Fluxo de mensagens no Sistema Millennium

1. Começa-se por indicar que existe 3 canais de comunicação no Millennium que são: X2X (X2X\_COM), MXS (MXS\_COM) e o novo OLM (OLM\_COM) que no programa sendout é identificado pelo campo PRCSRC do buffer:

        IF (X2X\_GAME\_STATE.EQ.X2X\_GAMES\_UP) THEN

C         CALL X2ADDPRO(BUF)                        ! MXSRV

          IF (HPRO(PRCSRC,BUF).EQ.OLM\_COM) THEN ! V05 - OLM

            CALL QUETRA(OLM,BUF)                    ! V05 - OLM (rever se deve usar o ABL ou é QUETRA)

C V05          IF (HPRO(PRCSRC,BUF).EQ.MXS\_COM) THEN     ! MXSRV

          ELSEIF (HPRO(PRCSRC,BUF).EQ.MXS\_COM) THEN     ! MXSRV

            CALL QUETRA(MXS,BUF)                    ! MXSRV

          ELSE                                      ! MXSRV

            CALL X2ADDPRO(BUF)                      ! MXSRV

          ENDIF                                     ! MXSRV

        ELSE

            CALL RELBUF(BUF)

        ENDIF

1º canal e mais antigo é X2X que se pode ver no programa X2RCVBUF no qual constrói o header da messagem/buffer

           IF (X2X\_I4\_STATION) THEN

          PRO(TERNUM,PROBUF)=TERMINAL\_NO

          PRO(LINENO,PROBUF)=STATION\_NO

       ELSE

          HPRO(TERNUM,PROBUF)=TERMINAL\_NO

          HPRO(LINENO,PROBUF)=STATION\_NO

       ENDIF

C          \*\*\*\*\* End V04 changes \*\*\*\*\*

       HPRO(PRCSRC,PROBUF)=X2X\_COM

       HPRO(PRCDST,PROBUF)=0

       HPRO(X2X\_CONNCTL\_OVR,PROBUF)=0

       HPRO(X2X\_DELIVER\_OVR,PROBUF)=0

       HPRO(X2X\_HOST\_ID,PROBUF)=HOST\_ID

       HPRO(TRCODE,PROBUF)=TYPREG

       HPRO(QUENUM,PROBUF)=QIN

       HPRO(MSGNUM,PROBUF)=0

       PRO(TIMOFF,PROBUF)=X2X\_LOOP\_TIME

       HPRO(INPLEN,PROBUF)=MES\_LEN

       HPRO(X2X\_DEST,PROBUF)=DEST\_LAYER

       HPRO(X2X\_LINK,PROBUF)=0

C          \*\*\*\*\* Start V04 changes \*\*\*\*\*

           IF (X2X\_I4\_STATION) THEN

          IF (IAND(DEST\_LAYER,X2DEST\_STATION).EQ.0)

     \*        PRO(LINENO,PROBUF)=SSAP !LINE # IS SSAP FOR XPORT

       ELSE

          IF (IAND(DEST\_LAYER,X2DEST\_STATION).EQ.0)

     \*        HPRO(LINENO,PROBUF)=SSAP !LINE # IS SSAP FOR XPORT

       ENDIF

Após construção do header da Mensagem enviado para a queue aplicacional INQUE pela subroutina QUEINP

IF (X2X\_GAME\_MODE.EQ.X2X\_GAMEM\_DAY) THEN

                    CALL QUEINP(PROBUF,ST)

Sendo a queue aplicacional INQUE como o próprio nome indica é de entrada a leitura da Mensagem/buffer é feita pelo programa ENCPRO

    CALL DQUINP(BUF\_NO)

Este programa é onde obrigatório a Mensagem passar (X2X,MXS) pois é onde é feito a logica de encriptação/desencriptação para o caso do (OLM) só passa mas já no percurso de saída da Mensagem. ENCPRO depois envia a mensagem para o Programa chamado de dispatcher que têm o nome de DISPAT pela subroutine ABL.

ABL(BUF\_NO,QUETAB(1,DIS),STATUS)

O programa DISPAT(dispatcher) é de grande importância na medida de que é onde se decide para onde a mensagem é reencaminha ou seja qual é programa correto para processar a mensagem, essa decisão é feita pela variável QUE que indica para qual quele aplicacional a mensagem é enviada. Para obter a QUE é chamado a subroutine GETQUEUE

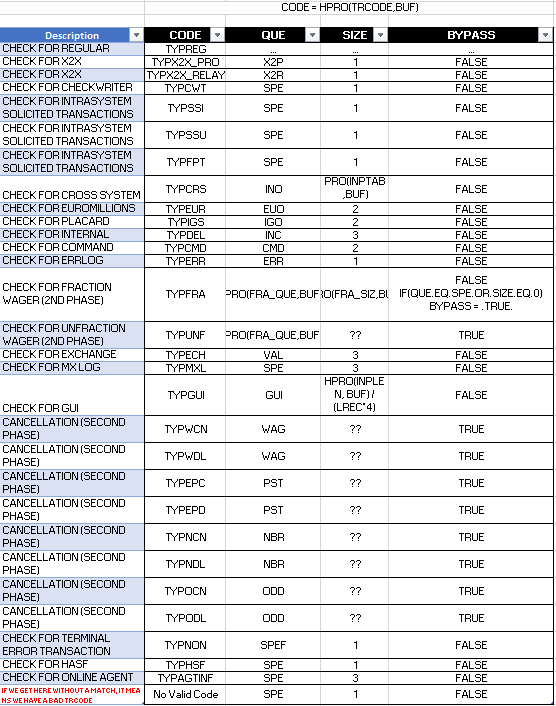
SUBROUTINE GETQUE(BUF,QUE,SIZE,BYPASS)

Sendo que o primeiro parâmetro é de entrada e é a mensagem em si, o 2º parâmetro que é de saída é a QUE como já foi referido, o 3º parâmetro SIZE que é de saída é referente ao número de segmentos do tmir (1,2 ou 3) por ultimo o BYPASS é também um parâmetro de saída para indicar se a mensagem vai voltar ou não ao Disptacher para voltar a ser processada (referente ao numero de serie).

QUE é obtida através do campo TRCODE da mensagem que indica o tipo de mensagem

CODE = HPRO(TRCODE,BUF)

Obtendo-se a seguinte tabela (ver documento MILL\_DATA\_STRUCTERS\_igs\_(DRAFT).xlsx na tab ***TRCODE-QUEUE Mapping*** e conjugação com a tab ***TRCODE types***):



Depois é enviado para a respetiva queue aplicacional em:

    CALL QUETRA(QUE,BUF)

Sendo a variável QUE dinâmica tomando os valores tabelados em cima.

Agora dependendo do tipo de Mensagem mas assumindo que é uma aposta do totoloto ou totobola é uma WAG então no programa WAGPRO:

TASK    = WAG

CALL DQUTRA(TASK,BUF)

Este programa é responsável por processar a mensagem de entrada e gerar a mensagem de saída em que vê qual é tipo de Jogo

GTYP = BPRO(BINPTAB+5,BUF)

A partir do tipo jogo faz o respetivo processamento

IF(GTYP.EQ.TLTO) THEN

            CALL DLOTTO(PRO(INPTAB,BUF),TRABUF,HPRO(INPLEN,BUF))

        ELSE IF(GTYP.EQ.TSPT) THEN

            CALL DSPORT(PRO(INPTAB,BUF),TRABUF,HPRO(INPLEN,BUF))

A mensagem de saída é construída na subroutine OUTWAG que como se pode ver escreve na secção OUTTAB do buffer já a indicar a direção do fluxo é de saída.

CALL OUTWAG(TRABUF,PRO(OUTTAB,BUF),HPRO(OUTLEN,BUF))

Neste momento só falta registar a transação/Transaction no TMF para esse efeito é enviado agora a mensagem para a queue aplicacional APU

        CALL QUETRA(APU, BUF) !send BUF/message to APU application queue

        CALL DQUTRA(TASK,BUF) !after sending the message to APU application queue can now remove the current application queue that is TASK=WAG

O programa que está a consumir esta queue aplicacional é o APULOG que

    TASK=APU

    CALL TOPQUE(TASK,BUF)

Obtendo o número de serie

XSER = IAND(PRO(WRKTAB,BUF),'3FFFFFFF'X)

Sendo o serial enviado para a subroutine APU\_WLOG:

C INPUT

C     SERIAL   -  TRANSACTION SERIAL NUMBER

C     RECORD   -  TRANSACTION LOG RECORD

    SUBROUTINE APU\_WLOG(SERIAL,RECORD)

Sendo posterior enviado para queue aplicacional LOG

    CALL QUETRA(LOG,BUF)

    CALL DQUTRA(TASK,BUF)

O programa a seguir é chamado de LOGGER neste programa não se está a obter a mensagem de nenhuma queue aplicacional nem mesmo da LOG onde a mensagem se encontra atualmente, pois é feito obtenção de dados da memoria escrita pelo APU\_WLOG (sendo a lógica implementado de muito baixo nível saído do pretendido nesta documentação não sendo por isso referido esses mesmos processos…). É neste momento (logger) onde é registado a mensagem no TMF (primario, backup e disco) de lembrar que a mensagem pode ocupar até 3 segmentos no máximo (48 bytes no total sendo 16 bytes por segmento) sendo a primeira posição da mensagem (posição do 1º segmento) o Numero de Série Interno.

Neste programa é chamado a subroutine LOGOUT

CALL LOGOUT(NUMOUT,HSER,HPPI)

Esta subroutine sim já está a usar a queue aplicacional LOG como se pode ver

    CALL DQUTRA(LOG,BUF)

Nesta subroutine é validado que se a mensagem for do tipo regular é enviado para a queue aplicacional **GAME\_OUTQUE** caso não seja validado que o TERMINAL é maior que 0 ou seja esteja definido ***TERMINAL=HPRO(TERNUM,BUF)*** sendo neste caso também redirecionado para queue aplicacional já referida **GAME\_OUTQUE**

        IF (HPRO(TRCODE,BUF).EQ.TYPREG) THEN

          CALL ABL(BUF,GAME\_OUTQUE,ST)

        ELSE

C

C SEND TO SPECIFIC TERMINAL

C

          IF (TERMINAL.GT.0) THEN

            IF(BTEST(AGTTAB(AGTTYP,TERMINAL),AGTMXT)) THEN    !MXSRV

                HPRO(PRCSRC,BUF)=MXS\_COM                        !MXSRV

            ENDIF                                             !MXSRV

            CALL ABL(BUF,GAME\_OUTQUE,ST)

          ELSE

Caso a mensagem não seja do tipo regular e não tenha definido um terminal quer dizer que é uma mensagem unsolicited ou broadcast então envia para todos os terminais MXS ou X2X.

C

C BROADCAST

C

             IF (HPRO(X2X\_DEST,BUF).NE.0) THEN

                IF (X2X\_GAME\_STATE.EQ.X2X\_GAMES\_UP) THEN

                   CALL X2ADDPRO(BUF)

                ELSE

                   CALL RELBUF(BUF)

                ENDIF

             ELSE

C

C SEND BUFFER TO ALL MXS TERMINALS

C

                    IF (HPRO(PRCSRC,BUF).EQ.MXS\_COM) THEN     ! MXSRV

                       CALL QUETRA(MXS,BUF)                   ! MXSRV

                    ELSE

O programa que está a consumir a mensagem enviado para queue aplicacional **GAME\_OUTQUE** não é mais nada que a ENCPRO pois não só trata das mensagens de entrada como as mensagens de saída também.

100 CONTINUE

    CALL RTL(OUTPUT\_BUF,GAME\_OUTQUE,ST)

Nesta secção do ENCPRO da mensagem de saída é chamado a subroutine **SENDOUT**

CALL SENDOUT(OUTPUT\_BUF)

Esta subroutine é onde se valida para qual canal a mensagem deve ser encaminhada se for do tipo **OLM\_COM** é enviado para o canal **OLM**, se for do tipo **MXS\_COM** é enviada para o canal MXS, caso contrário envia para o canal que sobra o **X2X** pois é implícito que é do tipo **X2X\_COM**.

        IF (X2X\_GAME\_STATE.EQ.X2X\_GAMES\_UP) THEN

C         CALL X2ADDPRO(BUF)                        ! MXSRV

          IF (HPRO(PRCSRC,BUF).EQ.OLM\_COM) THEN ! V05 - OLM

            CALL QUETRA(OLM,BUF)                    ! V05 - OLM (rever se deve usar o ABL ou é QUETRA)

C V05          IF (HPRO(PRCSRC,BUF).EQ.MXS\_COM) THEN     ! MXSRV

          ELSEIF (HPRO(PRCSRC,BUF).EQ.MXS\_COM) THEN     ! MXSRV

            CALL QUETRA(MXS,BUF)                    ! MXSRV

          ELSE                                      ! MXSRV

            CALL X2ADDPRO(BUF)                      ! MXSRV

          ENDIF                                     ! MXSRV

        ELSE

            CALL RELBUF(BUF)

        ENDIF

Diagrama provisório pelo qual se vai se baseado (final a ser feito pelo draw.io)

