09. Rules User Training

Overview

- What are rules?

- Types of rules
    - PEP-based rules
    - User-made rules
    - Delay rules

- How do user-made rules work?

What are rules?

- Good data management rests on policies

- For example:
  - “Checksums need to be checked every night”
  - “After an object gets uploaded, technical metadata should be added”
  - “Important data should have backups”

Rules are a way of automating and enforcing these policies

Why do we need to automate?

Workload

- Managing large collections manually takes a lot of work

Consistency

- Humans make (small) errors, for example when adding metadata

Human	Computer
‘Cat’	‘Cat’
‘cat’	‘Cat’
‘a cat’	‘Cat’

Efficient, reliable and consistent data management

KU LEUVEN

Types of rules

Rules can be classified by the way they are triggered:

	PEP-based rules	triggered by events
	User-made rules	triggered by users
	Delay rules	triggered by time

PEP-based rules

- PEP = Policy Enforcement Point = an event
    - E.g. ‘when a data object is removed’
- Actions in the rule are automatically undertaken when this PEP occurs
- Meant for automating community-wide policies
- Only available for admins

User-made rules

- A script written and executed by the user
- Meant for personal use

Delay rules

- Both PEP-based and user-made rules can be programmed for delayed execution

- Rules get executed based on time

• At a specified time	e.g. ’the 20th of Oktober 2020 00:00’
• At a specified interval	e.g. ‘every hour’
• Combinations	e.g. ‘every day until the 1st of January 2021’
	Etc…

How do user-made rules work?

- Easiest way: storing your rules in a file (e.g. testRule.r)
- Executed with the iRule command:

```
irule  –F  testRule.r
```

$\xrightarrow{}$ indicates that the rule is in a file

How do user-made rules work?

- File written in the iRODS rule language
- Syntax:

```
rulename(condition){
        actions
}

input   INPUT
output OUTPUT
```

How do user-made rules work?

Example

```
greetingRule{
          writeLine(“stdout”, “Hello *name !”);
}
input *name=“Jan”
output ruleExecOut
```

How do user-made rules work?

- Example

Not every rule needs a condition:
this rule is triggered unconditionally

```
greetingRule{
        writeLine(“stdout”, “Hello *name !”);
}
input *name=“Jan”
output ruleExecOut
```

Print output to the screen (instead of writing it to a log, for example)

- Output

Hello Jan!

More useful scenarios

- Applying metadata to a whole collection
- Checking data checksums every night and sending a mail if something is wrong
- Cleaning the trash folder every two weeks
- …

You can do this!

The next tutorial will teach you how…