

## epia.ep1

```

/*****
 * The software in this package is published under the terms of the GPL license
 * a copy of which has been included with this distribution in the license.txt file.
 *****/

/**
 * "Event Processing In Action" sample application
 */

/**
 * Keep last known driver location (for every driver)
 *
 * We use a named window and feed it from the GPSLocation stream
 * as named windows can be used accross statements and possibly further queried through
 * EsperJDBC or external tools.
 *
 * http://esper.codehaus.org/esper-3.2.0/doc_20091026/reference/en/html/epl_clauses.html#named_overview
 */
create window GPSLocationW.std:unique(driver)
as select * from GPSLocation;

insert into GPSLocationW
select * from GPSLocation;

/**
 * Specification subject to interpretation
 * - What if no driver nearby
 *   -> nothing happens - we'll have an Alert later
 * - What if driver join the system and/or gets nearby after DeliveryRequest was
   received
 *   -> the driver will not be able to join the already started bid
 *
 * We join incoming DeliveryRequest with known GPSLocation (from the named window).
 * The output stream is enriched with the 'manual' property out of the domain model
 (store, manual assignment).
 * The minimum ranking is tested out of the domain model as well.
 * The syntax below is using a Java based domain model (see also esper.cfg.xml)
 * and shows commented how we would do for an SQL RDBMS based domain model.
 *
 * http://esper.codehaus.org/esper-3.2.0/doc_20091026/reference/en/html/epl_clauses.html#joining_method
 *
 * http://esper.codehaus.org/esper-3.2.0/doc_20091026/reference/en/html/epl_clauses.html#histdata_overview
 */
insert into BidRequest(requestId, store, location, pickupTime, deliveryTime,
storeManual)
select d.requestId, d.store, d.location, d.pickupTime, d.deliveryTime, s.manual
from
    DeliveryRequest d unidirectional,
    GPSLocationW g
    //,sql:DomainDB['select ranking from Driver where driver = ${g.driver} and ranking
> ${d.minimumRanking}']
    ,method:Domain.driverRankLookup(g.driver) r
    ,method:Domain.isStoreManualLookup(d.store) s
where Geo.distanceKM(g.location, d.location) < 10
and r.ranking >= d.minimumRanking;

/**
 * Keep all DeliveryBid in a named window

```

epia.ep1

```
* Based upon completion the elements will get updated / removed later
*/
create window DeliveryBidW.win:keepall() as select requestId, store, driver,
pickupTime, 0 as ranking from DeliveryBid;

/**
* Feed the DeliveryBid and enrich the DeliveryBid with the driver ranking
*/
insert into DeliveryBidW select requestId, store, driver, pickupTime, ranking
from
    DeliveryBid d
    //,sql:DomainDB['select ranking from Driver where driver = ${g.driver}'] r
    ,method:Domain.driverRankLookup(d.driver) r;

/**
* Specification subject to interpretation
* - we keep the driver per pickup time and not per ranking, or per a more complex
domain level logic
*
* For an automatic store, 2 minutes after a BidRequest we assign the driver with the
earliest pickup
*/
on pattern[every b=BidRequest(storeManual=false) -> timer:interval(2 min)]
insert into Assignment
select d.*, b.deliveryTime as deliveryTime
from
    DeliveryBidW d
where requestId = b.requestId
order by pickupTime asc limit 1;
// BUG - onSelectExpr does not support limit clause

/**
* Specification subject to interpretation
* - we keep the top 5 drivers per ranking
*
* For a manual store, 2 minutes after a BidRequest we assign the top 5 drivers per
ranking
*/
on pattern[every b=BidRequest(storeManual=true) -> timer:interval(2 min)]
insert into AssignmentManual
select d.* from DeliveryBidW d where requestId=b.requestId order by ranking desc limit
5;

// TODO
// The AssignmentManual should be dealt with through some kind of store admin so as to
produce one Assignment out of the 5 AssignmentManual of a BidRequest

/**
* On completion of delivery we can remove from the DeliveryBid named window.
* This ensures that the DeliveryBid named window is kept around for whatever store
admin manual workflow.
*/
on DeliveryConfirmation dc
delete from DeliveryBidW d where d.requestId = dc.requestId;

// TODO - we likely need to remove from DeliveryBidW after a while - no matter
DeliveryConfirmation
// This depends on how the system would deal with Alerts and allow a store admin to
escalate / reassign / re-emit the bid.
// In this version the non completed DeliveryBid will hang around.

/**
* A named window where to keep various alerts
*

```

epia.ep1

```
* driver can be "" when information not available
*/
create window AlertW.win:keepall() as (requestId int, message String, driver String);

/**
 * No bid after 2 mins of a request
 */
insert into AlertW(requestId, message, driver)
select d.requestId, "no bidder", ""
from pattern[
    every d=DeliveryRequest -> (timer:interval(120 sec) and not DeliveryBid(requestId =
    d.requestId))
];

/**
 * No assignment on a manual store after 1 min of selected top drivers
 */
insert into AlertW(requestId, message, driver)
select a.requestId, "not assigned", ""
from pattern[
    every/*-distinct(a.requestId)*/ a=AssignmentManual -> (timer:interval(1 min) and
    not Assignment(requestId = a.requestId))
];

/**
 * Not picked up after 5 mins (300 secs) of the driver proposed pickup time
 */
insert into AlertW(requestId, message, driver)
select a.requestId, "not picked up", a.driver
from pattern[
    every a=Assignment -> (timer:interval(300 + (a.pickupTime-current_timestamp)/1000)
    and not PickupConfirmation(requestId = a.requestId))
];

/**
 * Not picked up after 10 mins (600 secs) of the request target delivery time
 */
insert into AlertW(requestId, message, driver)
select a.requestId, "not delivered", a.driver
from pattern[
    every a=Assignment -> (timer:interval(600 +
    (a.deliveryTime-current_timestamp)/1000) and not DeliveryConfirmation(requestId =
    a.requestId))
];

// TODO
// The specification is unclear on how an assignment gets finalized (success or not).
// We could keep all Assignment in a named window and flush them appropriately
// This would then be a starting point for computing driver statistics
```