Hands-on Session

ESC 36^{th} General Assembly

Valletta, Malta 2018

Status: *Working Draft*

Aurélien Dupont
European-Mediterranean Seismological Centre

Related project: *EPOS-Seismology*

July 18, 2018

Contents

1 Introduction

European-Mediterranean Seismological Centre (EMSC) was established in 1975 to provide aggregated and authoritative parametric earthquake information (location, magnitude, moment-tensor, damage assessment) for the European-Mediterranean region and serves as European coordination platform for the further development and integration of seismological products.

86 seismological agencies from *57* countries contribute data to EMSC, which is governed by a Coordination Bureau and Executive Council.

On an other side, EMSC is also the *European infrastructure for Seismological products in EPOS*[1] and provides information for *General Public*, *Earthquake Eyewitnesses* and *Scientists* (Fig. 1).

In the following, we focus on information for Scientists and especially on how to access *Data and Data Products* (what is so called DDSS in the EPOS framework).

This *Hands-on Session* aims to present how to recover seismic informations from:

- Website (on occassion, in event base),
- Web Services (on demand, in batch),
- Streaming Data (on continuous, in near real time).

Online ressources used in the following are summarized in the Table (1).

The examples and exercises presented are in *Python* language (version *2.7*).

The following packages are required and must be installed before running codes: *requests*, *tornado*, *obspy* and *pymongo*.

All useful material for the exercises proposed (i.e. chuncks of Python code) is available online. Please, clone the following repository: `https://github.com/EMSC-CSEM/Hands-on-Session`

[1] EPOS - The European Plate Observing System: `https://www.epos-ip.org/`

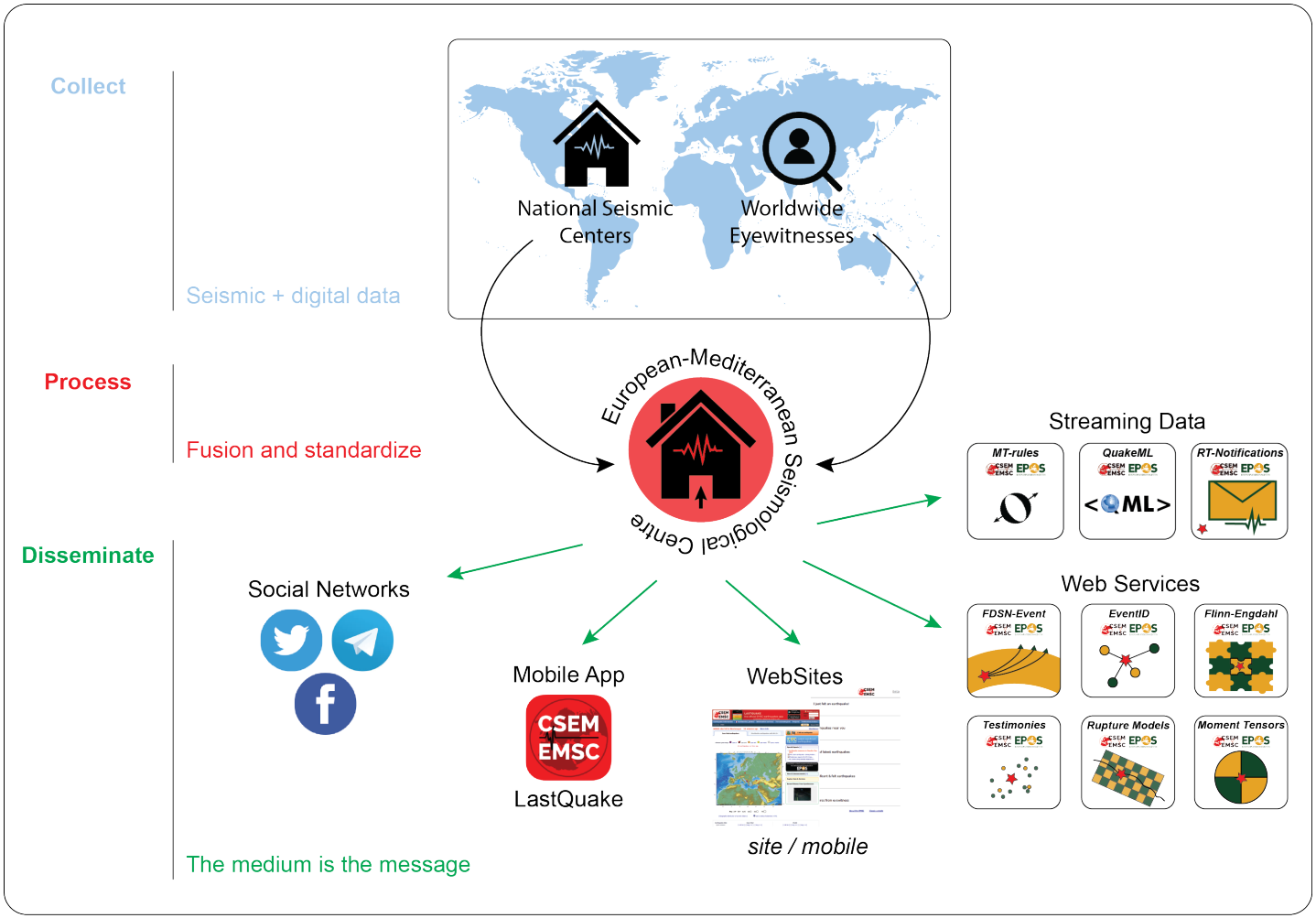

Figure 1: Overview of EMSC data Centre activities: EMSC collects seismic and digital data then, fusion and standardize information before to disseminate data and data products to *General Public*, *Earthquake Eyewitnesses* and *Scientists*. URLs of the dissemination channels are summarized in the Table (1).

Channel	URL
Twitter	https://twitter.com/lastquake
Facebook	https://fr-fr.facebook.com/EMSC.CSEM
Telegram	http://telegram.me/allquakes
Mobile App	https://itunes.apple.com + lastquake
	https://play.google.com + lastquake
Website	
Site	https://www.emsc-csem.org
	http://www.seismicportal.eu
Mobile	https://m.emsc.eu
Web Services	http://www.seismicportal.eu/webservices.html
Streaming Data	http://cerf.emsc-csem.org:8080
	http://www.seismicportal.eu/realtime.html
GitHub	https://github.com/emsc-csem

Table 1: URLs of the main dissemination channels used by the EMSC.

2 Operate the Websites

EMSC manages four main websites (Table 2):

Website	URL
Site	https://www.emsc-csem.org
Mobile	https://m.emsc.eu
Seismic Portal	http://www.seismicportal.eu
Test Platform	http://cerf.emsc-csem.org:8080

Table 2: URLs of the websites developped within EPOS.

In what follows, and as an example, we will focus on the recovery of seismic information from EMSC *Site* website.

2.1 Manually retrieves scientific data

Let's manually retrieve some parametric information (i.e. origin time and arrivals phases) composing a seismic event of interest (cf. Table 3) from *Site* website.

EMSC provides information for use by seismologists (https://www.emsc-csem.org/Earthquake/earthquake.php?id= + EVID + #scientific):

Date	Mw	Region	EVID
2011-03-11	9.0	Near east coast of Honshu, Japan	211443
2010-02-27	8.8	Offshore Maule, Chile	156286
2004-12-26	9.3	Off W coast of northern Sumatra	1974

Table 3: EVIDs of three earthquakes recorded in the EMSC information system.

e.g. https://www.emsc-csem.org/Earthquake/earthquake.php?id=211443#scientific

2.1.1 Exercice 1

Copy and paste the url above in a web browser to have a look of the datasets presented by EMSC.

Exercice #1

2.2 Automate figure retrieval

When an earthquake occurs, it can be useful to retrieve all the scientific information generated by EMSC's automated services.

For example, to make a scientific slideshow...in order to get a general view of the situation.

2.2.1 Exercice 2

We propose to test here a piece of code that helps to recover all the scientific material available from an EVID[2]/UNID[3] as input.

[2]EVID - event identifier for EMSC information system.
[3]UNID - unique identifier for Seismic Portal.

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/getevent_fromID_module_EMSC.py

» python *getevent_fromID_module_EMSC.py* 156286

Exercice #2

3 Using Web Services

In this section, we show how to combine the information coming from *web services* (ws) developped within EPOS (Fig. 4). For this, we propose a simple *use case* called **EUrythmics**[4].

Web Service	URL
FDSN-Event	http://www.seismicportal.eu/fdsn-wsevent.html
Flinn-Engdhal	http://www.seismicportal.eu/feregions.html
Rupture Models	http://www.seismicportal.eu/srcmodws
Testimonies	http://www.seismicportal.eu/testimonies-ws
EventID	http://www.seismicportal.eu/eventid
Moment Tensor	http://www.seismicportal.eu/mtws

Table 4: URLs of the web services developped within EPOS.

3.1 Use case description

After the occurence of a strong earthquake, as an EU scientist, I would like to:

- know all the relevant information from EMSC and,
- recover all seismological data and data products available.

[4]Eurythmy means "the art of movement"

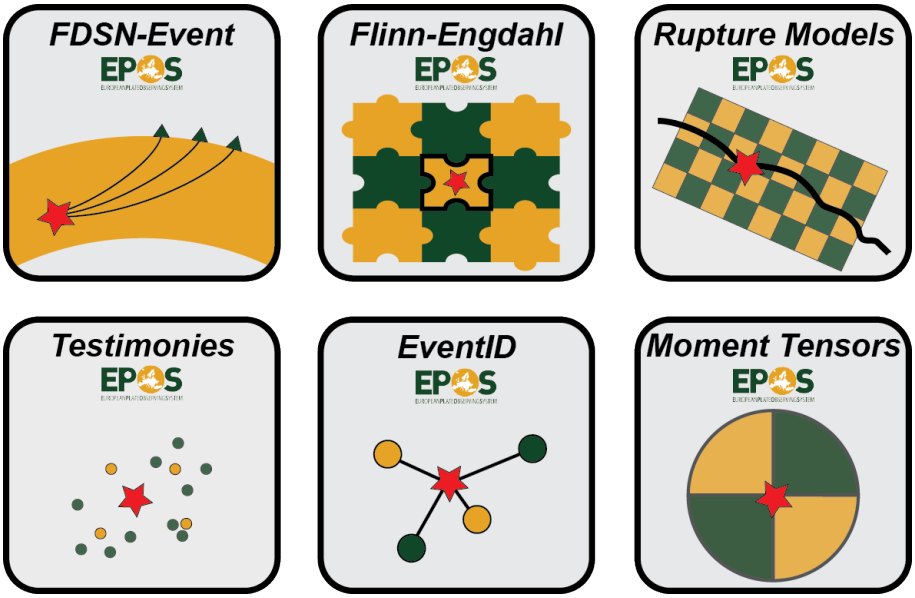

Figure 4: Web services developped in the framework of EPOS-Seismology. See table 4 for direct access to online services.

The use case is divided as follow:

1. Search for an earthquake (**i.e.** *FDSN-Event* ws)
   - Get origin parameters
   - Add origins and/or arrivals
2. Retrieve the Flinn Endghal region name (**i.e.** *Flinn-Engdhal* ws)
3. Collect Moment Tensors (**i.e.** *Moment Tensor* ws)
4. Collect Felt reports (**i.e.** *Testimonies ws*)
5. Identify if more informations are available from other seismological institutes (**i.e.** *EventID* ws)

Note that the Seismic Portal[5] is the main interactive gateway to access web services.

Figure 5: The FDSN-Event web service provides all the EMSC data available. Event information can include all origins and all arrivals reported to EMSC by its member institutions.

3.2 Search for earthquake parameters

As a first step, the only information you know is that the earthquake you are looking for is contained within a time slot:

e.g. between September 1st, 2017 and the end of October with a magnitude greater than 6.5.

This information is sufficient to be translated into a first FDSN-Event request:

url = http://www.seismicportal.eu/fdsnws/event/1/query?start=2017-09-01&end=2017-11-01&format=text&minmag=6.5

3.2.1 Exercice 3

Call (in command line[6]) the FDSN-event's web service:

```
wget -O result_Exercice_03.txt ${url}
cat result_Exercice_03.text
```

[5] http://www.seismicportal.eu/

[6] Note that each web service developped proposes also a query builder and a graphical user interface

Download and run the Shell script: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_03.sh

Exercice #3

The answer from the web service provides a list of *origin* parameters.

In the following, the earthquake studied will be the *Puebla*, Mexico (Mw 7.1) of September, 19th 2017 identifiable by its *unid*:

| Unid | 20170919_0000091 | Date time | 2017-09-19 18:14:38.5 UTC | Region | PUEBLA, MEXICO |

3.2.2 Get origin parameters

In order to know always more about the Puebla[7] earthquake, it possible to call again the FDSN-event's web service and to choose an ad hoc output data format (i.e *text*, *json* or *xml*) in the HTTP request.

3.2.3 Exercice 4

In the following, it is proposed to fetch *origin parameters* of the Puebla event:

```
main_event = {'unid':'20170919_0000091'}
```

Note that the main *Python functions* used in the following are attached in the *Annex 1*.

Three ways to call the FDSN-event's web service are presented which means three complementary ways to access the same differently formatted information:

[7] http://www.seismicportal.eu/eventdetails.html?unid=20170919_0000091

```
# ============================
# Download and parse (txt)
# ============================

url = "http://www.seismicportal.eu/fdsnws/event/1/query?eventid=
{unid}&format=text".format(unid=main_event['unid'])

res = geturl(url)
dataev = parsecsv(res['content'])
print("---parse-txt---\n{0}\n".format(dataev))
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_04_txt.py

```
# ============================
# Download and parse (json)
# ============================

url = "http://www.seismicportal.eu/fdsnws/event/1/query?eventid=
{unid}&format=json".format(unid=main_event['unid'])

res = geturl(url)
dataev = parsejson(res['content'])
eqinfo = dataev['properties']
print("---parse-json---\n{0}\n".format(eqinfo))
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_04_json.py

```
# ============================
# Download and parse (xml)
# ============================

url = "http://www.seismicportal.eu/fdsnws/event/1/query?eventid=
{unid}&format=xml".format(unid=main_event['unid'])

dataev = parsexml(url)
print("---parse-xml---\n{0}".format(dataev))
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_04_xml.py

Exercice #4 –

3.2.4 Add origins and/or arrivals

The FDSN-Event specifications define the parameter:

- includeallorigins (in order to include all origins),
- includearrivals (to include arrivals).

Note that in both case, these information are not included with the output *text* data format option.

In order to recover thoses additional parameters the *xml* output data format option shoud be chosen.

3.2.5 Exercice 5

Build the FDSN-Event's request that include all the origin parameters (i.e. *includeallorigins*) of the Puebla earthquake:

```
url = "http://www.seismicportal.eu/fdsnws/event/1/query\?eventid=
{unid}&format=xml&includeallorigins=true".format(unid=main_event['unid'
])
```

```
print "With origins"
dataev = read_events(url)
print dataev
for ev in dataev:
    print ev
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_05_xml.py

3.2.6 Exercice 6

Build the FDSN-Event's request that add to orign parameters all the arrivals (i.e. *includearrivals*) of the Puebla earthquake:

Exercice #5

```
url = "http://www.seismicportal.eu/fdsnws/event/1/query\?eventid=
{unid}&format=xml&includeallorigins=true&includearrivals=true".format(unid=
main_event['unid'])
```

```
print "With origins and arrivals"
dataev = read_events(url)
print dataev
for ev in dataev:
    print ev
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_06_xml.py

Exercice #6

3.3 Retrieve the Flinn-Endghal region name

This web service (Fig. 10) returns the Flinn-Endghal region name from a geographical location (longitude and latitude) as input, like the epicenter of an earthquake of interest.

3.3.1 Exercice 7

Build the Flinn-Endghal's request that returns the region names of the Puebla earthquake. Note that *origin parameters* of the Puebla event should be previously known:

```
url = "http://www.seismicportal.eu/fe_regions_ws/query?format=json&lat=
{lat}&lon={lon}"
```

Figure 10: The Flinn-Engdahl web service identifies the Flinn-Engdahl region from a geolocalisation entry point.

```
res = geturl(url.format(**eqinfo))
print parsejson(res['content'])
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_07_flinn.py

Exercice #7

3.4 Collect Moment Tensors

3.4.1 Exercice 8

Use the *Moment Tensors* web service (Fig. 12) to find all the Moment Tensors associated to the *Mw 7.1* Puebla earthquake.

Build the HTTP request: url = "http://www.seismicportal.eu/mtws/api/search?format=text&eventid={unid}"

And format the response of the call to the web service in order to obtain the full content or a more customised information:

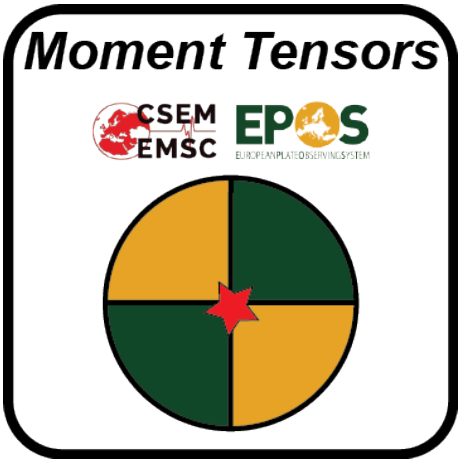

Figure 12: The Moment Tensors web service gives access to all the moment tensors collected at EMSC. The service's homepage shows a list of the latest moment tensors collected, sorted by earthquake, and with a graphical "beachball" representation. Users can build their own queries through a query builder.

```
** Full MT Contents **
res = geturl(url.format(unid=main_event['unid']))
print res['content'][:1000]

** Filtered Information **
mt_data = parsecsv(res['content'], usedict=True)
for mt in mt_data:
    print "Mt from {mt_source_catalog:10}, Strike: {mt_strike_1},
           Dip: {mt_dip_1}, Rake: {mt_rake_1}".format(**mt)
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_08_moment.py

Exercice #8

3.4.2 Exercice 9

Use ObsPy[8] to get a graphical representation (i.e. the traditional beachball plot) of a moment tensor of interest.

Suppose that you want to plot the beachball representation of the *USGS* solution:

Add the constraint *source_ catalog=USGS* to the previous *url* which is now rewrited: url = "http://www.seismicportal.eu/mtws/api/search\?source_catalog=USGS&eventid=20170919_0000091&format=text"

```
from obspy.imaging.beachball import beachball
# Get a list -- with only one element
mt = parsecsv(geturl(url)['content'], usedict=True)[0]
t = map(float, [mt['mt_mrr'], mt['mt_mtt'], mt['mt_mpp'],
                mt['mt_mrt'], mt['mt_mrp'], mt['mt_mtp']])
# Plot "USGS" Solution
beachball(t, width=200);
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_09_beachball.py

Exercice #9 –

3.5 Collect Felt Reports

Through EMSC's websites (site and mobile) and LastQuake mobile application, eyewitnesses are encourage to evaluate the earthquake's intensity they have felt. All this dataset gathered are made available via the Testimonies web service (Fig. 15).

[8] ObsPy is an open-source project dedicated to provide a Python framework for processing seismological data.

Figure 15: The Testimonies web service allows to recover all the felt reports collected from eywitnesses during an earthquake.

The search parameters are simple and looks like the other web service specifications. The main difference is found in the output data format. The only parameters allowed are *text* and *json* format.

3.5.1 Exercice 10

Build the HTTP request to call the *Testimonies* web service: url = http://www.seismicportal.eu/testimonies-ws/api/search?eventid={unid}&format=json

And format the response of the call to the web service in order to obtain the number of testimonies to recover:

```
tdata = parsejson(geturl(url.format(**main_event))['content'])}
print tdata
```

In a second step, modify slightly the url in order to recover intensity data points: url= http://www.seismicportal.eu/testimonies-ws/api/search?unids=[{unid}]&includeTestimonies=true

Note that the url's field *includeTestimonies* is added and set up *true*.

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_10_testimonies.py

Exercice #10

3.6 Mapping the event identifier

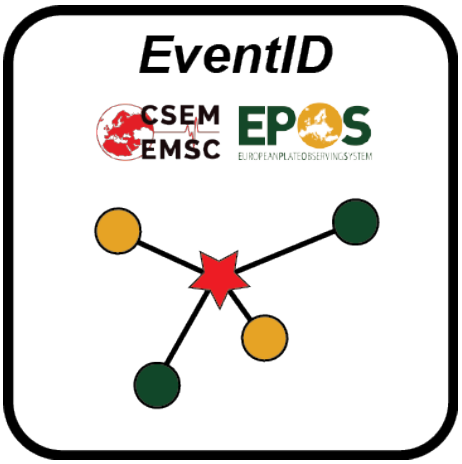

Figure 17: The EventID web service dynamically maps event identifiers to allow the identification of a same event between different seismological institutions.

All events of the *Seismic Portal* are assosciate to an *Unique Identifier* (UNID). And, on an other hand, many seismological institutes associate also an identifier (ID) to their seismic events. The basic idea of the *EventID*'s web service (Fig. 17) is to map dynamically all together the identifiers, for each seismic event of interest.

3.6.1 Exercice 11

In this exercise, we propose to use the UNID of the Puebla earthquake, in order to retrieve all the identifiers (IDs) of the partner seismological institutes. And to recover also in the same time through the URLs built the complementary seismological informations to the one previously collected on the Seismic Portal.

The *UNID* of the *Puebla, Mw 7.1*, Mexican earthquake is: *20170919_0000091*.

Partner seismological institutes can be: INGV, ISC, USGS, ...

As a first step, build the EventID's url:

```
url = "http://www.seismicportal.eu/eventid/api/convert?source_id=
{id}&source_catalog={source}&out_catalog={out}&format=text"
```

Then, call the web service and parse the response:

```
res = geturl(url.format(id=main_event['unid'],
                        source='UNID', out='all'))

iddata = parsecsv(res['content'], usedict=True)

for l in iddata:
    print "\n* Institute {#catalog:5}, ID: {eventid}\n
           url: {url}".format(**l)
```

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/Exercice_11_eventid.py

Exercice #11

As a way to discover more data and data products on the Mw 7.1 Puebla earthquake visite the URLs of the table (5).

Institute	**URL**
EMSC	https://www.emsc-csem.org/Earthquake/earthquake.php?id=619258
USGS	https://earthquake.usgs.gov/earthquakes/eventpage/us2000ar20
INGV	http://cnt.rm.ingv.it/event/17111031

Table 5: URLs obtained from the UNID (20170919_0000091), of the Mw 7.1 Puebla earthquake, mapped to the identifiers of other seismological institutes.

4 Listen Streaming Data

Within the EPOS project, EMSC tests and uses a new communication system called *HMB*. HMB is the acronym for *HttpMsgBus* which is a general-purpose HTTP-based message bus developed by Geofon[9]. This messaging uses the classical TCP port 80, widely open on networked computers.

This newest standardized way of exchanging offers the possibility to deliver the same information, at the same time, to all Seismological Institutes who are listening the bus (Fig. 19).

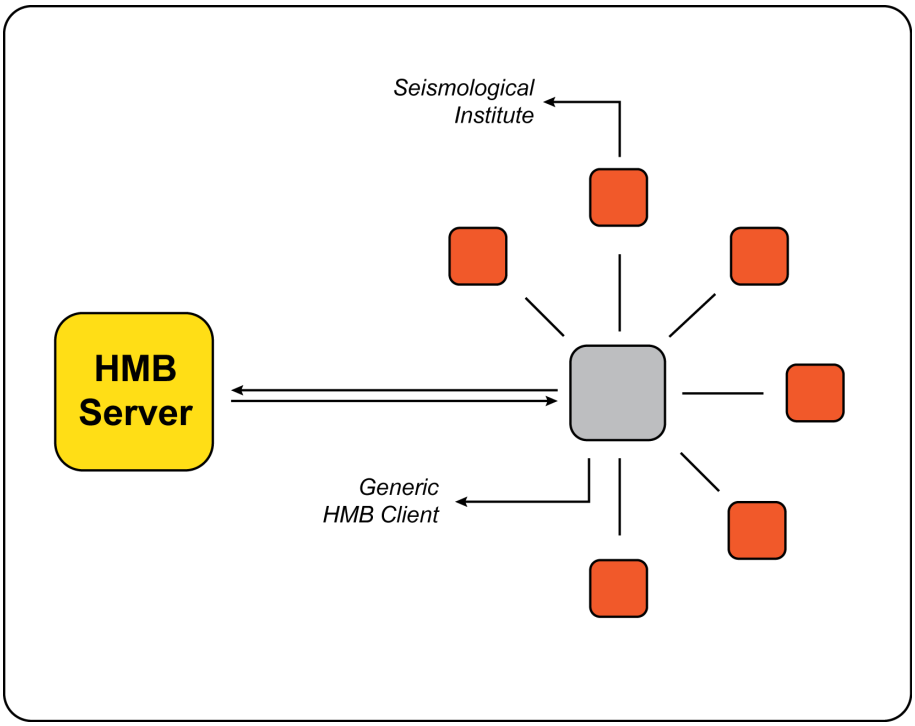

Figure 19: EMSC starts to disseminate real-time information through the HMB channel: HMB server is hosted at EMSC and distant clients can interact to the same bus with a common Python client. Transparency, rapidity and robustness as well as standardization are the four goals envisioned to foster the synergy between Seismological Institutes in Europe and to enhance their interoperability thanks to HMB messaging system.

In the following, it is proposed to interact with the HMB communication

[9] http://doi.org/10.5880/GFZ.2.4.2016.001

system where two types of near real-time applications are presented:

- The first one allows to retrieve a QuakeML stream and,
- The second one offers the possibility to recover (in function of dedicate ranking rules) a stream of Moment Tensors.

In a second step, we propose to test the Websocket protocol in order to interact directly with the Seismic Portal and to recover stream of earthquake events.

4.1 QuakeML data stream

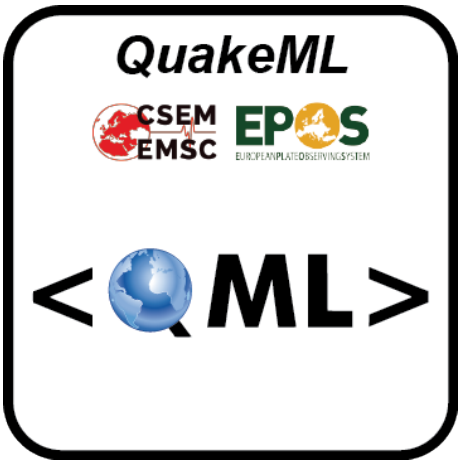

Figure 20: The QuakML data stream is a real time application of the EPOS-Seismology Test Platform that allows to recover, for each seismic event entering or beeing updated in the EMSC's information system, a QuakeML message.

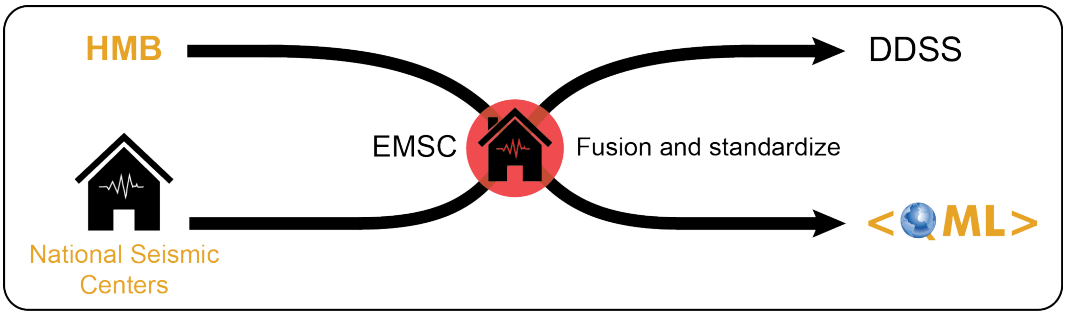

Figure 21: EMSC collects from National Seismic Centers parametric data (source parameters and phase picking). In a second step, EMSC fusion and standardize data. Then, seismic information is disseminates at the same rhythm as EMSC is digesting it: a QuakeML is automatically published on the HMB bus when an event is processed or updated.

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/receive_generic_QMLtest.py

4.2 Moment Tensors data stream

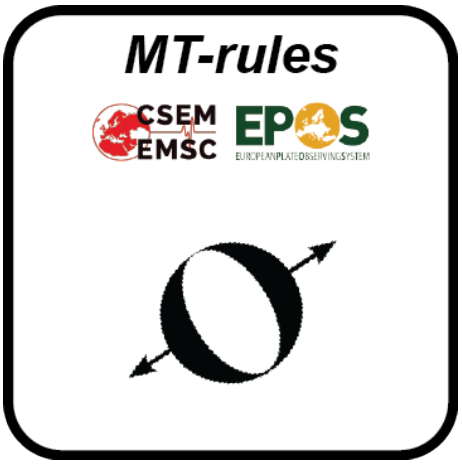

Figure 22: The Moment Tensors data stream is a real time application of the EPOS-Seismology Test Platform that allows to recover, for each seismic event entering or beeing updated in the EMSC's information system, a JSON message.

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/receive_generic_MTtest.py

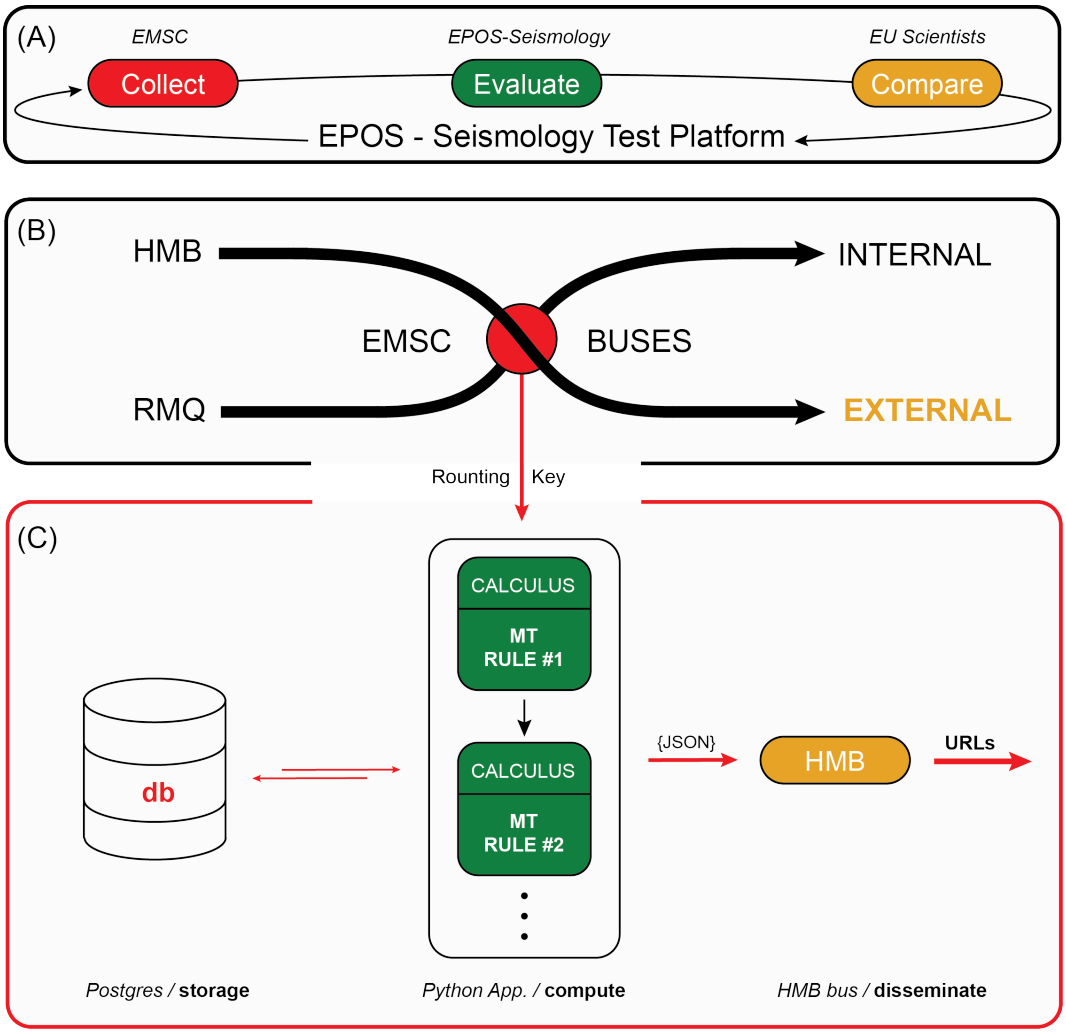

Figure 23: Diagram illustrating the application that streams Moment Tensors. Rules are calculated when a new event is published or updated on the internal EMSC's bus. There is a continuous interaction between buses, information database (for memory) and standardization process in order to produce a JSON message.

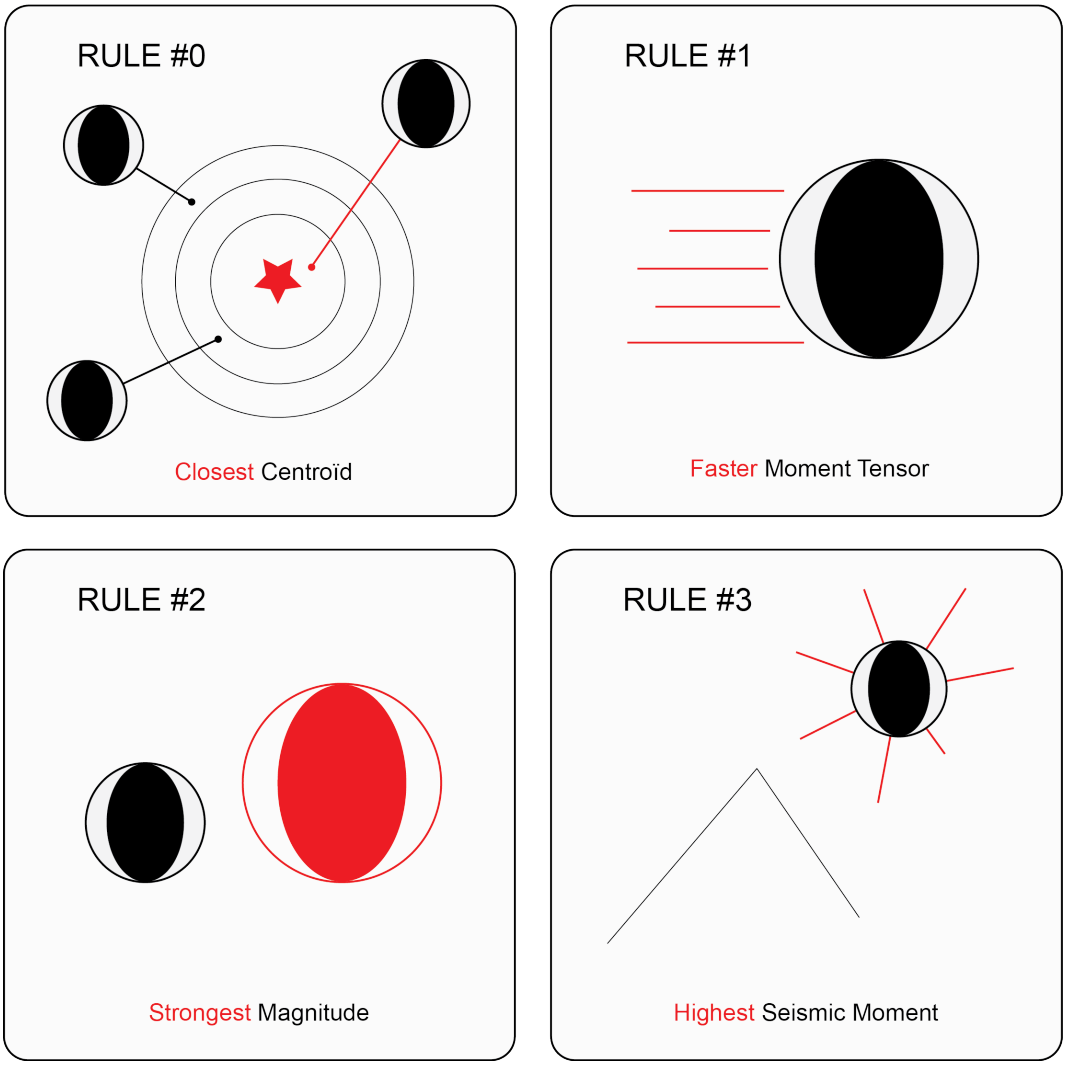

Figure 24: Four calculation rules are currently implemented in the Moment Tensor application hosted on the EPOS-Seismology Test Platform. Thoses rules are used for ranking the Moment Tensors stacked in the JSON files stream and published through HMB bus to Seismological Institutes.

4.3 Earthquake parameters data stream

Figure 25: The Web socket earthquake paremeters is a (near) real time application of the EPOS-Seismology Test Platform that allows to recover, for each seismic event entering or beeing updated in the EMSC's information system, a JSON message.

Using the Websocket protocol, users can be notified in (near) real-time of new events. A Json message is sent when an event is inserted or updated (Fig. 25).

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/near_Realtime_Notification.py

Download and run the Python code: https://github.com/EMSC-CSEM/Hands-on-Session/blob/master/seismicportal_listener.py

Annex 1 - Useful Python functions

```
# ------------------------------------------------
# Main Python functions used in EUrythmics
# ------------------------------------------------
# Func-Download
import requests
def geturl(url):
    res = requests.get(url, timeout=15)
    return {'status': res.status_code,
            'content': res.text}

# Func-Parse (txt)
import csv
from io import StringIO
def parsecsv(txt, usedict=False):
    if usedict:
        parser = csv.DictReader(StringIO(txt), delimiter='|')
    else:
        parser = csv.reader(StringIO(txt), delimiter='|')
    return [line for line in parser]

# Func-Parse (json)
import json
def parsejson(txt):
    return json.loads(txt)

# Func-Parse (xml)
from obspy import read_events
def parsexml(url):
   return read_events(url)
# ================================================
```