# Trust Metric Estimator API Reference Guide

## Short Description

The Trust Metric Estimator (TME) GE is a Bayesian computational model that helps system administrators identify and react to anomalies in user experiences.

## Licensing Issues

### Copyright

The Open specification of the TME GE is copyrighted to Athens University of Economics and Business (AUEB) and Athens Technology Center S.A. (ATC).

### Legal Notice

This specification is subject to the provisions of the OPTET Open Specification Legal Notice (implicit patents license).

## Overview of the Trust Metric Estimator GE

### Target Use

The TME GE is accepting notifications (atomic events) about end-user registrations, as well as, service performance to user requests in order to estimate individual user trust levels for a wide range of trustworthiness metrics. Then, by comparing the current trust level for each metric with the high and low threshold set by the administrator, a high/low alert is sent to the OPTET Management recipe for further action, if necessary.

### Position in the OPTET Lifecycle

As shown in Figure 1, the TME GE belongs to the run-time maintenance phase of a SW product lifecycle. However, it could also be used during the design phase for selecting the pair of service trustworthiness and price so that profits are maximised and later during the distribution and deployment phase to announce this price to the marketplace.

### User roles

This GE is intended for:

* The service administrator(s) who need to monitor user experience;
* The system designers who decide the system trustworthiness level;
* The service managers who select the retail price.

### Trust and Trustworthiness Requirements Addressed

The GE provides trust estimation for individual users that can be used by providers at several phases.

* At design-time for selecting the pair of service trustworthiness and price so that profits are maximized (her income minus the monetary cost that she faces to achieve a certain level of trustworthiness). The provider can define the trustworthiness level of the software and the retail price before making it available to the marketplace. The provider’s income consists of the price that the software/system is offered to multiplied by the number of trustors whose decision criterion is satisfied and thus buy it.
* At the distribution and deployment phase where the provider can announce the optimal price for the trustworthiness level selected during the design phase.
* At run-time for spotting trust concerns fast enough and allowing cost effective controls to be deployed so that a trust threshold is not violated after a number of future transactions.

## Detailed Open Specification

### Basic Features and Functionalities

Trust Metric Estimator offers the following features:

* Estimate a user’s trust level for each service separately and do so at several levels of detail in terms of:
	+ individual trustworthiness metrics or aggregate and
	+ time, i.e., trust evolution during a certain time window or a single trust value.
* Send alerts whenever any threshold for each metric is exceeded, so that an action is taken if necessary. Thresholds can refer to either high or low values and these can be defined separately for each metric.

### Architecture description

Figure 1 shows the architecture diagram of the TME GE.



Figure 1: The architecture of the Trust Metric Estimator Generic Enabler.

### Main Interactions

During setup:

* Administrator defines for each metric a threshold value, which is used for determining whether a particular request/transaction was successful, or not.
* Administrator defines high/low thresholds for each metric
* Administrator assigns (new) users to a segment

When the Complex Event Processor receives an atomic event from the Monitoring Adapter it is forwarded to TME Engine. Then:

* This atomic event can be a user registration request, a user service request with an outcome value for a supported trustworthiness metric,
* TME engine stores the atomic event to Measurement DB,
* TME engine characterizes the transaction as successful or not by comparing this outcome value to the respective success threshold,
* TME engine computes the updated trust level of user for that particular metric and stores it to the Customer Profile DB. In case the atomic event referred to a successful transaction, trust level is increased. Otherwise, trust level is decreased. The magnitude of trust change depends on the user’s segment.
* TME Engine checks whether an alert should be triggered

When the TME Engine receives a request for trust metric values or alerts from Management Adapter, the necessary data is retrieved from the Customer Profile DB.

When the TME Engine receives a request for trust metric values or details about service users from Optimal Control Selector directly, the necessary data is retrieved from the Customer Profile DB.

### Architectural Design Principles

This section describes the requirements for the design principles of the TME GE. These principles are summarised in the following:

* Simplicity: Provide a multi-purpose approach for the architectural design of the TME to enable the target audience realise the specifications of this GE.
* Domain driven design: Domain-Driven Design (DDD) is an approach to software development for complex needs by connecting the implementation to an evolving model. In domain-driven design, three basic uses determine the choice of a model, such as:
	+ The model and the design of the product are tightly bound. This binding of model and implementation also helps during maintenance and continuing development, because the code can be interpreted based on understanding the model.
	+ The model is the backbone of a language used by all team members
	+ The model is an agreed-upon way of structuring key elements in a project to facilitate partner collaboration

This model-based communication is not limited to diagrams in Unified Modelling Language (UML). To make most effective use of a model, it needs to pervade every medium of communication. It increases the utility of written text documents, as well as the informal diagrams and casual conversation reemphasized in agile processes. It improves communication through the code itself and through the tests for that code.

* Conformance to OPTET lifecycle: Describe the TME specifications to address the needs raised from the adopted of the OPTET engineering methodology and the OPTET Lifecycle phases.

### Dependencies to external Components

The following dependencies are recognised:

* Complex Event Processor for receiving atomic events;
* Management Adapter as a recipe implementation for sending alerts and receiving requests for trust metric values;
* Optimal Control Selector asset for receiving requests for User attributes, optimization window and current number of successes.

### Technologies and Specifications Used

With regard to the specifications, this Open Specification has used the JSR-000338 Java Persistence 2.1 Final Release

Regarding the technologies used:

* Hibernate (version 4.3.5.RELEASE), which is an implementation of the Java Persistence API (JPA) specification
* Java EE 7 SDK
* The Spring Web model-view-controller (MVC) framework, version 3.2.4.RELEASE
* Maven 3, a dependency management tool

## Open API Specifications

### API Overview

The following features are offered by the TME GE:

* Track trust level evolution

The trust computational model provided by the TME GE estimates a user’s trust level for each service separately and do so at several levels of detail in terms of

1. individual trustworthiness metrics or aggregate and
2. time, i.e., trust evolution during a certain time window or a single trust value
* Send alerts for trust concerns

The trust computational model provided by the TME GE sends alerts whenever any threshold for each metric is exceeded, so that an action is taken if necessary. Thresholds can refer to either high or low values and these can be defined separately for each metric.

### Intended Audience

The API offered by TME is used by an STS system to inject outcome/user/conversation events as well as to retrieve the computed trust values.

### Analysis of the exposed API operations

The TME GE operations are displayed in Figure 2.



Figure 2: Trust Metric Estimator – the analysis of the API operations

As shown in this figure, the following operations are made available:

**Method: /postRawEvent**

This is an API for injecting various types of supported events (user/conversation/outcome). This method acts as a dispatcher between the various event types.

 @RequestMapping(value = "/postRawEvent", method = RequestMethod.POST,

 produces = "application/json", consumes = "application/json",

 headers = {"Content-type=application/json"})

 public @ResponseBody

 String postRawEvent(@RequestBody String event)

Expected input:

{

 "name": "SessionOutcome",

 "sourceAssetType": "sensor",

 "sourceAssetId": "sensor3",

 "occurrenceTime":"02-04-2014 08:38:14",

 "userId":"1",

 "conversationId":"4",

 "responseTime":"401"

}

Expected output:

{ status: "ok" } If outcome event processed successfully or

{ status: "conversation doesn't exist error" } If target conversation doesn’t exists.

**Method: getTimeSpanOverallTrustAllUsersInteraction**

This is an API for querying the trust value HISTORY from TME, within an interaction for each user participating, within a time span.

 @RequestMapping(value = "/getTimespanOverallTrustAllUsersInteraction",

 method = RequestMethod.GET, produces = "application/json")

 public @ResponseBody

 String getTimespanOverallTrustAllUsersInteraction(

 @RequestParam(value = "target") String target,

 @RequestParam(value = "conversationId") Long conversationId,

 @RequestParam(value = "timeBegin") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date startDate,

 @RequestParam(value = "timeEnd") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date endDate)

Expected input:

/getTimespanOverallTrustAllUsersInteraction?target=TME&conversationId=1&timeBegin=2014-06-19+12:02:50&timeEnd=2015-06-29+12:02:50

Expected output:

{

 users: [

 {id: 4,

 trustValue: 0.615},

 {id: 1,

 trustValue: 0.765},

 {id: 2,

 trustValue: 0.765},

 {id: 7,

 trustValue: 0.5575},

 {id: 9,

 trustValue: 0.53}],

 conversationId: 1

}

**Method: getTimeSpanOverallTrustUserInteraction**

This is an API for querying, for a single user, the last trust value from TME (evaluated across all the trust metrics and all interactions he participates) within a time span.

 @RequestMapping(value = "/getTimespanOverallTrustUserInteraction",

 method = RequestMethod.GET, produces = "application/json")

 public @ResponseBody

 String getTimespanOverallTrustUserInteraction(

 @RequestParam(value = "target") String target,

 @RequestParam(value = "userId") Long userId,

 @RequestParam(value = "timeBegin") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date startDate,

 @RequestParam(value = "timeEnd") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date endDate)

Expected input:

/getTimespanOverallTrustUserInteraction?target=TME&userId=1&timeBegin=2014-06-10+17:25:39&timeEnd=2015-06-29+17:25:39

Expected output:

 {

 id: 1,

 trustValue: 0.765

 }

**Method: getTimeSpanTMETrustMetricValue**

This is an API for querying, for a single user, the last trust values of each single trust metric from TME, within a time span.

 @RequestMapping(value = "/getTimespanTMETrustMetricValue",

 method = RequestMethod.GET, produces = "application/json")

 public @ResponseBody

 String getTimespanTMETrustMetricValue(

 @RequestParam(value = "target") String target,

 @RequestParam(value = "userId") Long userId,

 @RequestParam(value = "metricName") String metricName,

 @RequestParam(value = "timeBegin") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date startDate,

 @RequestParam(value = "timeEnd") @DateTimeFormat(pattern = "yyyy-MM-dd HH:mm:ss") Date endDate)

Expected input:

/getTimespanTMETrustMetricValue?target=TME&userId=1&metricName=responseTime&timeBegin=2014-06-10+17:25:39&timeEnd=2015-06-29+17:25:39

Expected output:

 {

 id: 1,

 trustValue: 0.765

 }

## API Test Plan

The adoption of the test-driven-development (TDD) approach to software development is a way to ensure high quality for the API exposed. The test procedure described in this section aims at ensuring that the Open API exposed by the TME GE is validated against some test scenarios that intend to cover most parts of the functionality offered by TME GE.

The following projects have been used to drive the tests execution effectively:

* org.springframework.spring-test (3.2.4.RELEASE)
* org.mockito.mockito-core (1.9.5)

The execution of the test scenarios for TME GE is facilitated by the Apache Maven tool. The only thing required is to specify the appropriate Maven goal:

mvn clean test

then a Surefire report will be available containing the test results, as shown in Figure 3.



Figure 3: Trust Metric Estimator – Screenshot of the testing procedure with Apacahe Maven

We define the five API test scenarios, which are analysed in the following sections.

### TME test scenario ValidSegmentInitializationTest

**Features and Functionalities Addressed**

This test covers the functionality related to the creation and configuration of the Segment class instances, which represent the expected reactions to the evidences experienced during both the initial and evolution phases of engagement with a system.

**API Test Description**

This test validates the successful creation and initial configuration of the segment instances by checking various field values like the expected initial coefficient, the initial trust level etc.

**Expected Results and Sample Data**

Assert.assertEquals("HIGH\_TRUST", segment1.getName());

Assert.assertEquals("HIGHLY\_ACTIVE\_TRUST\_SEEKING", segment2.getName());

Assert.assertEquals("MEDIUM\_ACTIVE\_TRUST\_SEEKING", segment3.getName());

Assert.assertEquals("AMBIVALENT", segment4.getName());

Assert.assertEquals(7.244293721827499, segment1.getInitialCoefA(), DELTA);

Assert.assertEquals(7.0245, segment2.getInitialCoefA(), DELTA);

Assert.assertEquals(6.016690583406001, segment3.getInitialCoefA(), DELTA);

Assert.assertEquals(4.0791666665925, segment4.getInitialCoefA(), DELTA);

Assert.assertEquals(0.8255605381, segment1.getInitialTrustLevel(), DELTA);

Assert.assertEquals(0.7, segment2.getInitialTrustLevel(), DELTA);

Assert.assertEquals(0.6387144993, segment3.getInitialTrustLevel(), DELTA);

Assert.assertEquals(0.6111111111, segment4.getInitialTrustLevel(), DELTA);

### TME test scenario InjectUserEventTest

**Features and Functionalities Addressed**

This test covers the functionality related to the creation and configuration of the Trustor class instances.

**API Test Description**

The trustor creation action can be modelled as a special type of a TME event, specifically the 'UserNew' event type. The exposed TME API method '/postRawEvent' dispatches the various types of events to the appropriate handler. In this test scenario a new trustor is created, then wrapped as a 'UserNew' event and finally that event is injected to the TME backend.

**Expected Results and Sample Data**

Assert.assertEquals("{\"status\":\"OK\"}", status);

Assert.assertEquals("HIGHLY\_ACTIVE\_TRUST\_SEEKING", user.getSegment().getName());

Assert.assertEquals(1000L, user.getId(), DELTA);

### TME test scenario InjectConversationEventTest

**Features and Functionalities Addressed**

This test covers the functionality related to the creation and configuration of the ServiceInstance class instances, which represents the various conversations where the trustors are assigned.

**API Test Description**

The conversation creation action can be modelled as a special type of a TME event, specifically the 'ConversationNew' event type. The exposed TME API method '/postRawEvent' dispatches the various types of events to the appropriate handler. In this test scenario a new conversation is created, then wrapped as a 'ConversationNew' event and finally that event is injected to the TME backend. The initial configuration of a conversation may include a list of trustors belonging to that conversation. The associated trustors are validated expecting to belong to the newly created conversation.

**Expected Results and Sample Data**

Assert.assertEquals("{\"status\":\"OK\"}", status);

Assert.assertEquals(user2Id, iter.next().getId(), DELTA);

Assert.assertEquals(user1Id, iter.next().getId(), DELTA);

### TME test scenario TriggerAlertTest

**Features and Functionalities Addressed**

This test covers the functionality related to the triggering of alerts based on trust concerns.

**API Test Description**

To support this test scenario a newly created conversation accepts a set of outcome events, each one representing a failure. Those outcome events trigger the alteration of the trust level of the registered users. The trust computational model provided by TME GE send alerts whenever any threshold for each metric is exceeded, so that an action is taken if necessary. The expected behaviour for this test is to trigger the generation of an alert of type 'High' as a result of reaching the threshold for a specific metric. Note that the thresholds can refer to either high or low values and these can be defined separately for each metric. In case the registered users belong to different segments then the reaching of thresholds will occur with a different rate.

**Expected Results and Sample Data**

// inject outcome events

for (int j = 0; j < 10; j++) {

rawEvent = new RawEvent();

rawEvent.setName("SessionOutcome");

rawEvent.setConversationId(conversationId);

rawEvent.setResponseTime(400d);

tme.postRawEvent((new Gson()).toJson(rawEvent));

}

// check alert trigger

List<Alert> alerts = alertService.retrieveAlerts();

if (alerts.size() > 0) {

 Assert.assertEquals("HighTrust", alerts.get(0).getType().name());

}

### TME test scenario TrustLevelEstimationTest

**Features and Functionalities Addressed**

This test scenario covers the functionality regarding the evolution of trust levels based on the trust computational model provided by TME GE.

**API Test Description**

In this test scenario the triggering of an outcome event results in the trust level evolution during a certain time window or a single trust value. TME GE exposed API for querying, for a single user, the last trust value from TME (evaluated across all the trust metrics and all interactions he participates) within a time span is evaluated.

**Expected Results and Sample Data**

// post outcome event

rawEvent = new RawEvent();

rawEvent.setName("SessionOutcome");

rawEvent.setConversationId(conversationId);

rawEvent.setResponseTime(400d);

tme.postRawEvent((new Gson()).toJson(rawEvent));

// get trust level

String response = tme.getTimespanOverallTrustUserInteraction("TME", user1Id, start, end);

TMEResponse res = gson.fromJson(response, TMEResponse.class);

Assert.assertEquals("0.715", res.trustValue);

// get trust level

response = tme.getTimespanOverallTrustUserInteraction("TME", user2Id, start, end);

res = gson.fromJson(response, TMEResponse.class);

Assert.assertEquals("0.625", res.trustValue);