Framework for Member State time critical activities - “simple jobs”

1.1 Events

A database of events has been added to ECaccess. Events will be assigned a number and a name. A comment can describe the event. A command called ecevent is available to manipulate events. They will first be created. They can be restricted for use. Events will be triggered; they can be reset or deleted. Here follow more details on the command ecevent:

***Usage: ecevent [-create \|-send\|-clear\|-delete\|-grant\|-update] <MyNotification> ***
***[-comment "Comment for my notification (shown to the users)"] ***
***[-title "Comment for notification (shown to the operators)"] ***
***[-public] [-env “variables to pass "] [-seq <number>] ***
***[-notify\|-subsrcribe] [-users "list_of_users"] ***
[-at <date>\| -delay <arg>] [-format <arg>] [-ecport <arg>]

-create <MyNotification>: to create a new event, called MyNotification.
-comment “Comment”: adds a comment describing the event newly created
-public: the new event created can be subscribed to by any user. By default, only the owner of the event can subscribe to it.
-title “Comment”: adds a title describing the event newly created. This title will be displayed in the monitoring interface for the operators. It is compulsory when this event is to be monitored.
-metadata “data”: allows one to position the event in the monitoring interface available to the operators. This option is only available to UID emos. The main page of the monitoring interface shows all the events linked to the operational suite. We have subdivided this page into different areas. Firstly, we have defined 4 groups, one for the two main cycles at 00Z and at 12Z, one for the BC runs and one for the other runs. These groups will occupy one column on the monitoring page. Within these groups, we have defined families. For example for the runs at 00Z, one family defines the deterministic forecast run and another family defines the Ensemble Prediction system. Finally, within each family, one can define tasks. The tasks within one family should be ordered according to the schedule when the corresponding event will be started. The position parameters and descriptors to use are:

groupOrder=NN

groupName=<name>
familyOrder=MM
familyName=<name>
taskOrder=PP
taskName=<name>

For example:

-metadata "groupName=00Z_runs; groupOrder=10;
familyName=/od/mc/msjobs/00; familyOrder=10;
taskName=ms240; taskOrder=70"

Hint: use ecels to check the metadata for existing events and check the layout of the monitoring page under

http://ecgate.ecmwf.int:9080/do/events/

-send <MyNotification>: to send a signal to the event called <MyNotification>

-env "variables to pass ": pass the environment variables listed to the jobs subscribing to this event. The list should be of the format: "VAR1=val1; VAR2=val2; …"

-seq < number>: specifies the sequence number to notify the given event. This option is compulsory when sending and the number should be unique and increasing. The same event cannot be notified twice with the same number. For example, the date and time could be used as sequence number.

-at <date>: defines the date and time when the monitoring interface will cycle through, ready for the next notification of the event. <date> should be given using the format YYYYMMDDhhmmss. This format can be changed with the -format option. By default, there is no cycling on the monitoring interface; the status for an event will be updated when the next notification for this event is sent.

-format <arg>: defines the date format, as used with the -at option, The default format is like YYYYMMDDhhmmss.

-delay <arg>: defines the delay - starting from the notification of an event - after which the monitoring interface will cycle through, ready for the next notification of the event. By default, there is no cycling on the monitoring interface; the status for an event will be updated

when the next notification for this event is sent. The delay can be given in weeks (w), days (d), hours (h), minutes (m) or seconds (s), e.g. -delay 18h.

-grant <MyNotification>: ***to add or change the access for some users to the event called MyNotification***

- ***-users "list_of_users":*** ***grant or change access to an event to the given list of users. Commas should separate the UIDs. If a list of users is given without a "-subscribe" or "-notify", the access to the event for the given users will be removed.***
- ***-subscribe:*** ***to authorise the given users to subscribe to the event.***
- ***-notify:*** ***to authorise the given users to send notifications to the event.***

-clear <MyNotification>: ***to reset the sequence numbers of notification of the event called MyNotification and remove all jobs subscribing to the event. When an event is cleared, those users having jobs subscribed to the event will be notified by email.***

-update <MyNotification>: ***to update the settings of the event called MyNotification. This option can be used with the options -comment, -title, -metadata and -public which are described above.***

- ***-name <newName>:*** ***to rename the event.***

-delete <MyNotification>: ***to remove the event called MyNotification and remove all jobs subscribing to the event. When an event is deleted, those users having jobs subscribed to the event will be notified by email.***

[-ecport <arg>]: ***allows one to select the operational or test environment. <arg> is equal to 644 for the operational environment and equal to 9644 for the test environment. The default is 644.***

As one can see, the ecevent command already includes the necessary features for anybody to define their own events and possibly to generate simple dependencies between various tasks. Access to subscribe to or to send a notification to an event can also be granted to other users.

In the context of the current system of simple MS jobs via SMS, emos would, only once, create all the events corresponding to the different msjobs tasks defined in the operational suite. For example for the event an00h000:

```
% ecevent -create an00h000 -comment “At this stage, the operational \
analysis at 00UTC is complete.” -public -title “/od/o/msjobs/00/ms000”
```

And in the jobs msjobs, the call to msj_submit would be replace with a call to ecevent, to notify ECaccess and the jobs waiting for the event. For the event an00h000, the notification could happen with:

```
% ecevent -send an00h000 -env “MSJ_BASETIME=$MSJ_BASETIME;…” \
-seq $MSJ_YEAR$MSJ_MONTH$MSJ_DAY$MSJ_BASETIME -delay 15h
```

Note that some obsolete variables currently passed will not be available any longer, like MSJ_USER, MSJ_COUNTRY or MSJ_PATH. Only one user uses the MSJ_PATH variable to find out which event the job is related to. In order to allow the user to find this out, we suggest passing a variable defining the event, e.g. MSJ_EVENT=MSJ_DIR. We suggest that the variable MSJ_DIR in the operational suite is redefined to MSJ_EVENT.

Note also that the sequence number is unique for this event. If the operators would rerun a task msjob and if the command ecevent had succeeded in the first run, the second run would fail. The event should only be sent once. Only when the msjob tasks fail to complete the ecevent command should they then be rerun.