Oracle FLEXCUBE Direct Banking

Alerts
Release 12.0.3.0.0

Part No. E52543-01

April 2014

ORACLE®

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries:
Phone: +91 22 6718 3000
Fax: +91 22 6718 3001
www.oracle.com/financialservices/

Contents

Preface

Intended Audience

Oracle FLEXCUBE Direct Banking Development Alerts document in particular is targeted towards the following groups of users.

- Oracle FLEXCUBE Direct Banking Development Teams
- Oracle FLEXCUBE Direct Banking Implementation Teams
- Oracle FLEXCUBE Direct Banking Implementation Partners

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Access to OFSS Support

https://support.us.oracle.com

Structure

This alerts document is organized into the following categories:

- *Preface* gives information on the intended audience. It also describes the overall structure of the Alerts
- *Overview* provides brief description of the alerts and its classification.
- Alert Channels describes the channels on which alerts can be received by the user.
- Types of Alert and Alert Modes classifies alerts based on modes and channels
- Alert Architectural Flow describes the architecture of the alerts
- The subsequent chapters after Alert Architectural Flow describes the setup and a dummy example

Related Information Sources

For more information on Oracle FLEXCUBE Direct Banking Release 12.0.3.0.0, refer to the following documents:

- Oracle FLEXCUBE Direct Banking Licensing Guide.
- Oracle FLEXCUBE Direct Banking User Manuals.

Overview

Oracle FLEXCUBE Direct Banking is a multi channel e-banking platform with support for customer touch points like Internet, Mobile Phones, tablets and PDAs.

Alerts can be classified into the following types.

Sr. No	Alert Type	Alert Description
1.	Default Alerts	Alerts generated irrespective of any user/customer/account type, and are independent of alert subscriptions
2.	User Based Alerts	Alerts generated specific to a user activity
3.	Customer Based Alerts	Alerts are generated based on primary customer id linked to the user and alert subscription. For e.g. Beneficiary Alert
4.	Account Based Alerts	Alerts generated on a given account and alert subscription. For e.g. Balance Alert
5.	Authorization Alerts	Alerts generated based on the authorization which the user is performing For e.g. Mandate Alerts for all all eligible authorizers
6.	Transaction Alerts	Alerts generated for transactions that do not require authorization but still need alerts can be triggered via these alerts

Alert Channels

Oracle FLEXCUBE Direct Banking supports following three channels (modes) for sending the Alerts

- E-mail – These alerts are available to user in his email account.
- SMS – These alerts are available to user on his mobile.
- PUSH – These alerts are available to user on his mobile using Push notification functionality.
- On Screen – These alerts are available to business user in notification section on the dashboard and also in Mailbox.

Following table shows list of alerts and modes through which each alert is made available to the user.

Sr. No.	Alert Name	Email	SMS	PUSH	On Screen
1	Login Failed	✓	✓	X	X
2	Login Failed – Account Locked	✓	✓	X	X
3	Limit Utilisation	✓	✓	X	X
4	Limit Utilization Warning On Predefined Threshold	✓	✓	X	✓
5	Peer beneficiary registration alert	✓	✓	X	✓
6	Forex Alert	✓	✓	✓	✓
7	New Beneficiary Created Alert	✓	✓	X	✓
8	Term Deposit Opened Alert	✓	✓	✓	✓
9	Bill Pay Alert	✓	✓	✓	✓
10	Term Deposit Status Alert	✓	✓	✓	✓
11	Account Balance Alert	✓	✓	X	X
12	Account Status Alert	✓	✓	X	X
13	Clearing Cheque Returned Alert	✓	✓	✓	✓
14	Cheque Stop Alert	✓	✓	✓	✓
15	Funds Transfer Alert	✓	✓	✓	✓
16	Credit Alert	✓	✓	✓	✓
17	Debit Alert	✓	✓	✓	✓
18	Alert for the Successful Processing of Fund Transfers (Pay Later & SI)	✓	✓	✓	✓
19	User Created Alert	✓	✓		X
20	User Activated	✓	✓		X
21	Reset Password	✓	✓	X	✓*
22	Transaction Authorisation (Status Change Alerts) Alert	✓	✓	✓	X
23	Request Process Alert	✓	✓	✓	X
24	Alert To Beneficiary	✓	✓	✓	X
25	Channel Subscription/Unsubscription Alert (Self Subscription by User)	✓	✓	✓	X
26	All Default Host Alerts	✓	✓	X	✓

- * Only Transaction Password reset by Bank Administrator shall be eligible for On Screen Alerts. Login Password Reset Alerts shall not be available to be set as On Screen Alerts. Alert for Transaction Password Reset by Business User himself shall not be available to be set as On Screen Alert.
- The symbol ✓ indicates that the alert is configured at day 0 to be made available through the particular channel mode.
- The symbol X indicates that the day 0 configuration is to not provide the alert through the particular channel mode.

Types of Alerts

Oracle FLEXCUBE Direct Banking supports various alerts. The alerts can be summarized as follows

Alert	Subscribe Needed	Host Alerts	Internal Alerts	Alert Type
Default Alerts				
User Created Alert	N	N	Y	D
User Activated	N	N	Y	D
Reset Password		N	Y	D
Transaction Authorisation (Status Change Alerts) Alert	N	N	Y	D
Request Process Alert	N	N	Y	D
Alert To Beneficiary	N	N	Y	D
Channel Subscription/Unsubscription Alert (Self Subscription by User)	N	N	Y	D
All Default Host Alerts	N	N	Y	D
User Alerts				
Login Failed	N	N	Y	
Login Failed – Account Locked	N	N	Y	U
Limit Utilisation	N	N	Y	U
Limit Utilization Warning On Predefined Threshold	N	N	Y	U
Peer beneficiary registration alert	Y	N	Y	U
Forex Alert	Y	N	Y	U
Login Failed	**Y**	N	Y	U
Login Failed – Account Locked	**Y**	N	Y	U
Limit Utilisation	**Y**	N	Y	U
Limit Utilization Warning On Predefined Threshold	**Y**	N	Y	U
Customer Alerts				
New Beneficiary Created Alert	**Y**	N	Y	C
Term Deposit Opened Alert	**Y**	N	N	C
Bill Pay Alert	**Y**	N	N	C
Term Deposit Status Alert	**Y**	N	N	C
Account Alerts				
Account Balance Alert		Y	N	A
Account Status Alert		Y	N	A
Clearing Cheque Returned Alert		Y	N	A
Cheque Stop Alert		Y	N	A
Funds Transfer Alert		Y	N	A
Credit Alert		Y	N	A
Debit Alert		Y	N	A

Alert for the Successful Processing of Fund Transfers (Pay Later & SI)		Y	N	A
Authorization Alerts				
Initiation Alerts	N	N	Y	M
Authorization Alerts	N		Y	M

On basis of accessibility, alerts can be classified into following types.

- Default alert (D)
- Accounts based alerts (A)
- Customer based alerts (C)
- User based alerts (U)
- Mandate alerts (M)

All these types of alerts will be available to user through different modes (like Email, SMS) as per the database configuration.

The alerts which are configured for 'On Screen' mode can be further classified as given below

Default Host Alerts (H)	These types of alerts are generated by host for 'Inreraction module' and are available through host database link.
Default FCDB Alerts (I)	These types of alerts are default and internally generated by FCDB application.
Subscribed Host Alerts (S)	These types of alerts are generated by host and fetched through alert notification setup.
Subscribed FCDB Alerts (S)	These types of alerts are subscribed and internally generated by FCDB application.

All 'On screen' alerts are available in table, 'fcat_vw_communiactions'.

Alert Modes

The Alert System is designed to send relevant messages to its customers when certain alert specific event related to the customer happens

Alerts can be sent by the following three modes:

- E-mail
- SMS
- PUSH
- On Screen

A business user can register themselves for multiple Alerts through 'Alert' screen. Business user can select the channel, frequency, date range while creating alert. The Bank Administrator can also register the business users for the alert registration.

Subscribed Alerts can be provided at three levels:

- User Level
  A user level alert will be sent to the email address or mobile number as maintained at the user profile. This email address or mobile number can be overridden during alert registration, in which case, that particular alert will be sent to the email address or mobile number mentioned during alert registration.

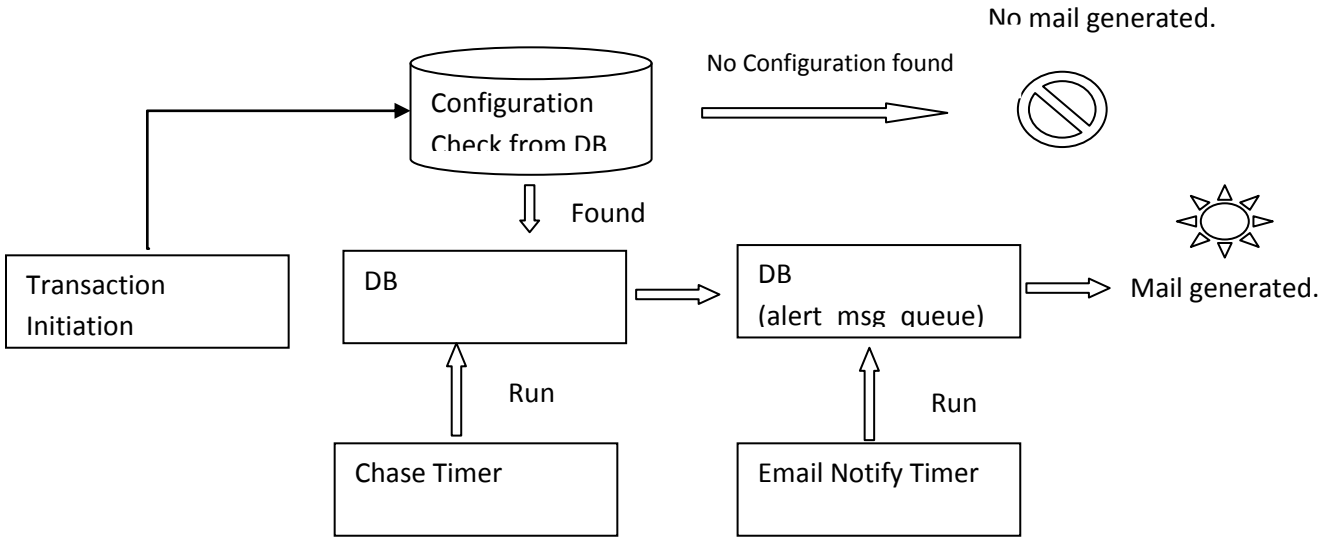
- Accounts Level
  An Accounts level alert will be sent to the email address as maintained at the user profile. This email address or mobile number can be overridden during alert registration, in which case, that particular alert will be sent to the email address or mobile number mentioned during alert registration.

- Customer Level
  A customer alert will be sent to the email address as mentioned in customer profile (of user's primary customer), e.g. on creation of a new beneficiary. If profile is not present, it will fetch email address from host (From fcc_vw_mstcorporate) .

Alerts Architectural Flow

The following steps need to taken to completed alert cycle as given below:

- ✓ While doing initiate/ transaction, we check alert configuration in DB if yes then store information related to specific alerts in DB otherwise no alert will be generated.
- ✓ Run the chase time to pick this information process through handler (implemented logic) and generate the alert with all required information into DB itself.
- ✓ Run the Email Notify timer to send generated alert from db to given email address.



Day Zero Alerts Setup

The Day zero configurations which need to done in the database on certain tables for which alerts to be generated are mentioned below in tabular form for ease of understanding. For alerts which require subscription, then user would be required to subscribe for alerts.

1. DB Configuration

Table Name	Column name	Data Type	Default Value	Comments
Mstalert	Is_subscribed	Char(1)	Z	This column is used to identify whether alert is subscribed or default. Y – Available for subscription N – Default alert Z – None; not available for alert generation
Mstalert	On_screen	Char(1)	N	This column is used to identify whether alert is onscreen or not. Y – Available for on screen alert generation N – Not available.
Mstchannelats	authrequired	Char(1)	N	This column is used to identify if transaction requires authorization or not. Y – This means for this idrequest and idchannel combination an alert will have to be generated
Mstusertypetxn	authalertrequired	Char(1)	N	This column has to be interpreted as whether alert is required for this user type & transaction (entity, channel, usertype, txn) Y – Alert is required for this entity, channel, transaction & user type combination
	alertparam	Varchar2 (4000)		This column is referred for transactions requiring authorization and would interact with host system. *STATUS~IDCHASE#* *STATUS1~IDCHASE1#* *STATUS2~IDCHASE2* Sample entry would like below, where1, 2, 3 are possible statuses from host system. *1~AUTHINICHASE#2~AUTHREJCHASE#3~AUTHREJCHASE*
Mstinitauthstatus	Alertrequired	Char(1)	N	Y – Alert has been configured based on current, next status, flgaction & authorization engine.
Mstinitauthtypes	Alertrequired	Char(1)	N	This is used to get information about authorization engine

				configured for that specific transaction.
Userchasestate	statedata	BLOB		This is used to store alert information after generated alert like channel, mail content information etc.
Msttimer	handoffclass	Varchar2 (255)		This is help to configured different type of timer .
Chaseconfig	chasehandlerclass	Varchar2 (255)		This is useful to get handler information used by timer.
msthostinterface	interface_impl	Varchar2 (255)		This is used to identify whether alert will generate through host or fcdb based upon diffenret adaptor currently localalertgenerationadapter
alert_msg_queue	Message	Varchar2 (4000)		This is used to store formatted mail generated from FCDB.

2. Subscription

Business user will be given an option to subscribe for the alert facility on user level, customer level & account level. While doing the transactions we check alert required configured in DB. A business user can register himself for multiple Alerts through Alert screen. Business user can select the channel while creating alert.

FCUBS Notification/Alert Setup

Currently below mentioned Account Based Alerts/Notifications are supported with FCUBS.

1. Account Balance
2. Account Status Change
3. Cheque Clearing Reject
4. Cheque Stop
5. TD Status Change
6. Funds Transfer
7. TD Open
8. Bill Pay

1. Steps to add New FCUBS Notification/Alert

1) Add entry in msthandoffs table with Processor Name as UBSAlertNotificationProcessor.java. This processor processes all the handoffs and generates alert by inserting into alert_msg_queue table. For any specific processing in NID mode new processor can be written and then configured here.
2) Modify Notifications.xsl to add the new handofftype added in step 1. This entry is based on FCUBS host version. If it is 10.X or 11.X then all handoffs come with root tag FCUBS_NOTIFICATION. So it can be added under <XSL:WHEN>. If host is FCUBS 7.X, then every handoff has it's own root tag. So that new handoff can be added under otherwise tag as per your handoff root tag. For any third party, tags can be added under xsl:otherwise.
3) There will be one timer FCCNotificationTimer running continuously which parses this notifications.xsl file with notification xml message received from FCUBS. After this it will get notification code (handoff type) and then it will call corresponding processor configured in msthandoffs table.
4) Queries required to be executed for handoff needs to be added in mstquery whose mapping will be done in appldata as described in step 6.
5) In Notifications.xsl, you need to specify the fields which you need to fetch from handoff. E.g. XREF, Hostref etc. Name of the fields should be same as in **NotificationDTO.java**.
   If you want to have some extra fields (which are not present in **NotificatinDTO.java**.) from database for example, you need to add the entries for those fields in Notifications.xsl as UDF under your Notification type as shown below:-

```
<udfFields>
<UDFDTO>
<udfName>benename </udfName>
<udfValue>josef</udfValue>
</UDFDTO>
</udfFields>
```

   Above needs to be added under **<NotificationDetailsDTO>**.

   Resulting notification will look like

```
<xsl:if test="//FCCACCSERVICE/HANDOFF_ACCBAL">
<NotificationDTO>
<notifCode>NOTIF_ACCBAL_ALERT_7X</notifCode>
<notificationdetails>
<NotificationDetailsDTO>
<codBranch><xsl:value-of select="//FCCACCSERVICE/HANDOFF_ACCBAL/BRN"/></codBranch>
```

<custAccountNo><xsl:value-of select="//FCCACCSERVICE/HANDOFF_ACCBAL/ACC"/></custAccountNo>

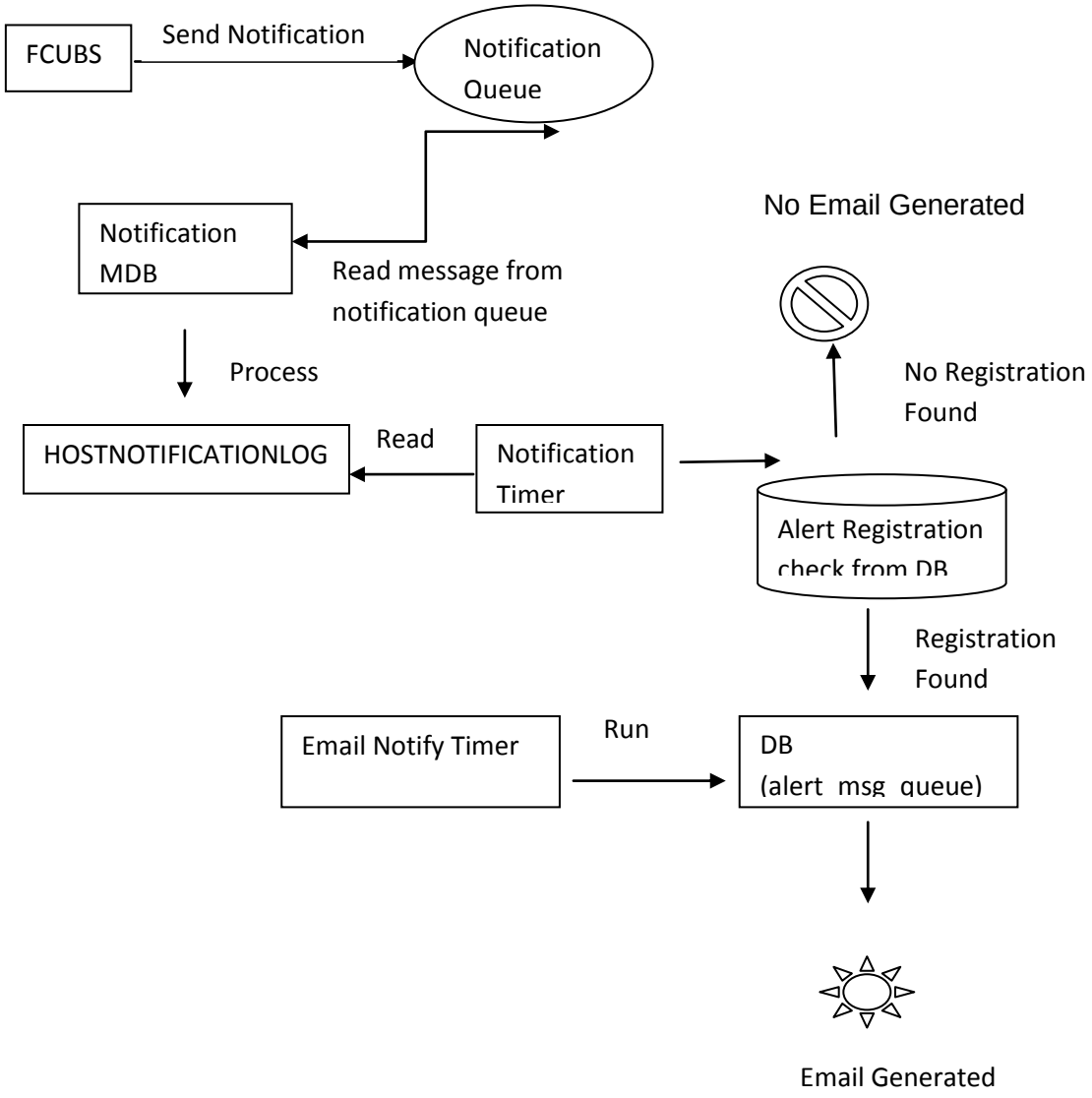
<udfFields>
<UDFDTO>
<udfName>fldauthtype</udfName>
<udfValue>hello</udfValue>
</UDFDTO>
</udfFields>
</NotificationDetailsDTO>
</notificationdetails>
</NotificationDTO>
</xsl:if>

6) The fields with udf value blank will be taken from query as specified below:-
    a. If your notification requires a query to be executed, then that query(**Query ID**) can be put in appldata under dataname **NOTIFICATION_QUERIES.** This query will be executed every time handoff comes. Arguments required by the query can be set in **NOTIFICATION_QUERIES_ARGS.** Values of these arguments will be taken from value set in **Notifications.xsl** or from the last query results. Result of the query is stored in the dto columns or in the udf fields if column is not defined for the same. But column name in the query should match either Field in DTO or udf specified in **Notifications.xsl.**
    b. Multiple queries can be executed for one Handoff. Query ID's can be placed hash separated in appldata under **NOTIFICATION_QUERIES.** Each query will be executed and result will be set in DTO fields.

7) For this new handoff new alert needs to be added in mstalert table.
8) Add new entry in MSTALERTCHANNELS for the channels supported like EMAIL, SMS, PUSH etc..
9) Add entry in MSTENTITYUSERALERTS for different user types like ECU, EN1, etc…
10) Do mapping between handoff type and idalert in appldata with dataname **UBS_NOTIFICATION_CODE.**
11) New XSL file needs to be written for new notification/alert and put it into alerts folder under datafiles.

2. Architecture/Flow of FCUBS Notification/Alert

1. There will be one Notification MDB deployed which will continuously listen on the queue where FCUBS sends notifications. Java file for this MDB is NotificationQueueAnonymousWaiter.java
2. After reading notification messages from notification queue MDB will insert each message in hostnotificationlog table. In this table notification xml message will be inserted in HOSTDATA column. Notification Code will be inserted in NOTIFCODE column. XREF will be inserted in IDENTITY column.
3. There will be one Notification Timer running which will continuously read poll this hostnotificationlog table.
4. Timer will parse Notifications.xsl on each notification xml message stored in HOSTDATA column. Depending on Notification Code, Timer will get instance of Processor configured in msthandoffs table and then call processRequest of that processor. Processor used currently is UBSAlertNotificationProcessor.java
5. This processor does processing of notification. It checks for users who have registered for the particular alert/notification and then generates alert for those users.


FCUBS
Send Notification
Notification Queue
Notification MDB
Read message from notification queue
Process
HOSTNOTIFICATIONLOG
Read
Notification Timer
No Email Generated
No Registration Found
Alert Registration check from DB
Registration Found
Email Notify Timer
Run
DB (alert_msg_queue)
Email Generated

Dummy Example

Case I: Transaction which does not require Authorization – Login Alert

1. User has to subscribe for login alert, which can be done either by bank admin or by business user. Once the user has subscribed this information is saved in mstuseralerts table.

2. When the user logs in into the system, application will check whether user has subscribed for login alert in mstuseralerts table**.** If the user has subscribed then required information is retrieved from this table and the information is logged into userchasestate table for further processing.

3. There is a timer service which picks all records from userchasestate table. The mapped handler class for this idchase is picked from chaseconfig table which appends any further processing data and makes a call to the adapter class. The adapter class is responsible to identify whether alert information is to be sent to host system or by FCDB. If alert has to be generated by FCDB then data is logged into alert_msg_queue.

4. Another timer service picks records from alert_msg_queue table and makes a call to corresponding notifier class as configured in msttimer. In case of email alerts call to SMTP server is made and information is sent. SMTP server in turn is responsible for sending the alert to the valid mail address of the user.

Case II: Transactions Requiring Authorization

1. When user logs in into the system and initiates a transaction, application checks if alert required is configured as Y in mstinitauthstatus table for given authorization engine, current status, next status & action. If alert required is configured as Y, then chase id is retrieved.
2. The application then checks if authalertrequired is configured as Y in mstusertypetxn for given entity, user type, channel & transaction. If the above criteria in 1 & 2 are met, then information specific to alert is generated by ListBasedAuthorizationEngine component, which populates data in userchasestate.

Mstinitauthtypes: Column value should be like below at initiation

- **Alertrequired = 'Y'**

Mstinitauthstatus:

- Idchase= ‘AUTHINITCHASE’
- Flgaction = ‘I’
- Flgcurrstatus=1
- Flgnextstatus=1
- Alertrequired = ‘Y’

Mstusertypetxn:

- AUTHALERTREQUIRED=Y

3. Follow step as given above from 3.

Case III: Transactions comes from host

1. This is the case when response comes from host on queue then application calls respectively adapter configured in msthostinterface table and invokes host service “HostResponseProcessingService“. This service help to get all information related to transaction came from host based upon idfcatrence no. from admintxnunauthdata table and check configuration alertrequired as Y in mstusertypetxn for given entity, user type, channel & transaction, If alert required is configured as Y, then chase id is retrieved combination of status and idchase from mstusertypetxn table then logged into userchasestate table for further processing.

   Mstusertypetxn: Column value should be like below at initiation

   - Alertparam**=**1~AUTHINITCHASE#2~AUTHREJCHASE#3~AUTHREJCHASE
   - AUTHALERTREQUIRED=Y

Follow step as given above from 3.