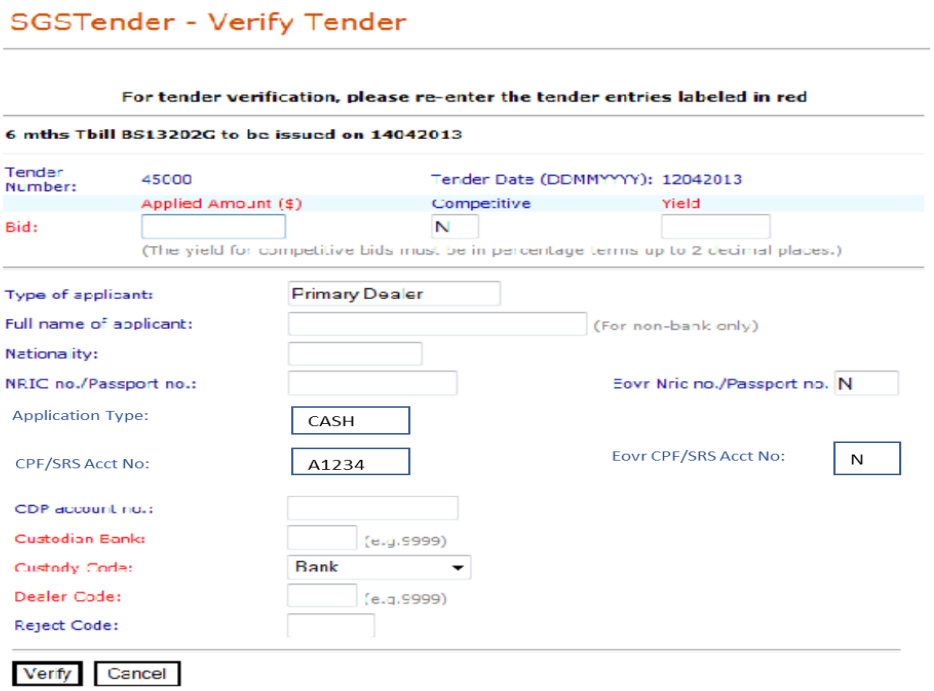
4.37 TenderResubmit.jsp



Verify tender

4.38 SelectVerifyTender.jsp

4.39 VerifyTender.jsp



4.40 ReVerifyTender.jsp

SGSTender - ReVerify Tender

6 mths Tbill BS13202E to be issued on 20042013

Please re-enter the entries labeled in red

We either had trouble understanding those fields, or need additional information.

- Unmatched Amount Applied, old value = 5000
- Unmatched Bid Yield, old value = 2.05
- Unmatched Custodian Bank, old value = 7375
- Unmatched Custody Code, old value = C
- Unmatched Dealer Code, old value = 5027

Tender Number: 45000 Tender Date (DDMMYYYY): 12042013

	Applied Amount ($)	Competitive	Yield
Bid:		N	

(The yield for competitive bids must be in percentage terms up to 2 decimal places.)

Type of applicant: Primary Dealer

Full name of applicant: (For non-bank only)

Nationality:

NRIC no./Passport no.: Eovr Nric no./Passport no. N

Application Type: CASH

CPF/SRS Acct No: A1234 Eovr CPF/SRS Acct No: N

CDP account no.:

Custodian Bank: (e.g.9999)

Custody Code: Bank

Dealer Code: (e.g.9999)

Reject Code:

Verify Cancel

4.41 ReleaseTender.jsp

SGSTender - Release Tender

6 mths Tbill BS13202G to be issued on 14042013

Tender Number 45000 | Tender Date (DDMMYYYY): 12042013

	Applied Amount ($)	Competitive	Yield
Bid:	51000	N	

(The yield for competitive bids must be in percentage terms up to 2 decimal places.)

Type of applicant: Primary Dealer

Full name of applicant: (For non-bank only)

Nationality:

NRIC no./Passport no.: Eovr Nric no./Passport no. N

Application Type: CASH

CPF/SRS Acct No: A1234 Eovr CPF/SRS Acct No: N

CDP account no.:

Custodian Bank: 7171 (e.g.9999)

Custody Code: Bank

Dealer Code 7171 (e.g.9999)

Reject Code

Release Cancel

4.42 TenderReleased.jsp

Transaction Succeeded.

- Tender number 45000 has been released. Please print a copy for reference.
- User Id: ITDLXM.
- Security Code: BS13202G.

Return Print

Browse tenders

4.43 BrowseAllTenders.jsp

Main | New Tender | Browse Tenders | Update Tender | View UnRel | Verify Tender | Reactivate | Trigger Retail Bid | Exit

SGSTender - List of Tenders

Click on the tender number to enquire:

Seq No.	Tender No.	Issue Code	Dealer Code	Cust Bank	Cust Code	Bid Yield	Amount Applied	Rel Status	Allot Status
1	600000	M100000F	7171	7171	B	1.22	1,000	Rel	Pending
2	600000	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
3	600001	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
4	600002	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
5	600003	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
6	600004	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
7	600005	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
8	600006	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
9	600007	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
10	600008	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
11	600009	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
12	600010	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
13	600011	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
14	600012	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
15	600013	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
16	600014	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
17	600015	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
18	600016	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
19	600017	M162317F	7171	7171	B	1.22	1,000	Rel	Pending
20	600018	M162317F	7171	7171	B	1.22	1,000	Rel	Pending

<< 1 >> of 2

4.44 ViewUnRelTenders.jsp

Seq No.	Tender No.	Issue Code	Dealer Code	Cust Bank	Cust Code	Bid Yield	Amount Applied
1	5001	BQ00003C	7533	7533	B	17.2	11,000,000
2	1	BQ06142N	7171	7171	B	0.0	2,000
3	10005	BQ09010J	7171	7171	B	1.0	5,000,000
4	10006	BQ09010J	7375	7375	B	1.0	50,000
5	10007	BQ09010J	7171	7171	B	1.2	100,000
6	10008	BQ09010J	7375	7375	B	1.0	1,000
7	10001	BQ09139N	7463	7463	B	1.2	133,500,000
8	1518	BQ99135E	7171	7171	B	0.59	100,000
9	1523	BQ99135E	7171	7171	B	0.6	19,000
10	1524	BQ99135E	9016	7171	B	0.0	11,000
11	1525	BQ99135E	7171	7171	B	0.0	12,000
12	1528	BQ99135E	9016	7171	B	0.0	2,000
13	1530	BQ99135E	9016	7171	B	0.55	1,000
14	1532	BQ99135E	7171	7171	B	0.0	1,000
15	1533	BQ99135E	7171	7171	B	0.0	1,000
16	1535	BQ99135E	7171	7171	B	0.56	10,000
17	1538	BQ99135E	7010	7010	B	0.54	1,000
18	1540	BQ99135E	7171	7171	B	0.0	12,000
19	1545	BQ99135E	2500	2500	B	0.54	9,000
20	1546	BQ99135E	7171	7171	B	0.0	5,000
21	1550	BQ99135E	7171	7171	B	0.54	40,000

Browse allot result

4.45 ~~SelectAllotResult.jsp~~ (Removed Jun 2013)

4.46 ~~ViewAllotResult.jsp~~ (Removed Jun 2013)

Reactivate tender

4.47 SelectTenderReactivate.jsp

4.48 ReActRelTender.jsp

SGSTender - Reactivate Tender

6 mths Tbill BS13202G to be issued on 14042013

Tender Number	45000	Tender Date (DDMMYYYY):	12042013
	Applied Amount ($)	Competitive	Yield
Bid:	51000	N	

(The yield for competitive bids must be in percentage terms up to 2 decimal places.)

Type of applicant:	Primary Dealer		
Full name of applicant:		(For non-bank only)	
Nationality:			
NRIC no./Passport no.:		Eovr Nric no./Passport no.	N
Application Type:	CASH		
CPF/SRS Acct No:	A1234	Eovr CPF/SRS Acct No:	N
CDP account no.:			
Custodian Bank:	7171 (e.g.9999)		
Custody Code:	Bank		
Dealer Code	7171 (e.g.9999)		
Reject Code			

Reactivate Cancel

4.49 TenderUnReleased.jsp

MAS Singapore Government Securities Auction (SGSAuction)

Main | New Tender | Browse Tenders | Update Tender | View UnRel | Verify Tender | View AllotResult | Reactivate | Exi

Transaction Succeeded.

- The tender 45000 has been unreleased. Please print a copy for reference.
- User Id: ADMINUSER3.
- Security Code: BS13202G.

Return Print

<u>Trigger retail bid (can access by Level 2 and Level 3 users only.)</u>

4.50 UploadTenderList.jsp

Singapore Government Securities Auction (SGSAuction)

Main | New Tender | Browse Tenders | Update Tender | View UnRel | Verify Tender | Reactivate | Trigger Retail Bid | Exit

SGSTender - Trigger Retail Bid

Click on the button to trigger retail bid batch:

File Name	
ea_dbs_ap1.txt.pgp	Delete
ea_dbs_ap2.txt.pgp	Delete
ea_dbs_ap3.txt.pgp	Delete
ea_ocbc_ap1.txt.pgp	Delete
ea_ocbc_ap2.txt.pgp	Delete
ea_ocbc_ap3.txt.pgp	Delete
ea_uob_ap1.txt.pgp	Delete
ea_uob_ap2.txt.pgp	Delete
ea_uob_ap3.txt.pgp	Delete
Trigger	

Acknowledgement file download

4.51 AcknowledgementFileDownloadMenu.jsp

Singapore Government Securities Auction (SGSAuction)

Main | New Tender | Browse Tenders | Update Tender | View UnRel | Verify Tender | Reactivate | Trigger Retail Bid | Exit

SGSTender - Acknowledgement File Download

Please enter processing date:

Download acknowledgement file for date 17/01/2024 (dd/mm/yyyy)

Submit Clear

4.52 AcknowledgementFileList.jsp

Main | New Tender | Browse Tenders | Update Tender | View UnRel | Verify Tender | Reactivate | Trigger Retail Bid | Exit

SGSTender - Acknowledgement File Download

Download Retail Bid Acknowledgement Submitted via eAuction on date 17/01/2024:

File Name	Acknowledgement File	
ea_dbs_ap3.txt.pgp	RT_MASSG.DBS.ABA.ea_dbs_aa3.txt.pgp.202401170015	Download
ea_uob_ap3.txt.pgp	RT_MASSG.UOBSFTP.ABA.ea_uob_aa3.txt.pgp.202401170021	Download
ea_ocbc_ap1.txt.pgp	RT_MASSG.OCBCSSB.ABA.ea_ocbc_aa1.txt.pgp.202401170016	Download
ea_dbs_ap2.txt.pgp	RT_MASSG.DBS.ABA.ea_dbs_aa2.txt.pgp.202401170014	Download
ea_uob_ap2.txt.pgp	RT_MASSG.UOBSFTP.ABA.ea_uob_aa2.txt.pgp.202401170020	Download
ea_ocbc_ap2.txt.pgp	RT_MASSG.OCBCSSB.ABA.ea_ocbc_aa2.txt.pgp.202401170017	Download
ea_uob_ap1.txt.pgp	RT_MASSG.UOBSFTP.ABA.ea_uob_aa1.txt.pgp.202401170019	Download
ea_dbs_ap1.txt.pgp	RT_MASSG.DBS.ABA.ea_dbs_aa1.txt.pgp.202401170013	Download
ea_ocbc_ap3.txt.pgp	RT_MASSG.OCBCSSB.ABA.ea_ocbc_aa3.txt.pgp.202401170018	Download

Auction

4.53 AuctionMenu.jsp

4.54 AuctionSecParaList.jsp (input parameters)

- Input Upper Bound, Lower Bound, MAS Max Amount (in mil) and reference point

4.55 AuctionSecParaList.jsp (verify parameters)

- Input Upper Bound, Lower Bound, MAS Max Amount (in mil) and reference point

Security Code	Security Description	Issue Date	Closing Date	Issue Size	Upper Bound	Lower Bound	Reference Point	MAS Max Amount (in mil)
N212100A	2-yr Bond	02/02/2013	21/01/2013	2,000,000,000	0.5	0.1	0.3	800
N512400A	5-yr Bond	01/02/2013	21/01/2013	1,500,000,000	0.5	0.1	0.3	800
NX12400A	10-yr Bond	10/03/2013	21/01/2013	2,000,000,000	0.5	0.1	0.3	800

4.56 AuctionSecList.jsp

- If auction is not running, auction run indicator = null

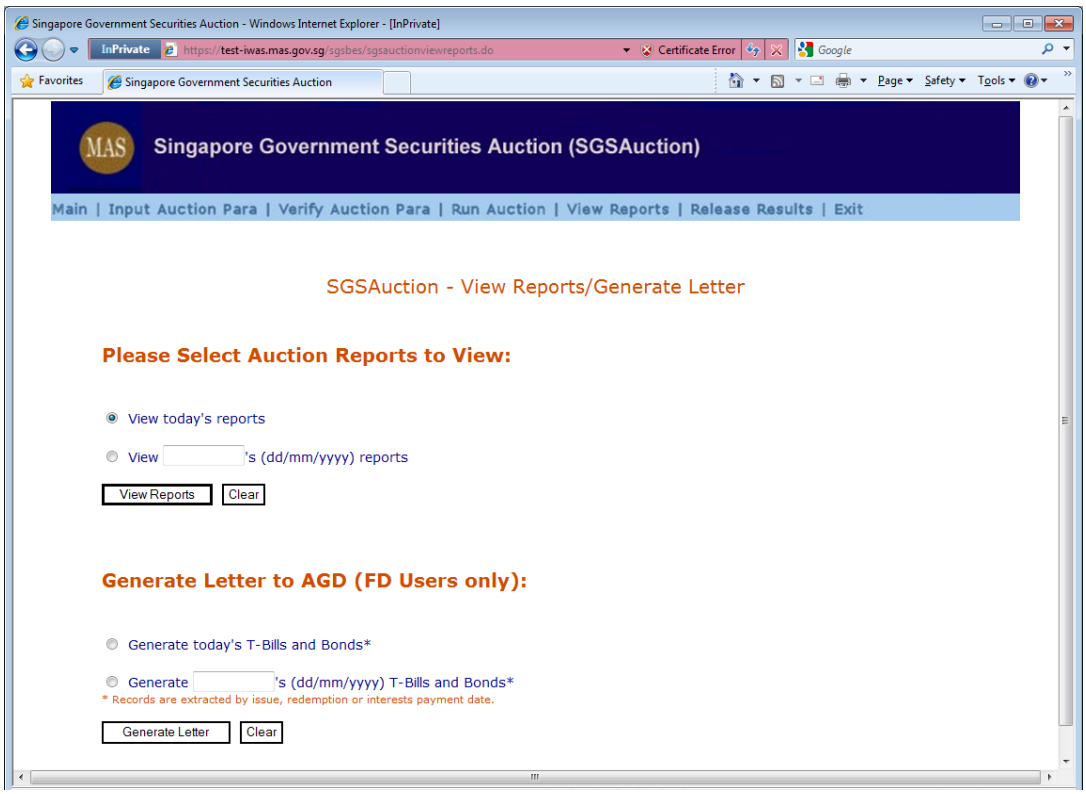
Security Code	Security Description	Issue Date	Closing Date	Issue Size	Reference Point	Upper Bound	Lower Bound	MAS Max Amount
B230906A	6M TBILL A TEST B2396	07/09/2023	06/09/2023	1,400,000,000				
NBBS300S	20YR TBILL TEST - CTG Testing	14/09/2023	06/09/2023	1,400,000,000	-	-	-	-
NBBS300T	20YR TBILL TEST- CTG Testing 1	14/09/2023	06/09/2023	1,400,000,000	-	-	-	-

- If auction is currently running, auction run indicator = R

- If auction is completed with failure, auction run indicator = F

- If auction is completed with success, auction run indicator = S

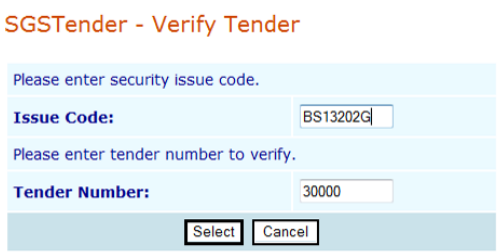
4.57 AuctionViewReportMenu.jsp



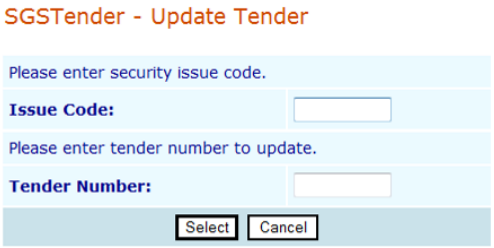
4.58 AuctionReportList.jsp

Report Name	Report Description
DECRYPTBIDS	Control Summary for Decrypt bids
MSABXT00	Control Summary for MSABXT00
MSABRT05_01	Summary of Application/Tender & Allotment of Government Securities
MSABRT05_01	Summary of Application/Tender & Allotment of Government Securities in .csv format
MSABRT05_02	Summary of Application/Tender & Allotment of Government Securities (Sorted by Custodian Banks)
MSABRT05_03	Summary of Application/Tender & Allotment of Government Securities by Bank & Non-Bank Categories
MSABRT05_03	Summary of Application/Tender & Allotment of Government Securities by Bank & Non-Bank Categories in .csv format
MSABRT08_01	Yield and Listing - List of bid yield and calculated price
MSABRT08_02	Yield and Listing - List of bid yield, calculated price, amount applied, amount allotted, etc.
MSABRT08_02	Yield and Listing - List of bid yield, calculated price, amount applied, amount allotted, etc., in .csv format
MSABRT06	Letters to Securities Dealers on Results of Tenders - **With effect from 17/11/2023, MSABRT06 report can be found on eApps.**
MSABRT11	Auction Allocation Reports
DetermineMedianYield	Median Yield Report

4.59 User Not Authorised Messages

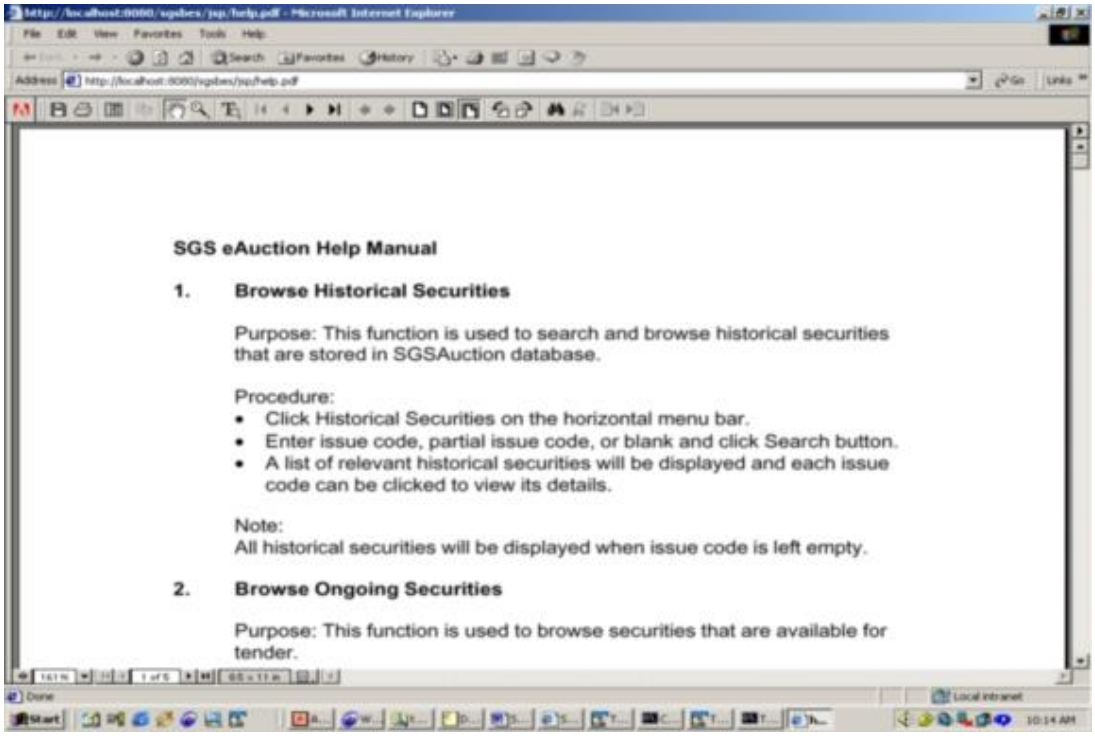


Tender must be verified and released by another authorised user.



Tender must be updated by another authorised user.

4.60 help.pdf



SGS eAuction Help Manual

1. Browse Historical Securities

Purpose: This function is used to search and browse historical securities that are stored in SGSAuction database.

Procedure:
- Click Historical Securities on the horizontal menu bar.
- Enter issue code, partial issue code, or blank and click Search button.
- A list of relevant historical securities will be displayed and each issue code can be clicked to view its details.

Note:
All historical securities will be displayed when issue code is left empty.

2. Browse Ongoing Securities

Purpose: This function is used to browse securities that are available for tender.

5 Data format

5.1 Security update file for MEPS+ and FMBS

The format of "secupdt.txt" and “secupdt_fmbs.txt” is:

S/N	Field Name	Format	Position		Remarks
			Fr	To	
1.	ISSUE_CODE	X(8)	1	8	E.g. NX00100H
2.	ISSUE_NO	9(1)	9	9	1 for new issuance 2 – 9 for re-opening
3.	ISSUE_TYPE	X(1)	10	10	Value is always space.
4.	CURR	X(3)	11	13	Value 'SGD'
5.	ISSUE_DESC	X(30)	14	43	Description of ISIN
6.	ISSUE_DATE	X(8)	44	51	YYYYMMDD
7.	TENDER_DATE	X(8)	52	59	YYYYMMDD
8.	QTY_OFFERED	9(13)	60	72	Total nominal amount for auction
9.	QTY_APPLIED	9(13)	73	85	
10.	**AVE_YIELD**	**S9(3)v9(2)**	**86**	**91**	**Average Yield with +/- sign**
11.	**CUT_OFF_YIELD**	**S9(3)v9(2)**	**92**	**97**	**Cutoff Yield with +/- sign**
12.	MATURITY_DATE	X(8)	**98**	**105**	YYYYMMDD
13.	PERCENT_COY	9(3)v9(2)	**106**	**110**	
14.	PERCENT_SUB	9(3)v9(2)	**111**	**115**	
15.	NC_PERCENT	9(3)v9(2)	**116**	**120**	
16.	NC_QTY_ALLOT	9(13)	**121**	**133**	
17.	INT_RATE	9(3)v9(4)	**134**	**140**	Contain zeroes for new ISIN. Contain coupon rate for re-open ISIN.
18.	TAX_STATUS	X(1)	**141**	**141**	Y or N
19.	CUT_OFF_YIELD_PRICE	9(3)v9(4)	**142**	**148**	
20.	AVE_YIELD_PRICE	9(3)v9(4)	**149**	**155**	

S/N	Field Name	Format	Position		Remarks
			Fr	To	
21.	CLOSING_PRICE	9(3)v9(4)	**156**	**162**	
22.	ISIN_CODE	X(12)	**163**	**174**	
23.	TENOR	9(3)	**175**	**177**	For T-Bills, will be in day(s) For Bonds, will be in year(s)
24.	ETENDER_IND	X(1)	**178**	**178**	'Y' if ISSUE_DATE>14012002
25.	MAS_APPLIED	9(13)	**179**	**191**	
26.	MAS_ALLOTTED	9(13)	**192**	**204**	
27.	INT_PAID_IND	X(1)	**205**	**205**	
28.	LAST_INT_DATE	X(8)	**206**	**213**	YYYYMMDD
29.	NEXT_INT_DATE	X(8)	**214**	**221**	YYYYMMDD
30.	ACCRUED_INT_DAYS	9(3)	**222**	**224**	000
31.	INT_DATE1	9(4)	**225**	**228**	MMDD
32.	INT_DATE2	9(4)	**229**	**232**	MMDD
33.	EX_INT_DATE	X(8)	**233**	**240**	YYYYMMDD

- Unused area will be padded with leading zeroes or spaces accordingly.
- '9' denotes field having numeric value(s).
- 'X' denotes field having alphanumeric value(s).
- 'S' denotes +/- sign.
- For Treasury Bills and MAS Bills, the fields filled will be up to number 26. For Bonds, up to number 33.
- If the record length is more than required, the additional characters will be ignored.

The change is on the format for AVE_YIELD and CUT_OFF_YIELD fields in 'secupdt.txt' interface file by including +/- sign in front of the figure to handle negative value.

Current format 9(3)v9(2)
Proposed format **S**9(3)v9(2)
where 'S' denotes +/- sign

eApps should provide the value in the following format:

123.45	→	+12345
23.45	→	+02345
3.4	→	+00340
0	→	+00000
-123.45	→	-12345
-23.45	→	-02345
-3.4	→	-00340

5.2 Auction Result for MEPS+

The format of "auction.txt" is:

S/N	Field Name	Format	Position		Remarks
			Line	Col	
1.	ISIN_CODE	X(12)	1	1	E.g. SG1234567890
2.	ISSUE_CODE	X(8)	1	2	E.g. NX00100H
3.	ISSUE_DATE	X(8)	1	3	YYYYMMDD
4.	REC_NUM	9(3)	1	4	Total number of records.
	Repetitive Auction Details				
5.	MBR_CODE	X(8)	2	1	E.g. OCBCSGSG
6.	BANK_CODE	9(4)	2	2	E.g. 7339
7.	CUSTODY_CODE	X(3)	2	3	E.g. TRD (Trade)
8.	PRICE	9(3)v9(5)	2	4	E.g. 12400500 (124.005)
9.	NOMINAL_AMT	9(13)	2	5	E.g. 0000000100000
10.	SETT_AMT	9(13)v9(2)	2	6	E.g. 000000010000000 ($100,000.00)
	End of Auction Indicator				
11.	End_Txn	X(6)	N	1	ENDTXN

- Field 5 to 10 will be repetitive for each auction detail record.
- Each auction detail record must start on a new line.
- All fields/columns should be delimited with a delimiter ';'
- '9' denotes field having numeric value(s).
- 'X' denotes field having alphanumeric value(s).
- All fields with decimal places must be padded with leading or trailing zeroes, up to the required decimal length.

- There will be 2 records for this txt file. One for CASH and the other for SRS. For each PDs, to sum the allotted amount for CASH and another for SRS.
- To encrypt this file and decrypt when sending to MEPS+ MQ.
- Sequence of list:
  - CASH
  - SRS

5.3 Bank Auction Result for FMBS

The format of "auction_fmbs.txt" is:

S/N	Field Name	Format	Position		Remarks
			Line	Col	
1.	ISIN_CODE	X(12)	1	1	E.g. SG1234567890
2.	ISSUE_CODE	X(8)	1	2	E.g. NX00100H
3.	ISSUE_DATE	X(8)	1	3	YYYYMMDD
4.	REC_NUM	9(3)	1	4	Total number of records, always 001
5.	MBR_CODE	X(8)	1	5	Always MASGSGSG
6.	BANK_CODE	9(4)	1	6	Always 2500
7.	CUSTODY_CODE	X(3)	1	7	Always TRD
8.	PRICE	9(3)v9(9)	1	8	E.g. 124005001234 (124.005001234), modified clean price
9.	NOMINAL_AMT	9(13)	1	9	E.g. 100000
10.	SETT_AMT	9(15)v9(2)	1	10	E.g. 10000000 ($100,000.00)

- Each auction record must start on a new line.
- All fields/columns should be delimited with a delimiter ';'
- ‘9’ denotes field having numeric value(s).
- ‘X’ denotes field having alphanumeric value(s).
- All fields with decimal places must be padded with trailing zeroes, up to the required decimal length.
- There will only be values if MAS is allotted bids.

6 Deployment

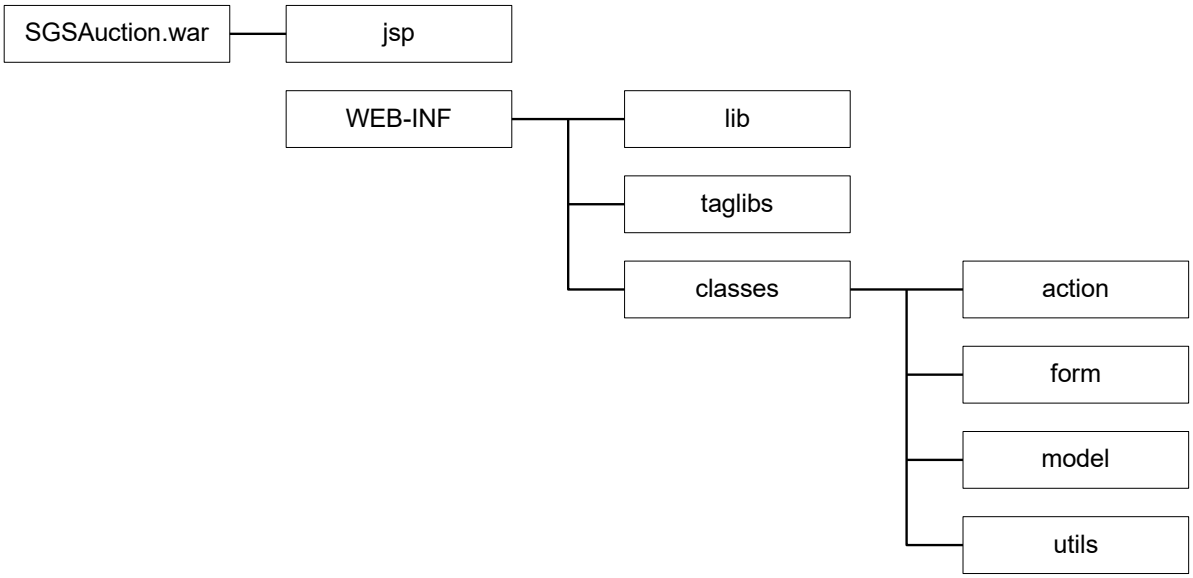
The deployment of SGS eAuction assumes all necessary hardware, operating systems, web servers, database server, Active Directory Authentication server and all network connections are in place already.

The deployment involves copying of web pages, copying of Java servlet classes, configuring interface with Active Directory Server and creating Oracle tables.

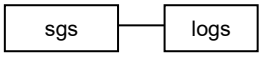
6.1 Copying of web pages and servlet files to WAS

6.1.1 Directory Structure on WAS

WEB related files are under directory "/was/AppServer/installedApps/SGSAuction.ear/sgsbes.war":

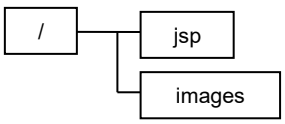


The configuration and error log files are under directory "/mas_appsw/was6/sgs":



The error log file, SGSAuctionError.log, is residing in this directory.

6.1.2 Directory Structure on HTTP Server



The access right to all directories should be "r—r—r—".

6.1.3 Copying of static web pages to HTTP Server

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp”:

- help.pdf

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/images”:

- blank.gif
- brn_logo.jpg
- computer.gif
- header_auction.jpg
- header_main.jpg
- header_tender.jpg
- header_user.jpg
- mas_logo.gif
- header_eappsuser.jpg
- header_usermaintenance.jpg

6.1.4 Copying of static web pages to WAS

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/”:

- error.jsp
- index.jsp
- LoginFailed.jsp
- Logoff.jsp
- Redirect.jsp
- SGSLogin.jsp
- Timeout.jsp
- UnauthorisedAction.jsp
- PageNotFound.jsp
- testDB.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp”:

- blank.jsp
- defaultLayout.jsp
- defaultLayoutNoTitle.jsp
- errormsg.jsp
- header.jsp
- header_nomenu.jsp
- header.jsp
- SGSMainMenu.jsp
- TransactionFailed.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/auction”:

- AuctionMenu.jsp
- AuctionReportList.jsp
- AuctionSecList.jsp
- AuctionSecParaList.jsp
- AuctionViewReportMenu.jsp
- header_auction.jsp
- message.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/auction/errormsg”:

- auction_errormsg1.jsp (removed Jun 2013)

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/browse”:

- SelectSecDealer.jsp
- SelectSecIssue.jsp
- ViewHistSec.jsp
- ViewOngoingSec.jsp
- ViewSecDealers.jsp
- ViewSecDetails.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/browse/errormsg”:

- Browse_errormsg.jsp (removed Jun 2013)

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/tender”:

- AddNewTender.jsp
- AddNewTenderFileAttach.jsp
- AddNewTenderMenu.jsp
- BrowseAllTenders.jsp
- BrowseTenderMenu.jsp
- header_tender.jsp
- ReActRelTender.jsp
- ReleaseTender.jsp
- ReVerifyTender.jsp
- SelectAllotResult.jsp
- SelectTenderAdd.jsp
- SelectTenderReactivate.jsp
- SelectTenderUpdate.jsp
- SelectVerifyTender.jsp
- SGSTenderMenu.jsp
- SGSUploadConfirm.jsp
- SGSUploadResult.jsp
- TenderAdded.jsp

- TenderDeleted.jsp
- TenderReleased.jsp
- TenderResubmit.jsp
- TenderUnReleased.jsp
- TenderUpdated.jsp
- UpdateTender.jsp
- VerifyTender.jsp
- ~~ViewAllotResult.jsp~~ (Removed Jun 2013)
- ~~ViewUnRelTenders.jsp~~ (Removed Jun 2013)

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/tender/errormsg”:

- tender_duplicate.jsp
- tender_errormsg1.jsp (removed Jun 2013)
- tender_errormsg2.jsp (removed Jun 2013)
- tender_noongoingsecurity.jsp
- tender_nosecurity.jsp
- tender_notender.jsp
- tender_sameuser id.jsp
- tender_verifiedtender.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/user”:

- AddNewUser.jsp
- BrowseUsers.jsp
- header_user.jsp
- SelectUser.jsp
- UpdateUser.jsp
- UserAdded.jsp
- UserMaintenance.jsp
- UserUpdated.jsp
- AddNewEappsUser.jsp
- BrowseEappsUsers.jsp
- EappsUserAdded.jsp
- EappsUserMaintenance.jsp
- EappsUserUpdated.jsp
- UpdateEappsUser.jsp
- UserMaintenanceMenu.jsp
- header_eappsuser.jsp
- header_usermaintenance.jsp

The following files should be copied to directory “SGSAuction.ear/sgsbes.war/jsp/user/errormsg”:

- editownuser id_errormsg.jsp
- subuser_errormsg.jsp
- updateuser_errormsg.jsp
- user_errormsg.jsp (removed Jun 2013)
- subeappsadduser_errormsg.jsp
- subeappsupdateuser_errormsg.jsp
- updateeappsuser_errormsg.jsp

6.1.5 Copying of Java classes to WAS

All the Java classes are copied to WAS machine.

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/action"

- Logoff.class
- MainMenu.class
- Timeout.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/action/auction"

- SGSAuctionDecryptBids.class
- SGSAuctionMenu.class
- SGSAuctionOutputReport.class
- SGSAuctionReleaseResult.class
- SGSAuctionRunAuction.class
- SGSAuctionRunAuctionOffline.class
- SGSAuctionSavePara.class
- SGSAuctionSecList.class
- ~~SGSAuctionSecParaList.class~~ (Removed Jun 2013)
- SGSAuctionSecParaInput.class (Added Jun 2013)
- SGSAuctionSecParaVerify.class (Added Jun 2013)
- SGSAuctionValidateBids.class
- SGSAuctionVerifyPara.class
- SGSAuctionViewReports.class
- SGSAuctionGenerateLetter.class (Added Jun 2013)

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/action/browse"

- SelectSecDealer.class
- SelectSecIssue.class
- ViewHistSec.class
- ViewOngoingSec.class
- ViewSecDealers.class
- ViewSecDetails.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/action/tender"

- AddNewTender.class
- AddNewTenderMenu.class
- AddTenderMethodSelect.class
- BrowseAllTenders.class

- BrowseTenderList.class
- MatchTender.class
- ReActRelTender.class
- SelectAllotResult.class
- SelectTender.class
- SGSTenderMenu.class
- SubTender.class
- TenderReleased.class
- TenderUnReleased.class
- UpdateTender.class
- UploadTenderConfirm.class
- UploadTenderResult.class
- VerifyTender.class
- ~~ViewAllotResult.class~~ (Removed Jun 2013)
- ~~ViewUnRelTenders.class~~ (Removed Jun 2013)

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/action/user"

- AddNewUser.class
- BrowseUsers.class
- SelectUser.class
- SubUser.class
- UpdateUser.class
- UserMaintenance.class
- UserMaintenanceMenu.class
- EappsUserMaintenance.class
- AddNewEappsUser.class
- BrowseEappsUsers.class
- SubEappsAddUser.class
- SubEappsUpdateUser.class
- UpdateEappsUser.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/form"

- LogonForm.class
- SearchForm.class
- SearchTenderForm.class
- TenderForm.class
- UserForm.class
- BrowseEappsUserForm.class
- EappsUserForm.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/model"

- BankMaster.class
- OneEappsTransaction.class

- OneTransaction.class
- SecurityMaster.class
- Transaction.class
- User.class
- AuthorisedAction.class (Added Jun 2013)
- EappsUser.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/sgsbes/auction"

- AuctionSystemLog.class
- DecryptBids.class
- DetermineMedianYield.class
- MSABAT00.class
- MSABAT01.class
- MSABAT02.class
- MSABIntf.class
- MSABReport.class
- MSABRT05.class
- MSABRT06.class
- MSABRT07.class
- MSABRT08.class
- MSABRT11.class
- MSABUT02.class
- MSABUT03.class
- MSABUT04.class
- MSABUT05.class
- MSABUT08.class
- MSABXT00.class
- ReleaseAuctionResult.class
- ReportAndLog.class
- RT11ApplicantSummary.class
- RunBankResult.class
- SetCorridorPara.class
- SgsbesException.class
- TransactionDescriptor.class
- Transactions.class
- GenerateLetter.class (Added Jun 2013)

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/sgseapps"

- PriceUtils.class
- SGSeAppsVariables.class
- Utils.class
- AuthorisedActionUtil.class (Added Jun 2013)

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/sgseapps/data"

- Decryptor.class
- Hex.class
- OddNoOfCharException.class
- ScrambleException.class
- ScrambleFacility.class
- ServiceLocator.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes/sgseapps/etender"

- ~~PriceUtils.class~~
- Securities.class
- CorridorPara.class

The following Java classes for SGS eAuction system should be copied to directory " SGSAuction.ear/sgsbes.war/WEB-INF/classes/utils/sqlmanager"

- DBUtils.class
- SQLBankMaster.class
- SQLManager.class
- SQLSecurityMaster.class
- SQLTransaction.class
- SQLUser.class
- SQLAuthorisedAction.class (Added Jun 2013)
- SQLEappsUser.class
- SQLPrimaryDealer.class

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/classes"

- application.properties

The following Java classes for SGS eAuction system should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF"

- struts-config.xml
- tiles-defs.xml
- web.xml

The following Java classes should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/lib":

- classes12.jar
- commons-beanutils.jar
- commons-collections.jar
- commons-dbcp.jar
- commons-digester.jar
- commons-fileupload.jar
- commons-lang.jar
- commons-logging.jar
- commons-pool.jar

- commons-resources.jar
- commons-services.jar
- commons-validator.jar
- dbconnbroker.jar
- gnu-regexp-1.0.8.jar
- jakarta-oro.jar
- jstl.jar
- sgsnet.jar
- standard.jar
- struts.jar
- sgsedt.jar (Added Jun 2013)
- log4j-1.2.8.jar (Added Jun 2013)

The following Java classes should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/taglibs":

- app.tld
- struts-bean.tld
- struts-html.tld
- struts-logic.tld
- struts-template.tld
- struts-tiles.tld

The following Java classes should be copied to directory "SGSAuction.ear/sgsbes.war/WEB-INF/taglibs/standard":

- c.tld
- fmt.tld
- sql.tld
- x.tld

6.1.6 Configuring interface with Active Directory (AD) server

The group "SGSAuction 2" (CN=SGSAuction 2,OU=SGSAuction,OU=Applications,OU=Resources,dc=mas,dc=gov,dc=sg) is defined in Active Directory server. All users of SGS eAuction are placed in this group.

6.2 Creating Oracle tables

Before creating Oracle tables, Oracle DBA should create necessary user ids and assign table space.

It involves the following steps:
- Run the DDL script file "CreateTable.sql" to create tables.
- Run a SQL script file "grant.sql" to grant access right to WAS users.

Subsequent tables are created with the following steps:
- Run the DDL script file "create_*AQAxxxx_TableName*.sql" to create tables and grant access rights to WAS users.

7 Appendix

7.1 NRIC No Validation Algorithm

Please refer to Algorithm purchased from ICA:
https://one.mas.gov.sg/dept/itd/M3/eApps/SGS%20eApps/Documentation/Agreements/NRIC-FIN%20ALGORITHM%20FORMULA%20CODE.doc

Formula for validation of NRIC (for Singaporeans) (obtained from payroll system)

	IC1	IC2	IC3	IC4	IC5	IC6	IC7	CHK
NRIC								

	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11
Check Digit Table	A	B	C	D	E	F	G	H	I	Z	J

Calculate :-

#NRIC-total = (IC1 * 2) + (IC2 * 7) + (IC3 * 6) + (IC4 * 5) + (IC5 * 4) + (IC6 * 3) + (IC7 * 2)

#Remainder = Mod (#NRIC-total ÷ 11)

i = 11 - #Remainder

*** For a valid NRIC, CKi = CHK

where CKi = (respective location of the alphabet in the Check Digit Table)

CHK = Last digit in NRIC

EMPLOYEE NRIC NO. CHECK DIGIT ROUTINE

LET THE NRIC NO. BE 1234567_

1) MULTIPLY THE NUMERIC DIGIT BY THE CORRESPONDING FACTORS -

2, 7, 6, 5, 4, 3, 2,

HENCE,	1	2	3	4	5	6	7
	*	*	*	*	*	*	*
	2	7	6	5	4	3	2
	2	14	18	20	20	18	14

2) COMPUTE THE TOTAL OF (1)

TOTAL = 2 + 14 + 18 + 20 + 20 + 18 + 14

= 106

3) DIVIDE TOTAL BY 11

106 / 11 = 9 REMAINDER 7

4) SUBTRACT REMAINDER FROM 11

11 - 7 = 4

5) THE VALID CHECK DIGIT SHOULD BE AS FOLLOWS -

REMAINDER	VALID CHECK DIGIT
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	Z
11	J

HENCE, THE VALID CHECK DIGIT SHOULD BE "D"
AND THE VALID NRIC NO. WOULD BE 1234567D

7.2 CPF No. Validation Algorithm

CONFIDENTIAL B
Compute check digit of CPF non-numeric a/c

Input Format

1st	2nd	3rd	4th	5th	6th	7th	8th	9th
S	X1	n	n	n	n	n	n	X2
T	X1	n	n	n	n	n	n	X2

Valid values :
1st char is either 'S' or 'T'.
2nd char is an alpha character.
3rd to 8th char is numeric character.
9th char is an alpha character.

Formula
1. Mulitply 3rd to 8th char with 7, 6, 5, 4, 3, 2 respectively
2. Sum the result of each mulitplication
3. **If 1st char is 'T', add 4 to the sum**
4. Add a weightage to the computed total. Please refer to Table 1.
5. Divide by 11 to obtain the remainder
6. Subtract the remainder from 11
   If the final result is 11, consider it as 0.
7. Check the final result(@) against the table found in Table 2 for 2nd char = 'A', 'B', 'C' or 'Y' and Table 3 for 2nd char ='D' or 'F' to determine the alpha character check digit for the 9th char.

Table 1 : Weightage

2nd Char value	Weightage
A	-1
B	1
C	4
D	0
F	0
Y	3

Table 2 : Reference Table for 9th Char value if 2nd char is = 'A', 'B', 'C' or 'Y'

@	9th char value
0	J
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H

9	I
10	Z

Table 3 : Reference Table for 9th Char value if 2nd char is = 'D' or 'F'

@	9th char value
0	A
1	B
2	C
3	E
4	J
5	K
6	L
7	N
8	P
9	X
10	V

Examples

1a. 1st char is 'S' and 2nd char is 'A'

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
	S	A	0	8	4	1	0	9	???
Multiply			7	6	5	4	3	2	
Sum			0	48	20	4	0	18	= 90
Add weightage (2nd char = 'A')									-1
Total									**89**
Divide by 11 to get Remainder									1
Final Result (11 - Remainder)								**= 11 - 1 = 10**	
Based on Table 2, 9th Char								= Z	

Result : **SA084109Z**

1b. 1st char is 'T' and 2nd char is 'A'

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
	T	A	0	8	4	1	0	9	???
Multiply			7	6	5	4	3	2	
Sum			0	48	20	4	0	18	= 90

Add weightage 4
(1st char = 'T') 4

Add weightage
(2nd char = 'A') -1

Total **93**

Divide by 11 to get
Remainder 5

Final Result (11 - Remainder) **= 11 - 5**
= 6

Based on Table 2, 9th Char = F

Result :**TA084109F**

2A. 1st char is 'S' and 2nd char is with 'B'

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
	S	B	0	0	0	0	0	1	???
Multiply			7	6	5	4	3	2	
Sum			0	0	0	0	0	2	= 2

Add weightage
(2nd char = 'B') 1

Total **3**

Divide by 11 to get
Remainder 3

Final Result (11 - Remainder) **= 11 - 3**
= 8

Based on Table 2, 9th Char = H

Result : **SB000001H**

2B. 1st char is 'T' and 2nd char is with 'B'

	1st	2nd	3rd	4th	5th	6th	7th	8th		9th
	T	B	0	0	0	0	0	1		???
Multiply			7	6	5	4	3	2		
Sum			0	0	0	0	0	2	=	2
Add weightage 4 (1st char = 'T')										4
Add weightage (2nd char = 'B')										1
Total										**7**
Divide by 11 to get Remainder										7
Final Result (11 - Remainder)										**= 11 - 7 = 4**
Based on Table 2, 9th Char										= D

Result : **TB000001D**

3A. 1st char is 'S' and 2nd char is with 'D'

	1st	2nd	3rd	4th	5th	6th	7th	8th		9th
	S	D	0	0	0	6	1	8		???
Multiply			7	6	5	4	3	2		
Sum			0	0	0	24	3	16	=	43
Add weightage (2nd char = 'D')										0
Total										**43**
Divide by 11 to get Remainder										10
Final Result (11 - Remainder)										**= 11 - 10 = 1**
Based on Table 2, 9th Char										= B

Result : **SD000618B**

3B. 1st char is 'T' and 2nd char is with 'D'

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
	S	D	0	0	0	6	1	8	???
Multiply			7	6	5	4	3	2	
Sum			0	0	0	24	3	16	= 43

Add weightage 4 (1st char = 'T')	4
Add weightage (2nd char = 'D')	0
Total	**47**
Divide by 11 to get Remainder	3
Final Result (11 - Remainder)	**= 11 - 3 = 8**
Based on Table 2, 9th Char	= P

Result : **TD000618P**

7.3 Bid Checksum Algorithm

- Algorithm for servlet SGSSubResult to calculate checksum:

Checksum	=	bid amount 1 + yield 1
	+	bid amount 2 + yield 2
	+	bid amount 3 + yield 3
	+	bid amount 4 + yield 4
	+	bid amount 5 + yield 5
	+	PD code + bank code
	+	transaction reference no.

- Algorithm for SGS Decryption Module to calculate overall checksum:

Overall checksum	=	bid checksum 1
	+	bid checksum 2
	+	….
	+	bid checksum n

7.4 Formulae used in SGS Daily Price calculation

7.4.1 Formulae for T-Bill

The formulae for price and yield calculations are as follows : -

$$\mathbf{D} = \frac{M}{365} \times R$$

$$\mathbf{P} = S\$100 - \left[\frac{M}{365} \times R \right]$$

Where

M	=	Days to maturity **(Settlement date* to maturity date)**
R	=	Annual rate of discount (yield), expressed as a percentage
D	=	Full discount per S$100 face value
P	=	Dollar price per S$100 face value

*** For New Issue of T-Bill, if settlement date is before issue date, then Days to maturity = Issue date to maturity date**

7.4.2 Formulae for Bond

7.4.2.1 Irregular Interest Period for New Issue

Normal

$$P = \left\{ \frac{RV}{\left(1 + \frac{Y}{200}\right)^{N - 1 + \frac{DSC}{E}}} \right\} + \left\{ \left\{ \sum_{K=2}^{N} \frac{\frac{CPN}{2}}{\left(1 + \frac{Y}{200}\right)^{K - 1 + \frac{DSC}{E}}} \right\} + \frac{FC}{\left(1 + \frac{Y}{200}\right)^{\frac{DSC}{E}}} \right.$$

$$- \mathbf{AI}$$

$$\mathbf{AI} = \left(\frac{CPN}{2} \times \frac{DIS}{E} \right) \qquad \mathbf{FC} = \left(\frac{CPN}{2} \times \frac{DIF}{E} \right)$$

Ex-Interest

$$P = \left\{ \frac{RV}{\left(1 + \frac{Y}{200}\right)^{N - 1 + \frac{DSC}{E}}} \right\} + \left\{ \sum_{K=2}^{N} \frac{\frac{CPN}{2}}{\left(1 + \frac{Y}{200}\right)^{K - 1 + \frac{DSC}{E}}} \right\} - \mathbf{AI}$$

$$\mathbf{AI} = - \left(\frac{CPN}{2} \times \frac{DSC}{E} \right)$$

where

FC	=	Amount of first interest (coupon) payment per S$100 face value
AI	=	Accrued interest per S$100 face value
DSC	=	Days between settlement date and first interest payment date
DIF	=	Days between issue date and first coupon date
DIS	=	Days between issue date and settlement date
E	=	Number of days in the first coupon period (6 months)
N	=	Number of semi-annual coupons payable between settlement date and maturity date
CPN	=	Annual coupon rate (as a percentage)
Y	=	Annual yield (as a percentage)
P	=	Dollar price per S$100 face value
RV	=	Redemption value

For New Issue of Bond only (not applicable to reopened issues)
If Settlement Date is before Issue date, then settlement date shd be equal to issue date.

7.4.2.2 With more than six months to maturity

Normal

$$P = \left\{ \frac{RV}{\left(1 + \frac{Y}{200}\right)^{N - 1 + \frac{DSC}{E}}} \right\} + \left\{ \sum_{K=1}^{N} \frac{\frac{CPN}{2}}{\left(1 + \frac{Y}{200}\right)^{K - 1 + \frac{DSC}{E}}} \right\} - AI$$

$$AI = \left(\frac{CPN}{2} \times \frac{DCS}{E} \right)$$

Ex-Interest

$$P = \left\{ \frac{RV}{\left(1 + \frac{Y}{200}\right)^{N - 1 + \frac{DSC}{E}}} \right\} + \left\{ \sum_{K=2}^{N} \frac{\frac{CPN}{2}}{\left(1 + \frac{Y}{200}\right)^{K - 1 + \frac{DSC}{E}}} \right\} - AI$$

$$AI = - \left(\frac{CPN}{2} \times \frac{DSC}{E} \right)$$

Where

AI	=Accrued interest per S$100 face value
DSM	=Days between settlement date and maturity date
DCS	=Days between beginning of current coupon period and settlement date
DSC	=E-DCS = days from settlement date to next 6-month coupon date
E	=Number of days in coupon period where settlement occurs
N	=number of sem-annual coupons payable between settlement date and maturity date
CPN	=Annual coupon rate (as a percentage)
Y	=Annual yield (as a percentage)
P	=Dollar price per S$100 face value
RV	=Redemption value

7.4.2.3 With six months or less to maturity

Normal

$$P = \left\{ \frac{100 \left\{ RV + \frac{CPN}{2} \right\}}{100 + \left\{ \frac{DSM}{E} \times \frac{Y}{2} \right\}} \right\} - AI$$

Where

$$AI = \left[\frac{CPN}{2} \times \frac{DCS}{E} \right]$$

Ex-Interest

$$P = \left\{ \frac{100 \times RV}{100 + \left\{ \frac{DSM}{E} \times \frac{Y}{2} \right\}} \right\} - AI$$

Where

$$AI = - \left[\frac{CPN}{2} \times \frac{DSC}{E} \right]$$

Where

AI =Accrued interest per S$100 face value
DSM =Days between settlement date and maturity date
DCS =Days between beginning of current coupon period and settlement date
DSC =E-DCS = days from settlement date to next 6-month coupon date
E =Number of days in coupon period where settlement occurs
N =number of semi-annual coupons payable between settlement date and maturity date
CPN =Annual coupon rate (as a percentage)
Y =Annual yield (as a percentage)
P =Dollar price per S$100 face value
RV =Redemption value

7.5 SGS Auction Safeguard Mechanism

Ceiling amounts to be determined by MMD before each auction

y

Non-competitive (a)	To be released in a strong auction (b)	Additional application in a weak auction Competitive at mid-pt yield + 25bps (c)

- (a) and (b) are non-competitive
- (a) and (b) may be zero
- (c) will represent the maximum amount that MAS can be alloted

Actual Allotment

y

Allotted in all auctions as non-competitive (a')	Allotted in normal and weak auctions as non-competitive (b')	Allotted in a weak auction only as a competitive bid (c')

- Actual allotment depends on how much other market non-comp applications add up to (x')
- Under Government Securities Regulations, (a') + (b') + (x') ≤ 40%
- See next page for different scenarios
- Y is set as an auction parameter, and is equivalent to c (maximum amount that MAS can be alloted)

Corridor Scenarios

SGS Allotment for Bonds			
Scenarios Allotment	**Strong Auction At 2.75 and lower**	**Normal Auction 2.76 - 3.24**	**Weak Auction At 3.25 and higher**
x' + a + b ≤ 40% where x' + a < 40%	a' = a (non-comp) b' = lowest prority, allotted remaining after releasing to: (i) comp bids at 2.75% and lower c' = 0 (re-run)	a' = a (non-comp) b' = b (non-comp) c' = 0	**Up to y** a' = a (non-comp) b' = b (non-comp) c' = Up to y - a - b (where c has lower priority vs other bids at 3.25)
x' + a + b > 40% where x' + a < 40%	a' = a (non-comp) b' = lowest priority, alloted remaining after releasing to: (i) non-comp x' + a, and (ii) comp bids at 2.75% and lower. c' = 0 (re-run)	a' = pro-rated b' = pro-rated c' = 0	**Up to y** a' = pro-rated b' = pro-rated c' = Up to y - a' - b' (where c has lower priority vs other bids at 3.25)
x' + a = 40%	a' = a (non-comp) b' = 0, since given lowest priority c' = 0 (re-run)		
x' + a > 40%	a' = pro-rated b' = 0, since given lowest priority c' = 0 (re-run)		

*Wrt Bid to cover, % of Competitive Applications at Cut-off Allotted, % of Non-Competitive Applications Allotted
For MAS **b** : Include total applied amount, regardless of final amt allotted
For MAS **c** : Include only final allotted amount, ie exclude any portion of unsuccessful bid.

7.6 Allocation Limit for bids

Non-Comp Limit (upper limit)
- 1% of amount offered (for PD)
- 1m (for non-PD, bills)
- 2m (for non-PD, bonds)
- Total cannot be >40% of amount offered

=> cumulative limit is based on per applicant per issue (applicant can apply more than 1 time)

Comp Limit
- (taken from AQA0001 table)
- 30% of amount offered (for PD)
- 15% of amount offered (for non-PD)

=> above limits inclusive of non-comp amount, but limit is set on comp amount.

Explanation when above limit (NC + C) is exceeded:
Case 1:
- E.g. Assume 100m bond issue.
- PD apply for NC 0.5m (NC limit not exceeded). PD apply for Comp 30m (Comp limit exceeded, 30.5m > 30%).
- The program will deduct the difference from Comp bid (deduct from highest bid).
- PD application will be cut to NC 0.5m, Comp 29.5m.

Case 2:
- E.g. Assume 100m bond issue.
- PD apply for NC 2m (NC limit exceeded, 2m > 1%). Application for NC is cut to 1m first. PD apply for Comp 30m (Comp limit exceeded, 30+1m > 30%).
- The program will deduct the difference from Comp bid (deduct from highest bid).
- PD application will be cut to NC 1m, Comp 29m.

Note: No limit set for MAS bills

Mas Max amount (normally round 40%)
- for bonds only
- for both comp and non-comp bids

Soft Limit (for Bonds, T-Bills and MAS Bills)
- Each PD should bid for at least (Issue Size/ No of PDs)
- Soft Limit to put as warning in report (`MSABRT05`), the auction will still carry on
- With effect from Nov 2014, the obligation limit (which is based on Dealer Code) will include bids put in PD on behalf of their customers as well.

Allot Indicator
- R (rejected)
- A (allotted)
- L (limit)
- N (not alloted)
- " " (for ?)
- "Z" is interium status for "A", will change back to "A" after end of run

(refer to: http://www.sgs.gov.sg/The-SGS-Market/Issuance-and-Auction-Conduct.aspx)

7.7 Sample Letter to AGD

The letter to AGD will contain information on new issuance, interests payment or redemption of SGS Bonds and/or T-Bills. The System uses the issuance/redemption/interests payment date entered by the users, to extract the information.

Please see the attached file below for a sample.

Letter to AGD
Sample_1.txt

7.7.1 Structure of Letter

Users are recommended to copy the contents of the text file into MS Word, change the font to Courier (size 10) and remove all spaces after paragraphs for printing. The text (in blue) change according to the transaction type and issue of records which are stored in the database.

```
17 May 2013  [Date when the letter is generated]
Accountant-General
Attn: Tan Boon Siong


     We append below details of redemption of Treasury Bills on 02
May 2013 [Date entered in AuctionViewReportMenu.jsp]:

Redemption:            182-DAY T/B 2012 BS13202C due 02/05/2013
ISIN Code:             SGSTBMDT0103
Nominal Amount:        $   1,000,000,000.00
Discount:              $   1,000,000,000.00
Net Proceeds:          $               0.00


     Please note that AG a/c X with us will be debited on 02 May
2013  [Date entered in AuctionViewReportMenu.jsp].



[Left blank for FD to enter the name before printing]
FINANCE DEPARTMENT
MONETARY AUTHORITY OF SINGAPORE
```

7.8 Authorised actions per user level

Module	Function	Action / Java Class	Description	Level 0	Level 2	Level 3	Level 5	Level 9	Level 8	Level 10
auction	Decrypt Bids	SGSAuctionDecryptBids	Decrypt bids			Y				
auction	Input Auction Para	SGSAuctionSavePara	Save parameters for SGS Auction		Y	Y[2]				
auction	Input Auction Para	SGSAuctionSecParaInput	Input Auction Parameters		Y	Y				
auction	Release Results	SGSAuctionReleaseResult	Release SGS Auction Results			Y				
auction	Run Auction	SGSAuctionRunAuction	Run SGS Auction			Y				
auction	Run Auction	SGSAuctionRunAuctionOffline	Run SGS Auction offline			Y				
auction	SGSAuction	SGSAuctionMenu	Access SGS Auction Menu		Y	Y	Y			
auction	Validate Bids	SGSAuctionValidateBids	Validate bids			Y				
auction	Verify Auction Para	SGSAuctionSecParaVerify	Verify Auction Parameters			Y				
auction	Verify Auction Para	SGSAuctionVerifyPara	Verify parameters for SGS Auction			Y				
auction	View Reports	SGSAuctionOutputReport	Export details of selected auction-related report as text			Y	Y			
auction	View Reports	SGSAuctionViewReports	Select and view auction-related reports			Y	Y			
auction	ViewReports	SGSAuctionGenerateLetter	Generate letter to AGD			Y				

[2] Amended Authorisation Level – Apr 2013

Module	Function	Action / Java Class	Description	Level 0	Level 2	Level 3	Level 5	Level 9	Level 8	Level 10
auction	View Securities for Auction	SGSAuctionSecList	View list of securities available for auction for the day			Y				
browse	Historical Securities	ViewHistSec	View list of historical securities		Y	Y	Y	Y		
browse	Ongoing Securities	SelectSecIssue	Select security issue		Y	Y	Y	Y		
browse	Ongoing Securities	ViewOngoingSec	View list of ongoing securities		Y	Y	Y	Y		
browse	Ongoing Securities	ViewSecDetails	View details of ongoing/historical securities		Y	Y	Y	Y		
browse	Security Dealers	SelectSecDealer	Select security dealer		Y	Y	Y	Y		
browse	Security Dealers	ViewSecDealers	View list of security dealers and their details by their accounts		Y	Y	Y	Y		
tender	Browse Tenders	BrowseAllTenders	View selected tenders by date		Y	Y				
tender	Browse Tenders	BrowseTenderList	Select tender for viewing		Y	Y				
tender	Browse Tenders	SelectTender	Select tender		Y	Y				
tender	Verify Tender	MatchTender	Match tender details			Y				
tender	New Tender	AddNewTender	Add new tender		Y	**Y**				
tender	New Tender	SubTender	Validate the tender transaction and update tender details records		Y	**Y**				

Module	Function	Action / Java Class	Description	Level 0	Level 2	Level 3	Level 5	Level 9	Level 8	Level 10
tender	New Tender	AddNewTenderMenu	Access add tender form		Y	Y				
tender	New Tender	AddTenderMethodSelect	Access form to select method of submitting a new tender		Y	Y				
tender	New Tender	UploadTenderConfirm	Validate uploaded file		Y	Y				
tender	New Tender	UploadTenderResult	Upload bids into the database		Y	Y				
tender	Reactivate	ReActRelTender	Reactivate released tender			Y				
tender	Release Tenders	TenderReleased	Release Tender			Y				
tender	Release Tenders	TenderUnReleased	Unrelease Tender			Y				
tender	SGSTender	SGSTenderMenu	Access SGS Tender menu		Y	Y				
tender	Update Tender	UpdateTender	Update/Delete tender		Y	Y				
tender	Verify Tender	VerifyTender	Verify tender			Y				
~~tender~~	~~View AllotResult~~	~~SelectAllotResult~~	~~Select allotment result for viewing~~	-	~~Y~~	~~Y~~	-	-		
~~tender~~	~~View AllotResult~~	~~ViewAllotResult~~	~~View selected/all allotment results~~	-	~~Y~~	~~Y~~	-	-		
tender	View UnRel	ViewUnRelTenders	View list of unreleased tenders		Y	Y				
user	Add User	AddNewUser	Access add/update user form					Y		

Module	Function	Action / Java Class	Description	Level 0	Level 2	Level 3	Level 5	Level 9	Level 8	Level 10
user	Add User	SubUser	Add/Update user record into database					Y		
user	Browse Users	BrowseUsers	View list of users					Y		
user	Browse Users	SelectUser	Select user to update					Y		
user	Update Users	UpdateUser	Access add/update user form					Y		
user	User Maintenance	UserMaintenanceMenu	Access user maintenance menu					Y	Y	Y
user	SGS Auction User Maintenance	UserMaintenance	Access SGS Auction user maintenance menu					Y		
user	SGS Auction User Maintenance	UserMaintenance	View Audit Log Report						Y	
eappsuser	eApps User Maintenance	EappsUserMaintenance	Access eApps user maintenance menu							Y
eappsuser	Add eApps User	AddNewEappsUser	Access add/update eApps user form							Y
eappsuser	Browse eApps Users	BrowseEappsUsers	View list of eApps users							Y
eappsuser	Update eApps Users	UpdateEappsUser	Access update/delete eApps user form							Y

7.9 FRN - Tender Submission and Auction

Please refer the document "FRN_SORA_SDS_v1.4.docx" for below details

- Tender Submission
- Run Auction
- Reports generated during Auction

7.10 eApps User Maintenance

7.10.1 Housekeeping

- If the System generated Token ID reaches 'ZxxxxT01' for a particular bank code as described in Section 3.2.15 SubEappsAddUser.java, need to check on the below and decide on the condition for housekeeping the data in DB table (ABA0030_CORP_PASS_MAPPING@MNET)
    - To get the list of Inactive eApps users for more than specified years (ABA0030_ACTIVE_IND='N' and ABA0030_LAST_LOGIN_DT > specified number of years)
    - To get the list of Active eApps Users for below scenarios
        - Users who are not logged into eApps application (ABA0030_ACTIVE_IND='Y' and ABA0030_LAST_LOGIN_DT=null)
        - Users who are not logged into eApps application for more than specified years (ABA0030_ACTIVE_IND='Y' and ABA0030_LAST_LOGIN_DT > specified number of years)

7.10.2 Steps to follow when Primary Dealer Bank code is changed

- When there is a change in the PD bank code,
    - Get the list of CorpPass user details to access eApps application from PD for CorpPass user registration under new bank code
    - An adhoc SQL script is written to update the new bank code in ABA0013_PRIMARY_DEALER table
- Before executing the adhoc SQL script, do the below steps in eAuction -> eApps User Maintenance
    - All the existing active eApps Users under the old Bank Code should be marked as Inactive/Deleted using Browse Users screen under eApps User Maintenance Menu
        - Select Active in the Drop down
        - Select the Bank Code from the Drop down (Old Bank Code)

        - Select all the Active users one by one and do the below
            - Set the users as 'Inactive' if the users have logged into eApps application earlier
            - Delete the users if the users have not logged into eApps application
    - 'No User Found' message must be displayed in Browse Users page when we select Active and the Old Bank code in the Drop down
- Now, execute the adhoc SQL script to update the new bank code in ABA0013_PRIMARY_DEALER table
- Register all the CorpPass User details received from the PD under the new bank code using Add User screen in eApps User Maintenance Menu
- All the newly registered CorpPass User under new bank code can be seen under Browse Users screen under eApps User Maintenance Menu by selecting Active and the New Bank code in the Drop down

7.11 CMTB and SGS (Infra) - Tender Submission and AGD Letter/Interface File

Please refer the document "CMTB_SGS(Infra)_SDS_v1.1.docx" for below details

- Tender Submission
- AGD Letter/Interface file

7.12 SGS Syndication

Please refer the document "Syndication_SDS_v1.0.docx" for below details

- SGS Syndication
    - Upload Syndication Files
    - Run Syndication
    - View Syndication Reports
    - Release Syndication Results
    - Cut-Off Price (Dirty) Computation
- AGD Letter/Interface file