Contents

[Show route processes per transaction: 1](#_Toc194844538)

[The issue: 1](#_Toc194844539)

[Example of result: 1](#_Toc194844540)

[Steps: 2](#_Toc194844541)

[Send the transaction 2](#_Toc194844542)

[$ snd\_ftr\_test\_in\_the\_fmt\_in.sh 3](#_Toc194844543)

[Get the traces 4](#_Toc194844544)

[$ getTrace.sh 4](#_Toc194844545)

[Script: getTrace.sh 4](#_Toc194844546)

[Script: test\_header.sh 5](#_Toc194844547)

# Version control

|  |  |
| --- | --- |
| Created by | Marcos Lara |
| When | 06-Apr-2025. |
| Objective | Instruction to use getTrace script |
| Observation | This is a draft version. I created this from memory, so I need to perform these steps to validate the entire document. |

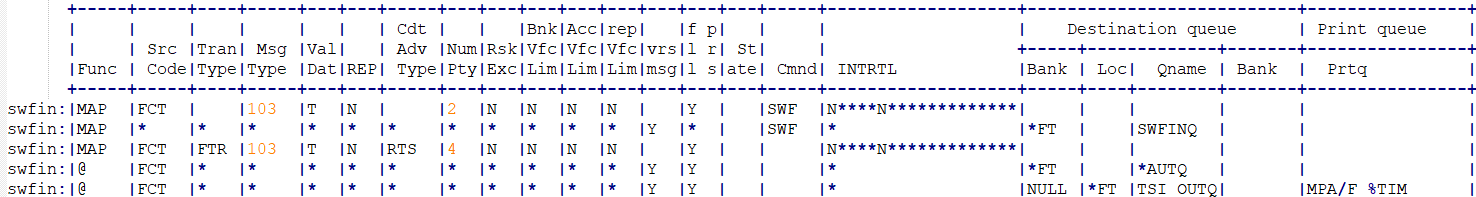
# Show route processes per transaction:

# The issue:

The sequence of scripts below attempts to address an issue related to missing time in analysing the routes taken by a message since it arrives at MTS.

# Example of result:

**Trn:** [**20210105-00000854**]



***Note:*** I know it is easy to read the trace files from your log directory, but if you have some help, you could save time analysing an issue. For example, if you had a formatted output for every transaction, it would indicate the route your message took and the processes involved, as shown in the example above.

It’s possible to run the script getTrace.sh after sending a transaction, which will output the list as shown above on the screen.

# Steps:

## Send the transaction

In the example below, I simulate a message being sent to MTS through the **FTR** record and the FMT process. You can run your test by manually editing the file ‘fmt\_in.dat’ to change the sequence number, or you can use some automation with the script below.

The script assumes that the file ‘fmt\_in.dat.model’ exists in the ‘input’ directory. Please let me know if you require additional information regarding this script.

Script: snd\_ftr\_test\_in\_the\_fmt\_in.sh

**$ cat snd\_ftr\_test\_in\_the\_fmt\_in.sh**

#!/bin/bash

source /home/**username**/scripts/libShare.sh

cd $AREA\_ROOT\_DIR/input

last=$(getLastTrn)

if [ ! -f fmt\_in.dat.model ];then

cp -p /usr/local/intranet/areas/**myarea**/input/**fmt\_in.dat.model** .

fi

# Calculate the last transaction sequential number

if [ ! -f FMT\_IN.SEQ ];then

linecmd -b asb -location MTRANS -line FMT\_IN -sh > /tmp/x14.txt 2>&1

c=`grep Next /tmp/x14.txt | awk '{ print $4 }'`

c=`expr $c - 1`

echo $c > FMT\_IN.SEQ

fi

c=`cat FMT\_IN.SEQ`

c=`expr $c + 1`

echo $c > FMT\_IN.SEQ

c=`printf "%.6d" $c`

# Adjust the sequential number into output file

sed 's/KKKKKK/'$c'/g' fmt\_in.dat.model > fmt\_in.dat

sleep 3

linecmd -bank asb -loc mtrans -line fmt\_in -show

linecmd -bank asb -loc mtrans -line fmt\_in -command trace

linecmd -bank asb -loc mtrans -line fmt\_in -up

linecmd -bank asb -loc mtrans -line fmt\_in -down

linecmd -bank asb -loc mtrans -line fmt\_in -show

last2=$(getLastTrn)

if [ "$last2" != "$last" ];then

showMessage "The Last Transaction number is: $last2"

msgp $last2

else

showMessage "The transaction was not created. Listing the log."

lf=`ls -ltr $AREA\_ROOT\_DIR/logs/\*fmt\* | tail -1 | awk '{ print $9 }'`

cat $lf

fi

***Note:***For your reference, please be aware that the account numbers have been modified to maintain their confidentiality. Additionally, note that **KKKKKK** will be replaced by the sequential number calculated by the script.

$ cat fmt\_in.dat.model

\*FTR **KKKKKK**20210105BRN00324 NRTNZDNZD000000000000.00S NN

2FTR 210105 SHA

3FTR000000000800.0000000000000000.9415670 T

5FTR 000000000000.00 000000000000.00 BRN

DFTR210105E 210105E 210105C 210105C 210105I 210105I

GFTRNZD

DBT=G/999999999999999

ORG= /888888888888888

-MR JAMIE BEHL

-7/23 NAPIER STREET

-FREEMANS BAY

-AUCKLAND

BBK=E/XYU063330

-SOMEBANK IN AUSTRALIA

-5, NEENAG STREET

-SOUTHPORT

BNF= /FR7777777777777777777777767,BNK=N

-Jamie Behl

-7/23 Napier Street,

-Freemans Bay

-Auckland, New Zealand

OBI=/RFB/NZ Acount

SRF=100292020**KKKKKK**

CTR= FERS IMT Reference Number: 12345600031741

- MR MARK DEBUSI

/FTR**0000KKKKKK**

Run the script:

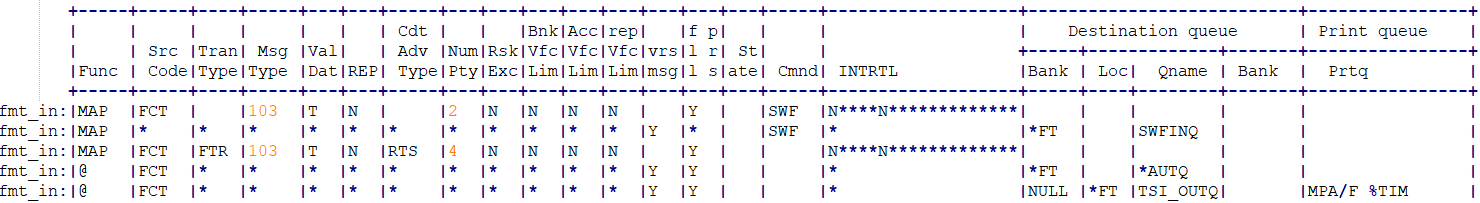
### $ snd\_ftr\_test\_in\_the\_fmt\_in.sh

## Get the traces

Then run the second script to check what route your message took.

### $ getTrace.sh

**Trn:** [**20210105-00000855**]

****

***Note:*** This is an example only. I will update this document with the results of a real test.

### Script: getTrace.sh

$ cat getTrace.sh

#!/bin/bash

#### this should be included in all scripts ###############

source libShare.sh

shname=$(getShellName $BASH\_SOURCE)

showmessage()

{

showMessage "$shname $1"

}

#### This should be included in all scripts ###############

showmessage "INF getTrace.sh Starting the process"

cd ~/scripts

for lx in `ls $AREA\_ROOT\_DIR/logs/\*.log`

do

showmessage "INF Running the command getTrace $lx"

java -cp . getTrace $lx > /dev/null

done

showmessage "INF Listing the generated files by transaction number.."

#

for lx in `ls ????.txt | sort`

do

showmessage "INF Trn: `echo $lx | sed 's/.txt//g'`."

test\_header.sh | sed '/^$/d'

cat $lx

echo

done

### Script: test\_header.sh

$ cat test\_header.sh

#!/bin/bash

cd ~/scripts

java -cp .:/usr/local/intranet/areas/**myarea**/scripts test\_header

if [ $? -ne 0 ];then

echo `date`: Error Running the file: test\_header.java

exit

fi

## More notes:

You may notice that most of the scripts were created using the old BASH notation for the shell, where the character ‘(grave accent or simply backtick) can be replaced by the sequence $(commands). Both methods work well; it's merely a matter of personal preference.