FInest – **F**uture **In**ternet **e**nabled optimi**s**ation of **t**ransport and logistics networks

D6.4

Prototypical implementation of event processing component

Project Acronym	FInest	
Project Title	Future Internet enabled optimisation of transport and logistics networks	
Project Number	285598	
Workpackage	#WP6 Proactive event driven monitoring	
Lead Beneficiary	IBM	
Editor	Fabiana Fournier	IBM
Contributors	Sarit Arcushin	IBM
	Guy Sharon	IBM
Reviewers	Rod Franklin	KN
	Clarissa Marquezan	UDE
Dissemination Level	Public	
Contractual Delivery Date	30/3/2013	
Actual Delivery Date	30/3/2013	
Version	V1.0	

Abstract

This report presents the forth deliverable of work package 6 "Proactive Event Driven Monitoring". The aim of this module within FInest is to provide event-driven monitoring capabilities to facilitate the end-to-end visibility of logistics processes and real-time information regarding actual execution. To this end, a proof-of-concept application has been built based on the FISH use case.

This document is complementary to the proof-of-concept demo and provides additional information to form a comprehensive understanding of how the event processing module works, its capabilities, and the results obtained.

The use case selected applies a wide range of rules operators and demonstrates the potential benefits of applying event processing techniques to logistic processes. The applied rules have been validated by the domain partners in terms of their relevance and correctness.

Document History

Version	Date	Comments
V0.1	20/11/12	First draft
V0.2	20/1/13	Updates after project technical meetings
V0.3	15/2/13	Updates after the technical face-to-face meeting in Essen
V0.4	1/3/13	Updates after project face-to-face meeting in Hamburg
V0.5	15/3/13	Updates after review
V1.0	30/3/13	Submitted version

Table of Contents

List of Tables

List of Figures

Acronyms

Acronym	Explanation
BCM	Business Collaboration Module
CEP	Complex Event Processing
ECM	E-Contracting Module
EPM	Event Processing Module
EPN	Event Processing Network
GE	Generic Enabler
FInest	Future Internet enabled optimisation of transport and logistics networks
IoT	Internet of Things
POC	Proof Of Concept
TCP	Transport Chain Plan
TEP	Transport Execution Plan
TPM	Transport Planning Module
WP	Work Package

1. Introduction

The FInest (Future Internet enabled optimisation of transport and logistics networks) project goal is to develop a platform to support optimizing the collaboration and integration within transport and logistics business networks. Four core modules have been designed to enable these collaborations: BCM (Business Collaboration Module), TPM (Transport Planning Module), ECM (E-Contracting Module), and EPM (Event Processing Module). To demonstrate potential benefits of the envisioned platform, four proof-of-concepts (POCs) have been built around one selected scenario, the FISH use case[1].

The goal of the Event Processing Module (EPM) is to detect and alert about situations of interest by the analysis of events data in real-time. This document describes the EPM implementation of the FISH use case and it is complementary to the EPM proof-of-concept.

It is important to note that an initial set of rules has been identified and submitted in D6.3 "Initial technical specification of event processing component"[2]. With this regard, this document provides a validation to the specification in D6.3 and a refinement to the event processing rules and Event Processing Network (EPN) defined in D6.3.

This document is organized as follows: Section 2 gives an overall overview of the event processing module proof-of-concept. Section 3 details the FISH use case implementation in terms of events and rules definitions, inputs, and outputs. It also shows the actual running for a specific set of input events. We conclude the report with a summary.

2. The EPM proof-of-concept

2.1. EPM interactions and main components

For background on event processing and a detailed specification please refer to D6.3[2]. We include here the minimal descriptions required to understand the demo constructs and to keep this document self-contained.

A high-level overview of the EPM and its interaction is depicted in Figure 1.

The EPM is composed of two main modules: the rules and events definitions created at build-time and the Complex Event Processing (CEP) engine that matches the events at run-time to the definitions and notifies regarding detected situations. Three potential sources of events have been identified in FInest: IoT (Internet of Things) sensors, backend systems, and events resulting from the actual execution of a transport plan and emitted from the BCM to the EPM.

In general, the EPM interacts mainly with the BCM that, during execution, passes relevant events to the EPM. The latter, in turn, notifies the BCM of any detected situation that can be alerted to the user. The EPM can also alter the information model of the business objects managed by the BCM. The EPM directly notifies the FInest front-end regarding proactive alerts,

[1] For a detailed description of this use case please refer to D2.3 available at http://www.finest-ppp.eu/

[2] Available at http://www.finest-ppp.eu/

i.e. future probabilistic events of interest. The rules definitions are initiated by the BCM, which gets the inputs from both the transport plan (TPM) and the contract (ECM). The inputs from the TPM are essentially the relevant transport plan and its legs, that is, the TCP (Transport Chain Plan) and its corresponding TEPs (Transport Execution Plan).

The FISH use case POC EPM has been implemented as a stand-alone module at this stage. As a result, the rules and events definitions, as well as the set of incoming events, have been manually created. In addition, the notifications and events resulting from the actual consumption of the events input and definitions are displayed in a dedicated dashboard (screenshots of this dashboard are shown in Figure 5 and Figure 6). Figure 1 depicts the POC EPM main modules and interactions.

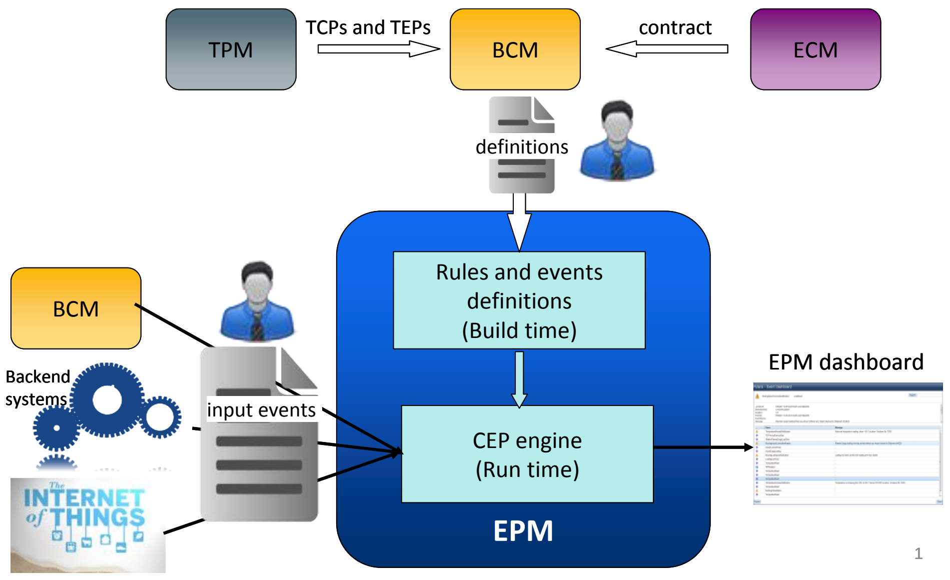


Figure 1: EPM POC main modules and interactions

2.2. EPM POC installation

The EPM POC has been implemented on top of the FI-WARE CEP GE (Generic Enabler) as detailed in the module specification in D6.3.

The event processing module has been installed in the KocSistem infrastructure on a windows server with ip address : 192.168.19.86

2.2.1. Software Requirements

The installation includes the following components:

1) latest Google Chrome browser
2) Finest EPM v1 (using java version 1.6 and up)
3) Finest FISH application files
    a. Application files for the proactive and the reactive scenarios (definitions)
    b. Simulation data file corresponding the FISH use case scenario (input set)

4) Event Dashboard for visualization of the EPM detected situations and alerts. A web application deployed on apache Tomcat version 7.0.2

2.2.2. Access to instantiation

Accessing the EPM POC application includes the following steps:

1) Access the KocSistem portal for Finest : https://sahne.kocsistem.com.tr/finest
2) Log in with a given username and password
3) Start a VPN connection via "network connection"
4) Open remote desktop connection to 192.168.19.86

2.2.3. Running the instantiation

Running the EPM POC includes the following steps:

1) Start Tomcat. C:\apache-tomcat\bin\startup.sh
2) Open Chrome Browser on : http://192.168.19.86:8080/dashboard/EventDashboard.html
3) Start Finest EPM on C:\Proton\launchProton.sh
4) Watch the resulting events, detected situations and alerts in the browser window

2.3. Methodology

The proof-of-concepts for the four modules in FInest have been progressively implemented and checked through bi-weekly technical meetings of the project. Due to the tight connection between the BCM and EPM, additional, BCM-EPM (work package 5 and 6 respectively) meetings took place on demand to revise and refine the on-going implementations.

In order to deliver a single consistent story board, in addition to the decision of implementing the same use case, all the relevant WPs (work packages) of the four core modules (i.e. WP5-8), have consumed the same TCP and TEPs files provided by the TPM.

The face-to-face technical meeting held in Essen, Germany, during 28-30/1/13 constituted a validation check point of the project POCs in general, and the EPM POC by the domain partners in terms of rules correctness and applicability to the use case, in particular.

3. The EPM proof-of-concept for the FISH use case

3.1. Events definition for the FISH use case

The selected FISH use case offers a rich set of possible events and rules of interest. The events set that was defined for the POC includes events from three possible producers (IoT, backend systems, and BCM). As previously noted, we defined one consumer for the EPM notifications, i.e., the FInest dashboard.

Table 1 describes the list of events and their attributes articulated for the FISH use case and their mapping to the corresponding attribute name and value in the TCP and TEPs xml files provided by the TPM (WP7).

Note that we denote by Populated in real-time, attribute values that are not part of the initial TEP, but are obtained as a result of the execution at run-time. Also note that the attribute values denoted in (orange) are actual attribute values as they appear in the TCP and TEPs xml files. The ShipmentID attribute serves as the unique identifier for an event (in our case 31).

Table 1: List of input (raw) events for the FISH use case

#	Event name	Attribute name	Attribute name and value in TEP
1.	TCP1ActualService Start	ShipmentId	Consignment. ID (ID31)
		CargoAmount	Consignment. ID. TotalTransportHandlingUnitQuantity (2 pieces)
		CargoWeight	Consignment. ID. GrossWeightMeasure (16,000)
		CargoWeightUOM	Consignment. ID. GrossWeightMeasure.unitCode (Kg)
		CargoVolume	Consignment. ID. GrossVolumeMeasure (60)
		CargoVolumeUOM	Consignment. ID. GrossVolumeMeasure.unitCode (M^3)
		ServiceStartTime	ServiceStartTimePeriod.StartDate (2012-05-06 10:00)
		ContractID	Transport.Contract.ID (ID20)

#	Event name	Attribute name	Attribute in TEP
2.	TCP1ActualService End	ShipmentId	Consignment. ID (ID31)

#	Event name	Attribute name	Attribute in TEP
3.	PlannedCargo Pickup	ShipmentId	Consignment. ID (ID31)
		PlannedCargoPickupTime	PlannedPickupTransportEvent. StartDate (2012-05-06 10:00)

#	Event name	Attribute name	Attribute in TEP
4.	ActualCargoPickup	ShipmentId	Consignment. ID (ID31)
		ActualCargoPickupTime	Populated in real-time

5.	ImportLicenseArrival	ShipmentId	Consignment. ID (ID31)
		CustomsId	Consignment.CustomsIdentification (ID630)
		ActualImportLicense ArrivalTime	Populated in real-time

6.	LoadingListArrival	ShipmentId	Consignment. ID
		LoadingListID	AdditionalDocumentReference.ID (where AdditionalDocumentreference.documentType=121 is " Transport loading instruction") (LoadingListID)
		ActualLoadingListArrival Time	Populated in real-time

7.	ActualCargoLoading	ShipmentId	Consignment. ID
		ActualCargoLoadingTime	Populated in real-time

8.	ActualCargoAmount	ShipmentId	Consignment. ID (ID=31)
		ActualAmount	Populated in real-time
		ActualVolume	Populated in real-time
		ActualWeight	Populated in real-time

9.	BookingCancellation	ShipmentId	Consignment. ID (ID=31)
		ActualCancellationTime	Populated in real-time

10.	TemperatureRead (RFID event)	ShipmentId	Consignment. ID (ID=31)
		ReadingValue	Populated in real-time by IoT
		ReadTimestamp	Populated in real-time by IoT
		Location	Populated in real-time by IoT
		SensorId	Populated in real-time by IoT

3.2. Rule set definition for the FISH use case

For the sake of simplicity and clarity, we have split the entire rule set into reactive and proactive sets. In reality, these two sets form one single definition set rule.

In addition, in order to distinguish between events that are known before execution and are part of the TEPs (e.g., pickup date or cargo amount ordered), and events that occur during execution (e.g., actual pickup date, cargo amount at pickup), we refer to *planned events* and *actual events* respectively. *Derived events* are the specific notifications displayed to the user via the dashboard as a result of a detected situation.

A brief description of the operators implemented in the FISH scenario:

Threshold (or *Filter*) – a predicate that is evaluated against an event; the event passes the filter if the predicate is evaluated to TRUE, and fails the filter if the predicate is evaluated to FALSE.

Absence – a specified event(s) must not occur within a predefined time window. The matching set in this case is empty.

Sequence - at least one instance of all participating event types must arrive in a specified order for the pattern to be matched.

All – at least one instance of all participating events must arrive for the pattern to be matched; the arrival order in this case is immaterial.

Trend – events need to satisfy a specific change over time of some observed value; this refers to the value of a specific attribute or attributes.

3.2.1. Reactive rules

In order to demonstrate the applicability of the selected scenario, we have defined the following set of rules (Table 2). In Table 2 we describe rules in a formal and informal way, the type of event processing agent that will be associated with a rule and the derived events of each rule. For template rules, rules whose parameters are instantiated per each TEP, the parameter values are given (in **orange**).

Table 2: Reactive rule set for the FISH use case

Rule name	MissingImportLicense
Rule (informal definition)	Import license was not received at the Alesund terminal X time units before planned loading
Rule definition	Absence(ImportLicenseArrival) X time units before PlannedCargoLoadingTime
Operator	Absence
Participating events	ImportLicenseArrival
Derived events	MissingImportLicenseNotification
Notification consumer	BCM
Template rule	Yes
Template parameters	X time units before planned loading **(X=24hs)**

Rule name	MissingLoadingList
Rule (informal definition)	Cargo loading list was not received at the Alesund terminal before the actual cargo loading starts.
Rule definition	Absence(LoadingListArrival) before ActualCargoLoadingTime
Operator	Absence
Participating events	LoadingListArrival
Derived events	MissingLoadingListNotification
Notification consumer	BCM
Template rule	No

Template parameters	N/A

Rule name	CargoPickupDelay
Rule (informal definition)	Cargo pickup did not happen as scheduled and there is a delay of Y unit times
Rule definition	Absence(ActualCargoPickup) Y time units after PlannedCargoPickupTime
Operator	Absence
Participating events	ActualCargoPickup
Derived events	DelayedPickupNotification
Notification consumer	BCM
Template rule	Yes
Template parameters	Y time units after scheduled pickup **(Y=1 hour)**

Comment: the same rule can be created and applied to other potential delays, i.e. loading, unloading, arrival.

Rule name	CargoDeviationObserved
Rule (informal definition)	Cargo amount deviation is greater than X % (both directions)
Rule definition	Deviation is observed if (ActualCargoAmount/Planned CargoAmount> X% OR ActualCargoAmount/Planned CargoAmount < X%)
Operator	Sequence
Participating events	TCP1ActualServiceStart ActualCargoAmount
Derived events	CargoAmountDeviationNotification
Notification consumer	BCM
Template rule	Yes
Template parameters	Deviation of X% from cargo amount that should be reported **(X=10%)**

Comment: The same rule is applied to Volume and Weight and fires CargoWeightDeviationNotification and CargoVolumeDeviationNotification respectively.

Rule name	BookingCancellationObserved
Rule (informal definition)	Booking cancellation occurred
Rule definition	Existence(BookingCancellation)
Operator	All
Participating events	BookingCancellation
Derived events	BookingCancellationNotification
Notification consumer	BCM
Template rule	No
Template parameters	N/A

Rule name	TemperatureAnomaly
Rule (informal definition)	A temperature read in the cargo storage area exceeds/goes under the allowed range.
Rule definition	Filter(TemperatureRead.ReadValue) if (NOT WITHIN [X,Y])
Operator	Threshold
Participating events	TemperatureRead
Derived events	TemperatureAnomaly notification
Notification consumer	BCM
Template rule	Yes
Template parameters	Allowed temperature range (from X to Y) **(Range [1-20])**

Rule name	TemperatureIncrease
Rule (informal definition)	A trend of temperature increase is observed in the cargo storage

	area
Rule definition	Trend(TemperatureRead.ReadValue) is increasing
Operator	Trend
Participating events	TemperatureRead
Derived events	TemperatureIncreaseNotification
Notification consumer	BCM
Template rule	No
Template parameters	N/A

3.2.2. Proactive rules

In a proactive event-driven application, at the conceptual level, two modules are required: a forecasting/predicting module, capable of predicting future probabilistic system states, and a (near) real-time decision module (for a detailed explanation on proactive event-driven computing and its specification refer to D6.3). The forecasting module relies on predictive models that, in combination with real-time pattern detections, produce a future (probabilistic) event. As the EPM POC is a stand alone application and we don't rely on historical data but on a very small set of input events, we cannot derive a predictive model. Therefore, we assume we have the forecasted predictions given and we illustrate the proactive principle by the two following rules that can be applied to notify ahead of time about a potential problem.

1. PredictedCoolingSystemMalfunction

In case there is an observed trend pattern by an IoT sensor in the temperature of a container, produce a notification with probability for significant risk to the freshness of the fish in that container. This notification can lead to two options: replacing the sensor immediately or waiting until the next station stop.

t1<t2<t3 => trend (TemperatureRead1, TemperatureRead2, TemperatureRead3)

PredictedSevereMalfunction

if 18<=t3<20 with certainty of 80%

Message: Malfunction in the cooling system of SensorID is predicted with certainty of 80%. It is recommended to immediately send a technician.

PredictedMalfunction

if 15<t3<18 with certainty of 60%

Message: Malfunction in the cooling system of SensorID is predicted with certainty of 60%. It is recommended to perform a maintenance activity in the next port-of-call.

2. PredictedCargoDeviation

In case of a significant deviation (30%-50%) in the cargo (number of containers, weight, or volume), produce a notification with probability for missing the planned ship due to overweight.

PredictedSevere<Amount/Weight/Volume> CapacityExceed

if CargoDeviation>1.5   with certainty of 85%

Message: Due to a severe cargo <Amount/weight/Volume> deviation of CargoDeviation%, a capacity exceed for ShipmentID is predicted with certainty of 85%. It is recommended to find an alternative plan for ShipmentID

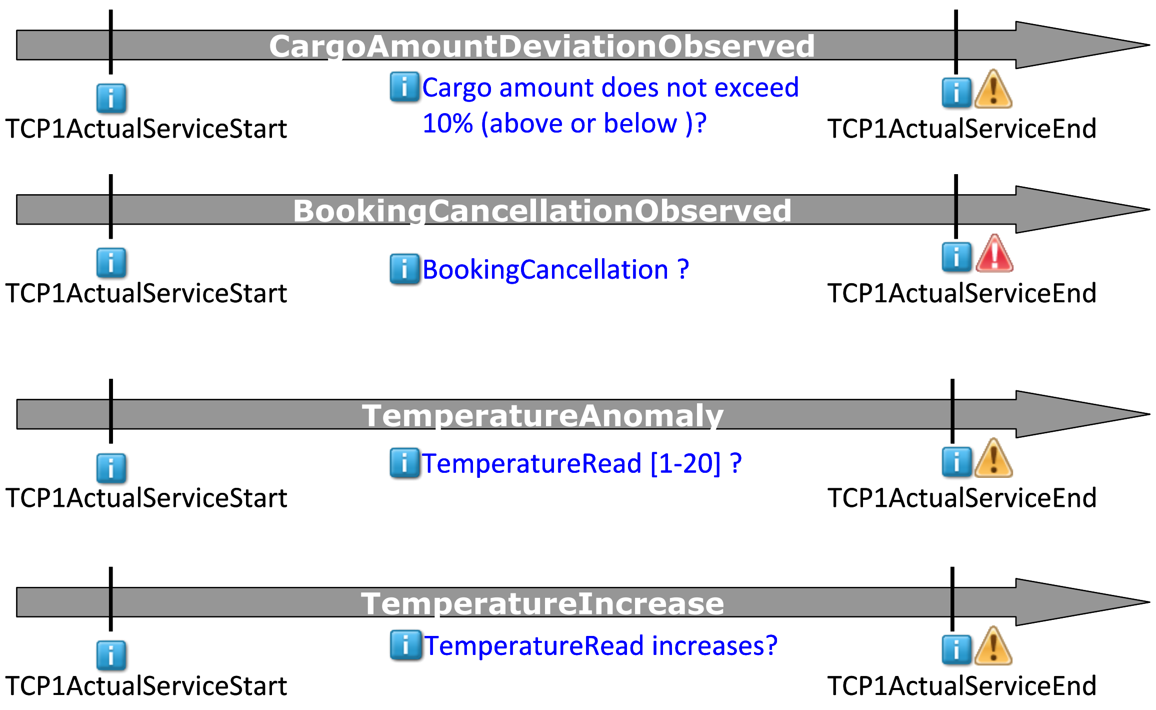
Predicted Amount/Weight/Volume CapacityExceed

if 1.3<CargoDeviation<=1.5   with certainty of 40%

Message: Cargo <Amount/weight/Volume> deviation of CargoDeviation% occurred. A capacity exceed for ShipmentID is predicted with certainty of 40%.

3.3. Contexts for the FISH use case

Context is a named specification of conditions that group event instances so that they can be processed in a related way. In our case, the entire rule set is relevant within the time period between TCP1ActualServiceStart and TCP1ActualServiceEnd events for an individual shipment (identified by ShipmentID). In other words, for each shipment we monitor all its relevant events during the entire trajectory, from when it is first picked-up at the customer, until it starts its journey overseas to Brazil. In case of a BookingCancellation event, it is assumed that this event causes termination of execution of all rules related to the relevant shipment (i.e., all opened contexts are closed). In addition, each rule can have a specific context that ends with the occurrence of a specific event as depicted in Figure 2.



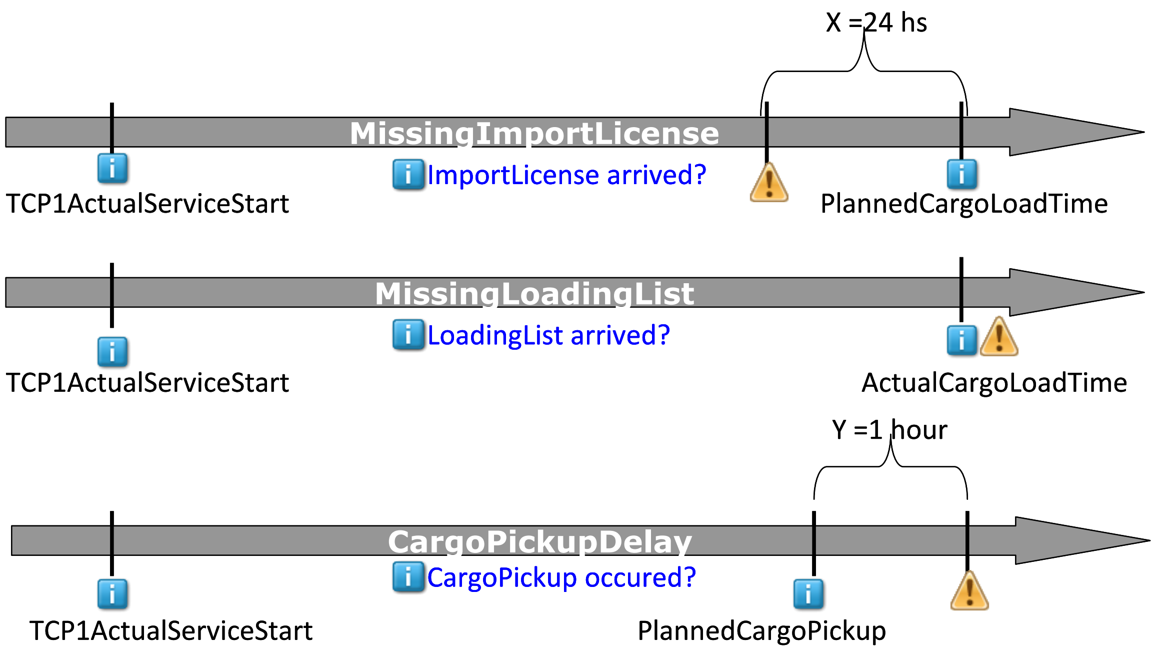


Figure 2: FISH use case rules context

For the two proactive rules their context is analogous to their reactive counterpart rule. In other words, the context of the PredictedCoolingSystemMalfunction rule is the same as the TemperatureIncrease rule and the context of the PredictedCargoDeviation rule is the same as for the CargoDeviationObserved.

3.4. Event processing network for the FISH use case

Table 3 describes the event processing network specified for the FISH use case, which encompasses the rule set given in Table 2 and the two proactive rules.

Table 3: Event processing network for the FISH use case

Event processing agent name	EPA Type	Input event types	Output event types	Context
MissingImportLicense	Absence	ImportLicenseArrival	MissingImportLicenseNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → X time units before PlannedCargoLoadingTime)
MissingLoadingList	Absence	LoadingListArrival	MissingLoadingListNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → ActualCargoLoadingTime)
CargoPickupDelay	Absence	ActualCargoPickup	DelayedPickupNotifications	EventInterval by ShipmentID (TCP1ActualServiceStart → PlannedCargoPickup + Y time units)
CargoAmountDeviationObserved	Sequence	TCP1ActualServiceStart ActualCargoAmount	CargoAmountDeviationNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)
BookingCancellationObserved	All (existence)	BookingCancellation	BookingCancellationNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)

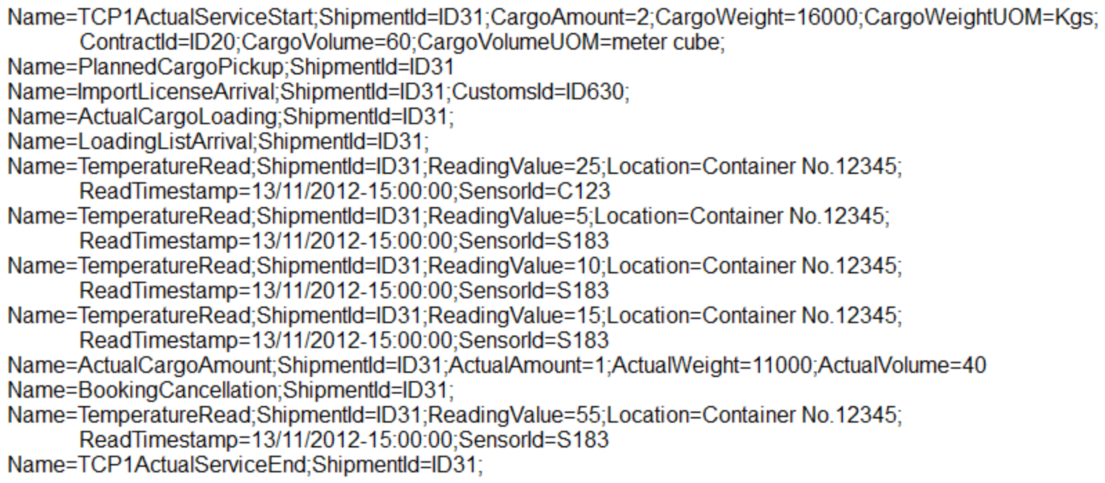
TemperatureAnomaly	Threshold (filter)	TemperatureRead	TemperatureAnomalyNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)
TemperatureIncrease	Trend	TemperatureRead	TemperatureIncreaseNotification	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)
Proactive EP agent name	**PRA Type**	**Input event types**	**Output event types**	**Context**
PredictedCoolingSystem Malfunction	Trend	TemperatureRead	PredictedSevereMalfunction PredictedMalfunction	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)
PredictedCargoDeviation	Sequence	TCP1ActualServiceStart ActualCargoAmount	PredictedSevere Amount/Weight/Volume CapacityExceed Predicted Amount/Weight/Volume CapacityExceed	EventInterval by ShipmentID (TCP1ActualServiceStart → TCP1ActualServiceEnd)

3.5. EPM proof-of-concept instantiation

For didactic purposes we split between the inputs and outputs events sets. In actual execution, this split of course doesn't exist, and the events along with the notifications are displayed in the dashboard as they actually occur. Furthermore, we defined a very small set of input events for a single shipment, sufficient to demonstrate all the rules defined. Obviously, in reality, we can get thousands of events related to many different shipments simultaneously.

3.5.1. Input events

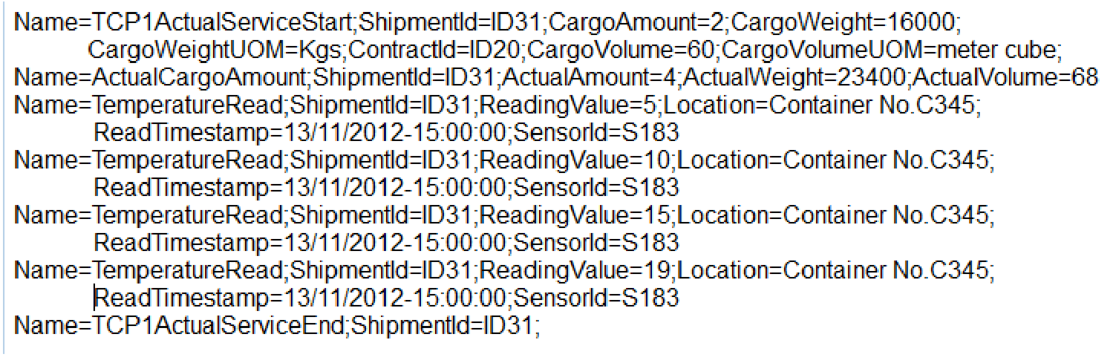
Figure 3 shows the set of events fed into the EPM run-time engine in this ordered.



```
Name=TCP1ActualServiceStart;ShipmentId=ID31;CargoAmount=2;CargoWeight=16000;CargoWeightUOM=Kgs;
        ContractId=ID20;CargoVolume=60;CargoVolumeUOM=meter cube;
Name=PlannedCargoPickup;ShipmentId=ID31
Name=ImportLicenseArrival;ShipmentId=ID31;CustomsId=ID630;
Name=ActualCargoLoading;ShipmentId=ID31;
Name=LoadingListArrival;ShipmentId=ID31;
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=25;Location=Container No.12345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=C123
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=5;Location=Container No.12345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=10;Location=Container No.12345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=15;Location=Container No.12345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=ActualCargoAmount;ShipmentId=ID31;ActualAmount=1;ActualWeight=11000;ActualVolume=40
Name=BookingCancellation;ShipmentId=ID31;
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=55;Location=Container No.12345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TCP1ActualServiceEnd;ShipmentId=ID31;
```

Figure 3: Input events for the reactive set of rules

Figure 4 shows the set of input events used to demonstrate the proactive rules defined.

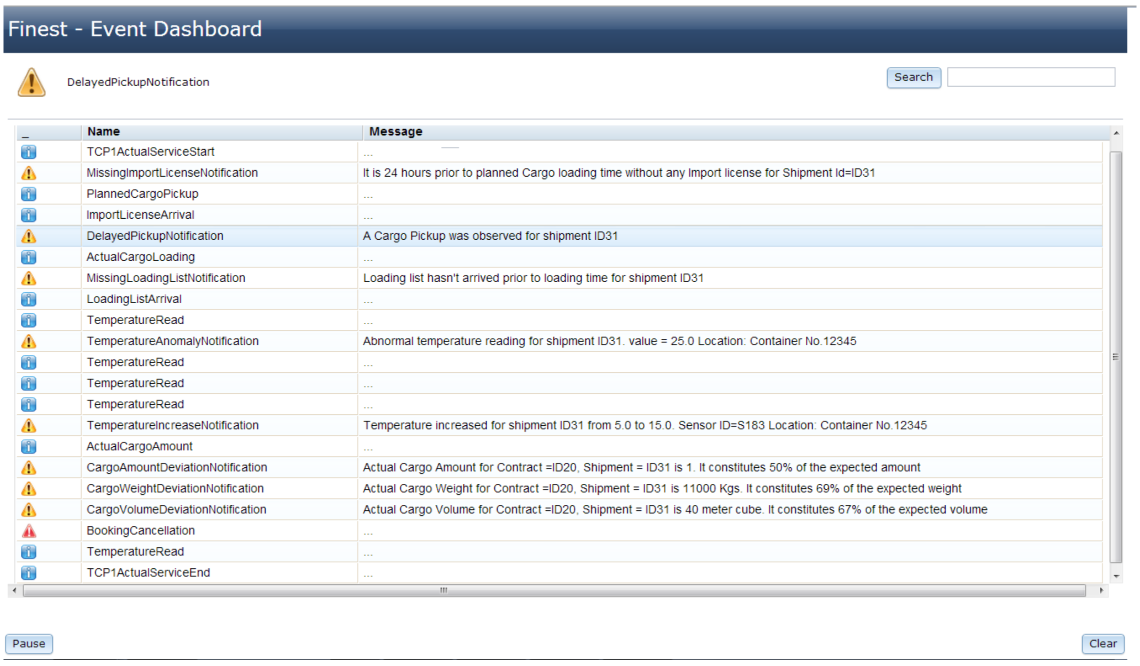


```
Name=TCP1ActualServiceStart;ShipmentId=ID31;CargoAmount=2;CargoWeight=16000;
        CargoWeightUOM=Kgs;ContractId=ID20;CargoVolume=60;CargoVolumeUOM=meter cube;
Name=ActualCargoAmount;ShipmentId=ID31;ActualAmount=4;ActualWeight=23400;ActualVolume=68
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=5;Location=Container No.C345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=10;Location=Container No.C345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=15;Location=Container No.C345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TemperatureRead;ShipmentId=ID31;ReadingValue=19;Location=Container No.C345;
        ReadTimestamp=13/11/2012-15:00:00;SensorId=S183
Name=TCP1ActualServiceEnd;ShipmentId=ID31;
```

Figure 4: Input events for the proactive set of rules

3.5.2. Output events

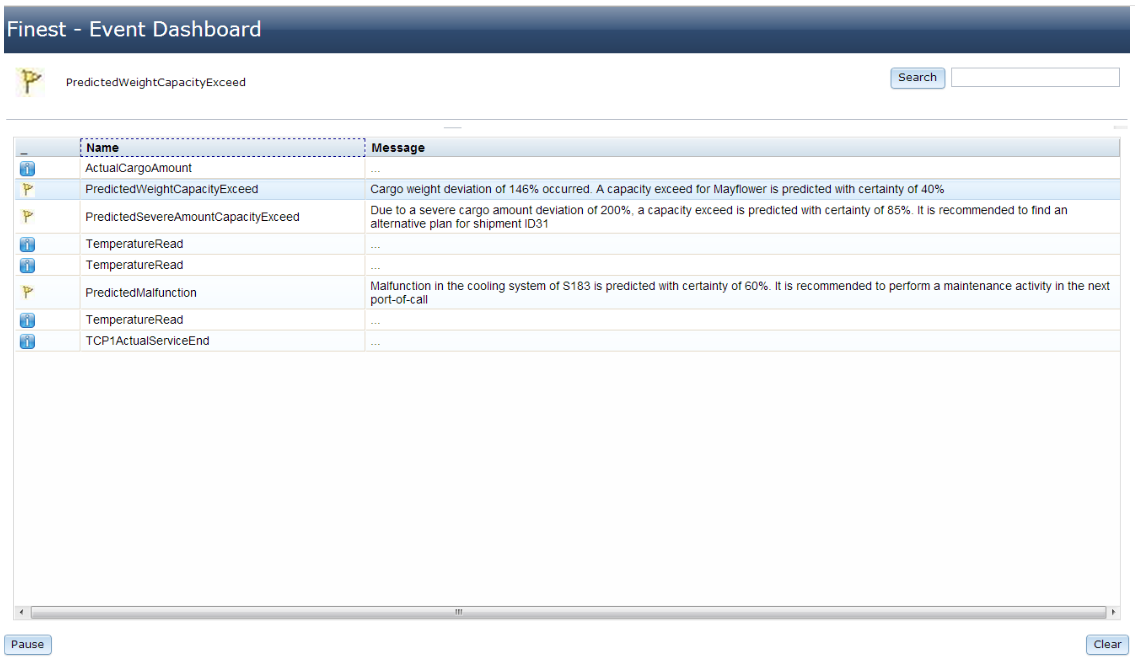
Again, for the sake of simplicity and clarity, we run first the input events for the reactive set of rules, and then we run the set of input events for the proactive set of rules. These events, along with the corresponding messages, are displayed in the FInest dashboard and shown in Figure 5 and Figure 6 respectively.


Finest - Event Dashboard

⚠ DelayedPickupNotification

	Name	Message
ℹ	TCP1ActualServiceStart	...
⚠	MissingImportLicenseNotification	It is 24 hours prior to planned Cargo loading time without any Import license for Shipment Id=ID31
ℹ	PlannedCargoPickup	...
ℹ	ImportLicenseArrival	...
⚠	DelayedPickupNotification	A Cargo Pickup was observed for shipment ID31
ℹ	ActualCargoLoading	...
⚠	MissingLoadingListNotification	Loading list hasn't arrived prior to loading time for shipment ID31
ℹ	LoadingListArrival	...
ℹ	TemperatureRead	...
⚠	TemperatureAnomalyNotification	Abnormal temperature reading for shipment ID31. value = 25.0 Location: Container No.12345
ℹ	TemperatureRead	...
ℹ	TemperatureRead	...
ℹ	TemperatureRead	...
⚠	TemperatureIncreaseNotification	Temperature increased for shipment ID31 from 5.0 to 15.0. Sensor ID=S183 Location: Container No.12345
ℹ	ActualCargoAmount	...
⚠	CargoAmountDeviationNotification	Actual Cargo Amount for Contract =ID20, Shipment = ID31 is 1. It constitutes 50% of the expected amount
⚠	CargoWeightDeviationNotification	Actual Cargo Weight for Contract =ID20, Shipment = ID31 is 11000 Kgs. It constitutes 69% of the expected weight
⚠	CargoVolumeDeviationNotification	Actual Cargo Volume for Contract =ID20, Shipment = ID31 is 40 meter cube. It constitutes 67% of the expected volume
⚠	BookingCancellation	...
ℹ	TemperatureRead	...
ℹ	TCP1ActualServiceEnd	...

Figure 5: Dashboard screenshot for the reactive events input set



_	Name	Message
ℹ	ActualCargoAmount	...
⚑	PredictedWeightCapacityExceed	Cargo weight deviation of 146% occurred. A capacity exceed for Mayflower is predicted with certainty of 40%
⚑	PredictedSevereAmountCapacityExceed	Due to a severe cargo amount deviation of 200%, a capacity exceed is predicted with certainty of 85%. It is recommended to find an alternative plan for shipment ID31
ℹ	TemperatureRead	...
ℹ	TemperatureRead	...
⚑	PredictedMalfunction	Malfunction in the cooling system of S183 is predicted with certainty of 60%. It is recommended to perform a maintenance activity in the next port-of-call
ℹ	TemperatureRead	...
ℹ	TCP1ActualServiceEnd	...

Figure 6: Dashboard screenshot for the reactive events input set

4. Summary

D6.4 describes the FISH use case POC implementation of the FInest event processing module. In essence, The FInest EPM POC implementation is based on the CEP GE and extended to provide proactive capabilities, and instantiated in the KocSistem cloud infrastructure. The implementation follows the architecture presented in D6.3 and validates it using a running example.

The scenario applied exhibits a wide range of rules operators and demonstrates the potential benefits of applying event processing techniques to logistics processes. The applied rules have been validated by the domain partners in terms of their relevance and correctness.