CEP

GUÍA DE REFERENCIA

NOVIEMBRE 2014

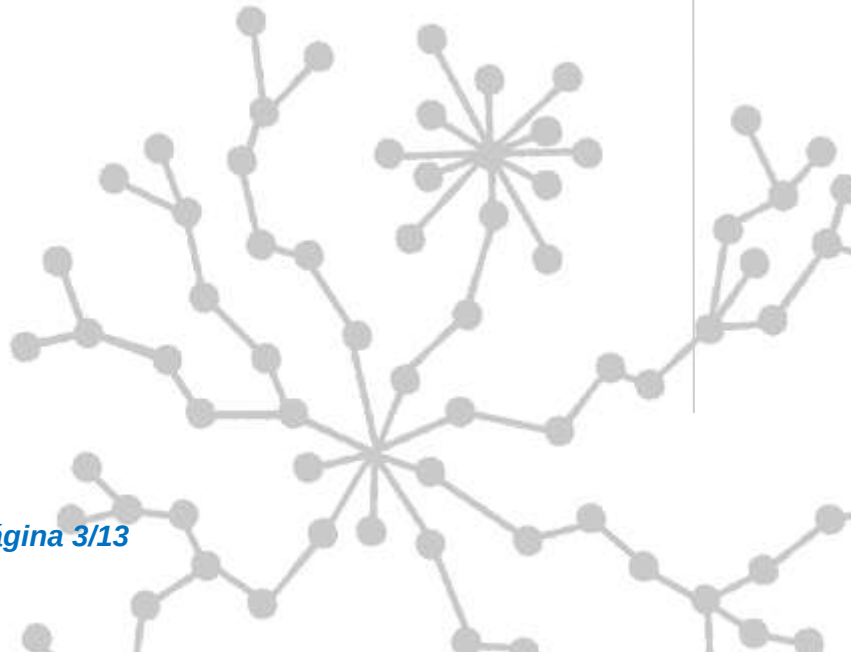
1 INDICE

2 INTRODUCCIÓN

2.1 Objetivos y alcance del presente documento

El presente documento es una guía de referencia que complementa y extiende la sintaxis del **WSO2 CEP**, que es el CEP de la *Reference Implementation* (RI) de Sofia2, y que ha sido elegido por cuestiones de rendimiento y licencia (**Apache 2.0**).

NOTA: Sofia2 soporta el funcionamiento con otros CEPs, ya que maneja los conceptos de forma independiente al CEP, en ese caso sólo sería necesario cambiar la sintaxis de la consulta CEP.



3 SINTAXIS DEL CEP DE SOFIA2

3.1 Conceptos previos

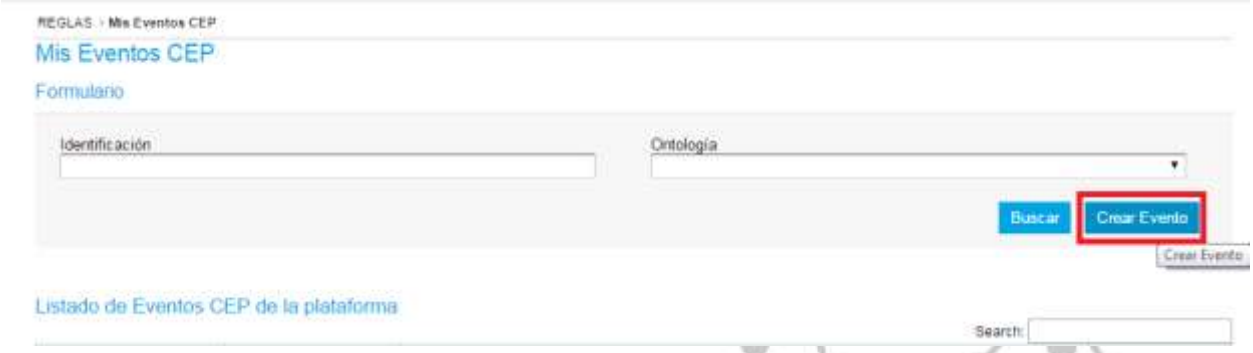
El CEP de Sofia2, basa su funcionamiento en la definición y captura de eventos.

- ***Event Stream***: Sucesión de eventos ordenados en el tiempo.
- ***Event Stream Definition***: Definición de un evento, representa un subconjunto de atributos de una ontología (nombre de atributo y tipo del mismo) y se define mediante un identificador único dentro de la plataforma.
- ***Event***: Un evento está asociado con un único *Event Stream*, y todos los eventos que forman el stream son iguales, es decir, tienen el mismo conjunto de atributos con el mismo tipo.
- ***Attribute***: atributo de la ontología, su nombre dentro de la definición del Evento es único y sólo soporta los siguientes tipos de datos: INT, LONG, BOOL, FLOAT, DOUBLE y STRING.
- ***Query***: Sentencia que combina eventos de entrada y que puede transformar o filtrar para devolver eventos de salida, utilizando para ello una sintaxis muy parecida a la de SQL.
- ***Partition***: Subconjunto de eventos procesado por una query.
- ***Partition Definition***: Definición de la partición, ésta puede hacerse basada en una variable o en un rango.

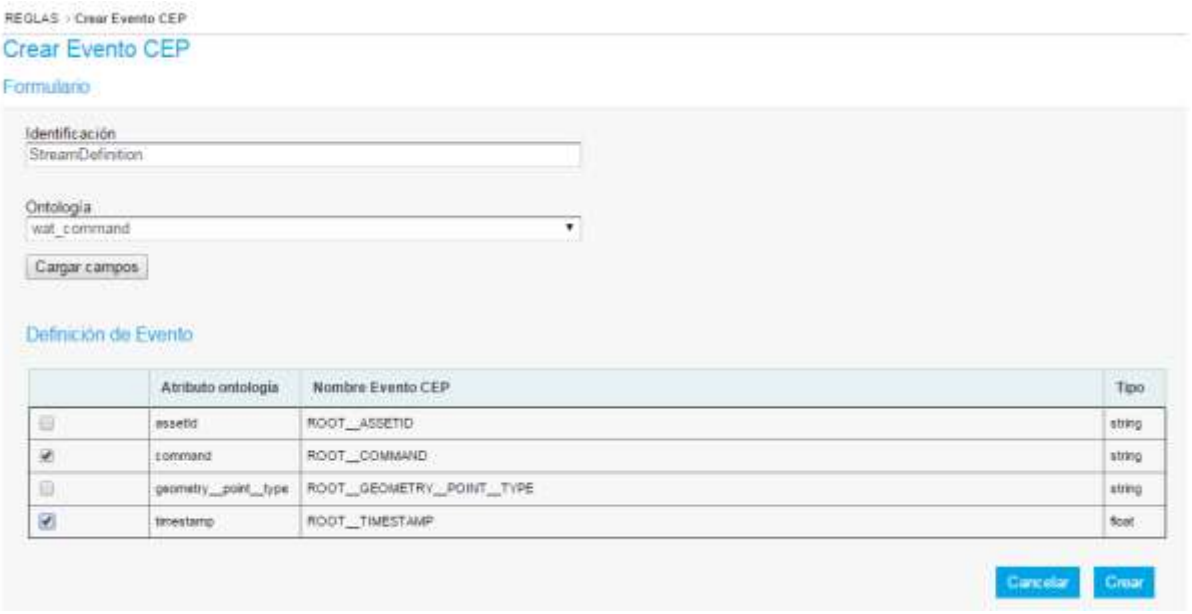
3.2 Definición de eventos en Sofia2

La definición de eventos en la plataforma mediante la consola es muy sencilla y abstrae de los detalles de la sintaxis del CEP.

Para ello en la opción de menú *Reglas* → *Mis Eventos CEP* podemos definir un evento de esta manera:



Dentro de la pantalla “Crear Evento CEP” le damos un identificador único y seleccionamos la ontología deseada, de la cual elegerimos únicamente aquellos atributos que nos interesen.



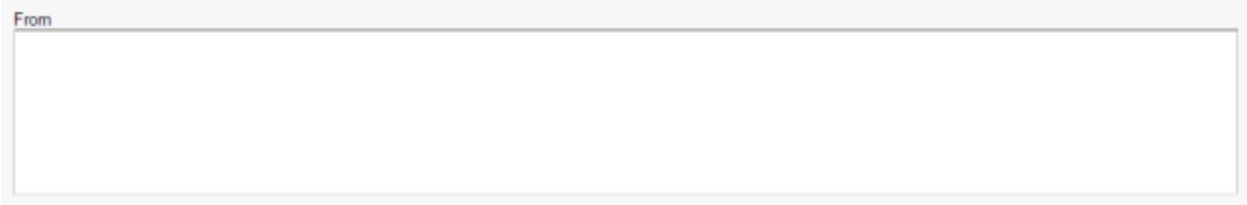
** El CEP de Sofia2 sólo mostrará aquellos atributos de la ontología cuyo tipo de datos es soportado por el CEP de la implementación usada (aquí la RI).*

3.3 Definición de Reglas en Sofia2 mediante la sintaxis del CEP.

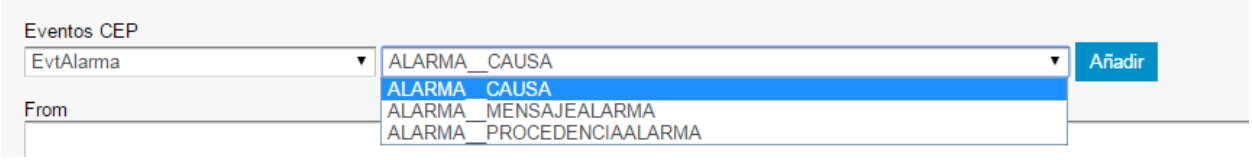
La definición de reglas en la plataforma se hace mediante la opción de menú *Reglas → Mis Reglas CEP*

Una vez dentro de la pantalla de alta de reglas podemos observar varios campos, todos ellos relacionados con la sintaxis del CEP.

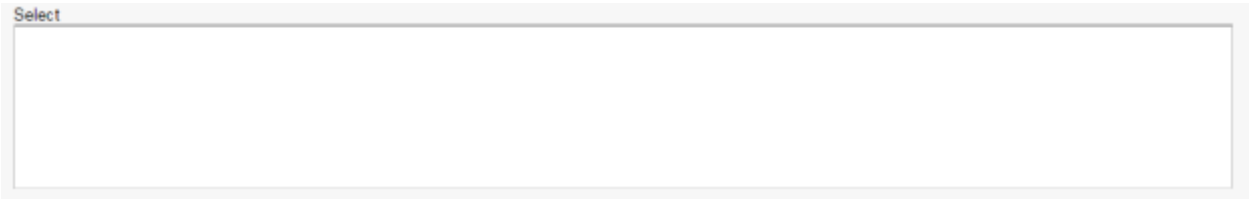
- **Campo FROM**: Mediante este campo se define la condición a aplicar al evento o los eventos de entrada a la regla o incluso la salida de otra regla. Aquí pueden definirse filtros, joins de eventos, patrones, secuencias, ventanas temporales, llamadas a funciones, etc…



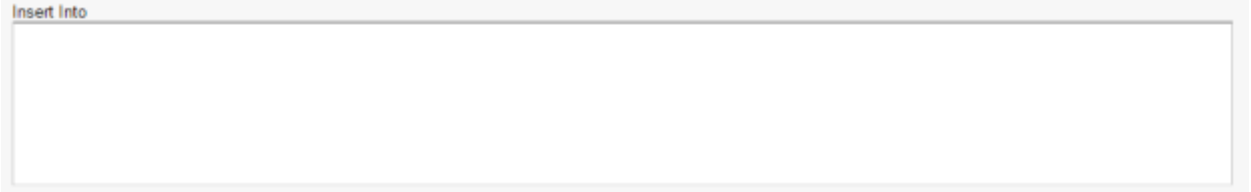
Los eventos disponibles en la plataforma para utilizarse en el FROM se muestran mediante el combo de eventos CEP, el cual nos muestra los eventos y los atributos del mismo.



- **Campo SELECT**: Este campo nos permite seleccionar aquellos atributos que nos interesen e incluso aplicar alguna transformación mediante funciones de agregación (count, avg, sum, max, min)



- **Campo INSERT INTO**: Se define el identificador (único en la plataforma) del evento de salida que va a contener los atributos de la claúsula SELECT



Una regla CEP en Sofia2 quedaría de la siguiente manera:

Que mediante la sintaxis del CEP sería:

```
from EVENTOESPIRA [ SENSOR_MEASURE >= 30]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA
```

3.3.1 Filters

Mediante los filtros podremos establecer condiciones en la clausula FROM utilizando los operadores de comparación:

- > , < , ==, >=, <=, ¡=

```
from EVENTOESPIRA [ SENSOR_MEASURE == 30]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA

from EVENTOESPIRA [ SENSOR_MEASURE >= 30]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA

from EVENTOESPIRA [ SENSOR_MEASURE != 30]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA
```

Mediante las palabras reservadas:

- *instanceof*, *contains*

```
from EVENTOESPIRA [ SENSOR_MEASURE instanceof float]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA

from EVENTOESPIRA [ SENSOR__ESPIRA_VELOCIDADMEDIA contains 'km']
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA
```

O mediante los operadores lógicos:

- *and, or, not*

```
from EVENTOESPIRA [ SENSOR__ESPIRA_VELOCIDADMEDIA contains 'km' and SENSOR_MEASURE >= 30]
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA
```

3.3.2 Windows

Mediante las ventanas se establecen subconjuntos a partir de un stream de eventos de entrada, de esta manera se pueden manejar los eventos capturados en dicha ventana para hacer cálculos específicos. Existen varios tipos de ventanas

- **Length Window**: Ventana que capturan los últimos '*n*' eventos que llegan al sistema, si alguno de ellos cumple la condición se emite un evento.

  ```
  from EVENTOESPIRA [ SENSOR_MEASURE >= 30]#window.length(10)
  select SENSOR_MEASURE as MEASURE
  insert into SALIDAESPERA
  ```

  En el ejemplo si llega la secuencia de eventos que satisfacen la condición (que la medida sea igual o superior a 30) 50, 50, 20, 30, 40, 10, 60 la regla emitirá los eventos con las medidas 50, 50, 30, 40 y 60

- **Time Window**: Ventana de tiempo, capturan los eventos que llegan en un intervalo de tiempo determinado devolviendo un evento que cumpla la condición especificada durante ese intervalo.

  ```
  from EVENTOESPIRA [ SENSOR_MEASURE > 50]#window.time(5 min)
  select avg(SENSOR_MEASURE) as MEASURE
  insert into SALIDAESPERA
  ```

- **Time Batch Window**: Ventana de tiempo que procesa en *batch* los eventos que llegan en un intervalo de tiempo, es decir, acumula los eventos que cumplen la condición especificada durante ese intervalo y emite un evento con el resultado acumulado cuando finaliza el tiempo marcado por la ventana.

  ```
  from EVENTOESPIRA [ SENSOR_MEASURE > 50]#window.timeBatch(1 min)
  select avg(SENSOR_MEASURE) as MEASURE
  insert into SALIDAESPERA
  ```

- **Length Batch Window**: Ventana que procesa en batch los últimos '*n*' eventos que llegan al sistema acumulando aquellos que cumplen la condición especificada, no devuelve evento de salida hasta que no llega el n-ésimo evento que cumple la condición.

```
from EVENTOESPIRA [ SENSOR_MEASURE > 50]#window.legthBatch(5)
select avg(SENSOR_MEASURE) as MEASURE
insert into SALIDAESPERA
```

  En este ejemplo según vayan llegando eventos al sistema cuya medida supere el valor 50 se van acumulando, cuando llegue el evento número 5 que cumple dicha condición se emitirá un evento con la media acumulada de las medidas.

- **Unique window**:

- **First Unique window**: ventana que mantiene aquellos eventos que llegan al sistema y que son únicos de acuerdo al atributo indicado.

```
from EVENTOESPIRA#window.firstUnique(SENSOR_MEASURE)
select avg(SENSOR_MEASURE) as MEASURE
insert into SALIDAESPERA
```

  En el ejemplo sólo se devolverían aquellos eventos cuyas medidas llegasen por primera vez al sistema, si llegase la secuencia de eventos cuyas medidas fueran: 50, 60, 60, 70, 50, 40, 40, 30, 60, 100, 101 devolvería los eventos 50, 60, 70, 40, 30, 100 y 101

- **External Time window**: ventana de tiempo cuyo procesamiento no hace uso de la hora del sistema, si no que hace uso de una fecha que se le pasa como parámetro

```
from EVENTOESPIRA [ SENSOR_MEASURE > 50]#window.externalTime(timeStamp,5 sec)
select avg(SENSOR_MEASURE) as MEASURE
insert into SALIDAESPERA
```

  En el ejemplo el parámetro "timeStamp"´es la fecha base que toma la regla para calcular las ventanas de tiempo.

3.3.3 Joins

Los joins toman dos[1] eventos como entrada, cada stream de eventos tiene que tener una ventana asociada. La salida generada se compone de un evento de cada stream

```
from EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 30]#window.time(1 min) as streamIzda
join EVENTOESPIRA [SENSOR__MEASURE >= 25]#window.time(1 min) as streamDcha
select SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

En el ejemplo cada evento de cada evento del stream que cumpla la condición en la ventana de tiempo dada devolverá un evento de salida, si llega la siguiente secuencia de eventos con las medidas siguientes

EVENTOTEMPERATURA → 20, 31, 30, 15, 40, 42, 33

EVENTOESPIRA → 30, 10, 12, 12, 10, 15, 25, 26, 33

La salida devuelta constará del producto cartesiano de los eventos del stream EVENTOTEMPERATURA que cumplen la condición en la ventana de tiempo indicada con los eventos del stream EVENTOESPIRA que cumplen la condición en la ventana de tiempo indicada

```
[31, 30] [30, 30] [40, 30] [42, 30] [33, 30]
[31, 25] [30, 25] [40, 25] [42, 25] [33, 25]
[31, 26] [30, 26] [40, 26] [42, 26] [33, 26]
[31, 33] [30, 33] [40, 33] [42, 33] [33, 33]
```

En el ejemplo anterior los eventos de cualquiera de los dos streams que cumplan la condición "disparará" la regla para que devuelva un evento de salida. Sin embargo este ejemplo considera que los eventos de uno y otro stream llegan en orden secuencial, es decir, primero llegan los eventos del stream EVENTOTEMPERATURA y luego los del stream EVENTOESPIRA, es por ello que la salida resultante aparece ordenada.

En la sintaxis del CEP hay un join que permite disparar la regla sólo cuando llegan los eventos de uno de los streams, en este caso el que lleva acompañada la palabra reservada *unidirectional.*

```
from EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 30]#window.time(1 min) as streamIzda unidirectional join
     EVENTOESPIRA#window.time(1 min) as streamDcha
select SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

En el ejemplo con la llegada de los streams siguientes

EVENTOESPIRA → 31, 10, 12, 12, 10, 15, 25, 26, 33

EVENTOTEMPERATURA → 20, 31, 30, 15, 40, 42, 33

Sólo los eventos que llegan para el stream EVENTOTEMPERATURA harán que la regla devuelva los siguientes resultados:

```
[31, 20] [31, 31] [31, 30] [31, 15] [31, 40] [31, 42] [31, 33] [31, 26] [31, 33]
[33, 20] [33, 31] [33, 30] [33, 15] [33, 40] [33, 42] [33, 33] [33, 26] [33, 33]
```

Si el unidirectional se aplicase en orden inverso, es decir

```
from EVENTOESPIRA[SENSOR__MEASURE >= 30]#window.time(1 min) as streamIzda unidirectional join
     EVENTOTEMPERATURA#window.time(1 min) as streamDcha
select SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

El stream de salida sería:

```
[31, 31] [31, 10] [31, 12] [31, 12] [31, 10] [31, 15] [31, 25] [31, 26] [31, 33]
[30, 31] [30, 10] [30, 12] [30, 12] [30, 10] [30, 15] [30, 25] [30, 26] [30, 33]
[40, 31] [40, 10] [40, 12] [40, 12] [40, 10] [40, 15] [40, 25] [40, 26] [40, 33]
[42, 31] [42, 10] [42, 12] [42, 12] [42, 10] [42, 15] [42, 25] [42, 26] [42, 33]
[33, 31] [33, 10] [33, 12] [33, 12] [33, 10] [33, 15] [33, 25] [33, 26] [33, 33]
```

[1]*WSO2 CEP no soporta joins de más de dos eventos*

3.3.4 Patterns

Los patrones de procesado se basan en uno o más input streams, comprueban eventos o condiciones sobre eventos de entrada de un input stream, estos input stream deben ser identificados de manera unívoca.

```
from e1=EVENTOESPIRA[SENSOR__MEASURE >= 30] -> e2=EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 33]
select e1.SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, e2.SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

En el ejemplo los eventos de entrada se distinquen por los identificadores e1 y e2, y mediante el operador "**->**" va a obtener como stream de salida la medida de cada evento (e1 y e2) siempre que cumpla que ha llegado un evento en el input stream EVENTOESPIRA cuya medida sea igual o superior a 30 seguido de otro evento recibido en el input stream EVENTOTEMPERATURA cuya medida sea igual o superior a 33.

```
InputStreams -> e1[21] e2[33] e1[15] e1[34] e1[40] e1[21] e1[30] e2[35] e1[30] e2[32] e1[12] e1[15] e1[12] e1[45] e1[30] e2[34]

OutputStream ->  SALIDA_SENSORES[30, 35]
```

Esta regla sólo se ejecutará una vez, es decir, una vez detecte la ocurrencia marcada por el patrón no volverá a ejecutarse, para que detecte todas las ocurrencias es necesario indicarlo mediante la palabra reservada ***every***

```
from every(e1=EVENTOESPIRA[SENSOR__MEASURE >= 30] -> e2=EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 33])
select e1.SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, e2.SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

Y la salida sería:

```
InputStreams -> e1[21] e2[33] e1[15] e1[34] e1[40] e1[21] e1[30] e2[35] e1[30] e2[32] e1[12] e1[15] e1[12] e1[45] e1[30] e2[34]

OutputStream ->  SALIDA_SENSORES[30, 35] SALIDA_SENSORES[30, 34]
```

Además se puede indicar que esté "escuchando" eventos durante un tiempo determinado si añadimos la palabra ***within***

```
from every(e1=EVENTOESPIRA[SENSOR__MEASURE >= 30] -> e2=EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 33])
select e1.SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, e2.SENSOR__MEASURE as MEDIDA_SENSOR
within 3000
insert into SALIDA_SENSORES
```

También pueden indicarse el número de eventos que queremos detectar de cada stream y que valores de éstos queremos devolver en la salida.

```
from e1=EVENTOESPIRA[SENSOR__MEASURE >= 30]<2:> ->
     e2=EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 33]<3>
select e1[0].SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, e2[0].SENSOR__MEASURE as MEDIDA_SENSOR
insert into SALIDA_SENSORES
```

En el ejemplo queremos detectar la llegada de 2 o más eventos del input stream e1 cuya medida sea igual o superior a 30 (<2:>) seguida de la llegada de exactamente 3 eventos del input stream e2 cuya medida sea igual o superior a 33. Además sólo se obtiene en la salida el primer elemento de cada una de las series e1[0] y e2[0]

3.3.5 Sequences

Con las *Sequences* pueden detectarse secuencias de eventos en un determinado orden sin que entre medias se den otros eventos distintos. Las *Sequences* se caracterizan por:

- Usan uno o más streams de eventos.
- Como entrada toma una secuencia de eventos definidos por una expresión regular (*, cero o más ocurrencias, +, una o más ocurrencias, ?, como mucho una ocurrencia)
- A las secuencias definidas, una o más, es necesario asignarles nombres que les definan de manera unívoca.
- La salida generada se forma nombrando los atributos de la secuencia detectada con la palabra reservada ‘as’.

```
from e1=EVENTOESPIRA[SENSOR__MEASURE >= 30]+
     e2=EVENTOTEMPERATURA[SENSORTEMPERATURA__MEASURE >= 33]?
insert into SALIDA_SENSORES
          e1[0].SENSORTEMPERATURA__MEASURE as MEDIDA_SENSOR_TEMP, e2.SENSOR__MEASURE as MEDIDA_SENSOR
```

En el ejemplo se toman dos event streams para definir la siguiente secuencia:

Para el input stream **e1** se va a detectar la ocurrencia de ***uno o más eventos seguidos*** cuya medida es igual o superior a treinta seguido de, como máximo, un evento del input stream **e2** cuya medida es igual o superior treinta y tres, para generar la salida que toma el primer evento detectada de **e1** seguido del evento detectado en **e2**.

Para la siguiente secuencia se obtendría:

```
e1(20) -> e1(21) -> e1(30) -> e1(21) -> e1(31) -> e2(33) -> e1(15)
SALIDA_SENSORES -> [30, null] [31, 33]
```

3.3.6 Event Absence

Una de las funcionalidades implementadas en las últimas releases de Sofia2 en la detección de ausencia de eventos, para ello se ha añadido a la sintaxis del CEP de Sofia2 la palabra reservada *eventAbsence*, su uso junto a las ventanas de tiempo permite detectar en un intervalo determinado la NO llegada de eventos.

Veamos un ejemplo:

```
from EVENTOESPIRA [ SENSOR_MEASURE > 50]#window.sofia:eventAbsence(20000)
select SENSOR_MEASURE as MEASURE
insert into SALIDAESPERA
```

En el ejemplo, la query detecta que en un intervalo de tiempo de 20 segundos (20000 ó 20 sec) no van a llegar eventos cuya medida supere el valor cincuenta, si esto ocurre se emite un “evento vacio” que indica esta circunstancia.

En la secuencia anterior se aprecia como en el intervalo de 20 segundos (01:12:25 – 01:12:45) no ha llegado ninguna medida que supere estrictamente el valor 50, con lo cual, transcurrida la ventana de tiempo se emite un evento especial [EMPTY_EVENT] que indica que se ha detectado la ausencia de este evento. Sin embargo en la ventana de tiempo entre las 01:12:45 y las 01:13:05 ha llegado un evento cuya medida supera el valor 50 (56) transcurrida esa ventana temporal no se emite ningún evento “vacío”.