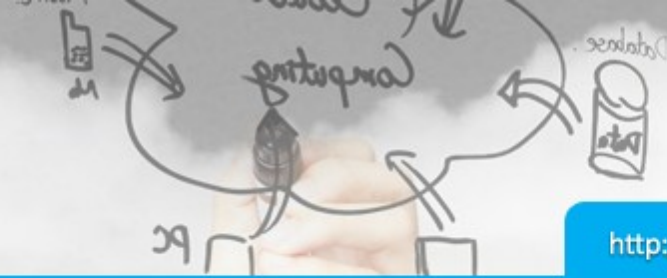




# Moteur de règles et plugins

Jérôme Pansanel et Emmanuel Medernach  
23 mars 2021



## Crédits

Cette présentation est basée sur la présentation cadre d'iRODS réalisée par Jason Coposky (directeur exécutif, consortium iRODS) :

- <https://slides.com/jasoncoposky>

## Anatomie d'un plugin pour le moteur de règle

### Une création récente

- Fonctionnement sous forme de plugin depuis la v4.2.0
- Permet d'étendre le moteur de règles (par ex. en utilisant de nouveaux langages)
- Plusieurs moteurs peuvent fonctionner de manière concurrente
- Chaque plugin doit définir 7 opérations :
  - start
  - stop
  - rule\_exists
  - list\_rules
  - exec\_rule
  - exec\_rule\_text
  - exec\_rule\_expression

## Fonctionnement du moteur

### Un outil flexible

- Une politique est créée de manière dynamique (dynamic policy enforcement point) avant et après chaque opération. Il correspond aux règles avec "\_pre" et "\_post"
- Le moteur va rechercher les règles avec le même nom que le PEP et l'exécuter.
- Dans une installation de base, 1200 PEPs dynamiques sont activées par une commande tel que **ils**. La majorité d'entre elles ne sont pas définies
- En cas de correspondance avec une PEP, elles sont exécutées dans l'ordre de chargement. La première est appelée, et le code de retour est interprété par le moteur de plugin. En cas de succès, les autres règles sont ignorées.
- Sinon, le moteur continue à passer les autres règles en revue jusqu'à avoir un succès. Le code de retour de la dernière règle est retourné au client.

# Configuration du système de plugins pour le moteur de règles

## Fichier `/etc/irods/server_config.json`

- Définit au format JSON dans une liste ordonnée (*array*)

```

"plugin_configuration": {
  "rule_engines": [
    {
      ...
    },
    ...
  ],
  ...
]
}
    
```

- L'ordre est important, car la priorité est donnée au premier composant lu

### RULE ENGINE PLUGIN FRAMEWORK

#### PLUGIN

##### RULEBASE

RULE (PEP)  
RULE (PEP)  
RULE (PEP)

##### RULEBASE

RULE (PEP)  
RULE (PEP)

#### PLUGIN

##### RULEBASE

RULE (PEP)  
RULE (PEP)

# Configuration du système de plugins pour le moteur de règles

## Anatomie de l'objet JSON

```
{  
  "instance_name": "<UNIQUE NAME>",  
  "plugin_name": "<DERIVED FROM SHARED OBJECT>",  
  "plugin_specific_configuration": {  
    <ANYTHING GOES HERE>  
  }  
  "shared_memory_instance": "<UNIQUE SHM NAME>"  
}
```

## Configuration des plugins pour le moteur de règle

```
{
  "instance_name": "irods_rule_engine_plugin-irods_rule_language-
instance",
  "plugin_name": "irods_rule_engine_plugin-irods_rule_language",
  "plugin_specific_configuration": {
    "re_data_variable_mapping_set": [
      "core"
    ],
    "re_function_name_mapping_set": [
      "core"
    ],
    "re_rulebase_set": [
      "core"
    ],
    "regexes_for_supported_peps": [
      "ac[ ^ ]*",
      "msi[ ^ ]*",
      "[ ^ ]*pep_[ ^ ]*_ (pre|post)"
    ]
  },
  "shared_memory_instance": "upgraded_legacy_re"
}
```

# Static Policy Enforcement Points

## Les PEPs historiques

- Langage propre
- Syntaxe proche du C
- Il est possible d'utiliser des expressions régulières, des opérations, des dictionnaires et des Language Integrated General Query (LIGQ)
- Possibilité d'appeler des règles (par ex. une fonction) et des microservices
- Voir le contenu du fichier `/etc/irods/core.re`

```
acPostProcForPut() {
    if("ufs_cache" == $KVPairs.rescName) {
        delay("<PLUSET>1s</PLUSET><EF>1h DOUBLE UNTIL SUCCESS OR 6 TIMES</EF>") {
            *CacheRescName = "comp_resc;ufs_cache";
            msisync_to_archive("*CacheRescName", $filePath, $objPath );
        }
    }
}
```



## Dynamic Policy Enforcement Points

### Une création récente

- Commence par « pep »
- Voir à la fin du fichier `/etc/irods/core.re`
- Type de PEPs recommandée pour faire fonctionner le moteur de plugins de manière optimale
- Pour obtenir la liste des PEPs dynamiques :  
[https://docs.irods.org/master/plugins/dynamic\\_policy\\_enforcement\\_points/](https://docs.irods.org/master/plugins/dynamic_policy_enforcement_points/)

```

pep_api_data_obj_put_post(*INSTANCE_NAME, *COMM, *DATAOBJINP, *BUFFER,
    *PORTAL_OPR_OUT) {
    writeLine("serverLog", *DATAOBJINP)
    writeLine("serverLog", *DATAOBJINP.destRescName)
}
    
```



# Questions ?

## Installation du moteur de règle Python

### Installation du paquet

```
$ sudo yum install -y irods-rule-engine-plugin-python
```

### Un coup d'oeil sur les nouvelles bibliothèques

```
$ ls /usr/lib/irods/plugins/rule_engines/  
libirods_rule_engine_plugin-cpp_default_policy.so  
libirods_rule_engine_plugin-irods_rule_language.so  
libirods_rule_engine_plugin-passthrough.so  
libirods_rule_engine_plugin-python.so
```

## Installation du moteur de règle Python

### Créer le fichier `/etc/irods/core.py`

```
$ sudo cp /etc/irods/core.py.template /etc/irods/core.py
```

### Éditer le fichier `/etc/irods/server_config.json`

- Insérer la nouvelle configuration de plugins avant le plugin du moteur de règle iRODS
- Cela permettra de servir les requêtes en premier

```
"rule_engines": [
  {
    "instance_name" : "irods_rule_engine_plugin-python-instance",
    "plugin_name" : "irods_rule_engine_plugin-python",
    "plugin_specific_configuration" : {}
  },
  {
    "instance_name": "irods_rule_engine_plugin-irods_rule_language-instance",
```

## Ajout d'un nouveau fichier de règle

### Éditer le fichier `/etc/irods/server_config.json`

```
"rule_engines": [
  ...
  ...
  {
    "instance_name": "irods_rule_engine_plugin-irods_rule_language-instance",
    "plugin_name": "irods_rule_engine_plugin-irods_rule_language",
    "plugin_specific_configuration": {
      "re_data_variable_mapping_set": [
        "core"
      ],
      "re_function_name_mapping_set": [
        "core"
      ],
      "re_rulebase_set": [
        "metadata",
        "core"
      ],
    ],
  },
],
```

```
$ cat /etc/irods/server_config.json | json_verify
```

## Ajout d'un nouveau fichier de règle

### Créer le fichier `/etc/irods/metadata.re`

```
add_metadata_to_objpath(*str, *objpath, *objtype) {  
    msiString2KeyValPair(*str, *kvp);  
    msiAssociateKeyValuePairsToObj(*kvp, *objpath, *objtype);  
}  
  
getSessionVar(*name, *output) {  
    *output = eval("str($"++*name++)");  
}
```

## Fichier /etc/irods/core.py

```
import exifread
import session_vars

def exif_python_rule(rule_args, callback, rei):
    file_path = str(rule_args[0])
    obj_path = str(rule_args[1])
    exiflist = []
    with open(file_path, 'rb') as f:
        tags = exifread.process_file(f, details=False)
        for (k, v) in tags.iteritems():
            if k not in ('JPEGThumbnail', 'TIFFThumbnail', 'Filename', 'EXIF
MakerNote'):
                exifpair = '{0}={1}'.format(k, v)
                exiflist.append(exifpair)
    exifstring = '%'.join(exiflist)
    callback.add_metadata_to_objpath(exifstring, obj_path, '-d')
    callback.writeLine('serverLog', 'PYTHON EXIF RULE complete')

def acPostProcForPut(rule_args, callback, rei):
    sv = session_vars.get_map(rei)
    phypath = sv['data_object']['file_path']
    objpath = sv['data_object']['object_path']
    if phypath[-4:] == '.jpg':
        callback.writeLine('serverLog', 'Exec EXIF Python Rule')
        remote_rule = "exif_python_rule('%s', '%s')" % \
            (phypath, objpath)
        callback.remoteExec('resource1-X.novalocal', '', remote_rule, '')
    callback.writeLine('serverLog', 'PYTHON - acPostProcForPut() complete')
```

## Tester, toujours tester !

### Test de la modification du fichier `/etc/irods/core.py`

```
$ python /etc/irods/core.py
Traceback (most recent call last):
  File "core.py", line 1, in <module>
    import exifread
ImportError: No module named exifread
$ wget \
http://archive.ubuntu.com/ubuntu/pool/universe/p/python-exif/python-
exif_2.1.2.orig.tar.gz
$ tar xzf python-exif_2.1.2.orig.tar.gz
$ cd exif-py-2.1.2/
$ python setup.py install
```



# Extraction des métadonnées

## Il faut un fichier avec des données EXIF

```

$ wget https://github.com/irods/irods_training/raw/master/stickers.jpg
$ iput stickers.jpg
$ ils -l stickers.jpg
  rbacon          0 rootResc;replResc;resource2      2157087 2021-01-06.21:03 &
stickers.jpg
  rbacon          1 rootResc;replResc;resource1      2157087 2021-01-06.21:03 &
stickers.jpg
$ imeta ls -d stickers.jpg
AVUs defined for dataObj /tempZone/home/rbacon/stickers.jpg:
attribute: Image Orientation
value: Horizontal (normal)
units:
----
attribute: EXIF SubjectArea
value: [2015, 1511, 2217, 1330]
units:
----
attribute: EXIF ColorSpace
value: sRGB
units:
...

```



# Questions ?