1 Primo esempio

In questo primo esempio si analizza una semplice situazione nella quale vengono fatte due richieste di accesso da parte di una certo "id". Il sistema avrà uno stato, dichiarato nel PAS, formato da un semplice intero, inizializzato a 0, chiamato "counter". Questo counter verrà incrementato di uno, ogni volta che l'accesso sarà consentito. L'accesso non sarà più consentito quando questo counter arriva a 5.

```
Request:{ Request1
  (name/id , "Lucrezia")
}
Request:{ Request2
  (name/id , "Lucrezia")
}

Policy example < permit-overrides
  target:equal("Lucrezia",name/id)
  rules:
    Rule access (
      permit target: less-than(status/counter, 5))
    obl:
      [permit M add(int counter, 1)]
>

PAS {
  Combined Decision : false ;
  Extended Indeterminate : false ;
  Java Package : "example" ;
  Requests To Evaluate : Request1, Request2 ;
  pep: deny-biased
  pdp: permit-unless-deny
  status: [(int counter = 0)]
  include accessNumber

}
```

2 Secondo Esempio

Questo esempio mostra come lo stato può essere rappresentato anche da un booleano. Il caso è simile al precedente, ovvero una certa persona, identificata da un certo id, tenta di accedere ad una risorsa, in questo caso un file, che può essere sia letto che modificato. In questo caso lo stato serve per avere un indicazione su quando il file è aperto in scrittura. Verrà autorizzato l'accesso in lettura, o scrittura, solo quando il file non è in scrittura, ovvero quando il booleano è false. Quando verrà autorizzata la richiesta di scrittura sul file, il booleano, precedentemente inizializzato a "False", verrà messo su "true".

```
Request:{ Request1
  (action/id, "write")
```

```
  (name/id , "Lucrezia")
}
Request:{ Request2
  (name/id , "Lucrezia")
  (action/id, "read")
}

PolicySet readWriteExample {permit-overrides
  target:equal("Lucrezia",name/id)
  policies:
    Policy read < permit-overrides
      rules:
        Rule read ( permit target: equal("read",action/id)
          && equal("false", status/isWriting) )
    >
    Policy write < deny-overrides
      rules:
        Rule  write (permit target: equal("write",action/id)
          && equal("false", status/isWriting)) >
    obl:
      [permit M flag(isWriting, true)]
}

PAS {
  Combined Decision : false ;
  Extended Indeterminate : false ;
  Java Package : "example" ;
  Requests To Evaluate : Request1, Request2 ;
  pep: deny-biased
  pdp: permit-unless-deny
  status: [(boolean isWriting = false)]
  include readWriteExample
}
```

3 Terzo esempio

In questo esempio si può vedere come viene gestito l'accesso su base temporale. Lo stato è rappresentato da un booleano, ed una data. Il booleano serve per indicare che si sta effettuando il primo accesso, mentre la data serve per memorizzare quando sarà possibile effettuare l'ultimo accesso. Quando verrà autorizzato il primo accesso, verrà eseguita una obligation che modificherà lo stato, salvando la data di ultimo accesso, e successivamente verrà messo il booleano a false. Quando si tenterà di fare un accesso successivo, si verificherà che la data di accesso, sia nel range della data salvata in precedenza.

```
Request:{ Request1
  (action/id, "read")
  (name/id , "Lucrezia")
}
```

```
Request:{ Request2
  (name/id , "Lucrezia")
  (action/id, "read")
}

PolicySet timeExample {  permit-overrides
  target:equal("Lucrezia",name/id)
  policies:
    Policy genericAccess < deny-overrides
      target:  less-than(status/lastAccess, today())
        && equal("false", status/firstAccess)
      rules:
        Rule genericAccess
        (permit target: equal("read",action/id) )
    >
    Policy firstAccess < deny-overrides
      target: equal("true", status/firstAccess)
      rules:
        Rule firstAccess
        (permit target: equal("read",action/id) ) >
    obl:
      [permit M sum-date(today + 30)]
}

PAS {
  Combined Decision : false ;
  Extended Indeterminate : false ;
  Java Package : "example" ;
  Requests To Evaluate : Request1, Request2 ;
  pep: deny-biased
  pdp: permit-unless-deny
  status: [(boolean firstAccess = false), (Date lastAccess)]
  include timeExample
}
```

4 Quarto esempio

In questo esempio viene usata una lista per memorizzare il comportamento passato. Il comportamento passato consiste nell'aver richiesto l'apertura di un determinato file. Questo, o eventuali n file, aperti verranno memorizzati in questa lista, e successivamente verrà garantito l'accesso solo a queste risorse.

```
Request:{ Request2
  (name/id , "Lucrezia")
  (action/id, "read")
  (file/id, "test.txt")
}
```

```
PolicySet funcExample {permit-overrides
  target:equal("Lucrezia",name/id)
  policies:
    Policy read <permit-overrides
      rules:
        Rule read ( permit target: equal("read",action/id)
          && equal("false", status/firstAccess))
    >
    obl:
      [permit M add-list(file/id, list), flag("false", firstAccess)]
    Policy funcControll < deny-overrides
      rules:
        Rule funcControll
        ( permit equal("true",list-element( status/funcList, file/id )))
    >

    obl:
      [deny M log("Funzione non permessa") ]

}

PAS {
  Combined Decision : false ;
  Extended Indeterminate : false ;
  Java Package : "example" ;
  Requests To Evaluate : Request1, Request2 ;
  pep: deny-biased
  pdp: permit-unless-deny
  status: [(list List), (boolean firstAccess)]
  include timeExample
}
```