# zEvent Mobile Application

# Administration Guide

*Version 1.0*

# Content:

1 Introduction ....... 4
2 zEvent Administration Dialog ....... 5
2.1 Load/Save ....... 5
2.2 Projects ....... 6
2.3 Users ....... 7
2.4 Groups ....... 8
2.5 Attributes ....... 8
2.6 Rules ....... 9
2.7 Create ....... 10
2.8 Event ....... 11
2.9 GetIp ....... 11
2.10 Rule test ....... 11
2.11 Event, Chart and JSON ....... 11
3 zEvent Quick Start ....... 12
3.1 Get the Administration Dialog ....... 12
3.2 Connect to IBM Bluemix ....... 12
3.3 Install IBM zEvent App ....... 13
3.4 Define a User in the Administration Dialog ....... 14
3.5 Send Push Message ....... 14
3.6 Save your Data ....... 16
4 Setup the z/OS Message Processing Facility for Notifications ....... 17
4.1 Overview ....... 17
4.2 The zEvent MPF Exit Module MPF4REXX ....... 18
5 Setup RMF Monitor III Batch as Event Provider ....... 20
5.1 Preface ....... 20
5.2 RMF Monitor III Batch and MPF Processing ....... 20
5.3 RMF Monitor III Batch and zEvent API Invocation ....... 23
5.4 zEvent Notifications with RMF Context ....... 25
5.5 zEvent Notifications with z/OSMF Context ....... 27
6 HTTPS Setup for zEvent Notifications ....... 29
6.1 Overview ....... 29
6.2 AT-TLS Configuration for zEvent ....... 29
7 zEvent API ....... 32
7.1 Parameters for Rule Selection ....... 32
7.2 Parameters to overwrite attributes ....... 33
7.3 Other parameters ....... 34
8 Push Interface ....... 35
8.1 Events Section ....... 37
8.2 Charts Section ....... 38

# Figures:

Figure 1: zEvent main menu ....... 5
Figure 2: Load panel ....... 6
Figure 3: Project settings ....... 7
Figure 4: User settings ....... 8
Figure 5: Attributes panel ....... 9
Figure 6: Rule settings ....... 10
Figure 7: zEvent API exec creation panel ....... 10
Figure 8: zEvent notification flow ....... 17
Figure 9: ZEVENT System REXX invocation ....... 18
Figure 10: System REXX parmlib member ....... 19
Figure 11: ERBM3B exit procedures ....... 20
Figure 12: ERBM3B example – RMF Monitor III CPC Report ....... 21
Figure 13: Sample rule for MPF processing ....... 22
Figure 14: Sample attributes for MPF processing ....... 22
Figure 15: Sample event for MPF processing ....... 23
Figure 16: Sample Rule for zEvent API processing ....... 23
Figure 17: Sample Attributes for zEvent API processing ....... 24
Figure 18: Invocation of the zEvent API via RMF Monitor III Batch ....... 24
Figure 19: zEvent notifications – RMF context ....... 25
Figure 20: Connection Label and notification source ....... 26
Figure 21: RMF context launch ....... 26
Figure 22: zEvent notifications – z/OSMF context ....... 27
Figure 23: z/OSMF Dashboards for userid *ibmdev* ....... 27
Figure 24: z/OSMF context launch ....... 28
Figure 25: PAGENT parameter example ....... 29
Figure 26: mybluemix  zEvent certificate chain ....... 30
Figure 27: Browser certificates (Firefox example) ....... 31
Figure 28: Keyring Database – Certificate List ....... 31
Figure 29: Sample *Event* and *Snapshot* tab ....... 36

# 1 Introduction

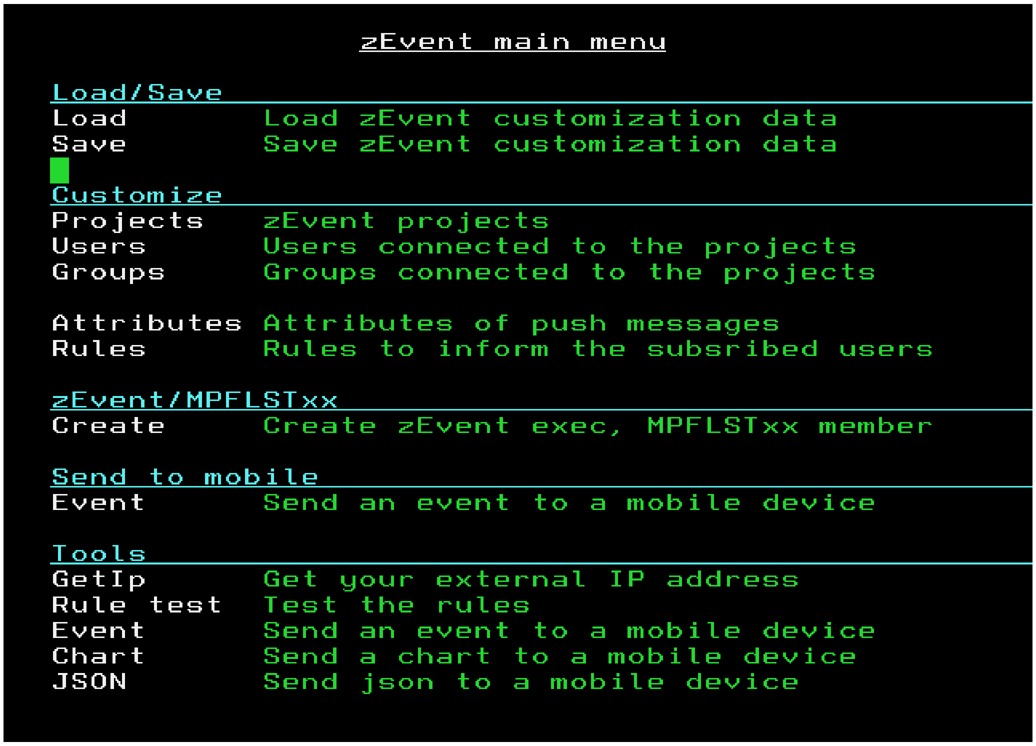
In addition to the IBM zEvent mobile application, a server-side facility is required to manage the submission of the zEvent notifications. This facility is the zEvent administration dialog, which is described in this document. The administration dialog allows to customize and automate all the parameters that are part of the notifications:

- Define users or groups of users that receive the notifications.
- Define rules that trigger a notification to specific users or groups.
- Configure the content of the notification.
- Enable z/OS console message ids to trigger notifications.

Once the user has performed the required configurations, the administration dialog generates a System REXX exec. This exec then connects to the platform specific push services, which are the Google Cloud Messaging Services (GCM) for Android or the Apple Push Notification Services (APNS) for iOS, in order to deliver the notifications for a certain event.

# 2 zEvent Administration Dialog

The zEvent administration dialog (cf. Figure 1) allows to customize zEvent data like users, devices, rules etc. and build the zEvent API REXX exec which can be used to send push messages to a mobile device. The zEvent exec (created by the administration dialog) can be called by another exec or an MPF exit.



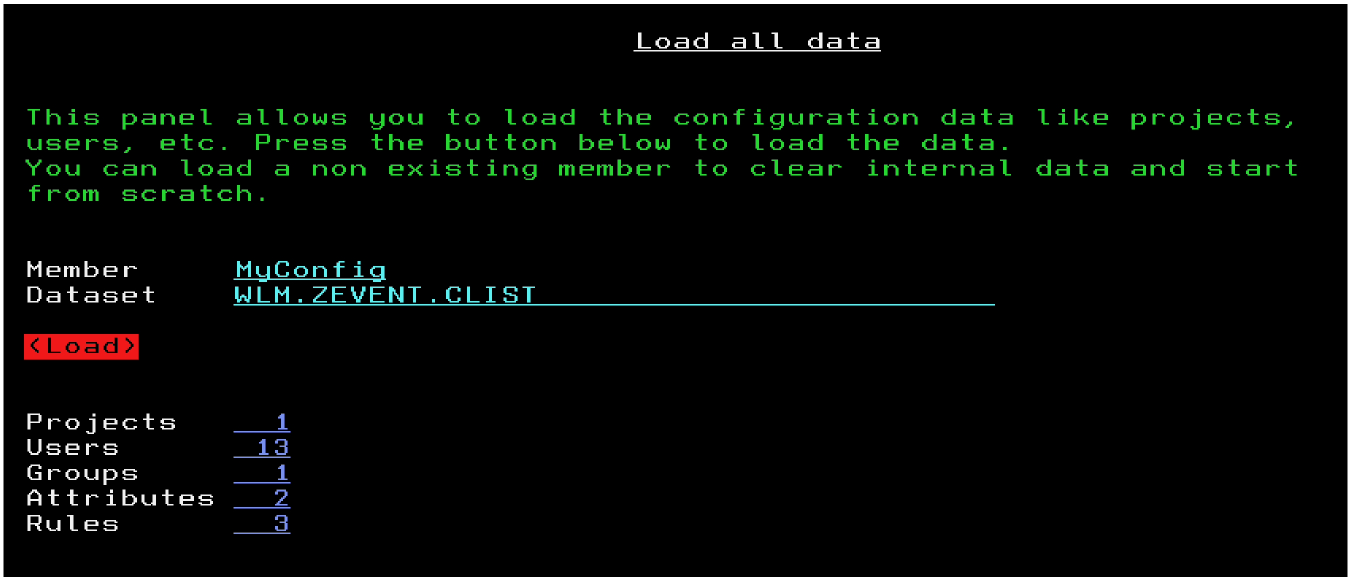
```
                   zEvent main menu

 Load/Save
 Load         Load zEvent customization data
 Save         Save zEvent customization data

 Customize
 Projects     zEvent projects
 Users        Users connected to the projects
 Groups       Groups connected to the projects

 Attributes Attributes of push messages
 Rules        Rules to inform the subsribed users

 zEvent/MPFLSTxx
 Create       Create zEvent exec, MPFLSTxx member

 Send to mobile
 Event        Send an event to a mobile device

 Tools
 GetIp        Get your external IP address
 Rule test    Test the rules
 Event        Send an event to a mobile device
 Chart        Send a chart to a mobile device
 JSON         Send json to a mobile device
```

Figure 1: zEvent main menu

## 2.1 Load/Save

The customization data you enter in the zEvent dialog, can be loaded/saved to a JSON encoded member of a partitioned data set. You can have one or more such members that can be seen as configuration profiles. These configuration members are used by the dialog exec, only.
The *DIALOG* exec will remember the most recently used configuration member and tries to load this member when the dialog is entered the next time. After a configuration member has been loaded, the number of users, rules, etc. is shown in the lower part of the panel (cf. Figure 2).
If you want to create a new configuration member, you can load an existing member and save it with a different name. To start from scratch you can load a non-existing member. The dialog exec will tell you that this member does not exist but as soon as you save the member it will be created.

You find the load option in the main menu (cf. Figure 1) of the administration dialog in the Load/Save section.



```
                                 Load all data

This panel allows you to load the configuration data like projects,
users, etc. Press the button below to load the data.
You can load a non existing member to clear internal data and start
from scratch.

Member       MyConfig
Dataset      WLM.ZEVENT.CLIST

<Load>

Projects        1
Users          13
Groups          1
Attributes      2
Rules           3
```

Figure 2: Load panel

## 2.2 Projects

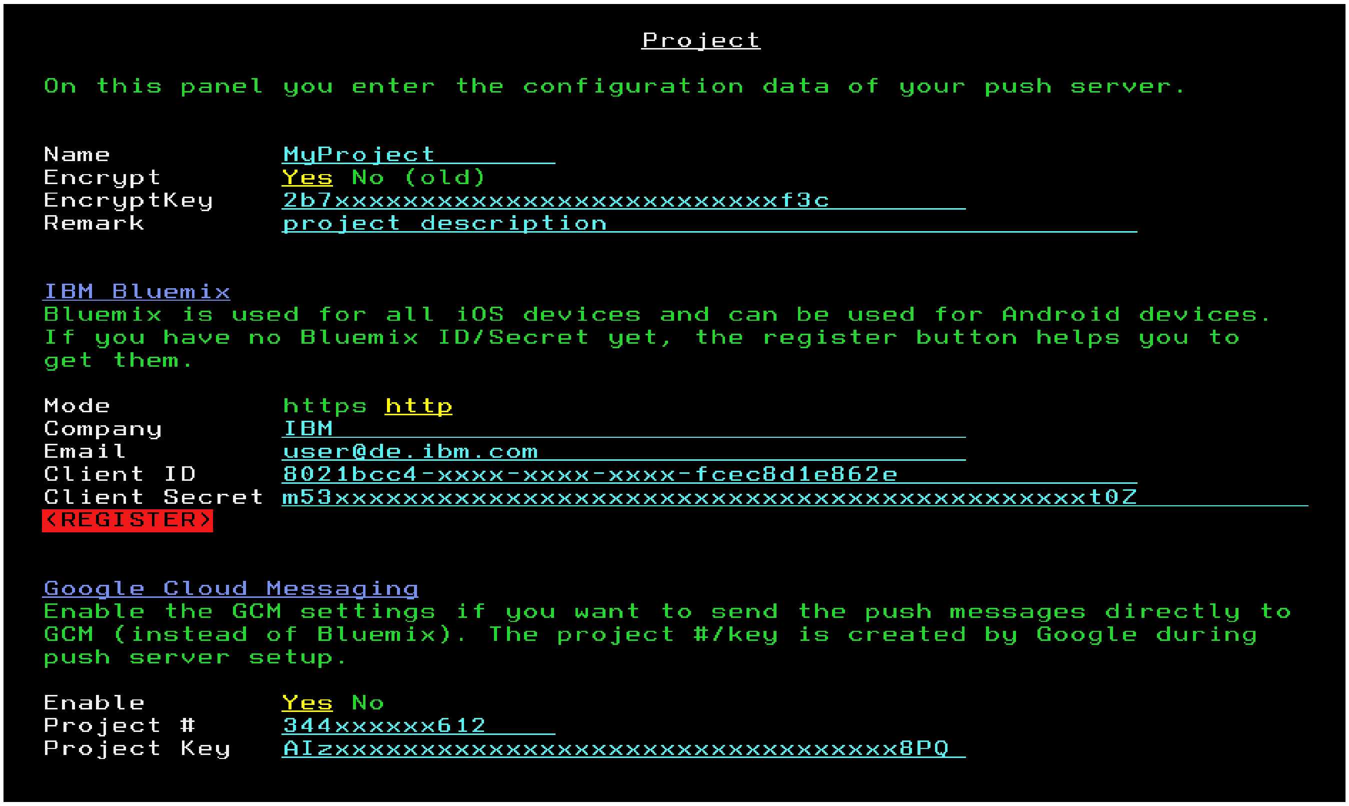
You need to define at least one *Project* that contains the attributes of the push servers. The *Users* (cf. chapter 2.3) which are connected to the same push server will have a reference to the same project.

If you decide to encrypt the messages before you send them to the device, you need to specify an encryption key. This key must be identical with the one that is specified in the zEvent app.

For the push server there are two choices, the IBM Bluemix server and the Google Cloud Messaging (GCM) service. All push messages sent to an Apple device will be pushed using the IBM Bluemix server but for Android devices you have the choice to use Bluemix or GCM.
To use the Bluemix server, you need to register once to get a *Client ID* and a *Client Secret*. Each push message will get authorized using this information. After specifying the company name and email address you can use the register button to get the client id/secret.
It is highly recommended to use the HTTPS protocol for the communication to the Bluemix server. The HTTPS protocol can only be used if the AT-TLS setup has been performed on the z/OS system (cf. chapter 6 for details).

If you want to use GCM directly, you need to have a Google account and set up your own push server. During this setup you will receive a **project number** and a **project authorization key**.
For more details refer to the following URL: https://developers.google.com/cloud-messaging/

The project settings example below (cf. Figure 3) shows all settings which are possible for a project. As long as there is no encryption key specified in your zEvent app, you need to select *Encrypt*=No. If you did not setup your own Google push server ( you don’t need one ) you need to select *Enable*=No in the Google Cloud Messaging section.

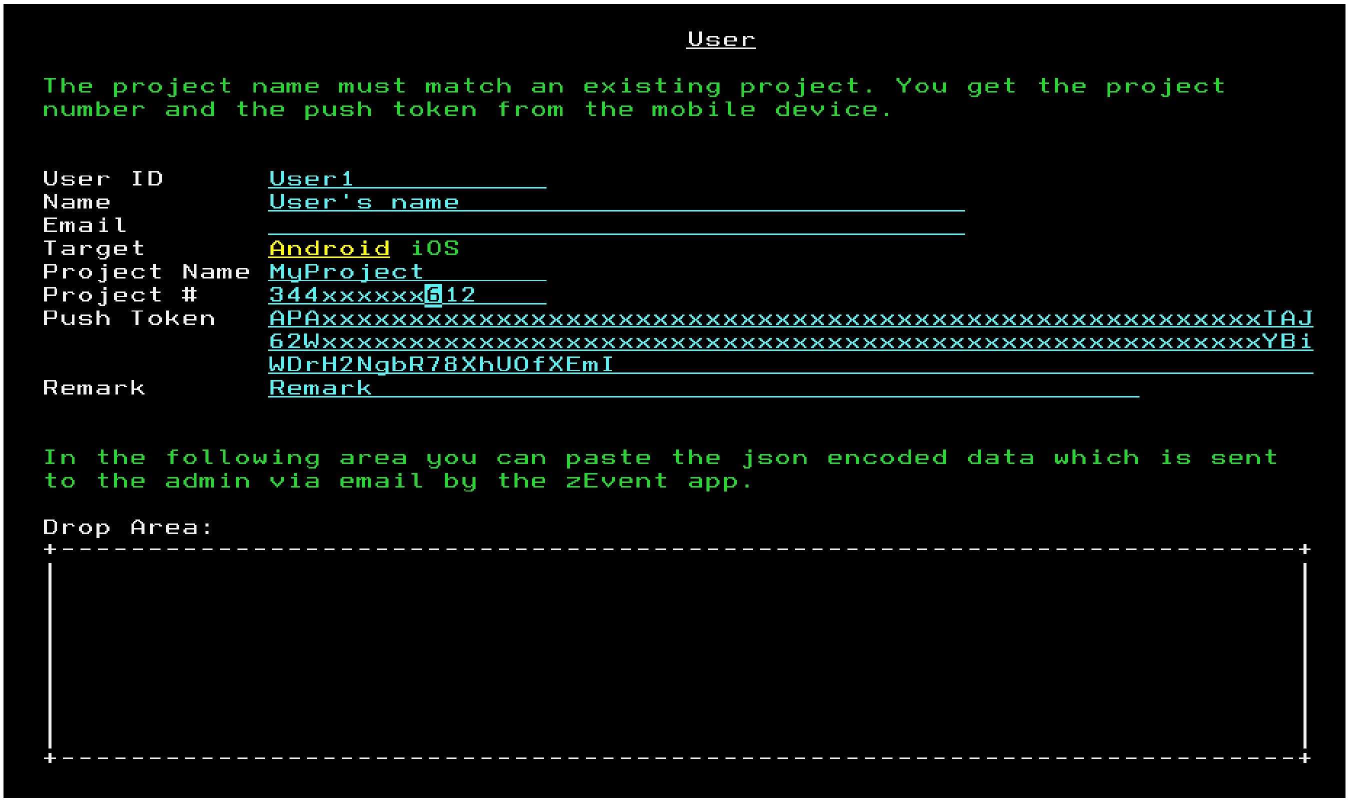


```
                                    Project

  On this panel you enter the configuration data of your push server.


  Name          MyProject
  Encrypt       Yes No (old)
  EncryptKey    2b7xxxxxxxxxxxxxxxxxxxxxxxxxxxxf3c
  Remark        project description


  IBM Bluemix
  Bluemix is used for all iOS devices and can be used for Android devices.
  If you have no Bluemix ID/Secret yet, the register button helps you to
  get them.

  Mode          https http
  Company       IBM
  Email         user@de.ibm.com
  Client ID     8021bcc4-xxxx-xxxx-xxxx-fcec8d1e862e
  Client Secret m53xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxt0Z
  <REGISTER>


  Google Cloud Messaging
  Enable the GCM settings if you want to send the push messages directly to
  GCM (instead of Bluemix). The project #/key is created by Google during
  push server setup.

  Enable        Yes No
  Project #     344xxxxxx612
  Project Key   AIzxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8PQ
```

Figure 3: Project settings

## 2.3 Users

You need to define a user for each device. A user needs to be connected to a project by specifying the name of an existing project.
The zEvent app is sending an email to the administrator when the connect button in the push settings of the app is used. In this email there is a JSON encoded string which can be pasted into the **Drop Area** of the user panel (cf. Figure 4). Therefore it is not necessary to type the push token by hand.



```
                                  User

  The project name must match an existing project. You get the project
  number and the push token from the mobile device.


  User ID      User1
  Name         User's name
  Email
  Target       Android iOS
  Project Name MyProject
  Project #    344xxxxxx612
  Push Token   APAxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxTAJ
               62WxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxYBi
               WDrH2NgbR78XhUOfXEmI
  Remark       Remark


  In the following area you can paste the json encoded data which is sent
  to the admin via email by the zEvent app.

  Drop Area:
  +-----------------------------------------------------------------------+
  |                                                                       |
  |                                                                       |
  |                                                                       |
  |                                                                       |
  |                                                                       |
  |                                                                       |
  |                                                                       |
  +-----------------------------------------------------------------------+
```

Figure 4: User settings

## 2.4 Groups

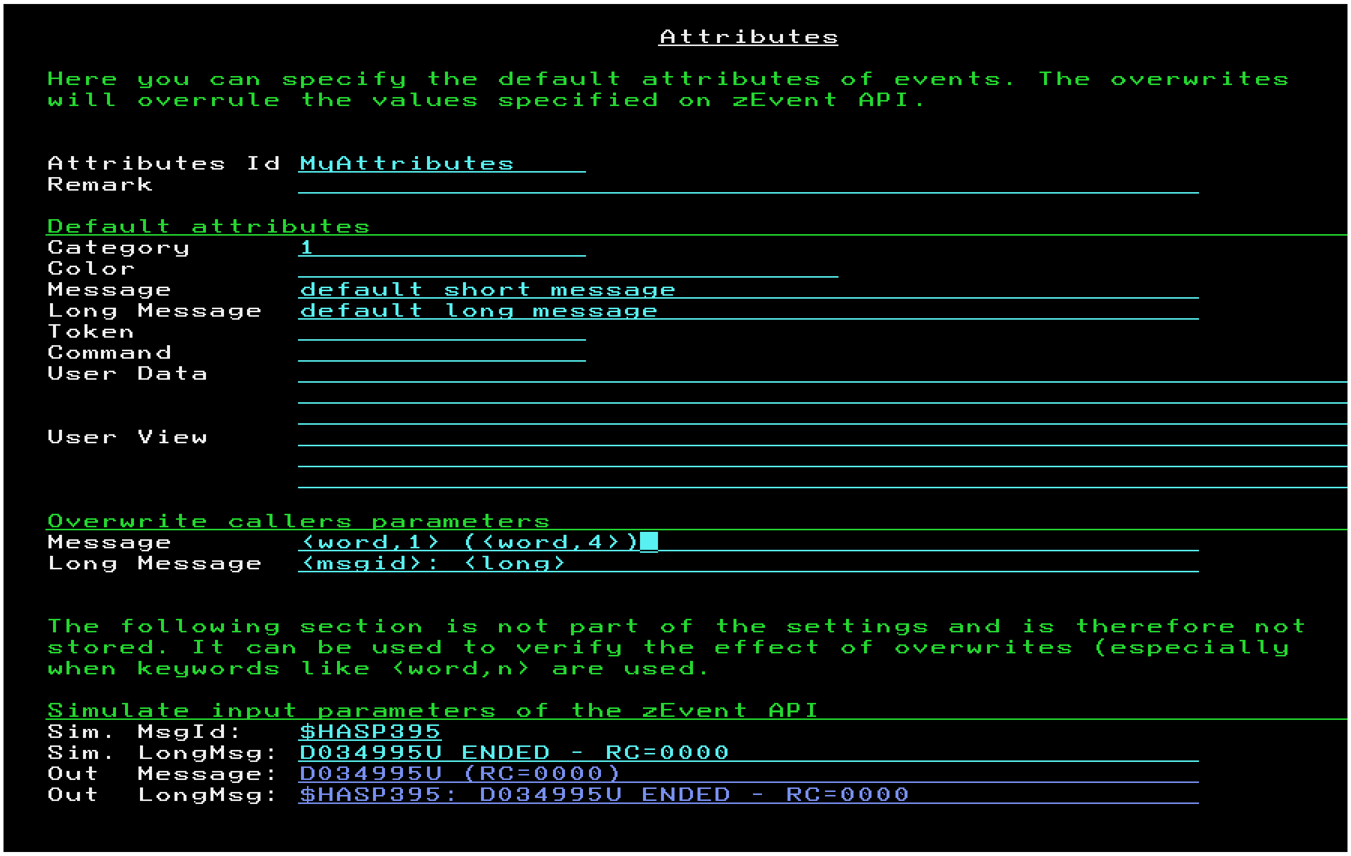
User groups consist of one or more users.

## 2.5 Attributes

The zEvent exec created by the dialog has parameters like category, color, etc. to tailor the push message. The attributes provide defaults for the not specified parameters (Note: Token and Command are not implemented, yet).
In the User data field you can enter JSON formatted key/value pairs which are shown in the zEvent app without interpreting their content (e.g. *{"key1":"val1"}, {"key2":"val2"}*).
The *User View* field allows to specify link information which points for example to a specific dashboard on the z/OSMF tab of the zEvent app. For an example how to use *User View*, see Chapter 5.4 and 5.5.
Additionally it is possible to overwrite the short and the long message by using the overwrite fields of the attributes. Within the overwrite fields you can use one or more of the following keywords in your text:

| Keyword | Meaning |
|---|---|
| <MSGID> | Message id of the long message specified on zEvent API |
| <LONG> | Complete long message from API |
| <LEFT,nn> | The first nn characters of the long message |
| <RIGHT,nn> | The last nn characters of the long message |
| <WORD,n> | The n th word of the long message |
| <PART,s,n> | N characters from long message starting at position s |

The last section of the panel allows to simulate a call to the zEvent API with a user defined message. This allows to verify the effect of the attributes. In the example below (cf. Figure 5) you can see that the overwrite section of the attributes has been used to customize a console message before it is sent to the mobile device.



```
                              Attributes

Here you can specify the default attributes of events. The overwrites
will overrule the values specified on zEvent API.


Attributes Id MyAttributes
Remark

Default attributes
Category      1
Color
Message       default short message
Long Message  default long message
Token
Command
User Data



User View




Overwrite callers parameters
Message       <word,1> (<word,4>)
Long Message  <msgid>: <long>


The following section is not part of the settings and is therefore not
stored. It can be used to verify the effect of overwrites (especially
when keywords like <word,n> are used.

Simulate input parameters of the zEvent API
Sim. MsgId:   $HASP395
Sim. LongMsg: D034995U ENDED - RC=0000
Out  Message: D034995U (RC=0000)
Out  LongMsg: $HASP395: D034995U ENDED - RC=0000
```

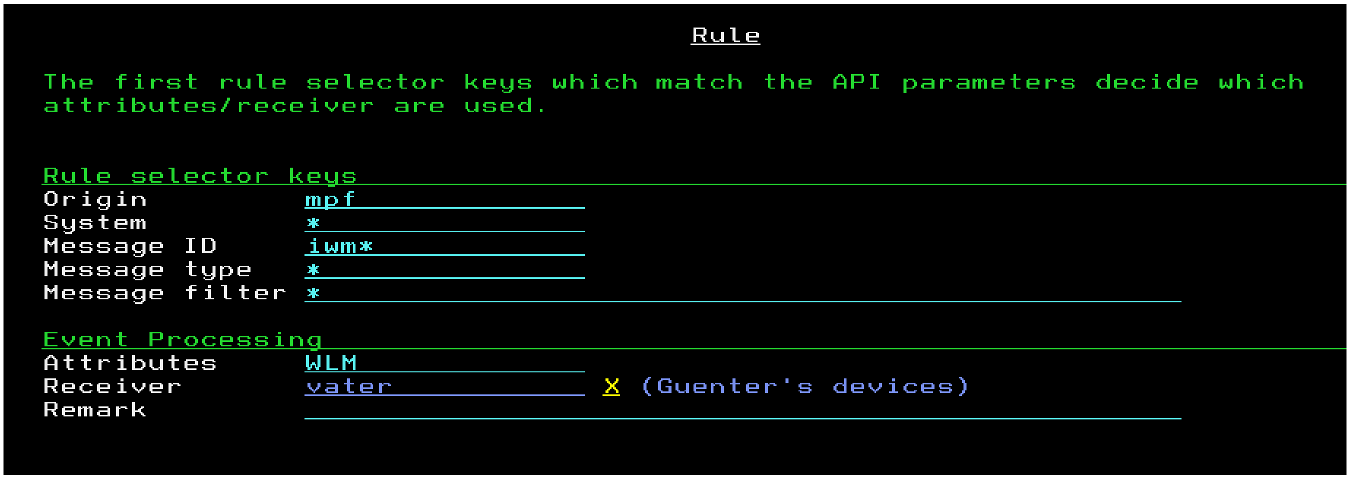
Figure 5: Attributes panel

## 2.6 Rules

The Rules (cf. Figure 6) allow to define which users are the receivers of the individual messages. If the parameters of the zEvent exec match a rule, a push message will be sent to the specified receiver using the specified attributes. If there is an asterisk in a field, the rule is considered as matching for any zEvent parameter value.

In the *Message filter* field you can enter a string which needs to be part of the long message or you can use one or more of the following keywords to define a more complex filter:

| Keyword | Meaning |
|---|---|
| OR(text) | One of all OR texts needs to exist in the long message |
| AND(text) | All of the AND texts need to exist in the long message |
| NOT(text) | The NOT text must not exist in the long message |



```
                              Rule

 The first rule selector keys which match the API parameters decide which
 attributes/receiver are used.


 Rule selector keys
 Origin         mpf
 System         *
 Message ID     iwm*
 Message type   *
 Message filter *

 Event Processing
 Attributes     WLM
 Receiver       vater            X (Guenter's devices)
 Remark
```

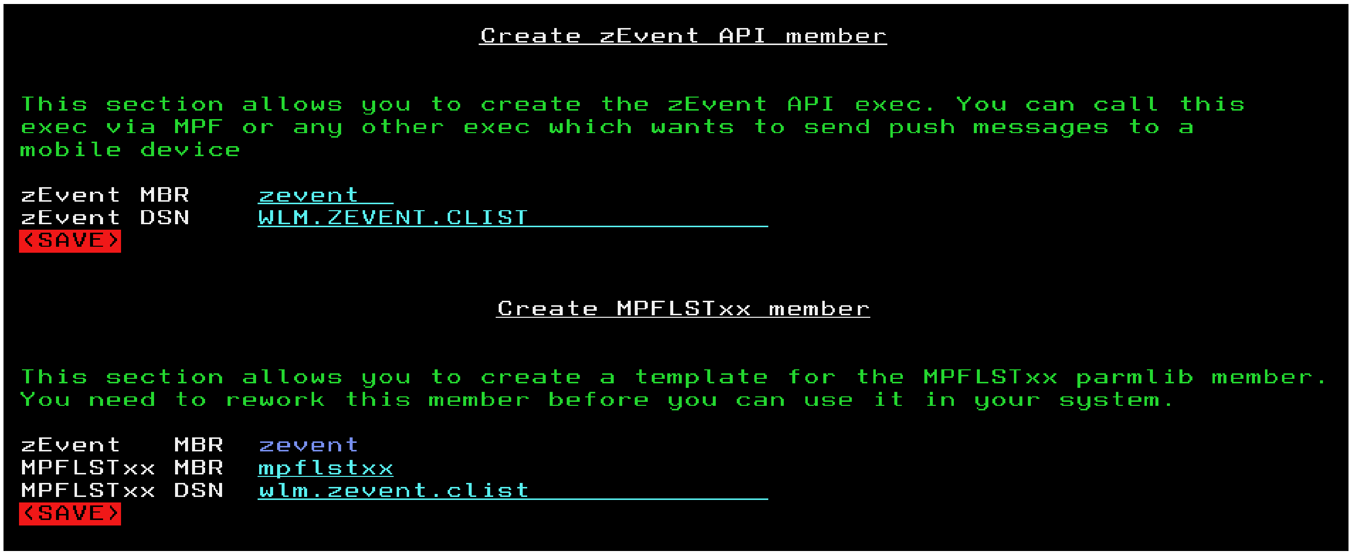
Figure 6: Rule settings

## 2.7 Create

After the configuration is complete, you can create the zEvent API exec (cf. Figure 7). This exec can be invoked to send push messages based on caller's parameters. All the rules, attributes, etc are stored in this exec, so caller's parameters are used to find a matching rule and choose the attributes and receivers.

**Note:** If you change settings in the dialog ( users, rules, etc ) and just save the changes to the JSON file by using the load/save option of the dialog, the changes have no effect in the zEvent API exec until you create a new zEvent API exec.

If you want to use the zEvent exec as an MPF exit you can also build a template for the MPFLSTxx member which is based on your rules. You need a MPFLSTxx member which forwards the messages you are interested in to the zEvent exec. The zEvent exec will then search for a matching rule and send a push message to the receivers (for more details see chapter 4).



```
                       Create zEvent API member


This section allows you to create the zEvent API exec. You can call this
exec via MPF or any other exec which wants to send push messages to a
mobile device

zEvent MBR     zevent
zEvent DSN     WLM.ZEVENT.CLIST
<SAVE>


                        Create MPFLSTxx member


This section allows you to create a template for the MPFLSTxx parmlib member.
You need to rework this member before you can use it in your system.

zEvent   MBR   zevent
MPFLSTxx MBR   mpflstxx
MPFLSTxx DSN   wlm.zevent.clist
<SAVE>
```

Figure 7: zEvent API exec creation panel

## 2.8 Event

The event option allows you to send a message to one or more users. This is helpful to test your setup before creating the zEvent API exec. After sending a push message to a device you can switch to the TRACE panel at the top of the screen to see what data has been sent to the push server and what response was returned.

## 2.9 GetIp

If you choose to push your messages directly to your Google push server, you might want to use the optional security feature from Google which allows only your hosts to push to your push server. To configure this security feature, you need the IP address of the host(s) which sends the push messages to GCM. That means, only registered ip addresses are allowed to push messages. To help you to find out your external IP address, the GetIp option tries to consult an internet web site which returns your external IP address.
When your internet gateway uses different ip addresses you may get different ip addresses when you use this function multiple times. In this case you need to register all your ip addresses in the setup of the Google push server.

## 2.10 Rule test

The Rule test allows to simulate zEvent API calls and you can verify if the selected rule is the one that you expect. Furthermore you can see if the defaults and the overwrites work as you expected. This is especially useful if you use keywords in the *Message filter* field of rules or in the overwrite section of the *Attributes* panel.

## 2.11 Event, Chart and JSON

The Event, Chart and JSON option allow you to send event messages and charts to a mobile device. This is for testing purposes. You can switch on/off the encryption and you can check the trace to see the communication data.

# 3 zEvent Quick Start

By following the steps in this chapter you should be able to send your first message to the zEvent app.

## 3.1 Get the Administration Dialog

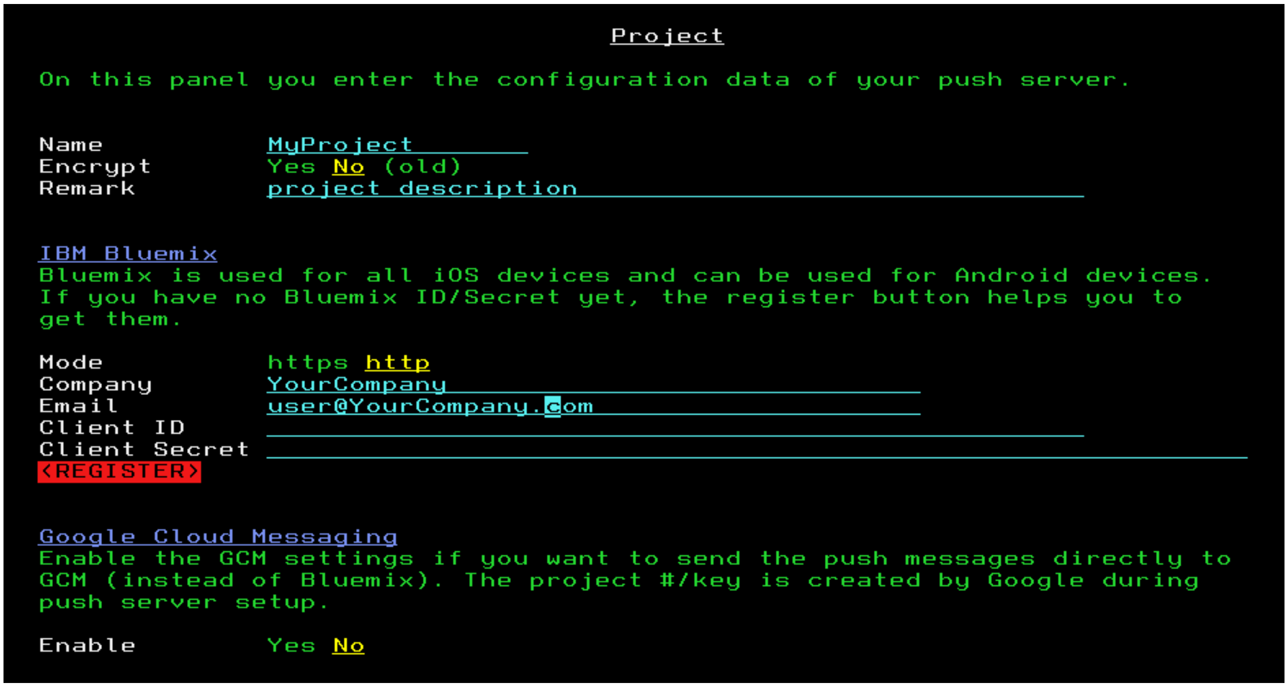
To send push messages from z/OS to the zEvent app, you need the zEvent Administration Dialog. The administration dialog is an ISPF application which allows you to define projects, users, etc. and to create a zEvent API exec. You can download the administration dialog from the zEvent home page and upload it to your host system:

1. Download the file IBMDEV.ZEVENT.EXEC.VxRx (binary file) from the zEvent website (www.ibm.com/systems/z/os/zos/features/zevent/)
2. Upload (ftp) the file in binary mode to a preallocated host dataset in fixed block 80 format
3. Receive from this dataset into a dataset of your choice:
   TSO RECEIVE INDATASET(‘hlq.ZEVENT.EXEC.VxRx’)

## 3.2 Connect to IBM Bluemix

Since your network settings need to allow outbound connections to the internet, the first step is to check if communication to the IBM Bluemix server is possible:

1. Execute the DIALOG exec
2. Select the *Projects* option and edit (E) the predefined entry
3. Define a name for your project
4. Leave **Encrypt=No**, **Mode=http** and Google Cloud Messaging **Enable=No** for now
5. To get a client id/secret, provide your company’s name and your email address and select *REGISTER*. This will fill the client id/secret fields:



```
                                  Project

 On this panel you enter the configuration data of your push server.


 Name           MyProject
 Encrypt        Yes No (old)
 Remark         project description


 IBM Bluemix
 Bluemix is used for all iOS devices and can be used for Android devices.
 If you have no Bluemix ID/Secret yet, the register button helps you to
 get them.

 Mode           https http
 Company        YourCompany
 Email          user@YourCompany.com
 Client ID
 Client Secret
 <REGISTER>


 Google Cloud Messaging
 Enable the GCM settings if you want to send the push messages directly to
 GCM (instead of Bluemix). The project #/key is created by Google during
 push server setup.

 Enable         Yes No
```

6. Go back to the main menu (F3)

## 3.3 Install IBM zEvent App

In the previous step you verified if you are able to connect to IBM Bluemix. Now you need a mobile device which will receive your push messages.

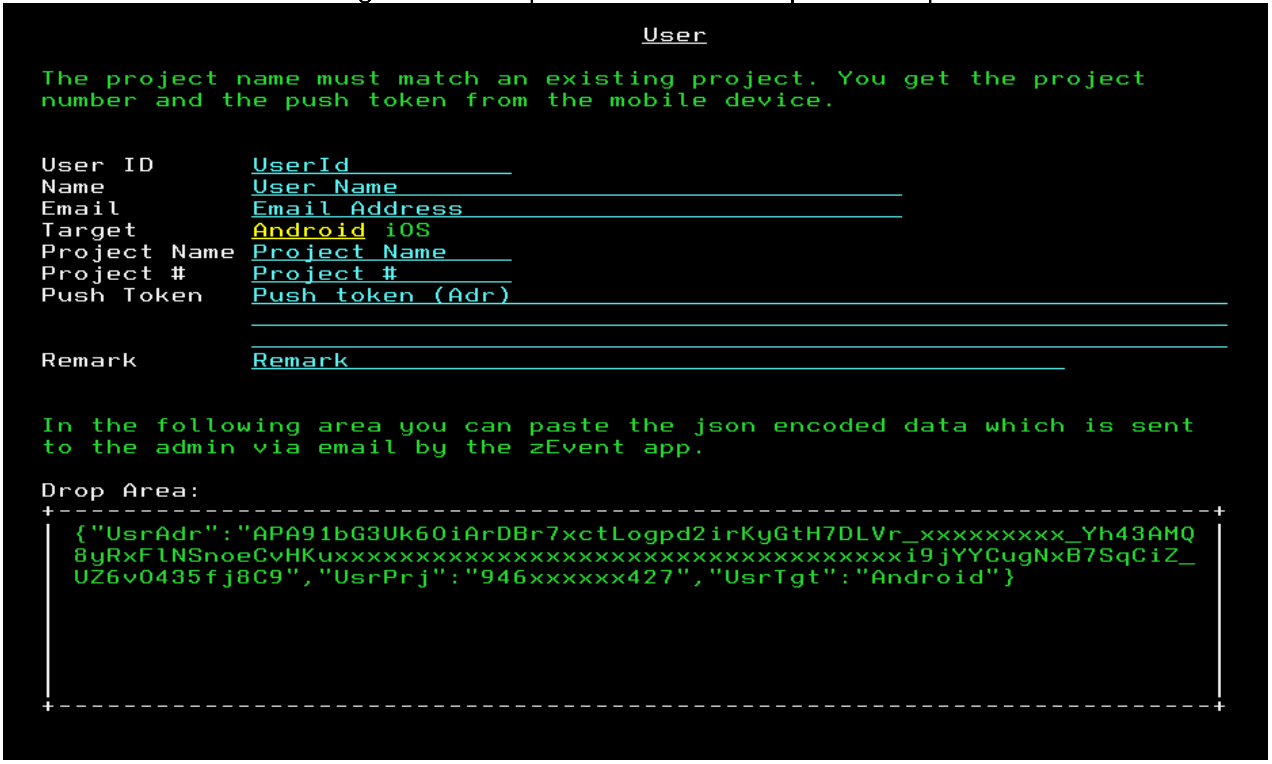
Install the zEvent APP which can be found in the app stores of Apple and Google.

1. Start the zEvent app
2. Navigate to the *Settings* menu
3. Select the *Push* option
4. Android only: Enter the *Project #* 344593236612 manually or use *QR Scan* with the following QR-Code to get the project number of IBM Bluemix :
5. Enter (at least) the admin email of the person who has access to the zEvent Administration Dialog
6. Use Connect and select your email client in the selection dialog. If your email client is configured correctly, the admin should get an email with the device data
7. Open the received email because the included data is needed in the next step

### 3.4 Define a User in the Administration Dialog

The zEvent App sends an email to the administrator with the required information to create a user.

1. Go back to your Administration Dialog on the host
2. Select the **Users** option and edit (E) the predefined entry
3. Define a **UserId** for the new user
4. The email which has been sent to the administrator has a JSON encoded string with the user information. Paste this string into the drop area on the users panel and press <Enter>:



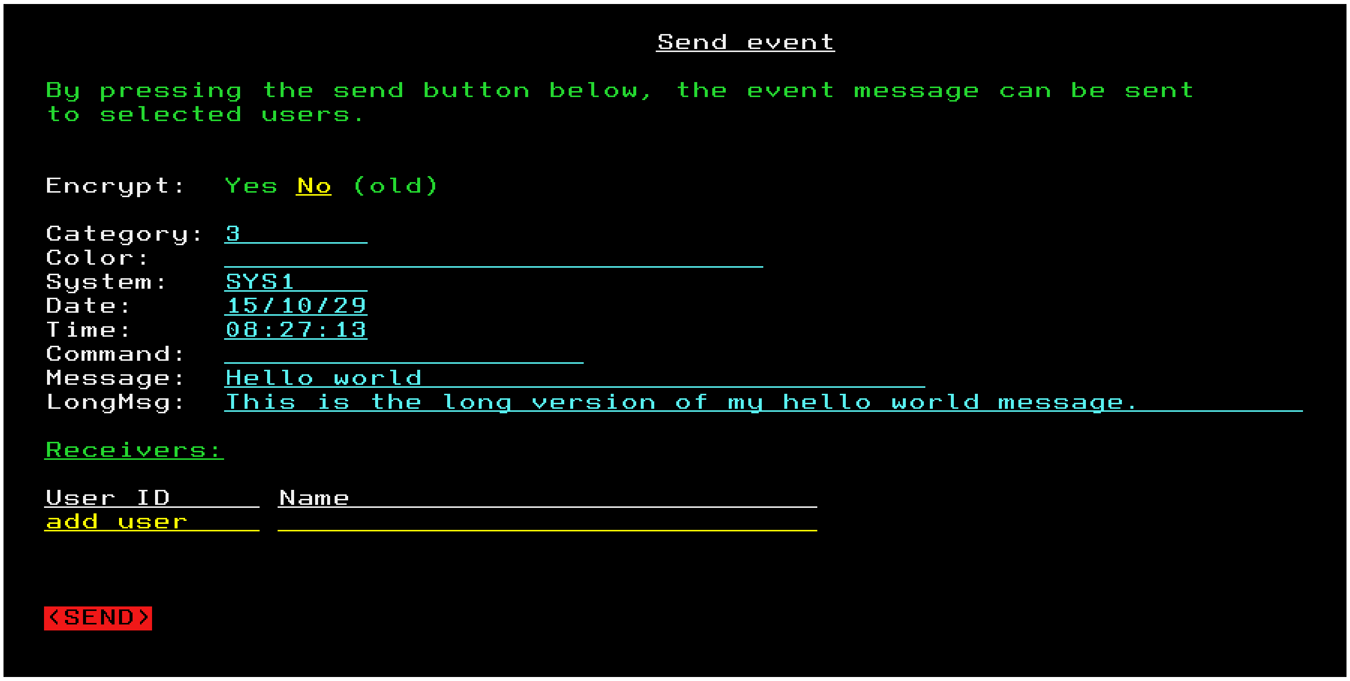
```
                                User

 The project name must match an existing project. You get the project
 number and the push token from the mobile device.


 User ID       UserId
 Name          User Name
 Email         Email Address
 Target        Android iOS
 Project Name  Project Name
 Project #     Project #
 Push Token    Push token (Adr)



 Remark        Remark


 In the following area you can paste the json encoded data which is sent
 to the admin via email by the zEvent app.

 Drop Area:
 +------------------------------------------------------------------------+
 | {"UsrAdr":"APA91bG3Uk6OiArDBr7xctLogpd2irKyGtH7DLVr_xxxxxxxxx_Yh43AMQ  |
 | 8yRxFlNSnoeCvHKuxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxi9jYYCugNxB7SqCiZ_  |
 | UZ6v0435fj8C9","UsrPrj":"946xxxxxx427","UsrTgt":"Android"}             |
 |                                                                        |
 |                                                                        |
 |                                                                        |
 |                                                                        |
 |                                                                        |
 +------------------------------------------------------------------------+
```

5. Go back to the main menu (F3)

### 3.5 Send Push Message

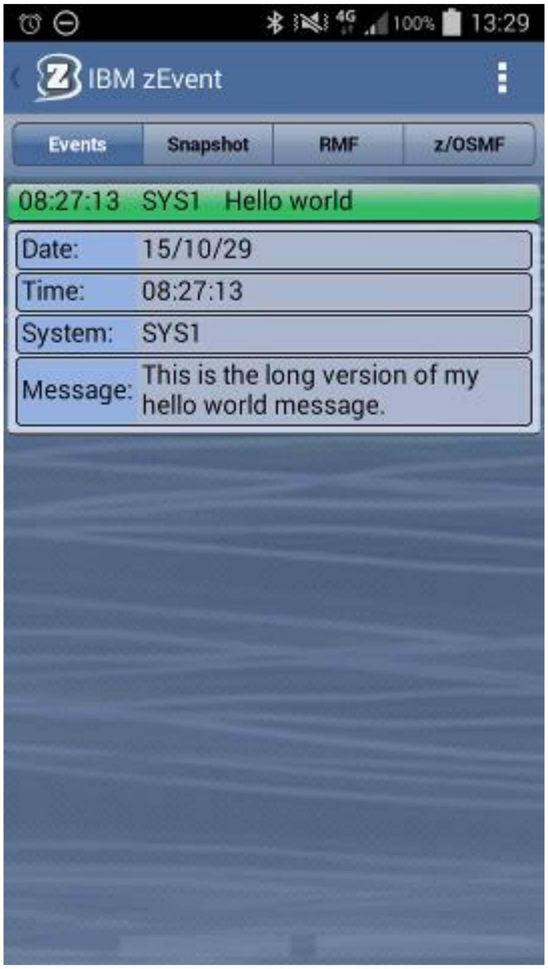
Now you have the minimal setup to be able to send push messages to your mobile device. Navigate to the *Event* option in the *Send to mobile* section.

Fill in the Message and LongMsg field, add a user (<Enter> on *add user* line ) and *SEND* the message to your mobile device.



```
                                  Send event

By pressing the send button below, the event message can be sent
to selected users.

Encrypt:  Yes No (old)

Category: 3
Color:
System:   SYS1
Date:     15/10/29
Time:     08:27:13
Command:
Message:  Hello world
LongMsg:  This is the long version of my hello world message.

Receivers:

User ID     Name
add user

<SEND>
```

Now you should see your push message in the *Events* tab:



08:27:13 SYS1 Hello world

Date: 15/10/29

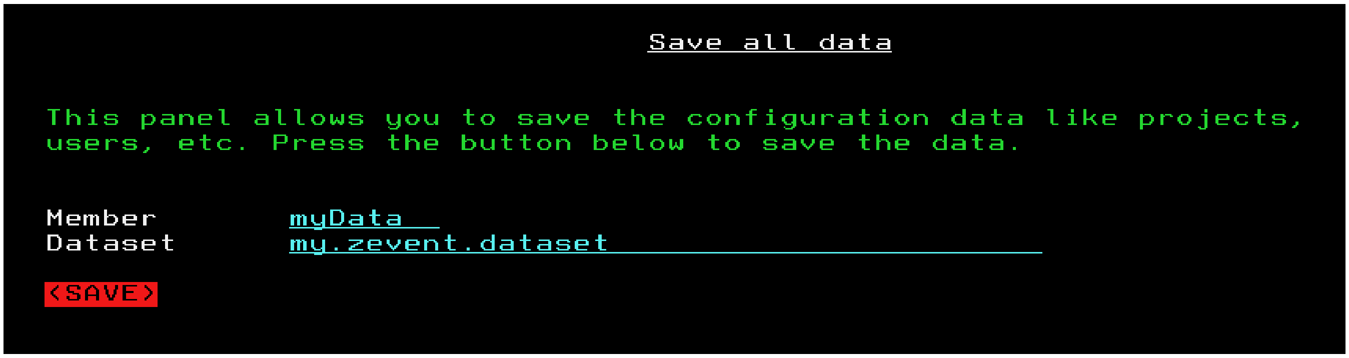
Time: 08:27:13

System: SYS1

Message: This is the long version of my hello world message.

### 3.6 Save your Data

The configuration data you entered in the administration dialog can be saved in a JSON encoded member. The **save** option can be found in the **Load/Save** section of the main menu:



```
                                    Save all data


  This panel allows you to save the configuration data like projects,
  users, etc. Press the button below to save the data.


  Member        myData
  Dataset       my.zevent.dataset

  <SAVE>
```

# 4 Setup the z/OS Message Processing Facility for Notifications

## 4.1 Overview

As described in chapter 2 the submission of zEvent notifications to the mobile devices is performed by a System REXX exec which is generated as result of the specifications provided with the zEvent administration dialog. The caller of the System REXX exec can be any component on the z/OS system that obeys to the calling conventions of the API for this exec.
Figure 8 shows that one potential caller of the REXX exec can be an MPF exit module.
This approach allows that any message on the operator console can trigger the submission of a notification. Hence, the installation might decide to write a specific MPF exit module in order to react on critical system events. But this is not needed with zEvent: it is more convenient to exploit the sample MPF exit module MPF4REXX, which is delivered as part of the zEvent package (.cf chapter 4.2).

You can download the module MPF4REXX module from the zEvent website and upload it to your z/OS system:

1. Download the file IBMDEV.ZEVENT.LINKLIB.VxRx (binary file) from the zEvent website ([www.ibm.com/systems/z/os/zos/features/zevent/](www.ibm.com/systems/z/os/zos/features/zevent/))
2. Upload (ftp) the file in binary mode to a preallocated host dataset in fixed block 80 format
3. Receive from this dataset into a dataset of your choice:
   TSO RECEIVE INDATASET(‘hlq.ZEVENT.LINKLIB.VxRx’)

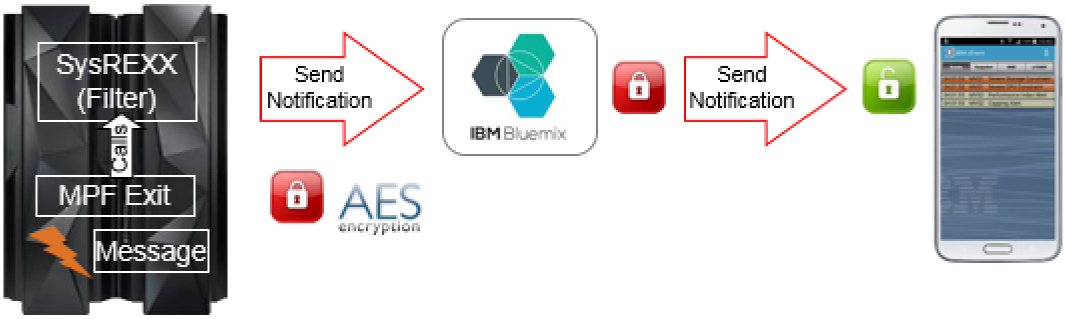


Figure 8: zEvent notification flow

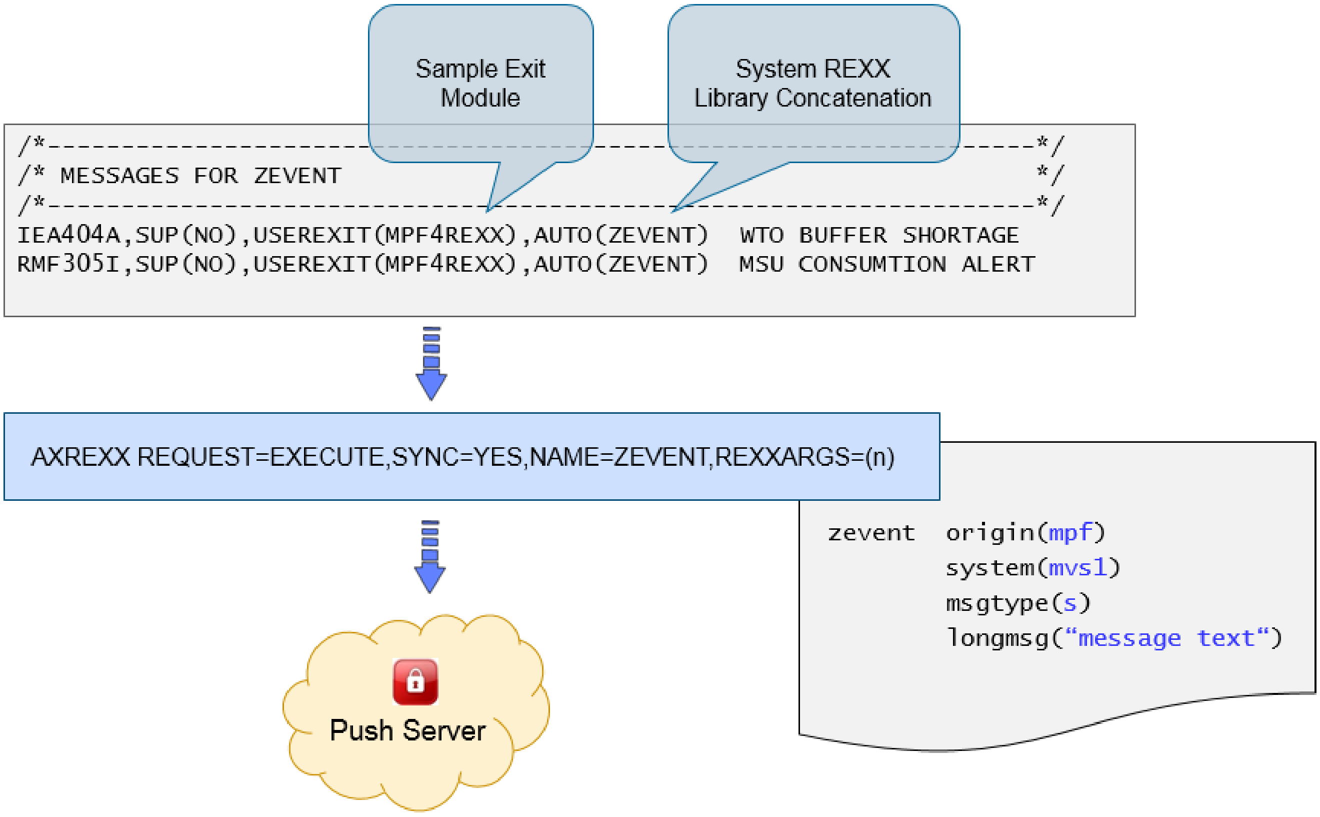
## 4.2 The zEvent MPF Exit Module MPF4REXX

The purpose of this module is to transfer control to the zEvent System REXX exec in case a specific message with a certain message id appears on the operator console.
As shown in Figure 9, this is achieved by means of the AXREXX authorized assembler service.

In this context it is recommended to define the specific TSO userid *ZEVENT* on the z/OS system.
In case the MPF4REXX module can find this RACF userid, the zEvent System REXX will be invoked on behalf the *ZEVENT* userid.
This is important for two reasons:

- The installation can take care that the *ZEVENT* user has RACF read access to the library where the zEvent System REXX exec is located.
- In case HTTPS communication is configured for message delivery the installation can specify within the AT-TLS policy that the communication initiated by the *ZEVENT* user needs to be intercepted and promoted to HTTP secure communication.

However, in case no *ZEVENT* userid has been defined by the installation the zEvent System REXX will be invoked on behalf of the userid specified with the AXRUSER parameter in the active AXRxx parmlib member (.cf also Figure 10).



```
/*-------------------------------------------------------------------*/
/* MESSAGES FOR ZEVENT                                                */
/*-------------------------------------------------------------------*/
IEA404A,SUP(NO),USEREXIT(MPF4REXX),AUTO(ZEVENT)  WTO BUFFER SHORTAGE
RMF305I,SUP(NO),USEREXIT(MPF4REXX),AUTO(ZEVENT)  MSU CONSUMTION ALERT
```

```
AXREXX REQUEST=EXECUTE,SYNC=YES,NAME=ZEVENT,REXXARGS=(n)
```

```
zevent  origin(mpf)
        system(mvs1)
        msgtype(s)
        longmsg("message text")
```

Figure 9: ZEVENT System REXX invocation

The MPF sample exit module can be treated as any other z/OS MPF user exit: it must reside in a linklist library and APF authorization is required as well. Furthermore the name specified with the AUTO parameter in the MPFLSTxx member must match the name of the REXX target module used in the zEvent setup and customization dialog.
Furthermore, in the active AXRxx parmlib member a line needs to be added which contains the dataset where the zEvent API exec resides (cf. Figure 10).

```
/*********************************************************************/
/* AXRZE - The SYSREXX parmlib member                               */
/*                                                                   */
/*********************************************************************/

CPF('REXX&MVSPID.',SYSPLEX) /* REXXNN AS A SYSPLEX WIDE CPF VALUE    */
AXRUSER(AXRUSER)            /* SECURITY=AXRUSER RESULTS IN THE EXEC
                               RUNNING IN A SECURITY ENVIRONMENT
                               DEFINED BY THE USERID AXRUSER         */
REXXLIB ADD DSN(SYS1.SAXREXEC)   /* 2ND DS OF REXXLIB CONCATENATION */
REXXLIB ADD DSN(ZEVENT.SAXREXEC) /* LIBRARY OF ZEVENT API EXEC       */
```

Figure 10: System REXX parmlib member

In case the MPF4REXX exit module cannot find the ZEVENT System REXX exec specified with the AUTO parameter the following console message will be displayed:

*ZEVENT: MPF4REXX CALL TO SYSTEM REXX FAILED*

In case of success the following message will be issued:

*ZEVENT: MPF4REXX CALL TO SYSTEM REXX SUCCESSFUL*

# 5 Setup RMF Monitor III Batch as Event Provider

## 5.1 Preface

Beyond the possibility to utilize any existing console message or message id as a trigger for a zEvent notification, an installation can also setup other z/OS components than MPF as event provider.
E.g. a z/OS performance monitoring facility can inspect selected key metrics continuously and submit push notifications based on the comparison of certain metric values and defined thresholds.

## 5.2 RMF Monitor III Batch and MPF Processing

The RMF Monitor III Batch Reporter is a standard component of the z/OS optional priced feature Resource Measurement Facility aka RMF.
In a nutshell, once the ERBM3B Started Task is active, a certain RMF Monitor III report is invoked for each Monitor III interval or reporting range. Thereby the installation is able to parse the values contained in the report and react on defined thresholds by means of exit procedures.

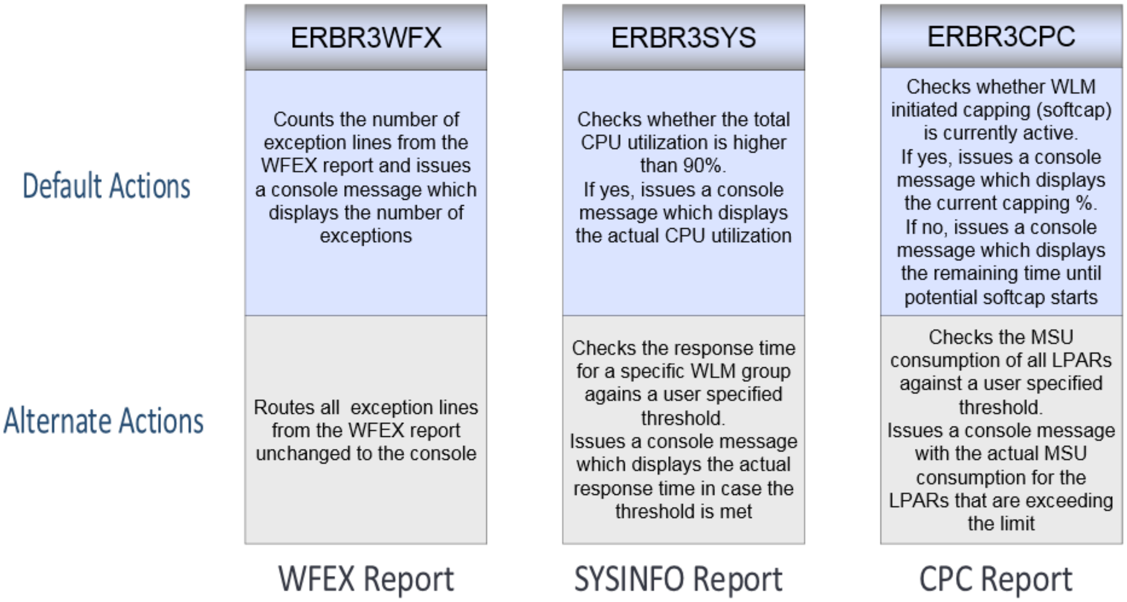
For more details about RMF Monitor III Batch refer to the following documents:

- RMF Users Guide, Chapter 21. Client Server Enabling
  http://publibz.boulder.ibm.com/epubs/pdf/erb2ug10.pdf
- The RMF2WTO Secret – From Exceptions to console Messages
  ftp://public.dhe.ibm.com/eserver/zserieszos/rmf/RMF2WTO.pdf

RMF delivers in SYS1.SERBCLS three sample exit procedures for the following Monitor III reports:

| | |
|---|---|
| • CPC | ERBR3CPC |
| • SYSINFO | ERBR3SYS |
| • WFEX | ERBR3WFX |

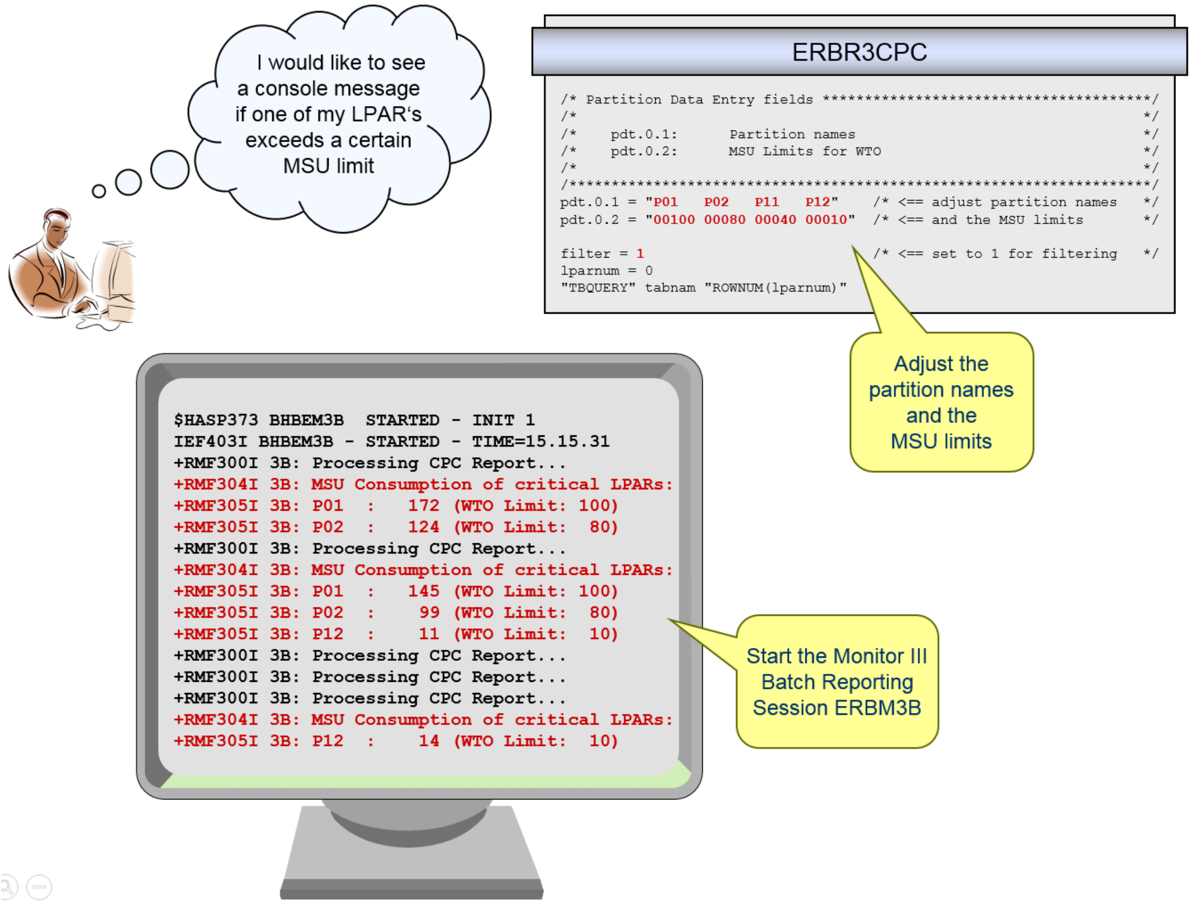
Depending on the report type, the actions shown in Figure 11 actions can be performed. All actions are associated with a console message and a specific message id.



| | ERBR3WFX | ERBR3SYS | ERBR3CPC |
|---|---|---|---|
| Default Actions | Counts the number of exception lines from the WFEX report and issues a console message which displays the number of exceptions | Checks whether the total CPU utilization is higher than 90%.<br>If yes, issues a console message which displays the actual CPU utilization | Checks whether WLM initiated capping (softcap) is currently active.<br>If yes, issues a console message which displays the current capping %.<br>If no, issues a console message which displays the remaining time until potential softcap starts |
| Alternate Actions | Routes all exception lines from the WFEX report unchanged to the console | Checks the response time for a specific WLM group agains a user specified threshold.<br>Issues a console message which displays the actual response time in case the threshold is met | Checks the MSU consumption of all LPARs against a user specified threshold.<br>Issues a console message with the actual MSU consumption for the LPARs that are exceeding the limit |
| | WFEX Report | SYSINFO Report | CPC Report |

Figure 11: ERBM3B exit procedures

Figure 12 shows an example for procedure ERBR3CPC where the Monitor III CPC report is processed and console message RMF305I is issued for every partition where the specified MSU threshold is exceeded.


I would like to see a console message if one of my LPAR's exceeds a certain MSU limit

ERBR3CPC

```
/* Partition Data Entry fields ***************************************/
/*                                                                   */
/*    pdt.0.1:      Partition names                                  */
/*    pdt.0.2:      MSU Limits for WTO                               */
/*                                                                   */
/*********************************************************************/
pdt.0.1 = "P01   P02   P11   P12"    /* <== adjust partition names   */
pdt.0.2 = "00100 00080 00040 00010"  /* <== and the MSU limits       */

filter = 1                           /* <== set to 1 for filtering   */
lparnum = 0
"TBQUERY" tabnam "ROWNUM(lparnum)"
```

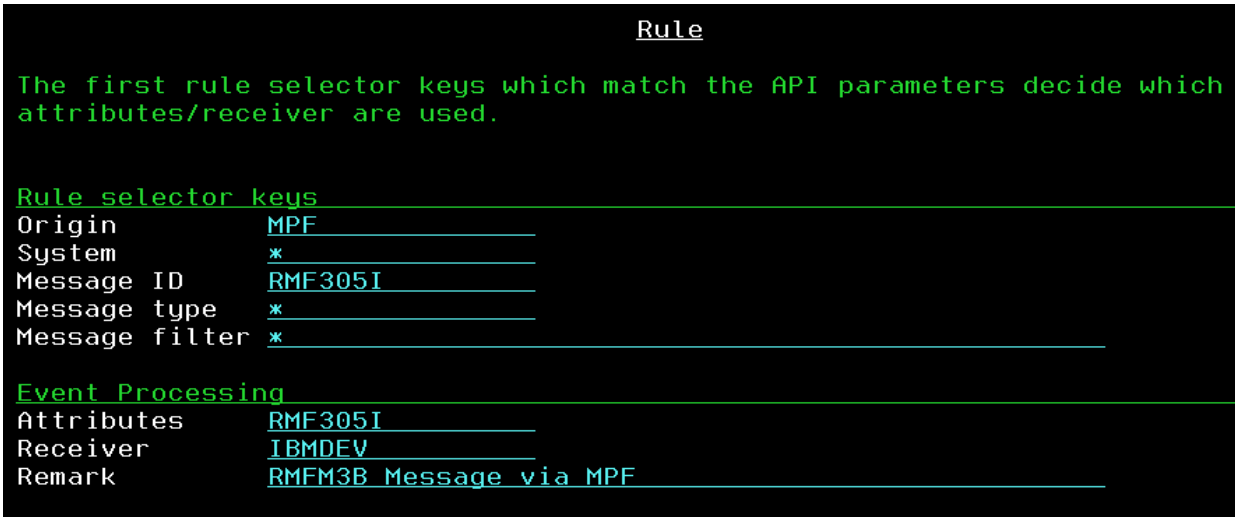
Adjust the partition names and the MSU limits

```
$HASP373 BHBEM3B  STARTED - INIT 1
IEF403I BHBEM3B - STARTED - TIME=15.15.31
+RMF300I 3B: Processing CPC Report...
+RMF304I 3B: MSU Consumption of critical LPARs:
+RMF305I 3B: P01   :   172 (WTO Limit: 100)
+RMF305I 3B: P02   :   124 (WTO Limit:  80)
+RMF300I 3B: Processing CPC Report...
+RMF304I 3B: MSU Consumption of critical LPARs:
+RMF305I 3B: P01   :   145 (WTO Limit: 100)
+RMF305I 3B: P02   :    99 (WTO Limit:  80)
+RMF305I 3B: P12   :    11 (WTO Limit:  10)
+RMF300I 3B: Processing CPC Report...
+RMF300I 3B: Processing CPC Report...
+RMF300I 3B: Processing CPC Report...
+RMF304I 3B: MSU Consumption of critical LPARs:
+RMF305I 3B: P12   :    14 (WTO Limit:  10)
```

Start the Monitor III Batch Reporting Session ERBM3B

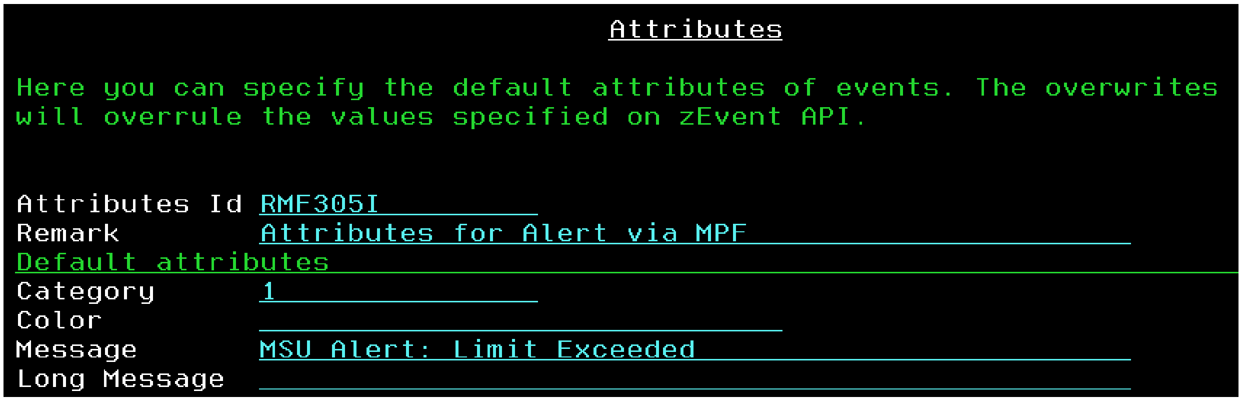
Figure 12: ERBM3B example – RMF Monitor III CPC Report

Now let’s combine both facilities – RMF Monitor III Batch and zEvent notifications:
Whenever message RMF305I appears on the console a notification to a certain mobile device is supposed to be submitted.
This can be achieved by means of the zEvent Administration Dialog as described in chapter 2.
What we need is a corresponding rule together with specific attributes based on the console message RMF305I. The figures below show an example implementation:



```
                                 Rule

The first rule selector keys which match the API parameters decide which
attributes/receiver are used.


Rule selector keys
Origin         MPF
System         *
Message ID     RMF305I
Message type   *
Message filter *

Event Processing
Attributes     RMF305I
Receiver       IBMDEV
Remark         RMFM3B Message via MPF
```

Figure 13: Sample rule for MPF processing

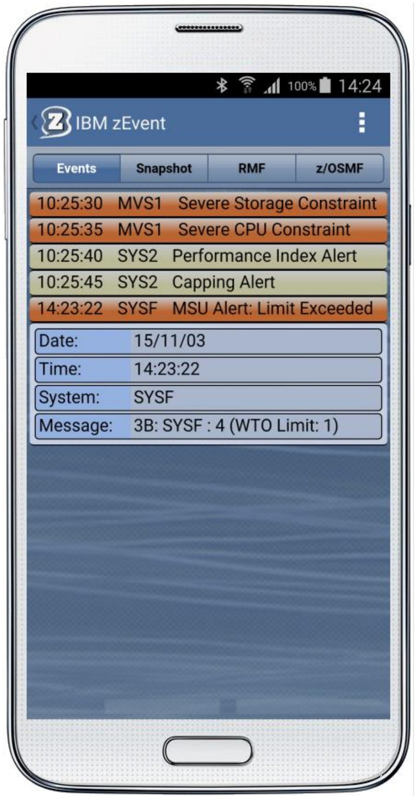


```
                                 Attributes

Here you can specify the default attributes of events. The overwrites
will overrule the values specified on zEvent API.


Attributes Id RMF305I
Remark        Attributes for Alert via MPF
Default attributes
Category      1
Color
Message       MSU Alert: Limit Exceeded
Long Message
```

Figure 14: Sample attributes for MPF processing

When the ZEVENT exec based on the specifications above has been created and the setup for the MPF exit module MPF4REXX has been performed as described in chapter 4 the ZEVENT exec is called for all partitions where the specified MSU threshold is exceeded.

As final result a push notification as shown in Figure 15 is submitted to the receivers of the corresponding rule specified in the Administration Dialog.



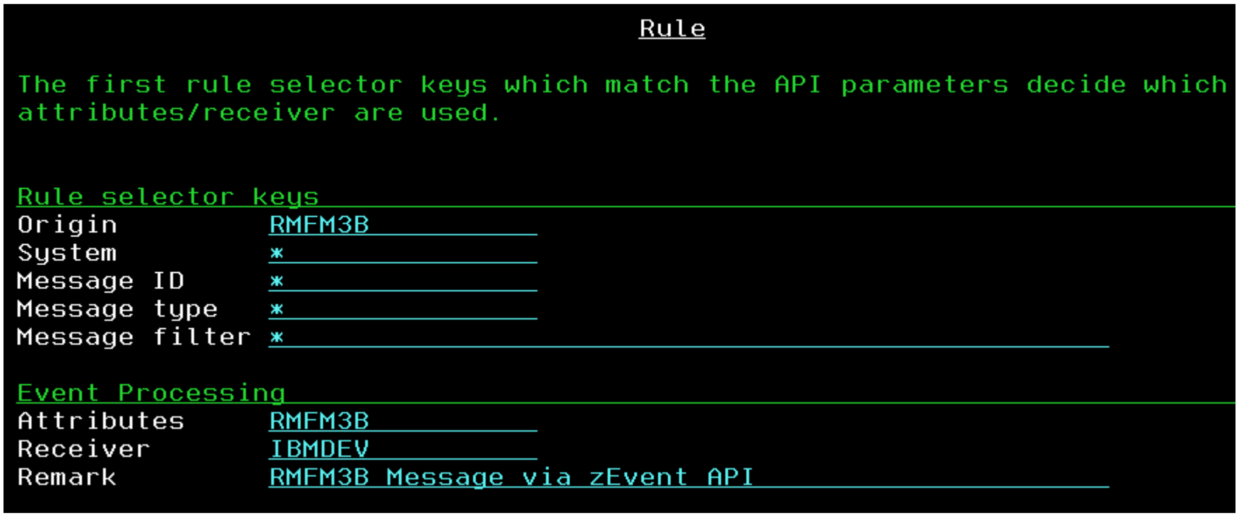
```
10:25:30   MVS1   Severe Storage Constraint
10:25:35   MVS1   Severe CPU Constraint
10:25:40   SYS2   Performance Index Alert
10:25:45   SYS2   Capping Alert
14:23:22   SYSF   MSU Alert: Limit Exceeded
```

| | |
|---|---|
| Date: | 15/11/03 |
| Time: | 14:23:22 |
| System: | SYSF |
| Message: | 3B: SYSF : 4 (WTO Limit: 1) |

Figure 15: Sample event for MPF processing

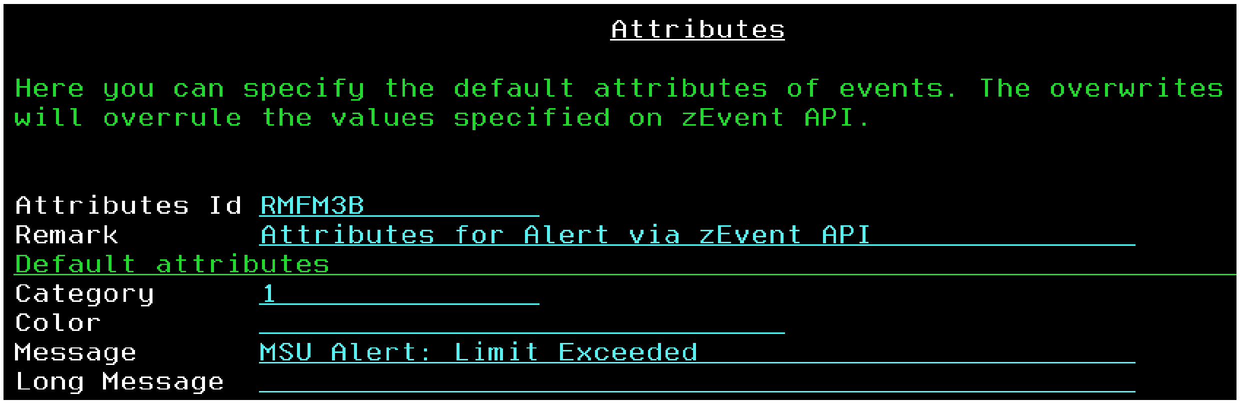
### 5.3 RMF Monitor III Batch and zEvent API Invocation

As an alternative to the MPF based processing, the RMF Monitor III Batch facility can invoke directly the ZEVENT exec which has been generated with the Administration Dialog.
This is the preferred method when no console messages are required and just a notification about the exceeded threshold needs to be submitted to the mobile device.
In this case the corresponding rule and attributes may look like as follows:



```
                                    Rule

The first rule selector keys which match the API parameters decide which
attributes/receiver are used.


Rule selector keys
Origin          RMFM3B
System          *
Message ID      *
Message type    *
Message filter  *

Event Processing
Attributes      RMFM3B
Receiver        IBMDEV
Remark          RMFM3B Message via zEvent API
```

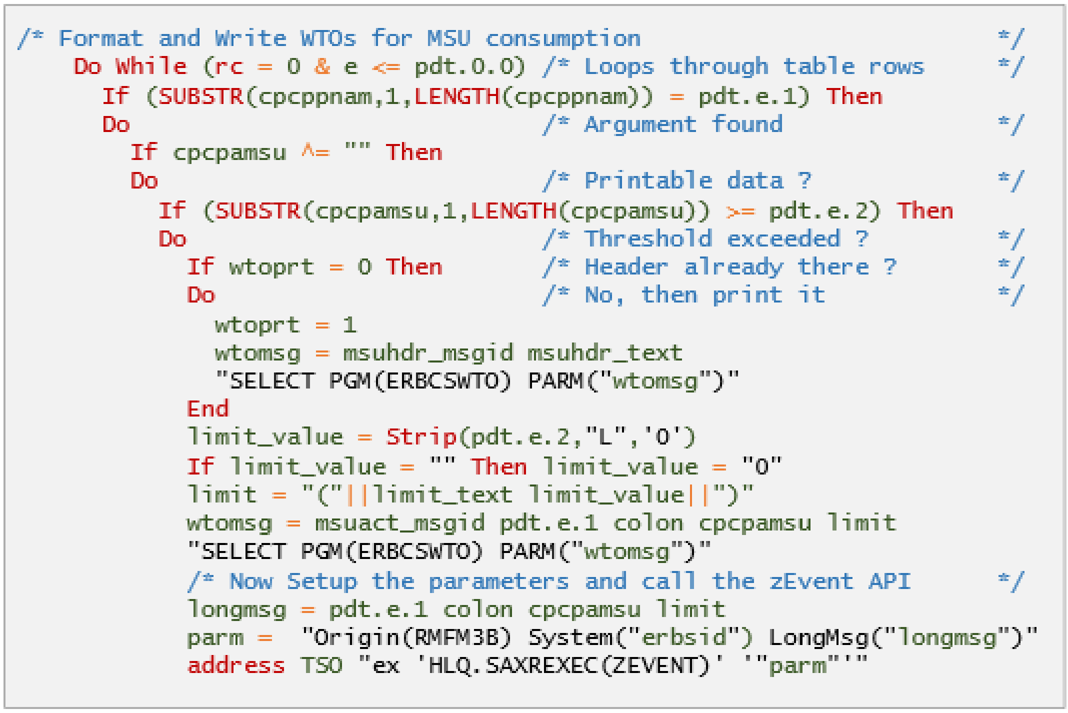
Figure 16: Sample Rule for zEvent API processing



```
                                 Attributes

Here you can specify the default attributes of events. The overwrites
will overrule the values specified on zEvent API.


Attributes Id RMFM3B
Remark        Attributes for Alert via zEvent API
Default attributes
Category      1
Color
Message       MSU Alert: Limit Exceeded
Long Message
```

Figure 17: Sample Attributes for zEvent API processing

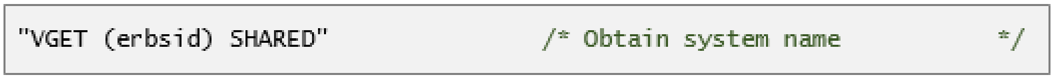
In order to call the zEvent API properly, the Monitor III Batch procedure ERBR3CPC needs to be slightly extended:
Since we want to invoke the zEvent API now directly, the generation of console messages with module ERBCSWTO is not needed anymore. Optionally the corresponding statements can be removed.
As shown in Figure 18, the last three lines have been inserted to setup the parameters and call the zEvent API.



```
/* Format and Write WTOs for MSU consumption                             */
    Do While (rc = 0 & e <= pdt.0.0) /* Loops through table rows         */
      If (SUBSTR(cpcppnam,1,LENGTH(cpcppnam)) = pdt.e.1) Then
      Do                             /* Argument found                   */
        If cpcpamsu ^= "" Then
        Do                           /* Printable data ?                 */
          If (SUBSTR(cpcpamsu,1,LENGTH(cpcpamsu)) >= pdt.e.2) Then
          Do                         /* Threshold exceeded ?             */
            If wtoprt = 0 Then       /* Header already there ?           */
            Do                       /* No, then print it                */
              wtoprt = 1
              wtomsg = msuhdr_msgid msuhdr_text
              "SELECT PGM(ERBCSWTO) PARM("wtomsg")"
            End
            limit_value = Strip(pdt.e.2,"L",'0')
            If limit_value = "" Then limit_value = "0"
            limit = "("||limit_text limit_value||")"
            wtomsg = msuact_msgid pdt.e.1 colon cpcpamsu limit
            "SELECT PGM(ERBCSWTO) PARM("wtomsg")"
            /* Now Setup the parameters and call the zEvent API          */
            longmsg = pdt.e.1 colon cpcpamsu limit
            parm =  "Origin(RMFM3B) System("erbsid") LongMsg("longmsg")"
            address TSO "ex 'HLQ.SAXREXEC(ZEVENT)' '"parm"'"
```

Figure 18: Invocation of the zEvent API via RMF Monitor III Batch

Please note that the z/OS system id is passed as part of the parameter list. For this reason the variable *erbsid* needs to be retrieved in advance of the call. This can be achieved with the following statement:

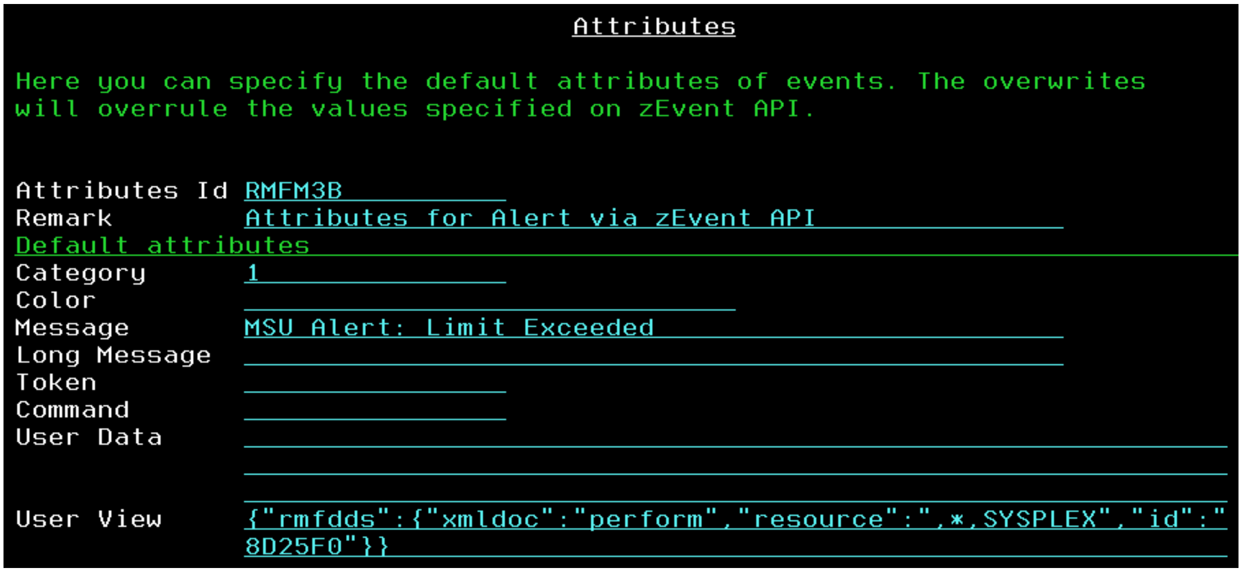


```
"VGET (erbsid) SHARED"                /* Obtain system name              */
```

When this changes are completed, the ERBM3B procedure can be restarted. Once a threshold is exceeded, the notification will be sent to the mobile device and the message on the event tab of the zEvent app is exactly the same than shown in Figure 15.
The only difference is that the zEvent API has been called directly without any exploitation of z/OS MPF components.

## 5.4 zEvent Notifications with RMF Context

zEvent push notifications can be extended with a variety of additional information. This feature can be used to provide context sensitive links to the monitoring facilities of the zEvent mobile app.
Hence, the receiver of a notification can switch instantly to the RMF tab or to the z/OSMF tab and analyze selected performance metrics in order to decide whether the notification requires an immediate action or not.
The additional information can be provided by means of the zEvent Administration Dialog in terms of JSON strings. The content of the JSON string is specified together with the *Attributes* definition as *User View* parameter (.cf Figure 19).



```
                              Attributes

Here you can specify the default attributes of events. The overwrites
will overrule the values specified on zEvent API.


Attributes Id RMFM3B
Remark        Attributes for Alert via zEvent API
Default attributes
Category      1
Color
Message       MSU Alert: Limit Exceeded
Long Message
Token
Command
User Data



User View     {"rmfdds":{"xmldoc":"perform","resource":",*,SYSPLEX","id":"
              8D25F0"}}
```

Figure 19: zEvent notifications – RMF context

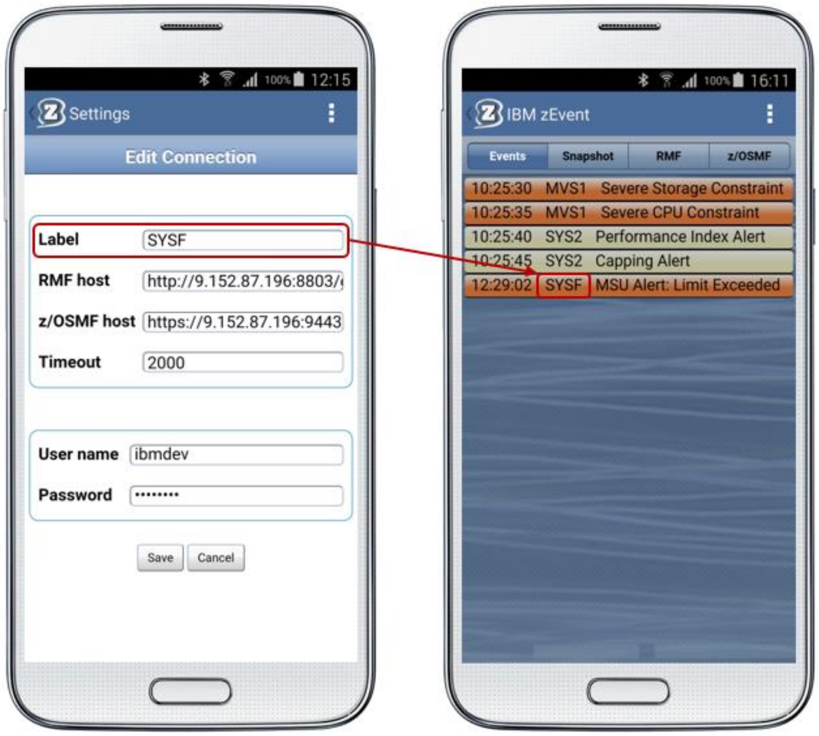
On the mobile device, the *User View* specification is recognized and translated to a URL suffix which is used to build the outbound request URL for the RMF Distributed Data Server (aka DDS).

Example:

```
http://ip.rmf:8803/gpm/perform.xml?resource=",*,SYSPLEX"&id=8D25F0
```

For more details about the RMF DDS API refer to the RMF Programmers Guide, Chapter 3, Accessing performance data using the RMF Distributed Data Server.

As shown in Figure 20, for a successful context launch it must be ensured that the label of the connection profile matches the system name as source of the push notification.

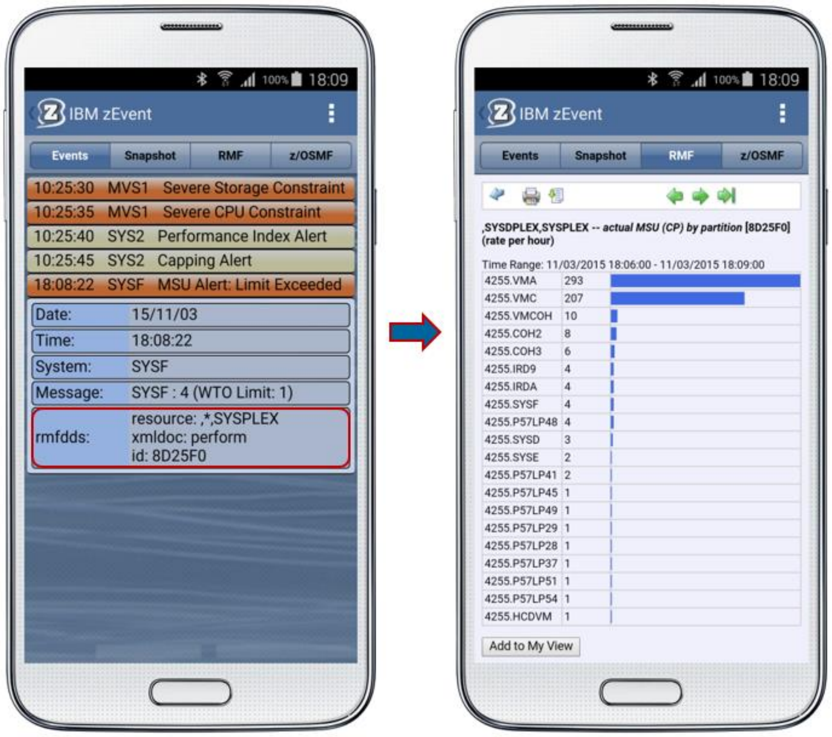


Label: SYSF
RMF host: http://9.152.87.196:8803/
z/OSMF host: https://9.152.87.196:9443
Timeout: 2000
User name: ibmdev
Password: ••••••••

10:25:30 MVS1 Severe Storage Constraint
10:25:35 MVS1 Severe CPU Constraint
10:25:40 SYS2 Performance Index Alert
10:25:45 SYS2 Capping Alert
12:29:02 SYSF MSU Alert: Limit Exceeded

Figure 20: Connection Label and notification source

The actual RMF context launch is shown in Figure 21: Once the user clicks on the expanded notification and selects the *open RMF* option, the zEvent app switches to the *RMF* tab and displays the required metric.



10:25:30 MVS1 Severe Storage Constraint
10:25:35 MVS1 Severe CPU Constraint
10:25:40 SYS2 Performance Index Alert
10:25:45 SYS2 Capping Alert
18:08:22 SYSF MSU Alert: Limit Exceeded

Date: 15/11/03
Time: 18:08:22
System: SYSF
Message: SYSF : 4 (WTO Limit: 1)
rmfdds: resource: ,*,SYSPLEX<br>xmldoc: perform<br>id: 8D25F0

,SYSDPLEX,SYSPLEX -- *actual MSU (CP) by partition* [8D25F0] (rate per hour)

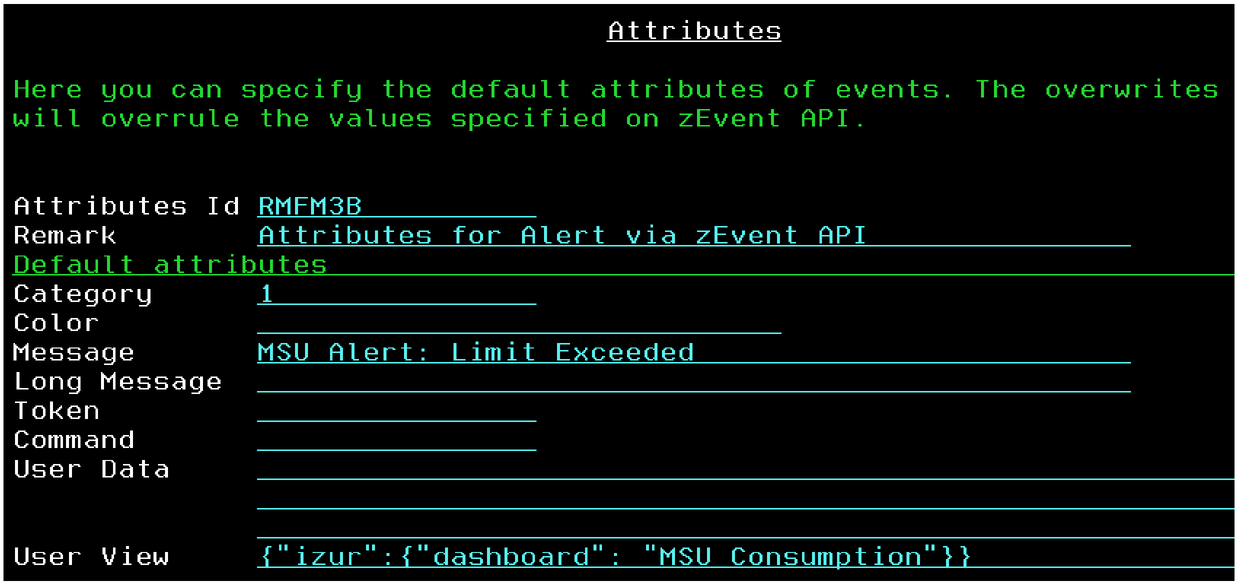
Time Range: 11/03/2015 18:06:00 - 11/03/2015 18:09:00

| | |
|---|---|
| 4255.VMA | 293 |
| 4255.VMC | 207 |
| 4255.VMCOH | 10 |
| 4255.COH2 | 8 |
| 4255.COH3 | 6 |
| 4255.IRD9 | 4 |
| 4255.IRDA | 4 |
| 4255.SYSF | 4 |
| 4255.P57LP48 | 4 |
| 4255.SYSD | 3 |
| 4255.SYSE | 2 |
| 4255.P57LP41 | 2 |
| 4255.P57LP45 | 1 |
| 4255.P57LP49 | 1 |
| 4255.P57LP29 | 1 |
| 4255.P57LP28 | 1 |
| 4255.P57LP37 | 1 |
| 4255.P57LP51 | 1 |
| 4255.P57LP54 | 1 |
| 4255.HCDVM | 1 |

Figure 21: RMF context launch

## 5.5 zEvent Notifications with z/OSMF Context

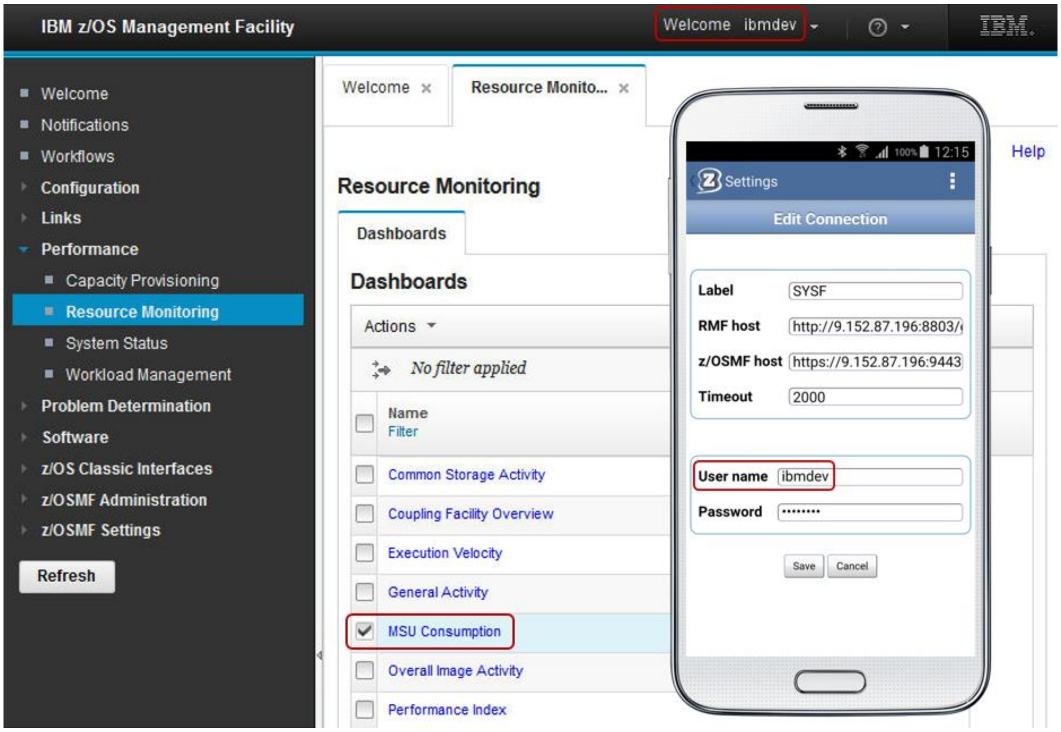
Accordingly to RMF context, the *User View* specification of the *Attributes* definition supports also z/OSMF context by using *izur* as component name and *dashboard* as keyword (Figure 22).



```
                                  Attributes

Here you can specify the default attributes of events. The overwrites
will overrule the values specified on zEvent API.


Attributes Id RMFM3B
Remark        Attributes for Alert via zEvent API
Default attributes
Category      1
Color
Message       MSU Alert: Limit Exceeded
Long Message
Token
Command
User Data



User View     {"izur":{"dashboard": "MSU Consumption"}}
```

Figure 22: zEvent notifications – z/OSMF context

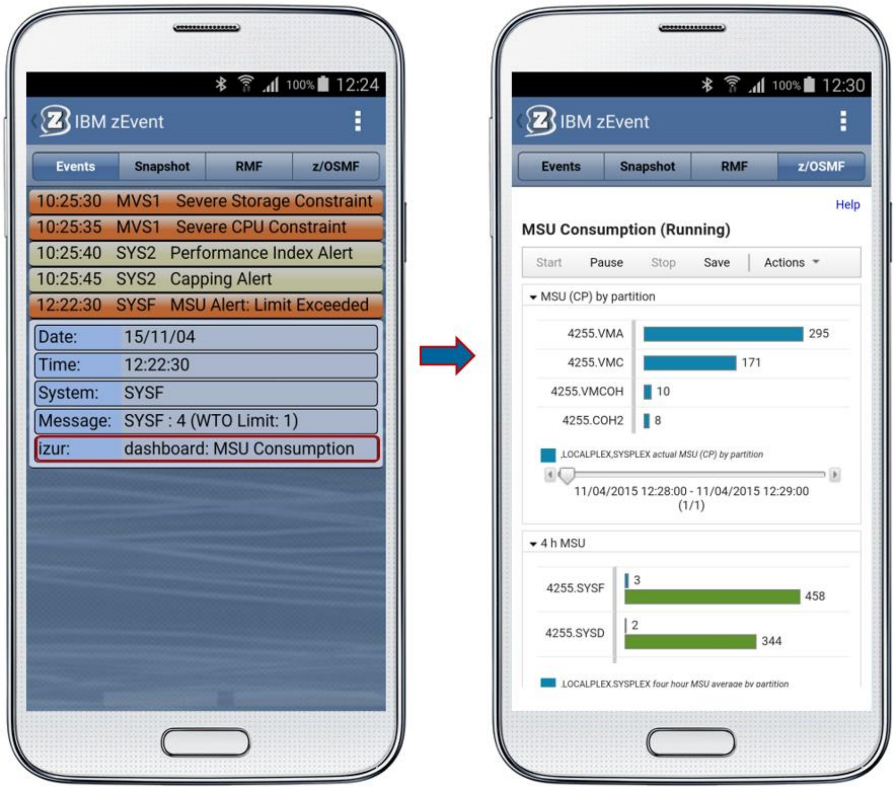
The dashboard name following the *dashboard* keyword can refer to any existing z/OSMF dashboard that is defined for the z/OSMF user id which has been provided with the login information of the zEvent connection profile. Specifically, the dashboard *MSU Consumption* must actually appear in the dashboard list of the z/OSMF userid *ibmdev* (Figure 23).



Label SYSF
RMF host http://9.152.87.196:8803/(
z/OSMF host https://9.152.87.196:9443
Timeout 2000
User name ibmdev
Password ••••••••

Figure 23: z/OSMF Dashboards for userid *ibmdev*

The actual z/OSMF context launch is shown in Figure 24: Once the user expands the notification and clicks *open z/OSMF*, the zEvent app switches to the z/OSMF tab and displays the required dashboard.



| | | |
|---|---|---|
| 10:25:30 | MVS1 | Severe Storage Constraint |
| 10:25:35 | MVS1 | Severe CPU Constraint |
| 10:25:40 | SYS2 | Performance Index Alert |
| 10:25:45 | SYS2 | Capping Alert |
| 12:22:30 | SYSF | MSU Alert: Limit Exceeded |

| | |
|---|---|
| Date: | 15/11/04 |
| Time: | 12:22:30 |
| System: | SYSF |
| Message: | SYSF : 4 (WTO Limit: 1) |
| izur: | dashboard: MSU Consumption |

Figure 24: z/OSMF context launch

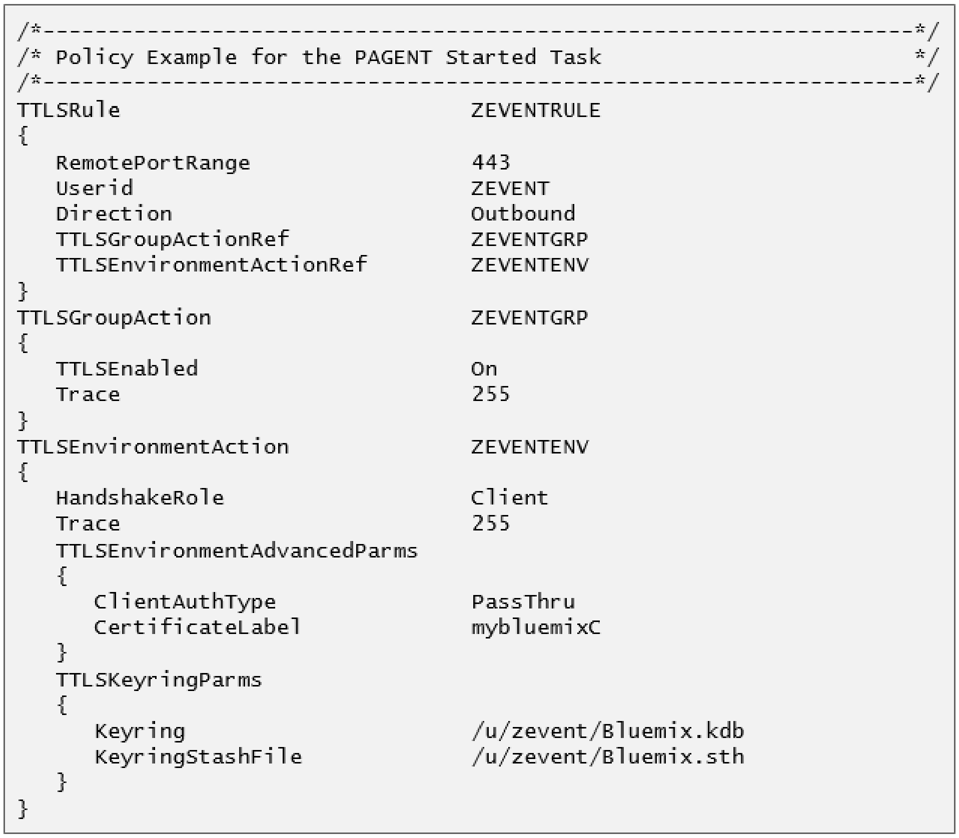
# 6 HTTPS Setup for zEvent Notifications

## 6.1 Overview

The zEvent REXX exec allows to use HTTP communication, but this is only intended for testing purposes until the HTTPS setup is complete.
Since zEvent does not support HTTPS natively, you need to exploit the z/OS Application Transparent Transport Layer Security (AT-TLS) in order to setup the HTTPS based communication.

## 6.2 AT-TLS Configuration for zEvent

In general the AT-TLS configuration for zEvent can be performed in the same way than for any other z/OS component which needs to be enabled for secure communication via AT-TLS.
The installation specifies the AT-TLS parameters by means of a policy for the PAGENT started task. An example for an AT-TLS policy with special regard to zEvent is shown in Figure 25.



```
/*--------------------------------------------------------------------*/
/* Policy Example for the PAGENT Started Task                         */
/*--------------------------------------------------------------------*/
TTLSRule                           ZEVENTRULE
{
   RemotePortRange                 443
   Userid                          ZEVENT
   Direction                       Outbound
   TTLSGroupActionRef              ZEVENTGRP
   TTLSEnvironmentActionRef        ZEVENTENV
}
TTLSGroupAction                    ZEVENTGRP
{
   TTLSEnabled                     On
   Trace                           255
}
TTLSEnvironmentAction              ZEVENTENV
{
   HandshakeRole                   Client
   Trace                           255
   TTLSEnvironmentAdvancedParms
   {
      ClientAuthType               PassThru
      CertificateLabel             mybluemixC
   }
   TTLSKeyringParms
   {
      Keyring                      /u/zevent/Bluemix.kdb
      KeyringStashFile             /u/zevent/Bluemix.sth
   }
}
```

Figure 25: PAGENT parameter example

It is recommended to specify the ZEVENT userid within the TTLSRule when the ZEVENT exec is invoked by the MPF4REXX module. In this case the PAGENT will intercept all HTTP requests where the originating user is the ZEVENT user.

For environments where the ZEVENT exec is invoked by other components than MPF, alternatively the Jobname parameter can be used in the AT-TLS policy in order to determine which requests should be intercepted resp. converted to HTTPS by the PAGENT.

In addition the installation needs to propagate to the PAGENT the required certificates for the HTTPS based communication. For detailed information how this can be achieved, refer to the AT-TLS related descriptions in the z/OS Communication Server IP Configuration Guide and the Security Server RACF System Programmer's Guide.
The example in Figure 25. refers to a quick setup for certificates using the OMVS *gskkyman* utility to create a keyring database and a keyring stash file.
In order to enable the SSL based communication to the Bluemix server, the keyring database must contain the mybluemix.net certificate chain as shown in Figure 26.

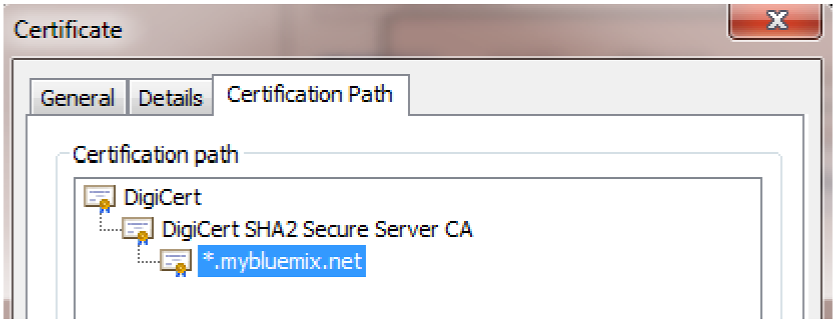


Figure 26: mybluemix  zEvent certificate chain

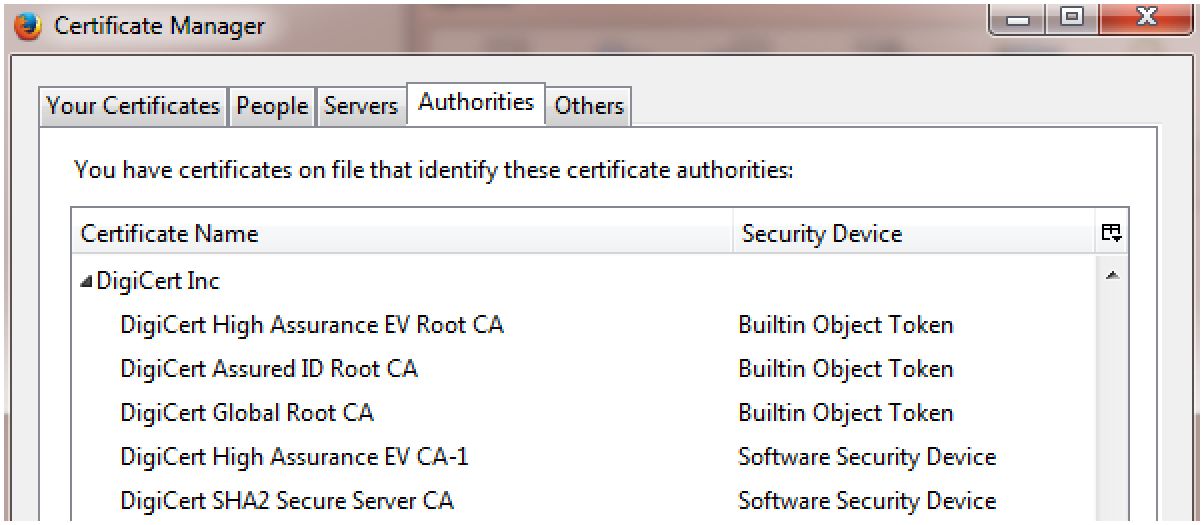
The certificates can be obtained by performing the following steps on your workstation:

Get the mybluemix zEvent certificate:

- Open the URL https://ibm-zevent.mybluemix.net
- Right click and choose “View Page Info”
- Select “Security”
- Click the “View Certificate” button
- Open the “Details” tab
- Click the “Export” Button

The following Certificates can be exported from your browsers certificate store (e.g. Firefox):

- DigiCert Global Root CA
- DigiCert SHA2 Secure Server CA

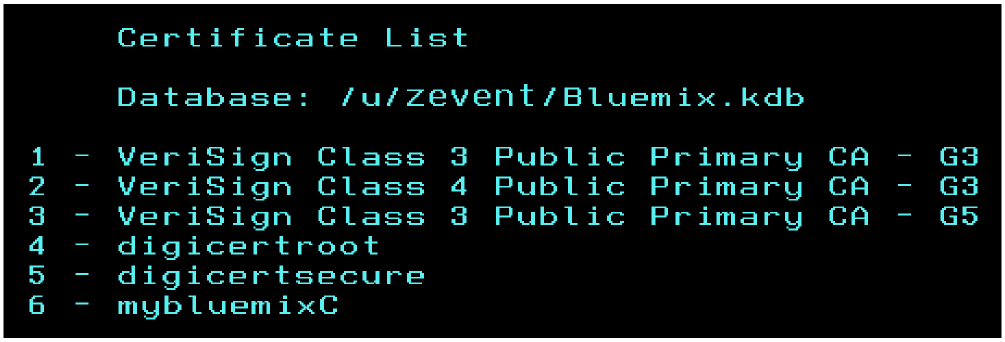


You have certificates on file that identify these certificate authorities:

| Certificate Name | Security Device |
|---|---|
| DigiCert Inc | |
| DigiCert High Assurance EV Root CA | Builtin Object Token |
| DigiCert Assured ID Root CA | Builtin Object Token |
| DigiCert Global Root CA | Builtin Object Token |
| DigiCert High Assurance EV CA-1 | Software Security Device |
| DigiCert SHA2 Secure Server CA | Software Security Device |

Figure 27: Browser certificates (Firefox example)

Once the certificates are extracted they can be uploaded in ASCII mode to a USS directory on the z/OS system (e.g. with FTP).
Now the OMVS *gskkyman* utility can be used to import the certificates into the keyring database in descending order. Then the certificates should appear in the certificate list of the keyring database:



```
    Certificate List

    Database: /u/zevent/Bluemix.kdb

1 - VeriSign Class 3 Public Primary CA - G3
2 - VeriSign Class 4 Public Primary CA - G3
3 - VeriSign Class 3 Public Primary CA - G5
4 - digicertroot
5 - digicertsecure
6 - mybluemixC
```

Figure 28: Keyring Database – Certificate List

Furthermore the label of the Bluemix certificate (example: mybluemixC) must match the label specified with *CertificateLabel* parameter in the AT-TLS policy.

This is everything needed with regard to the zEvent AT-TLS configuration.
After restarting the PAGENT started task the AT-TLS component should be able to intercept and convert the zEvent related communication successfully to the HTTPS protocol.

# 7 zEvent API

The zEvent API exec is built by the zEvent Administration Dialog and can be used to push messages to a mobile device. Since the API exec has all rules, users, etc. integrated at creation time, it does not need to load external data. The parameters of the exec are used to find a matching rule and the attributes of the matching rule are used to complete the parameters with defaults and overwrites.

## 7.1 Parameters for Rule Selection

If you have defined the proper rules, you just need one or more of the following four parameters to identify the desired rule. The API exec will find the rule and use the receiver and the attributes which are specified in this rule.

The main reason for these parameters is to select a rule. As soon as the zEvent API finds a rule matching these parameters, a push message is sent to the receivers specified in the rule.

| Parameter | Description |
|---|---|
| **ORIGIN** | Identifies the caller of the API |
| **SYSTEM** | Name of the issuing host system |
| **MSGID** | Message identifier |
| **MSGTYPE** | The type of a push message |

**Origin:**
A name which identifies the caller of the API. This allows to distinguish between multiple callers when evaluating the rules.

**System:**
Name of the issuing system. You could also send messages on behalf of another system and specify the name of that system. If no system is specified, the API exec uses the content of REXX environment variable *SYSNODE* instead.

**MsgId:**
The message id identifies one specific message.

**MsgType:**
The message type allows to subdivide your messages into different types.

## 7.2 Parameters to overwrite attributes

The following parameters are normally specified in the rule section of the administrative dialog and selected by the rules, but you can overwrite these values via parameter:

| Parameter | Description |
|---|---|
| **CATEGORY** | Can be used to classify event messages |
| **COLOR** | Color of an entry in the events/charts tab of the zEvent app |
| **MESSAGE** | Short message which will be displayed in the event list of the app |
| **LONGMSG** | Long message will be shown if you expand a message in event list |
| **USERDATA** | Allows to add user data to event messages |
| **USERVIEW** | Specifies the metric or dashboard which will be shown in the RMF or z/OSMF tab |
| **TOKEN** | Not implemented yet |
| **COMMAND** | Not implemented yet |

**Category**:
The categories 1, 2 and 3 will default to the colors red, yellow and green (if color parameter is omitted). Basically the category is a text that can be freely assigned by the caller. Since there are no filter options in the zEvent app yet, the best use of category is to set the predefined colors red, yellow and green.

**Color:**
Specifies the color of an entry in the events/charts tab. The color can be specified in one of the following formats: rgba(222,0,0,0.55), rgb(222,0,0), red, #ff0000

**Message:**
Is the short message which will be displayed in the event list of the zEvent app

**LongMsg:**
Long message will be shown if you expand a specific message in the event list. It can contain html statements like <br> (for line break) in the message text.

**UserData:**
The user data allows to specify key/value pairs which are shown in the event list of the zEvent app. The key/value pairs need to be encoded in JSON format (e.g. {"key1":"val1"},{"key2":"val2"}).

**UserView:**
Specifies the metric or dashboard which will be shown in the RMF or z/OSMF tab. For an example how to use *User View*, see chapter 5.4 or 5.5.

**Token:**
Not implemented yet

**Command:**
Not implemented yet

## 7.3 Other parameters

| Parameter | Description |
|---|---|
| **DATE** | Current date |
| **TIME** | Current time |
| **CHART** | Allows to send a pre-configured chart in JSON format |
| **RECEIVER** | Send a push event to a specific device |
| **RC** | Return code of the API |

**Date:**
The current date. It can be empty if you want the zEvent API exec to fill in the date

**Time:**
The current time. It can be empty if you want the zEvent exec to fill in the time

**Chart:**
Allows to send a pre-configured chart in JSON format

**Receiver:**
You can specify a user or group of users for the push message which is known by the zEvent API exec. Therefore it needs to be defined in the zEvent administrative dialog.

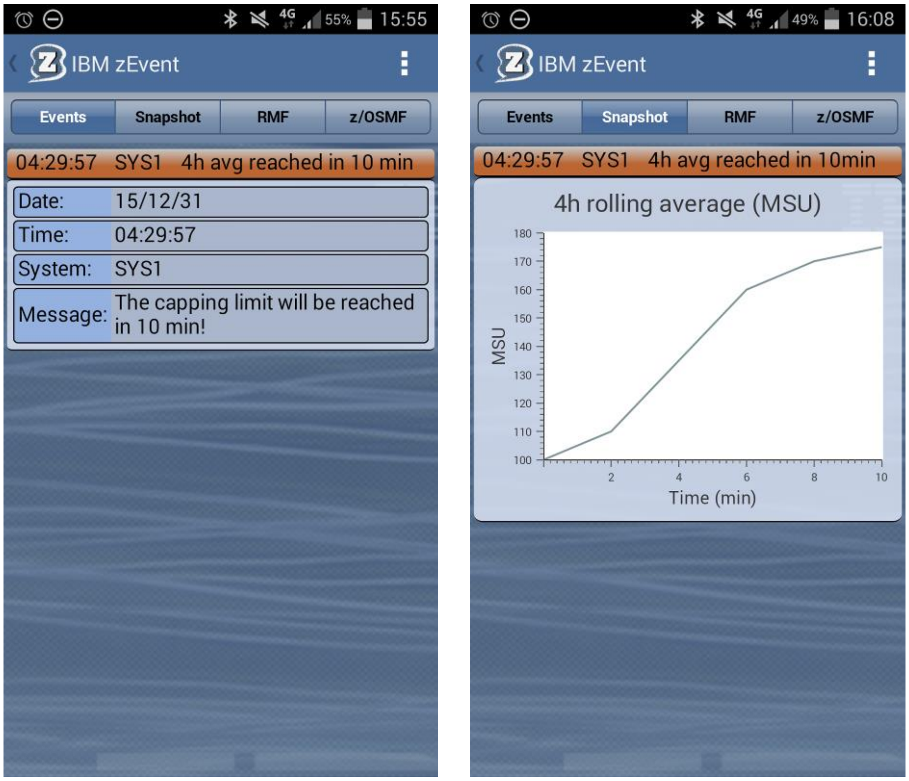
**RC:**
Return code of the zEvent API exec.

# 8 Push Interface

The push interface describes the data which is pushed to the mobile device. The payload can contain an *events* and/or a *charts* section. The *events* are shown on the *Events* tab and the *charts* are shown on the *Snapshot* tab of the zEvent app.

```
{
     "events" : [ {
          "dte" : "15/12/31",
          "tme" : "04:29:57",
          "sys" : "SYS1",
          "msg" : "4h avg reached in 10 min",
          "lng" : "The capping limit will be reached in 10 min!",
          …
     } ],

     "charts" : [ {
          "dte" : "15/12/31",
          "tme" : "04:29:57",
          "sys" : "SYS1",
          "msg" : "4h avg reached in 10min",
          "ttl" : "4h rolling average (MSU)",
          "xax" : {
               "ttl" : "Time (min)",
               "lbl" : [0, 2, 4, 6, 8, 10]
          },
          "yax" : {
               "ttl" : "MSU"
          },
          "ser" : [ {
               "ttl" : "avg",
               "typ" : "Lines",
               "val" : [100, 110, 135, 160, 170, 175]
          } ],
          …
     } ]
}
```

The above specifications will create a push message which contains an event and a chart. Figure 29 shows the corresponding results in the zEvent app.



04:29:57 SYS1 4h avg reached in 10 min

| | |
|---|---|
| Date: | 15/12/31 |
| Time: | 04:29:57 |
| System: | SYS1 |
| Message: | The capping limit will be reached in 10 min! |

04:29:57 SYS1 4h avg reached in 10min

Figure 29: Sample *Event* and *Snapshot* tab

## 8.1 Events Section

An *events* section is an array of one or more events. When you push an event to a mobile device, a new entry on the *Events* tab is created which shows the time, system and message attribute of the event. After expanding the new entry you will see other available information about the event.

**Syntax:**

```
"events" : [ {
     "dte" : "15/12/31",
     "tme" : "04:29:57",
     "sys" : "SYS1",
     "cat" : "2",
     "col" : "red",
     "msg" : "event 1 (of event list)",
     "lng" : "long message",
     "tok" : "myToken",
     "cmd" : "",
     "usr" : [{"key1":"val1"},{"key2":"val2"}],
     "viw" : {"zOSMF" : "myView"}
} ]
```

**Keywords:**

| Keyword | Meaning | Description |
|---|---|---|
| **dte** | Date | Date of event creation |
| **tme** | Time | Time of event creation |
| **sys** | System | Issuing system |
| **cat** | Category | Category. If the color attribute is not set, category 1, 2, 3 are translated to color red, yellow, green. |
| **col** | Color | rgba(222,0,0,0.55), rgb(222,0,0), red, #ff0000 |
| **msg** | Message | Short message |
| **lng** | Long Msg. | Long message |
| **tok** | Token | Not implemented yet |
| **cmd** | Command | Not implemented yet |
| **usr** | User | User data ( Key:Value pairs ) |
| **viw** | View | When opening another tab (RMF, z/OSMF, etc.) from this event, the view allows to start a specific view, dashboard, etc. For an example how to use *User View*, see chapter 5.4 or 5.5. |

## 8.2 Charts Section

A *charts* section is an array of one or more charts. When you push a chart to a mobile device, a new entry on the *Charts* tab is created which shows the time, system and message attribute of the chart. After expanding the new entry you see the chart and eventually other available data.

**Syntax:**

```
"charts" : [ {
     "dte" : "15/12/31",
     "tme" : "04:45:57",
     "sys" : "SYS4",
     "cat" : "1",
     "col" : "red",
     "msg" : "chart message",
     "ttl" : "chart 1 (of chart list)",
     "tok" : "myToken",
     "cmd" : "",
     "lgd" : false,
     "xax" : {
          "ttl" : "Time (h)",
          "lbl" : [ 1, 2, 3 ]
     },
     "yax" : {
          "ttl" : "MSU"
     },
     "ser" : [ {
          "ttl" : "CPU1",
          "typ" : "Lines",
          "col" : "#665260B4",
          "mrk" : false,
          "val" : [ 22, 16, 25 ]
     } ],
     "usr" : [{"key1":"val1"},{"key2":"val2"}]
} ]
```

**Keywords:**

| Keyword | Meaning | Description |
|---|---|---|
| dte | Date | Date of chart creation |
| tme | Time | Time of chart creation |
| sys | System | Issuing system |
| cat | Category | Category. If the color attribute is not set, category 1, 2, 3 are translated to color red, yellow, green. |
| col | Color | rgba(222,0,0,0.55), rgb(222,0,0), red, #ff0000 |
| msg | Message | Short message |
| ttl | Title | Title of the chart |
| tok | Token | Not implemented yet |
| cmd | Command | Not implemented yet |

| | | |
|---|---|---|
| lgd | Legend | Not implemented yet ( show/hide legend ) |
| xax | X Axes | Keeps the x-axes information |
| xax-ttl | Title | x-axes title |
| xax-lbl | Label | x-axes labels |
| yax | Y Axes | Keeps the y-axes information |
| yax-ttl | Title | y-axes title |
| ser | Series | Keeps the series information |
| ser-ttl | Title | Series title |
| ser-typ | Type | Series type (Lines, Columns) |
| ser-col | Color | Series color |
| ser-mrk | Marker | Use markers for this series (true, false) |
| ser-val | Values | Values of this series |
| usr | User | User data ( Key:Value pairs ) |