

# ITDB's User manual

This manual describes how to demonstrate the ITDB. In the demo, the functionality of the ITDB's components will be shown in order:

1. The Mediator
2. The Intrusion Detector
3. The Repair Manager
4. The Containment & Uncontainment Manager
5. The SSM
6. The Isolation Manager (part of the Mediator)

ITDB demo programs also includes

1. A credit card application that is used to generate data into ITDB.
2. Simulator programs that is used to generate transactions in order to simulate the real world environment.

The figure below illustrates all the ITDB execution files.

Name	
sql	SQL Scripts (Used by the Mediator)
AutoTrns - inct Usr 1.exe	Transaction Simulators
AutoTrns - inct Usr 2.exe	
AutoTrns - SSM.exe	
AutoTrns - Usr 11.exe	
AutoTrns - Usr 4.exe	
AutoTrns - Usr 5.exe	
CatReg.exe	
CorbaTest.exe	CORBA Testing Program
DCM.exe	Damage Uncontainment Manager
ID.exe	Intrusion Detector
MediateCorba.ref	
OciMediate.exe	Mediator
OCIMEDIATE.HLP	
OciMediate.opt	
POCI.exe	Credit Card Application
RepairMngr.exe	Repair Manager
simdetect.opt	
sqlnet.log	
SSM.exe	SSM
TranPatt.ctr	
Transaction.dat	Transaction Patterns (Used by the Mediator)
TrnsType_Log.txt	Transaction Log
Unconfine_Log.txt	Transaction Type (Used by the DCM.exe)
UnDefine.dat	Undefined Transaction Log

## Demo Procedures

Before starting the demo, please run the 'cleanup' script in Sqlplus.

```
SQL> conn mediate/mediate@orcinew <Enter>
SQL> set serveroutput on size 20000 <Enter>
SQL> exec cleanup; <Enter>
```

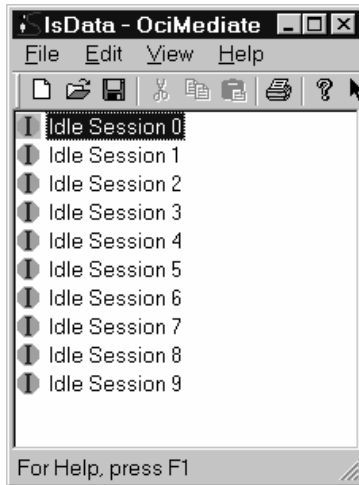
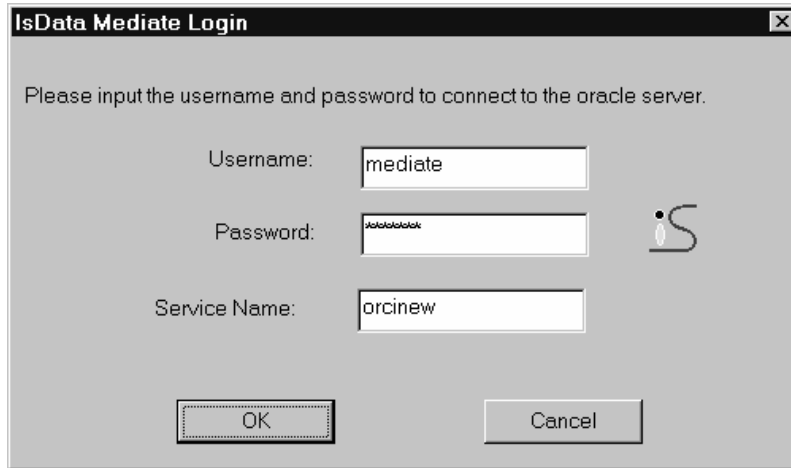
## 1. The Mediator

1.1 Describe the functionality of the Mediator.

1.2 Show the screen interface and the meaning of each items e.g. the icon, menu.

1.3 Steps

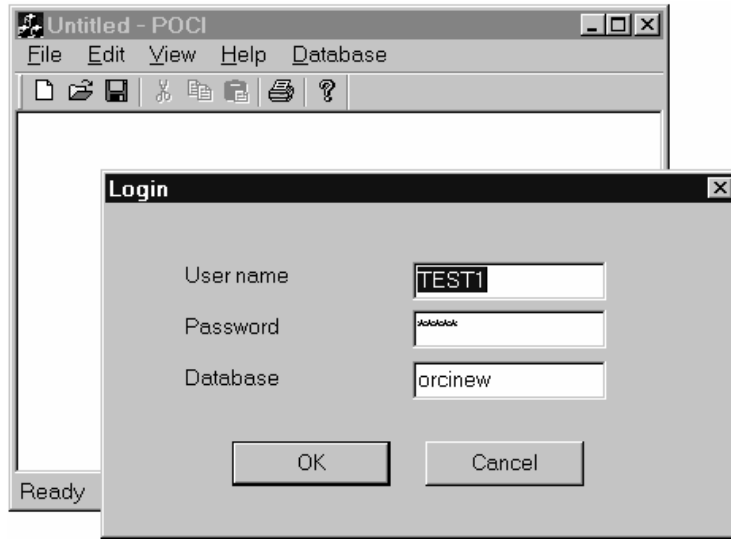
1. Start the Mediator (Mediate.exe) and login to the database as the mediate user.



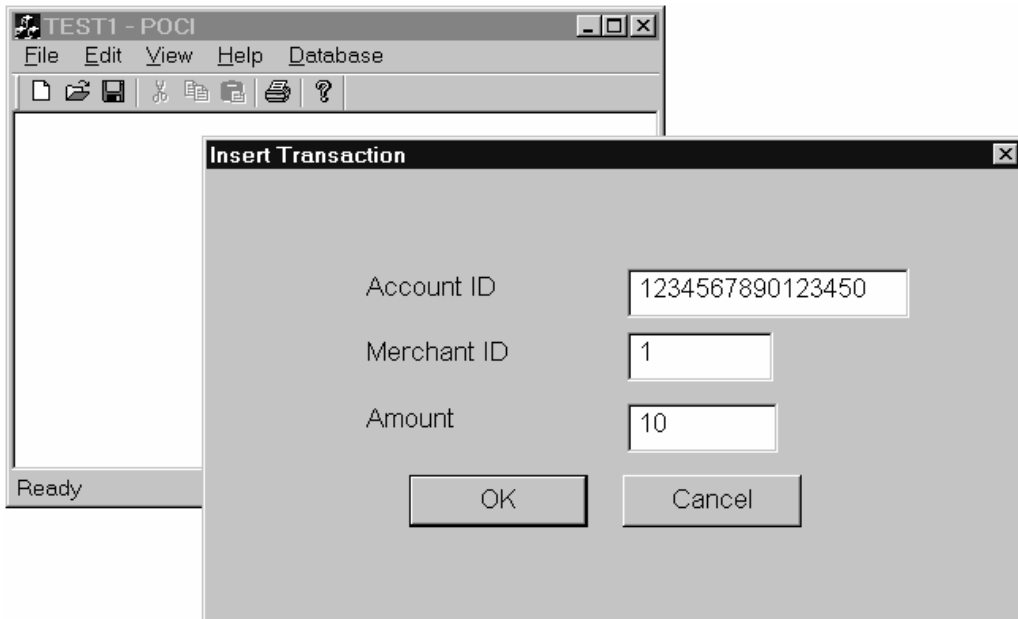
The Mediator's screen interface

2. Start the Credit Card Application (Poci.exe)

3. Login to the database as test1-99 user (the application connect and submit transactions using the service provided by the Mediator).



4. Submit a normal transaction to the database.  
User Account\_ID: From 1234567890-5 except 1234567891  
Merchant\_ID: 1 or 2  
Amount: 1-100 (amount over 1000 will be determined as malicious)

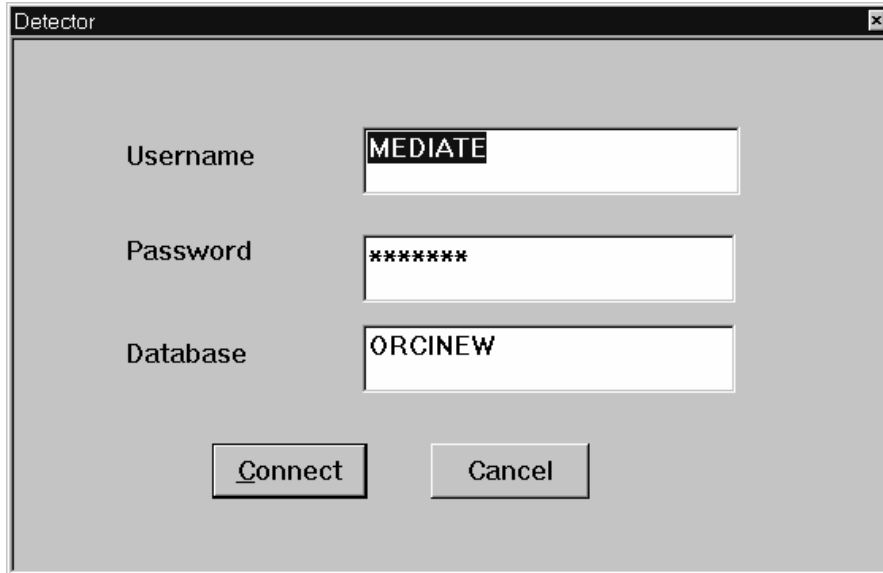


## 2. The Intrusion Detector

2.1 Describe the functionality & the interface of the Intrusion Detector.

### 2.2 Steps

1. Continue from the previous step.
2. Start the Intrusion Detector and login to the database.



Detector

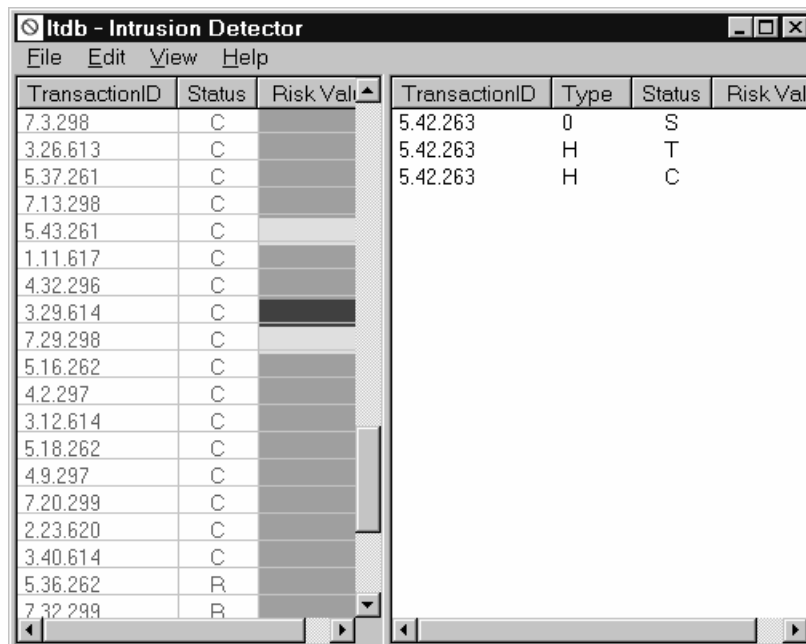
Username: MEDIATE

Password: \*\*\*\*\*

Database: ORCINEW

Connect Cancel

3. Submit transactions that have different risk to the database.
  - Innocent Transaction (Green color)
  - Suspicious Transaction (Yellow color)



ltdb - Intrusion Detector

File Edit View Help

TransactionID	Status	Risk Val
7.3.298	C	
3.26.613	C	
5.37.261	C	
7.13.298	C	
5.43.261	C	
1.11.617	C	
4.32.296	C	
3.29.614	C	
7.29.298	C	
5.16.262	C	
4.2.297	C	
3.12.614	C	
5.18.262	C	
4.9.297	C	
7.20.299	C	
2.23.620	C	
3.40.614	C	
5.36.262	R	
7.32.299	R	

TransactionID	Type	Status	Risk Val
5.42.263	I	S	
5.42.263	H	T	
5.42.263	H	C	

4. Explain the meaning of each item's color how it is related to the risk value.

### 3. The Repair Manager

3.1 Describe the functionality & the interface of the Intrusion Detector.

3.2 Steps

1. Continue from the previous step.

2. Start the Repair Manager (The Repair Manager always automatically login to the database).

3. Start three Transaction Simulators program (User - Test4, Test5 and Test11).

4. Use the Credit Card Application (Poci.exe) to submit a malicious transaction.

User Account\_ID: From 1234567890-5 except 1234567891

Merchant\_ID: 1 or 2

Amount: >= 1000

5. Show a table that contains the repair process details. The tables include the Mediate.write\_log and the Rpmgr.Repair\_det\_logs table.

XID	STEPID	TABLE_NAME	RECORD_ID	COL_NAME	OLD_VALUE	NEW_VALUE	OP_CODE
2.2.638	1135	TRANSACTIONS	8851	AMT		1000	I
2.2.638	1135	TRANSACTIONS	8851	ACCOUNT_ID		1234567890123450	I
2.2.638	1135	TRANSACTIONS	8851	MERCHANT_ID		1	I
2.2.638	1135	TRANSACTIONS	8851	TRANID		8851	I
2.2.638	1134	ACCOUNTS	1234567890123450	ACC_BALANCE	131834	132834	U
2.2.638	1135	TRANSACTIONS	8851	SALE_DATE		26/09/2002 01:09:33	I
2.2.638	1135	TRANSACTIONS	8851	POST_DATE		26/09/2002 01:09:33	I
2.2.638	1135	TRANSACTIONS	8851	USER_ID		TEST1	I
2.2.638							C

XID	STEPID	TABLE_NAME	RECORD_ID	COL_NAME	OLD_VALUE	NEW_VALUE	OP_CODE
5.46.280	1168	TRANSACTIONS	8851	MERCHANT_ID		1	D
5.46.280	1168	TRANSACTIONS	8851	ACCOUNT_ID		1234567890123450	D
5.46.280	1168	TRANSACTIONS	8851	AMT		1000	D
5.46.280	1168	TRANSACTIONS	8851	SALE_DATE		26/09/2002 01:09:33	D
5.46.280	1168	TRANSACTIONS	8851	POST_DATE		26/09/2002 01:09:33	D
5.46.280	1168	TRANSACTIONS	8851	USER_ID		TEST1	D
5.46.280	1168	TRANSACTIONS	8851	TRANID		8851	D
5.46.280	1167	ACCOUNTS	1234567890123450	ACC_BALANCE	132834	131834	U

## 4. The Containment & Uncontainment Manager

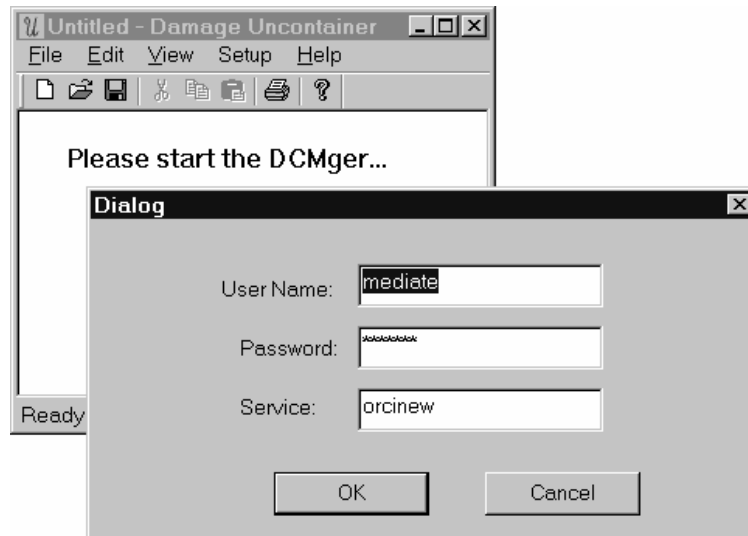
4.1 Describe the functionality & the interface of the Uncontainment Manager.

### 4.2 Steps

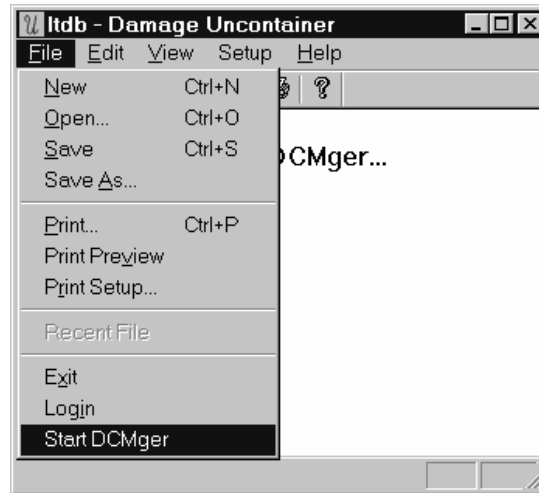
1. Continue from the previous steps.
2. Start the Uncontainment Manager (DCM.exe).



3. Login to the database.



4. Start the process.

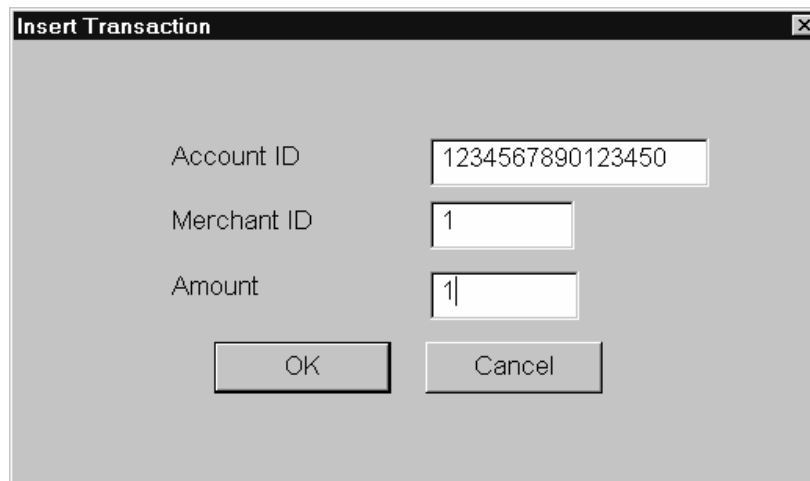


5. Submit a malicious transaction from the Credit Card Application (POCI.exe). For example, you might enter the following data:

User Account\_ID: From 1234567890-5 except 1234567891  
Merchant\_ID: 1 or 2  
Amount: 1000

6. Submit a transaction that will be blocked from the containment process. For example:

User Account\_ID: same as above  
Merchant\_ID: same as above  
Amount: 1





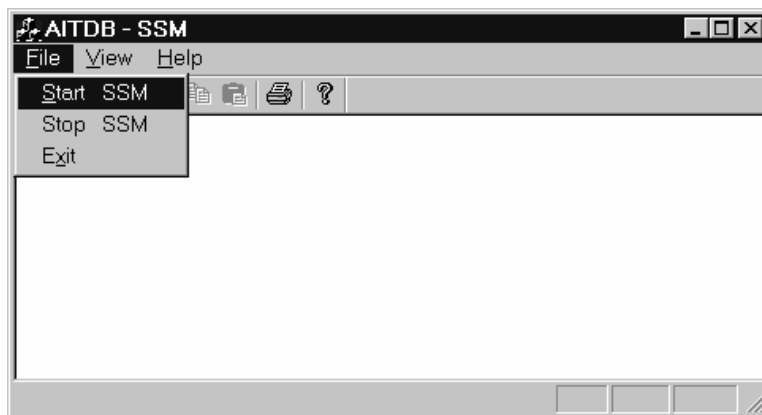
7. Wait until the repair process is finished.
8. Submit the transaction that contains the blocked item (in 4.4) again. Explain why the transaction is allowed at this time.
  - Note: the Containment Manager is implemented as a part of the Mediator.

## 5. The SSM

5.1 Describe the functionality, the parameters that are used in this version.

5.2 Steps

1. Continue from the previous step.
2. Run and start the SSM process.



3. Show the events table, adaptation-rules table, conditions table and actions table.
4. Use the Credit Card Application (Poci.exe) to submit a malicious transaction.



5. Show the SSM's interface. Illustrate that how parameters dynamically changed --the integrity level, the transaction submission rate etc.

Parameter Name	Value	Priority	Updated By	Update Time
INTEGRITY_LEVEL	0.95	L	MEDIATOR	SEP 26,15:01
TRNS_SUBMS_RATE	18	L	MEDIATOR	SEP 26,15:01
TRNS_DELAY	1	L	MEDIATOR	SEP 26,15:01
NUM_AFFECTED_TRNS	0	L	CLEANUP	SEP 26,14:58
NUM_MALICIOUS_TRNS	2	L	ID	SEP 26,15:01
NUM_AFFECTED_ITEMS	0	L	CLEANUP	SEP 26,14:58
NUM_MALICIOUS_ITEMS	16	L	ID	SEP 26,15:01
NUM_BLOCKED_ITEMS	3	L	MEDIATOR	SEP 26,15:01
CONTAINMENT_PHASE	ONE_PHASE	L	MEDIATOR	FEB 11,10:29
SUSPICIOUS_THRESHOLD	4	L	DEF	FEB 11,10:29
SUSPICION_RATE	65	L	MEDIATOR	FEB 12,12:03
SUSPICIOUS_TO_MALICIOUS_...	0.57	L	MEDIATOR	FEB 12,12:03
FALSE_ALARM_RATE	5	L	ABC	FEB 11,10:29
AVG_RESPONSE_TIME	12	L	ABC	FEB 11,10:29
TIME_INTERVAL_ANALYZER	1500	L	MEDIATOR	FEB 11,10:29
TIME_INTERVAL_LISTENER	1500	L	MEDIATOR	FEB 11,10:29
TIME_INTERVAL_SSM_POLLING	700	L	TEST	FEB 11,10:29
TIMEOUT_GET_MSG	2	L	TEST	FEB 11,10:29
TIMEOUT_SSM_GET_MSG	2	L	TEST	FEB 11,10:29

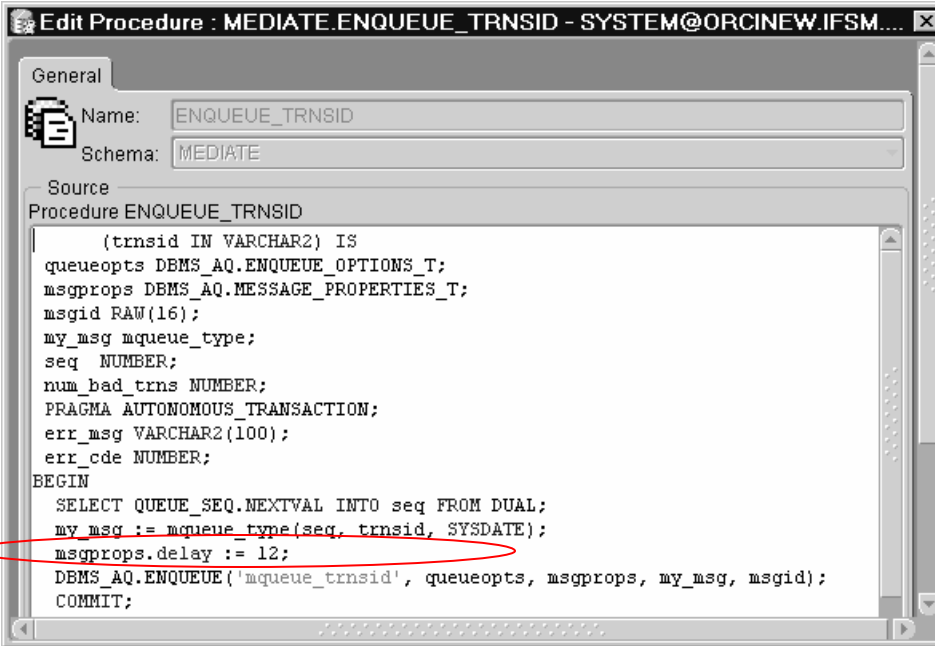
6. Submit the second malicious transaction.

7. Notice that when the integrity level  $\leq 0.95$  and Transaction Submission Rate  $> 10$  the Transaction Delay Time will be increased by 1 second. The condition is kept in the AITDB\_RULE\_CONDS table.

RID	COND_NO	PAREN...	VAR_1	OPERATOR	VAR_2	PAREN...	LNK
1	1	(	INTEGRITY_LEVEL	$\leq$	0.95		AND
1	2		TRNS_SUBMS_RATE	$>$	10		AND
1	3		TRNS_DELAY	$<$	5	)	

8. When the repair process finish, the Transaction Delay Time will be decreased by 1 second.

Note - If the transaction delay time is not changed, increase the delay time (for example: 12->15) in the mediate.enqueue\_trnsid script as shown below.



```

Edit Procedure : MEDIATE.ENQUEUE_TRNSID - SYSTEM@ORCINNEW.IFSM...
General
Name: ENQUEUE_TRNSID
Schema: MEDIATE
Source
Procedure ENQUEUE_TRNSID
(trnsid IN VARCHAR2) IS
queueopts DBMS_AQ.ENQUEUE_OPTIONS_T;
msgprops DBMS_AQ.MESSAGE_PROPERTIES_T;
msgid RAW(16);
my_msg mqueue_type;
seq NUMBER;
num_bad_trns NUMBER;
PRAGMA AUTONOMOUS TRANSACTION;
err_msg VARCHAR2(100);
err_cde NUMBER;
BEGIN
SELECT QUEUE_SEQ.NEXTVAL INTO seq FROM DUAL;
my_msg := mqueue_type(seq, trnsid, SYSDATE);
msgprops.delay := 12;
DBMS_AQ.ENQUEUE('mqueue_trnsid', queueopts, msgprops, my_msg, msgid);
COMMIT;

```

## 6. Isolation Manager

In this demo, we will use test user "test9" to show how the user is isolated and merged back by this system.

We will manually input six transactions including both suspicious and innocent ones for user test9. (For simplicity, we use risk value of transaction to represent the risk of user, for example, risk value 40 means suspicious, risk value of 0 innocent). While other two programs will simulate other two users "test2" and "test3" and keep submitting innocent transactions.

The transactions of users "test2" and "test3" continuously update balance of accounts '1234567890123457' and '1234567890123458' like this "update accounts set acc\_balance=acc\_balance+1". While test9 will submit several shopping-like transactions: a new transaction record in transactions table and balance update on accounts table. The accounts on which test9 will update include these two accounts (xxx57,xxx58) and some others.

User test9 will be isolated after the first transaction and will be merged back after several suspicious transactions and the final innocent transaction. We will see that some of transactions during the isolation period will be back in main tables and some updates will be discarded because of conflicts with the operation of other users.

**Programs:**  
mediator

intrusion detector  
simulator (2)  
test9 interface

**Tables to check for isolation:**

Main table: Accounts; Transactions

Isolation tables: Accounts\_test9; Transactions\_test9.