Monitor III Batch (MON3B) – User's Guide

Version 1.0

Content:

Figures:

1. Introduction

Monitor III Batch (M3B) is a tool which allows to continuously monitor the system state by retrieving data from RMF Monitor III, to process the information and to send, push notifications to the zEvent mobile application and WTOs to the operator console. M3B is a REXX program which provides a programmable interface for the end user. The function retrieves RMF Monitor III reports via the ERB3XDRS API and allows the user to specify which data should be evaluated for generating events. Events are user defined notifications and snapshots (graphics) which can be sent to the zEvent app as well as WTOs for the operator console. The programming interface supports REXX control statements to process the report data and to specify the conditions under which the notifications are sent to end devices. The function relies on the zEvent notification API and the zEvent apps for iOS or Android.

Annotation: This program is not identical with the RMF Monitor III Batch session which is described in the RMF User's Guide.

For more information refer to the zEvent website
http://www-03.ibm.com/systems/z/os/zos/tools/zevent/

2. Quick Start

2.1 Prerequisites

In order to use MON3B efficiently it is necessary to setup zEvent on the same z/OS systems from which you want to use the function and to install the zEvent app on your iOS or Android mobile phone. MON3B uses the zEvent REXX API. You must perform the following steps before you can run MON3B:

- zEvent mobile app
  - Download the zEvent mobile app either for your iOS or your Android device from the respective app store. Install the app and follow the instructions in the zEvent User's Guide to setup you mobile device: ftp://public.dhe.ibm.com/eserver/zseries/zos/wlm/zEvent_User_Guide_v1.0.pdf
- zEvent on z/OS
  - You must also perform either the HTTPS or AT-TLS setup to connect your z/OS host to the push mechanisms. This is described in the zEvent administration guide: ftp://public.dhe.ibm.com/eserver/zseries/zos/wlm/zEvent_Administration_Guide_v1.0.pdf
- RMF Monitor III
  - You must run the RMF Monitor III data gatherer session (RMFGAT) on your system
- Your TSO userid (from which you submit the MON3B batch job) must be able to access RMF Monitor III reporter sessions.

2.2 Setting Up Monitor III Batch

Now you can setup MON3B on your z/OS system. MON3B is distributed with the following TSO datasets which need to be uploaded to your TSO userid:

- RMFM3.CEXEC.BIN contains the compiled MON3B REXX exec
- RMFM3.LINKLIB.BIN contains support programs required by MON3B
- RMFM3JCL.BIN contains sample job control to easily start using the function

First you must transfer the datasets in binary format. One possibility is to open a command window and use the following FTP command which is shown for RMFM3.CEXEC.BIN as example:

```
FTP your-zos-system
<enter: userid, password>
QUOTE SITE PRIM=1 SEC=1 CYL LRECL=80 RECFM=FB BLKSIZE=3120
IMAGE
PUT RMFM3.CEXEC.BIN
200 Port request OK.
125 Storing data set BVAU.T113.CEXEC.BIN
250 Transfer completed successfully.
```

Figure 1: FTP commands to transfer z/OS datasets

After transferring the datasets you must convert them to z/OS datasets:

```
RECEIVE INDSN('UID.RMFM3.CEXEC.BIN') and press ENTER
```

Figure 2: Restore z/OS host dataset

The original datasets are restored and can now be used. Running Monitor III Batch on z/OS

3. Running Monitor III Batch

After transferring the datasets and assuming you also use RMF Monitor III in your installation you can use MON3B by modifying and submitting the sample jobs in RMFM3.JCL or defining your own job control. You must also setup Rules in the zEvent host dialog to define the receivers of your messages and you must generate the zEvent REXX command which is used by MON3B to send messages to the zEvent mobile app.

3.1 Setting up MON3B in the zEvent Host Dialog

You must define at least one Rule for the zEvent REXX API which allows the push services to identify the sender and receiver of events. Figure 3: Setting up Rules for MON3B shows the zEvent Rules dialog with a set of rules for MPF exist and Monitor III batch function (marked in red). It necessary to provide the following information for a MON3B rule:

- The origin meaning where the caller is. In this document the abbreviation M3B is used as origin for the Monitor III batch function.
- The message identifier.
- The receiver.

For all other columns a star can be specified and the attributes column can remain empty. It is also possible to define default attributes. Please refer to the zEvent administration guide on how to do this. It is not required to run MON3B for defining the rule.

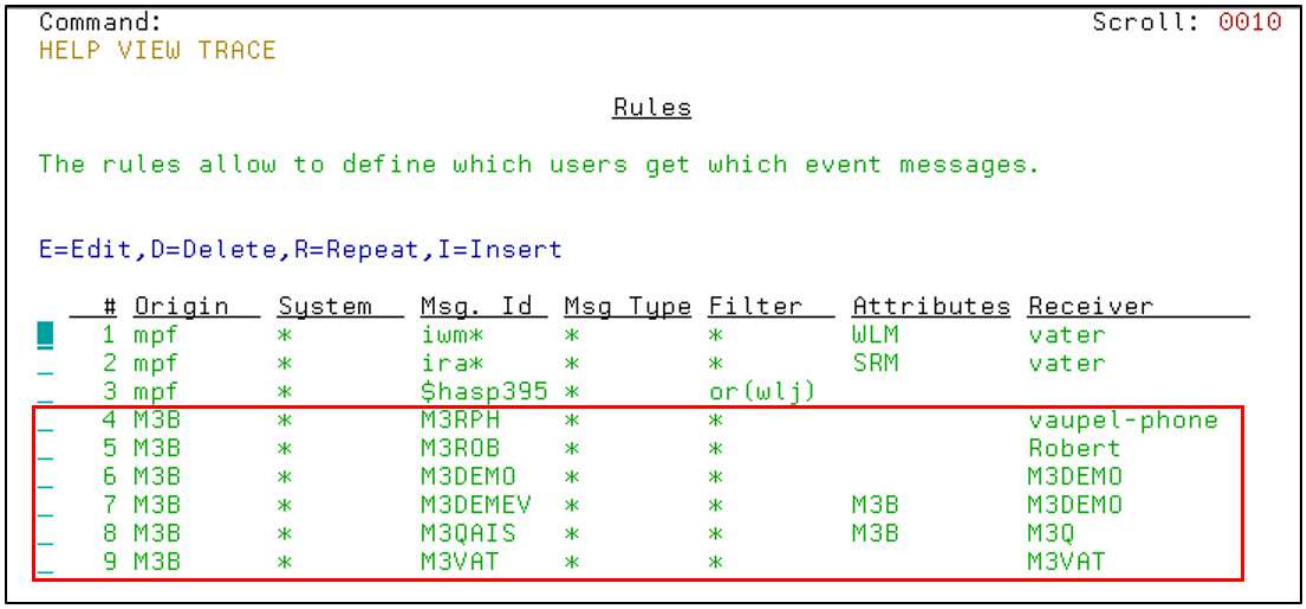

Figure 3: Setting up Rules for MON3B

As an example for a rule definition we select the fifth rule (M3ROB):

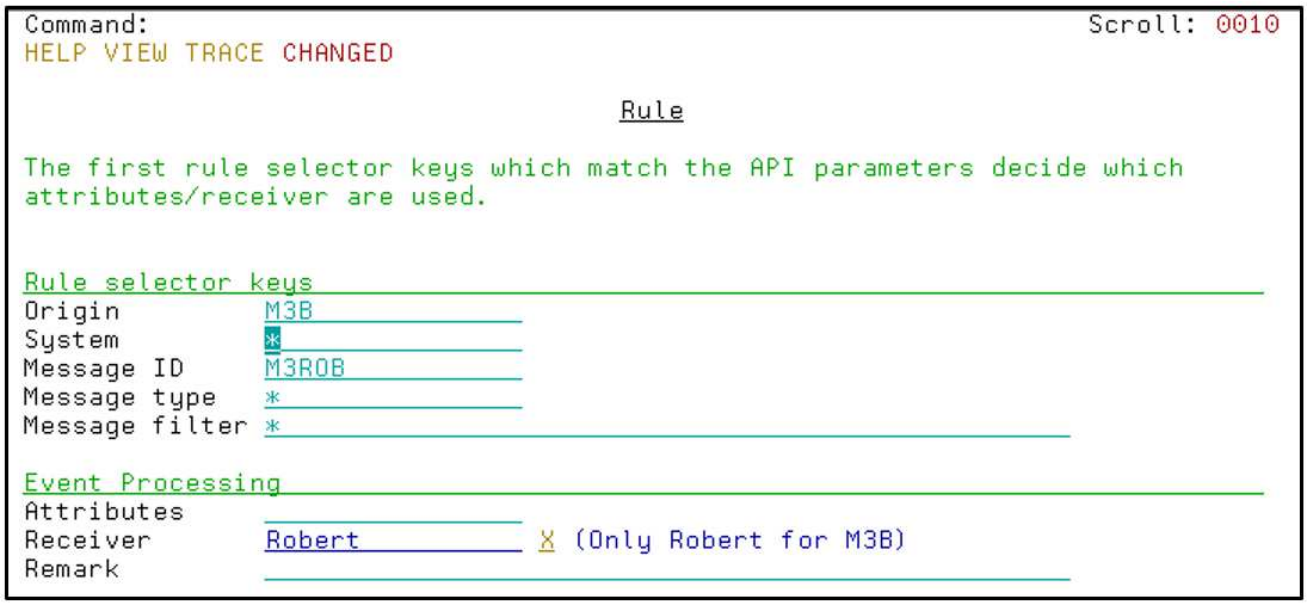

Figure 4: Rule definition

The receiver is named Robert and can be either the name of a smartphone or a group of smartphones. In this example the smartphones are already registered and contained in the group Robert (see Figure 5: Sample Group Definition)

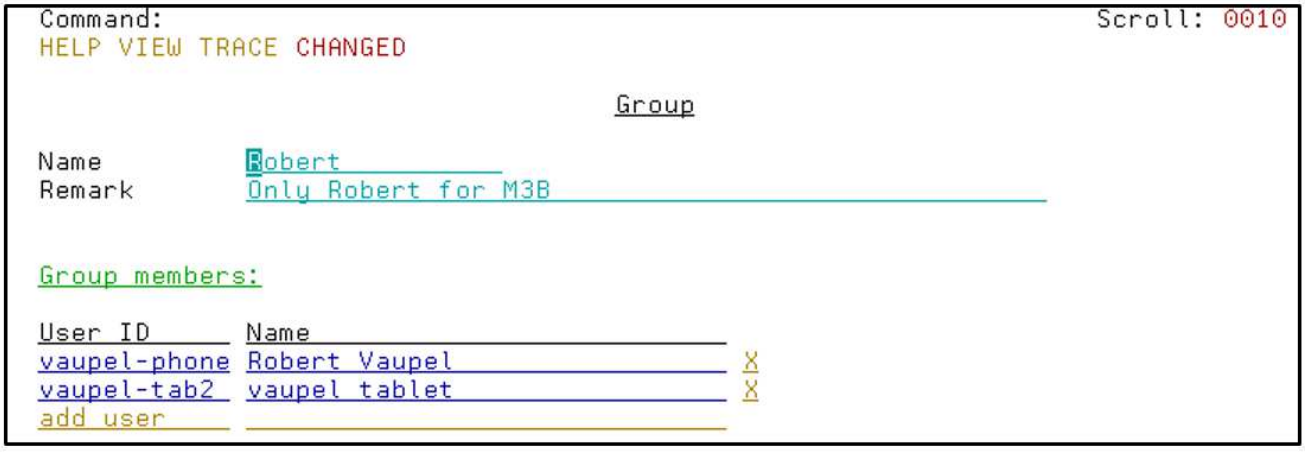

Figure 5: Sample Group Definition

Important: The rule qualifier or message id needs to be remembered. In our example this was named M3ROB. We have to define this name in MON3B batch job to send the message to the associated group of mobile phones.

3.2 Creating the zEvent REXX command

Finally the zEvent REXX command must be created. Figure 6: Creating the zEvent REXX command shows the dialog for generating the REXX and for saving it to a CLIST library. The dataset name of the CLIST library needs to be referred to in the job control to run MON3B.

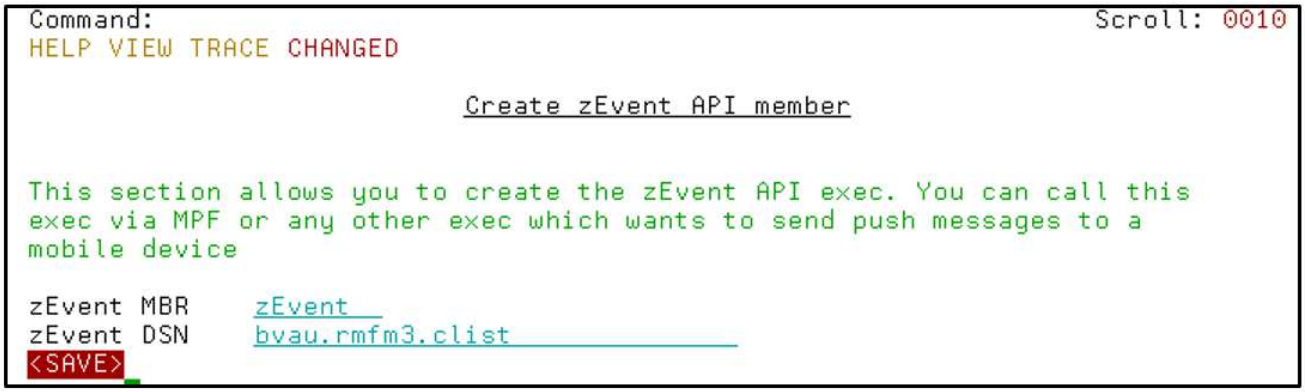

Figure 6: Creating the zEvent REXX command

3.3 Monitor III Batch Sample Job Control

The job control dataset contains a set of sample job controls to run Monitor III Batch and to send events to the zEvent apps which are defined as receivers or to issue WTOs to the operator console. In this chapter we will just describe the job control part which is rather simple and in the next chapter the programming environment for Monitor III Batch. Figure 7: Sample Job Control shows the few job control statements which are required to run Monitor III Batch:

- A jobcard, the sample job is named MON3CJOB

- The JOBPARM parameter is just optional and can be deleted
- The execution statement (EXEC) to establish a batch TSO session and to invoke the MON3B REXX command
- The STEPLIB statement for the supporting modules. This can be deleted if the modules are part of the LINKLIB concatenation
- The SYSPROC statement which defines the library for MON3B

```
//MON3CJOB JOB '?,?',,NOTIFY=&SYSUID,REGION=0M,
//         MSGLEVEL=(1,1),MSGCLASS=H,CLASS=C
/*JOBPARM SYSAFF=*
//*************************************************
//MON3BTCH EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50,
//  PARM='MON3B '
//STEPLIB  DD   DISP=SHR,DSN=<uid>.RMFM3.LINKLIB
//SYSPROC  DD   DISP=SHR,DSN=<uid>.RMFM3.CEXEC
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSTERM  DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//SYSIN    DD   DUMMY
//OUTDSN   DD   DUMMY
//REPORT   DD   DUMMY
//PARMS    DD   *
! Here starts the control information
```

Figure 7: Sample Job Control

The interesting part is now the programming of the REXX command which is described in the next main chapter

4. Programming Monitor III Batch

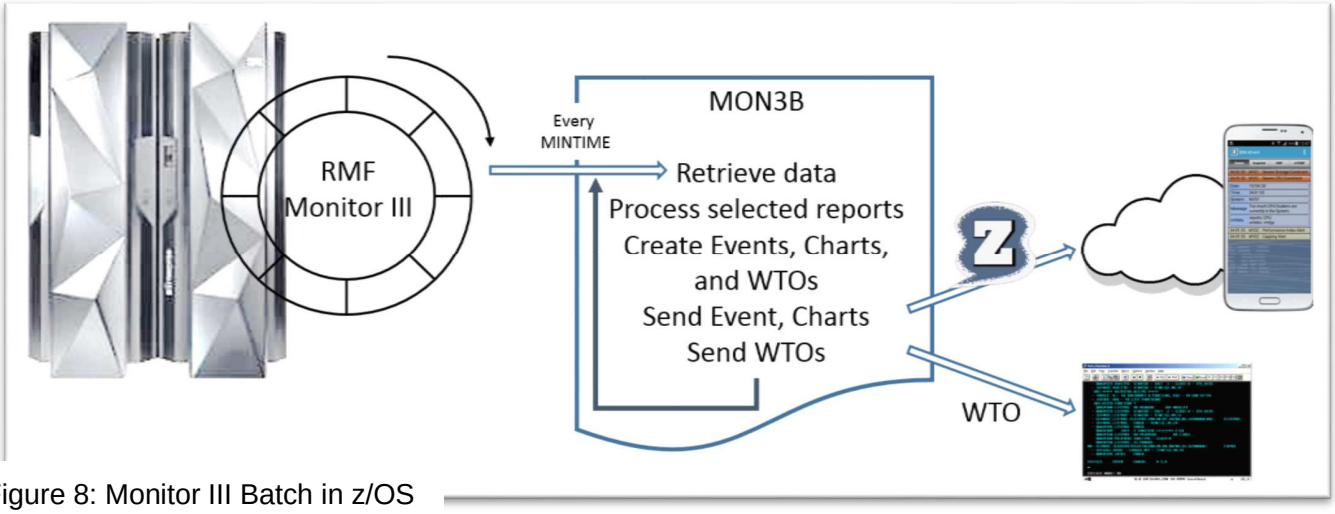

Figure 8: Monitor III Batch in z/OS

Figure 8: Monitor III Batch in z/OS shows the relationship between RMF Monitor III data gatherer and Monitor III batch. RMF Monitor III periodically collects data and saves it into its wrap around buffer at the end of the Monitor III Mintime. When Monitor III batch is started it access the RMF Monitor III wrap around buffer through interface ERB3XDRS and retrieves the current set of samples. The set of samples is formatted into table layout with the help of RMF Monitor III services and then available for the batch function to access the reports, retrieve and process data, create events, charts and WTOs and send them to the mobile phones by using the zEvent REXX API or to the operator console.

4.1 MON3B Processing Concept

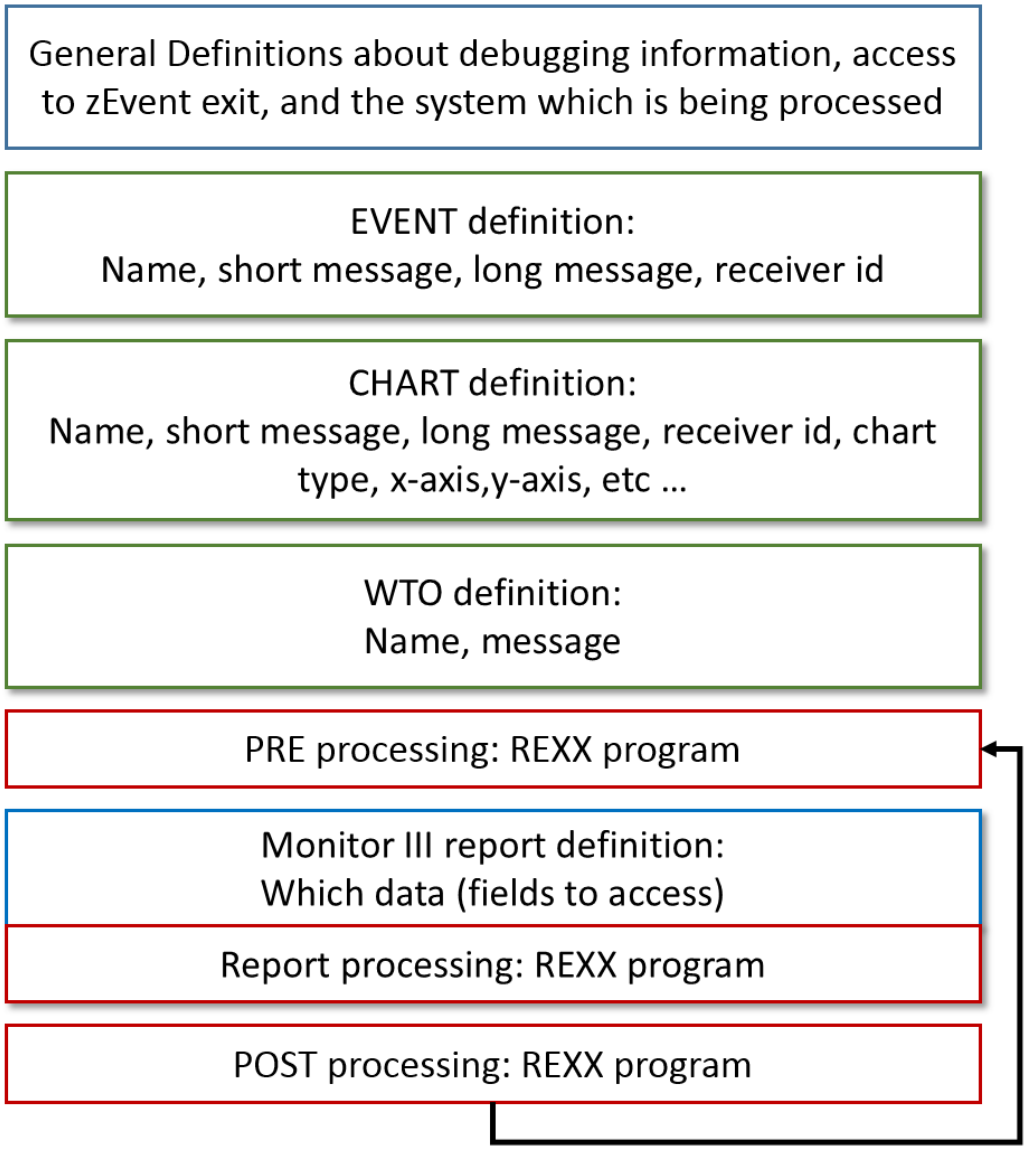

Figure 9: MON3B Processing Concept

Figure 9: MON3B Processing Concept shows the possible elements of a Monitor III Batch program. These elements follow the PARMS control statements in Figure 7: Sample Job Control. Usually the

program contains one or multiple general processing statements which are directives for the MON3B REXX command followed by at least one EVENT, CHART, or WTO definition. It is also possible to have multiple of such definitions and it is not required to have all of these elements. The processing logic follows marked in red rectangles in the graphic. The program retrieves Monitor III data at the end of each Mintime. First a pre-processing section allows to set and establish common variables. In the next step the desired reports are processed. For the reports it is also necessary to define which fields should be accessed. The fields are described in the RMF Programmer's Guide available at

http://publibz.boulder.ibm.com/epubs/pdf/erb2pg10.pdf

The field definitions which can be used by MON3B are described in *Chapter 8: Monitor III data reporter tables*.

Figure 9: MON3B Processing Concept shows the Monitor III report definitions and processing control statements together. All sample programs which are shipped show this structure but it is not required to have the definitions together. At the end a post processing section allows to specify logic which issues the events, charts and WTOs conditionally to the desired end user devices.

4.2 MON3B Example: Monitoring Job Resource Consumption

We will now look at a simple example which monitors jobs and address spaces in a z/OS environment and which sends events based on the job CPU consumption. In addition a snapshot chart is also send at the end of each Monitor III Mintime listing the 10 jobs with the highest CPU consumption in the system.

```
//MON3CJOB JOB '?,?',,NOTIFY=&SYSUID,REGION=0M,
//         MSGLEVEL=(1,1),MSGCLASS=H,CLASS=C
/*JOBPARM SYSAFF=*
//*************************************************
//MON3BTCH EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50,
//  PARM='MON3B '
//STEPLIB  DD   DISP=SHR,DSN=<uid>.RMFM3.LINKLIB
//SYSPROC  DD   DISP=SHR,DSN=<uid>.RMFM3.CEXEC
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSTERM  DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//SYSIN    DD   DUMMY
//OUTDSN   DD   DUMMY
//REPORT   DD   DUMMY
//PARMS    DD   *
! This is a comment
! DEBUG,PRINTPARMS,PRINTREPS
DEBUG,PRINTPARMS,PRINTINFO
! Event Definitions
EVDSN,<uid>.RMFM3.CLIST
EVENT,JOB,CAT,#jobcatg
EVENT,JOB,SMSG,#jobsmsg
EVENT,JOB,LMSG,#longmsg
EVENT,JOB,RID,M3RPH
! Chart definition
CHART,JOB,COND,1
CHART,JOB,RID,M3RPH
CHART,JOB,CAT,3
```

```
CHART,JOB,MSG,'JOB consumption'
CHART,JOB,TIT,'AQFT JOB consumption'
CHART,JOB,LGD,false
CHART,JOB,YTIT,'Utilization'
CHART,JOB,XTIT,'Jobs'
CHART,JOB,XPTS,10
CHART,JOB,XAXS,E,ORDER,TOP,1
CHART,JOB,YAXS,1,APPL%,#350D08,Columns,false
PRE,$BEGIN
   #alert1 = 30
   #alert2 = 70
   #alert3 = 90
   #resl.0 = 0
   if datatype(#cnt,'N') = 1 Then #cnt = #cnt + 1
   else #cnt = 1
   say left('-',60,'-')
   say 'Interval#: 'right('0000'#cnt,4)
   say left('-',60,'-')
   #Rc = $InitChart('JOB')
$END
REP,PROC,T,PRCPJOB,,$USEPREVIFEMPTY
REP,PROC,T,PRCPASI,X,$NOP
REP,PROC,T,PRCPTYPE,,$NOP
REP,PROC,T,PRCPSVCL,,$NOP
REP,PROC,T,PRCPEAPP,,$NOP
REP,PROC,P,T,$BEGIN
   #resstr = ''
   #psbl = 0
   #cptyp = 0
   if datatype(#PRCPEAPP,'N') = 1 then #psbl = 1
   if strip(#PRCPTYPE) = 'CP' then #cptyp = 1
   if #psbl & #PRCPEAPP > #alert1 then do
     #resstr = 'Job= '#PRCPJOB||' uses '||#PRCPEAPP||' Appl%'
     #re2str = 'JobName= '#PRCPJOB||', '
     #re2str = #re2str||'ASID#= '#PRCPASI||', '
     #re2str = #re2str||'Service Class = '||#PRCPSVCL||', '
     #re2str = #re2str||'Application Utilization = '||,
               #PRCPEAPP||'%'
     #rescat = 4
     if #PRCPEAPP > #alert2 Then #rescat = 2
     if #PRCPEAPP > #alert3 Then #rescat = 1
     #i = #resl.0+1
     #resl.0 = #i
     #resl.#i = #resstr
     #re2l.#i = #re2str
     #recl.#i = #rescat
   end
   if #cptyp then #Rc = $AddElement('JOB',#PRCPJOB,#PRCPEAPP)
$END
```

```
POST,$BEGIN
   do #i = 1 to #resl.0
     #jobsmsg = #resl.#i
     #longmsg = #re2l.#i
     #jobcatg = #recl.#i
     #Rc = $SENDEVENT(JOB)
     say 'After SENDEVENT: 'time()
   end
$END
```

Figure 10: Monitoring Job CPU Consumption

Figure 10: Monitoring Job CPU Consumption shows the complete job control with all necessary control statements to monitor job and address space consumption based on the RMF Monitor III PROC report. We will describe the control statements now and the full syntax in the next chapter:

- A comment line starts with a !
- DEBUG allows to print certain information during the program execution
  - PRINTPARAMS the input parameter at the beginning of the processing
  - PRINTREPS the content of the variables obtained from the Monitor III reports every Mintime. This is usually as long as you develop a new control sequence
  - PRINTINFO some information during the program execution
- EVDSN specifies the CLIST dataset which contains the zEvent exec. Theoretically this can also be specified in the SYSPROC DD statement but then the CLIST and CXEC datasets need to have the same dataset attributes.
- EVENT defines an event which will be sent to the mobile phones
- CHART defines a CHART or snapshot sent to the mobile phones
- REP specifies the reports from which the Monitor III data is extracted
- PRE is a pre-processing control sequence at the beginning of Monitor III Mintime cycle. This is usually necessary to initialize control variables
- POST is a post processing control sequence at the end of the data retrieval and processing cycle.

EVENT Definition

```
EVENT,JOB,CAT,#jobcatg
EVENT,JOB,SMSG,#jobsmsg
EVENT,JOB,LMSG,#longmsg
EVENT,JOB,RID,M3RPH
```

The first parameter for each event definition is the name of the event. The name in this example is JOB. All event statements with the same name describe one event. In addition the attributes of the event need to be specified:

- CAT is the category of the event. The category will be used in this example to give the event a certain color on the mobile phone.
- SMSG is the short message which is displayed on the event screen.
- LMSG is the long message which is displayed when the event is opened on the mobile phone

- RID is the receiver identifier. In this example we use M3RPH. This identifier is defined as a message id for the program M3B in the rules section of the zEvent ISPF dialog, see Figure 3: Setting up Rules for MON3B.

The values for CAT, SMSG, LMSG are variables: #jobcatg, #jobsmsg, and #joblmsg which are set in the REXX program sections of the input stream

Recommendation: It is recommend to use a number sign # in front of all variables which are defined and used in the control parameter sections because the REXX program never uses the # sign for any variable. Therefore variables in the control parameter section cannot interfere with any variables in the REXX program.

Figure 11: Example for a Job event

The event is sent at the end of the processing in the POST processing section via the $SENDEVENT macro. MON3B provides some external routines which are available in the control program section. In this example 3 steam variables #resl. #re2l. and #recl. Are set in the report processing section when the job meet a certain criteria. At the end of the control parameter section a loop evaluates the context of the steam variables and sends for each entry which represents a job meeting the criteria an event to the mobile phone users.

Figure 11: Example for a Job event shows one event which meets the criteria in the processing section of the PROC report on the mobile phone. The short message (SMSG) is displayed in the message header and the long message in the Message box after the message has been opened on the mobile phone. The category is set to not critical in this example and therefore the message is displayed in blue. Date, Time and System is inserted automatically by MON3B.

CHART definition

```
! Chart definition
CHART,JOB,COND,1
CHART,JOB,RID,M3RPH
CHART,JOB,CAT,3
CHART,JOB,MSG,'JOB consumption'
CHART,JOB,TIT,'AQFT JOB consumption'
CHART,JOB,LGD,false
CHART,JOB,YTIT,'Utilization'
CHART,JOB,XTIT,'Jobs'
CHART,JOB,XPTS,10
CHART,JOB,XAXS,E,ORDER,TOP,1
CHART,JOB,YAXS,1,APPL%,#350D08,Columns,false
```

Defining a chart is a little more complex then defining an event. Again the chart also has a name which is in our example also JOB. Please notice that you can use the same name for different entities. Then a chart requires some more attributes:

- COND is an optional attribute which is also available for EVENTs. It defines a condition whether which is evaluated to 0 (=NO [default]) or 1 (=YES). If it is not specified the chart can still be send via a macro $SENDCHART similar to $SENDEVENT in the previous example. In our example we want to send the chart at the end of each Mintime so we just define a 1 for the send condition of the chart.
- RID is the receiver identifier and again we use M3RPH because we want to send the chart and the events to the same set of mobile phones.
- CAT is fixed this time. We don't want any highlighting so we just define 3.
- MSG is the message title for the snapshot screen.
- TIT is the title of the chart.
- LGD defines whether a legend should be displayed or not. Because we create a chart with only one data item we don't need a legend.
- YTIT is the y-axis title
- XTIT is the x-axis title
- XPTS defines how many data items are displayed on the x-axis. In our example we want to display the 10 jobs with the highest CPU consumption in the system.
- XAXS defines the type of x-axis. Two possibilities exist: an x-axis which contains elements like job as shown in our example or an x-axis which consists of a time series. In our example we want elements (E = jobs). Now we have to define a sort order ORDER. We use TOP for top down and specify a 1 for the y-axis series (there is only 1).

- YAXS defines the y-axis series. We only use 1 series, so the first parameter defines the series number which is 1. The y-axis series legend name would be APPL%. Before we decided not to display it (LGD) but we still define it here. The series color comes next in RGB format. We want to display the series as a column chart without any markers.

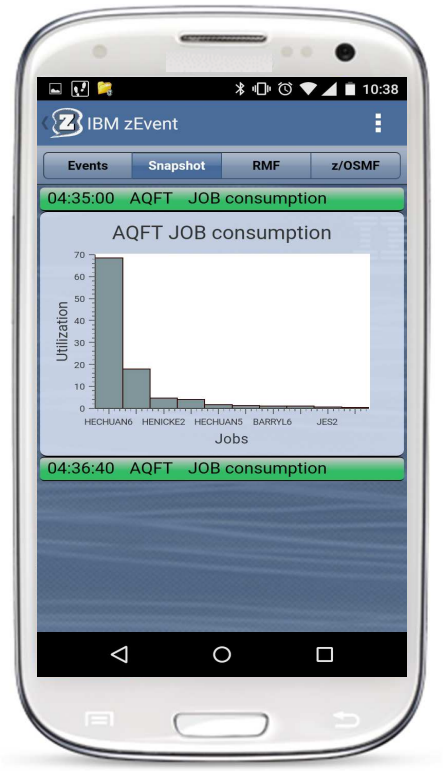

Figure 12: Example for the Job Chart

The nice part of this comprehensive declaration is that we don't have to do much now to create and send the chart. In the PRE processing section we define **#Rc = $InitChart('JOB')** which initializes our chart at the beginning of each Mintime and the reporting section we just add the elements to the charts: **#Rc = $AddElement('JOB',#PRCPJOB,#PRCPEAPP)**. The add element adds each job by name plus the EAPPL% value as y-axis value to the chart. The sorting comes automatically. Also the charts is automatically send for the jobs with the 10 highest EAPPL% values at the end of the Mintime because of the COND statement in the chart declaration.

Figure 12: Example for the Job Chart shows the sample chart on the snapshot screen on the mobile phone for the same RMF Monitor III Mintime than the event shown on Figure 11: Example for a Job event.

REP Definition

The report section specifies which RMF Monitor III reports should be processed. It is possible to process multiple Monitor III reports and also to combine data from multiple reports. In order to do this right it is necessary to layout the sequence of the report definitions. The report processing sections are executed in the sequence they are defined.

The report definition must have the name of the Monitor III report. Please refer to the RMF Programmer's Guide for the correct report name. In our example we extract data from the Monitor III PROC report.

The main definition of the report sections is for the report variables from which data is extracted and the report processing section. Therefore the next attribute after the report name is either where the variable is defined: **T** for the table section of the report or **H** for the header section or whether the following is the report processing section **P**. For T and H the next parameter is the variable name. This must be exactly the name as described in the RMF Programmer's Guide. The next variable describes how the value contained in the variable should be represented. Many table values use K, M, or G to save the space in the ISPF report for large numbers or M, H for time values. If the type value is omitted the value from the variable is used as is. An **N** converts a numeric value to a number which means leading K, M, or G suffixes cause a multiplication of the numeric value by 1000, 1000000 and so forth. A **T** converts a time value to seconds and **X** a numeric value to a hexadecimal representation. For understanding the

possible values types it is recommended to look at the actual RMF Monitor III reports and the RMF Report Analysis manual.

```
REP,PROC,T,PRCPJOB,,$USEPREVIFEMPTY
REP,PROC,T,PRCPASI,X,$NOP
REP,PROC,T,PRCPTYPE,,$NOP
REP,PROC,T,PRCPSVCL,,$NOP
REP,PROC,T,PRCPEAPP,,$NOP
REP,PROC,P,T,$BEGIN
   #resstr = ''
   #psbl = 0
   #cptyp = 0
   if datatype(#PRCPEAPP,'N') = 1 then #psbl = 1
   if strip(#PRCPTYPE) = 'CP' then #cptyp = 1
   if #psbl & #PRCPEAPP > #alert1 then do
     #resstr = 'Job= '#PRCPJOB||' uses '||#PRCPEAPP||' Appl%'
     #re2str = 'JobName= '#PRCPJOB||', '
     #re2str = #re2str||'ASID#= '#PRCPASI||', '
     #re2str = #re2str||'Service Class = '||#PRCPSVCL||', '
     #re2str = #re2str||'Application Utilization = '||,
               #PRCPEAPP||'%'
     #rescat = 4
     if #PRCPEAPP > #alert2 Then #rescat = 2
     if #PRCPEAPP > #alert3 Then #rescat = 1
     #i = #resl.0+1
     #resl.0 = #i
     #resl.#i = #resstr
     #re2l.#i = #re2str
     #recl.#i = #rescat
   end
   if #cptyp then #Rc = $AddElement('JOB',#PRCPJOB,#PRCPEAPP)
$END
```

The table of RMF Monitor III report is presented in a large array to the MON3B batch program. That means processing this table means that each row is being examined. For each row the table variables which are defined in the control section of the job control are extracted and then the processing section of the report is executed for each row.

Sometimes it is possible that information in the report tables spread across multiple lines. On subsequent lines some information is left blank which increases the readability of the report. For example on the PROC may display two lines per job or address space if the job or address space uses regular CPs and zIIPs. An example is shown in Figure 13: Example of Job on the PROC report for the RMF Monitor III data gatherer address space which uses regular CPs and zIIPs.

```
            Service  CPU  DLY USG EAppl  ----------- Holding Job(s) -----------
Jobname  CX Class    Type  %   %    %     %  Name      %  Name      %  Name

RMFGAT   SO SYSSTC   CP     0   0 0.411
                     IIP    0   0 0.006
```

Figure 13: Example of Job on the PROC report

At this point it must be noted that the tables which are retrieved by the REXX program are already formatted for display. Therefore on consecutive lines some information might be missing. The MON3B REXX program provides an internal attribute ($USEPREVIFEMPTY) which allows to copy a value from the previous line to the current line. This attributes is especially meaningful for name values which are often omitted on subsequent lines in the report table.

The report processing section is denoted by **P** followed by a **T** or **H** which refers to either the table body or the table header. It is possible to have multiple report processing sections for the same report definition. These sections are executed in the sequence they are defined in the job control, except that sections which refer to the header (**H**) are always executed before the table sections (**T**).

REXX Scripts

All REXX processing statement can either consist of just one line or a small script. A script contains multiple lines it enhances the readability and it is required to implement conditional statements and loops. A script always starts with the internal directive **$BEGIN** and the last line of the script must only contain the internal directive **$END**. For executing the script MON3B creates one REXX line which is presented to the REXX INTERPRET statement.

In our example we only use one script for the report section. In order to understand the implementation is it necessary to look at all scripts in the JCL control statements.

PRE processing script

simply defines 3 alert conditions which are used in the report processing section. Then it counts the

```
PRE,$BEGIN
   #alert1 = 30
   #alert2 = 70
   #alert3 = 90
   #resl.0 = 0
   if datatype(#cnt,'N') = 1 Then #cnt = #cnt + 1
   else #cnt = 1
   say left('-',60,'-')
   say 'Interval#: 'right('0000'#cnt,4)
   say left('-',60,'-')
   #Rc = $InitChart('JOB')
$END
```

number of RMF Monitor III Mintimes and writes a line to the job output for debugging purposes. Finally it initializes the job chart.

REP processing script

In our example the report processing section primarily evaluates the EAPPL% value for regular CPs of a job to determine whether the event should be sent and then creates the short and long messages.

```
REP,PROC,P,T,$BEGIN
   #resstr = ''
   #psbl = 0
   #cptyp = 0
   if datatype(#PRCPEAPP,'N') = 1 then #psbl = 1
   if strip(#PRCPTYPE) = 'CP' then #cptyp = 1
   if #psbl & #PRCPEAPP > #alert1 then do
     #resstr = 'Job= '#PRCPJOB||' uses '||#PRCPEAPP||' Appl%'
     #re2str = 'JobName= '#PRCPJOB||', '
     #re2str = #re2str||'ASID#= '#PRCPASI||', '
     #re2str = #re2str||'Service Class = '||#PRCPSVCL||', '
     #re2str = #re2str||'Application Utilization = '||,
               #PRCPEAPP||'%'
     #rescat = 4
     if #PRCPEAPP > #alert2 Then #rescat = 2
     if #PRCPEAPP > #alert3 Then #rescat = 1
     #i = #resl.0+1
     #resl.0 = #i
     #resl.#i = #resstr
     #re2l.#i = #re2str
     #recl.#i = #rescat
   end
   if #cptyp then #Rc = $AddElement('JOB',#PRCPJOB,#PRCPEAPP)
$END
```

Based on the previous defined alert values the category of the event is specified. The processor type is used to determine whether the line should be processed at all (zIIPs are skipped) and whether the EAPPL% value is added to the chart.

POST processing script

```
POST,$BEGIN
   do #i = 1 to #resl.0
     #jobsmsg = #resl.#i
     #longmsg = #re2l.#i
     #jobcatg = #recl.#i
     #Rc = $SENDEVENT(JOB)
     say 'After SENDEVENT: 'time()
   end
$END
```

In the post processing script the content of the steam variables are copied to the variables which have been used in the EVENT definition and the events are sent to the mobile devices. It would also be possible to send the events directly in the report processing section. In our example the use of a steam variable would allow to further process the information which has been obtained from the PROC report for example by adding information from other reports, etc…

4.3 MON3B Control Parameter Reference

This section describes the available control parameter. The general structure of control parameters is depicted in Figure 9: MON3B Processing Concept and an example is described in MON3B Example: Monitoring Job Resource Consumption.

General Definitions

DEBUG,<parameter>[,<parameter>]
Prints information which is helpful to understand the program and to debug the control flow

Parameters:
PRINTPARMS = print input parameters once after startup
PRINTREPS = print Monitor III report data at each Monitor III Mintime
PRINTINFO = print runtime information

INTERPRET,<LONG|SINGLE>
Defines how the REXX statements are presented to the INTERPRET REXX statement

Possible Values:
LONG: this is the default. All rexx statements between a $BEGIN and $END sequence are concatenated to to one REXX string. This allows to program complex IF statements and loops
SINGLE: each control statement is processed individually.
This does not allow to program loops

Processing

<PRE|POST>,rexx control statement
PRE: REXX control statements which are executed at the begin of each Mintime.
The intention is to use this for global initializations.
POST: REXX control statements which are executed at the end of each Mintime.
The intention is to use this to finally process the results and to send events, snapshots and WTOs based on conditions
<rexx> = either a valid rexx statement or REXX control statement sequence
The REXX control sequence is started by a $BEGIN statement and ended by a $END statement

Processing statements can also be placed for REPort, EVENT, CHART, and WTO definitions.

Report Definition

The report definition defines the RMF Monitor III report which is used to extract data from. The report is presented to the program in a table format. The table is already in a format very similar to what is displayed in the Monitor III reporter session. Numbers are often abbreviated by K,M, or G in order to

save space on the ISPF screen and time values are often marked with a S, M, and H scale. The data can be accessed by Monitor III variables of the report. The variables are described in the RMF Programmer's guide. Therefore the name of the report must be the same as described in the programmers' guide.

It is possible to have multiple report definitions. They are processed in the order they are defined. Each report is presented to the program row by row. Therefore the content of the variables changes for each row. Information which should be kept for later processing must be saved in control variables. Please refer to MON3B Example: Monitoring Job Resource Consumption for an example.

LOCSYS,systemname
Name of the local system (usually not necessary to specify)
REPSYS,systemname
Name of the system from which the report should be obtained. This is only required if it should be different from the local system.

REP,<report>,<H|T>,<field>,<type>,<condition>
<report> = report name, e.g. PROC, SYSSUM, CPC, ...
<H|T> = H: header field, T: table field (see Programmers' Guide)
<field> = field name, use a valid name from the Programmers' Guide
<type> = blank (use as is) | N (convert to number) | T (convert to time value in seconds)
<condition>= a condition on how to fill the data
for example: $**USEPREVIFEMPTY** - use last value if empty

REP,<report>,P,<H|T>,<rexx control statement>
<report> = report name, e.g. PROC, SYSSUM, CPC, ...
P = processing statement
<H|T> = H: header field, T: table field
<rexx> = rexx control statement which is either executed after all header variables are read for a report or after all table variables for a table row have been read

Event definition

Specifies the events which should be sent to the smartphone. It is possible to specify multiple events. All events have a unique name. It is also possible to specify no event and either only CHARTs or WTOs (or nothing at all but then also nothing is sent).

EVDSN,datasetname = name of dataset for zEvent exec

EVENT,<name>,SMSG,<message>	= short message	[required]
EVENT,<name>,LMSG,<long message>	= long message	[required]
EVENT,<name>,CAT,<category>	= category: 1,2,3,...	[recommended]
EVENT,<name>,DFRM,<dateformat>	= U|E (U=default)	[optional]
EVENT,<name>,RID,<receiver>	= receiver qualifier or msgid	[required]
EVENT,<name>,UDATA,<key>,<val>	= Userdata key value pair = multiple can be defined	[optional]
EVENT,<name>,COND,<condition>	= when to issue the message this is a valid rexx statement	[optional]

Chart Definition

CHART,<name>,CAT,<category>	= category: 1,2,3,...	[recommended]
CHART,<name>,COL,<color>	= color: red, white, etc ... or #RRBBYY	[optional]
CHART,<name>,MSG,<message>	= chart event message	[required]
CHART,<name>,TIT,<title>	= chart title	[optional]
CHART,<name>,LGD,<legend>	= legend: {true,false,t,f,1,0}	[optional]
CHART,<name>,YTIT,<y-axis title>	= text string	[optional]
CHART,<name>,XTIT,<x-axis title>	= text string	[optional]
CHART,<name>,XAXS,<x-axis type>[,ORDER,{TOP,NO,BOT},<key>]		[required]
<x-axis type>	= T time or E element series	[required]
ORDER	= for element series only	[only for E]
TOP	= highest value to the left	[only for E]
NO	= no sorting just as it comes	[only for E]
BOT	= lowest value to the left	[only for E]
<key>	= yaxs number which is used as sort key	[only for E]
CHART,<name>,XPTS,<x-axis points>	= x-axis points	[required]
CHART,<name>,YAXS,<num>,<title>,<color>,<type>,<marker>		[required]
<num>	= Y series number	
<title>	= Y series legend	
<color>	= Y series color	
<type>	= Lines, Columns	
<marker>	= {true,false,t,f,1,0}	
CHART,<name>,RID,<receiver>	= receiver qualifier	[required]
CHART,<name>,COND,<condition>	= when to issue the chart	[optional]

WTO Definition

One or multiple WTO definitions can be specified. For issuing a WTO it is necessary that the user (and batch program) has the permission to send WTOs to the operator console. The WTO definition is rather simple and the user must create the message text completely on its own.

WTO,<msgid>,MSG,<msgtext>	= WTO definition
WTO,<msgid>,COND,<condition>	= when to issue the WTO