Functional Integration Profile Audit Trail Event

Version 00.10

Technologieplattform Smart Grids Austria • Fachhochschule Technikum Wien • OFFIS e.V • AICO EDV-Beratung GmbH • Sprecher Automation GmbH • Tiani Spirit GmbH

Document Information	
Title	
Editor	IES Team
Authors	IES Team
Description	
Last Changes	22.06.2018
sClassification	☐ **RED** – Sensible Information, Access only for: ☐ **YELLOW** – Restricted, Access only for: ☐ **GREEN** – for project-internal usage ☒ WHITE – public

Version History			
Version	**Date**	**Changes from**	**Comment**

Acknowledgements
This paper is a result of the IES project funded by the Austrian Climate and Energy Fund, administrated by the Austrian Research Promotion Agency (FFG) under contract number 853693. It has been prepared in the course of work-packages two, three and four to outline the big picture and to identify the different interoperability issues of smart energy systems. The editors would like to thank all the contributing team members of the IES project for their invaluable contribution of knowledge, experience and support toward a better joint understanding of the complexities involved in safe and smart energy systems.

Table of Content

1 About the Document

A **Technical Framework** represents a technical specification, which is integrated into a predefined document structure. Please note that a technical framework does not equal a new standard. It rather describes the normalised use and application of existing standards and practices to avoid interoperability issues. Integration Profiles state constraints/recommendations that define how to apply standards and good practice to realise a specific feature of a Business Function in an important interoperability fashion. The technical framework is embedded in a business domain overview, which is accessible from the project homepage at http://www.iesaustria.at. The concept is based on the IHE technical framework that subdivides a technical framework into two part: volume 1 for an informative and volume 2 for a normative description. This document describes volume 2.

The document structure of the technical framework is as follows:

Volume 1:

- Business Case Overview (informative)
  - Typical use cases
  - Relevant meta-actors
  - Related standards
- Business Functions (informative)
  - Describe the interoperability issues with the IEC 62559 Use Case Methodology
  - Use Case diagrams

Volume 2:

- Integration Profiles (informative and normative)
  - Technical solution for a specific interoperability issue from the Business Function
  - Definition of transactions that are needed
  - Definition of actors that are involved
- Transactions (normative)
  - Specification of actors that shall be implemented
  - Specification of the IT standards and how options/variants shall be used

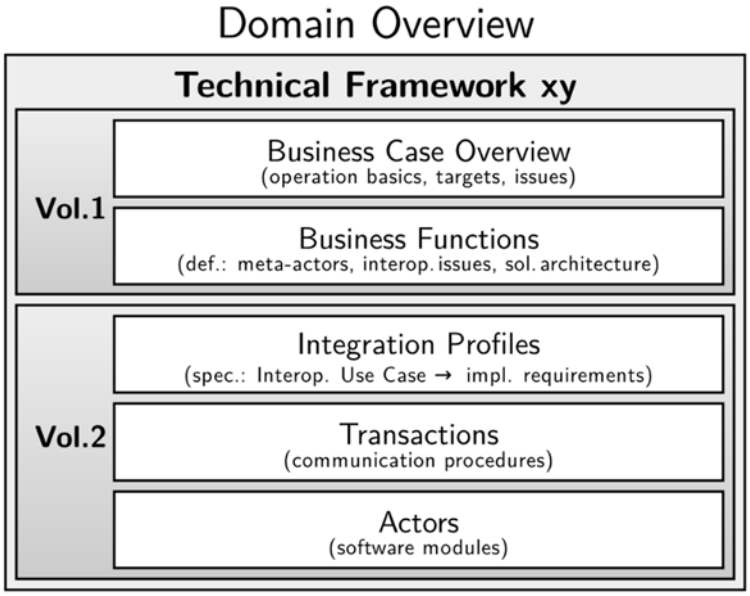

Figure 1: Structure of the Document (IES Technical Framework Template)

2 Definitions

Actor
is a functional software component of a system that executes transactions with other actors as defined in an Integration Profile.

Business Case
is the economic viable application of an idea or technology.

Business Function
is a feature required to be realised for a Business Case to work.

Conformance Testing
is a standalone process to ensure that the implementation conforms to specified standards and profiles, i.e. the implementations outputs and response are checked against rules and patterns.

Integration Profile
is the specification required to realise a part of a Business Function (or combination thereof) in an interoperable fashion (normalised).

Interoperability Testing
is a process to check whether the system interacts effectively with foreign systems, i.e. when different vendors meet to test their interfaces against each other (e.g. Connectathon).

Interoperability Use Case
is a (part of a) Business Function that relies on data exchange between different actors according to an Integration Profile (i.e. where interoperability is required).

Meta-Actor
is the composition (grouping) of all the functional components (actors) that the Meta-Actor is required to integrate in order to perform all the Business Functions related to it (according to the Use Case Diagram). It could be a human operator, but typically it is a software component embedded in some device that provides an interface to some communication infrastructure.

Transaction
is the specification of a set of messages (1..n) exchanged between at least two actors that realise the Use Case specific information exchange (in one or both directions, in a strict or loose order) as specified by an Integration Profile.

Operational Use Case
is a (part of a) Business Function that describes an activity not involving any data exchange between actors. Operational Use Cases are mentioned in the Technical Framework, but not considered by Integration Profiles because per se they do not raise interoperability problems.

3 Functional Integration Profile: Audit Trail Event

In this profile, mainly the data structure is defined that is needed to ensure the documentation of authentication and security aspects in the data exchange between actors. These data include administrative staff to understand the communication between actors and to track possible errors or other interesting events. This profile gives a short overview on the IHE "Audit Record Event" and defines the format of the data to be collected and the minimum set of attributes to be captured by the systems for subsequent use by a review application. So, the data should include records of who accessed data, when, for what action, from where, and which records are involved. At this point, no behavioural requirements are specified for when audit messages are generated, or for what action should be taken on their receipt.

Table 1: Dependencies among Integration Profiles (bundling with external IPs)

Integration Profile	Depends on	Dependency Type	Purpose
Audit Trail Event	IHE - Consistent Time	For a consistent logging, a timestamp is needed	To ensure consistency among timestamps
Audit Trail Event	Check server status	Each communication between two actors shall be logged	Logging actors activity
Audit Trail Event	Set/Get 61850 Data Attribute/Object	Each communication between two actors shall be logged	Logging actors activity
Audit Trail Event	Establish a secure connection	Each communication between two actors shall be logged	Logging actors activity
Audit Trail Event	61850 Send Planned Schedule	Each communication between two actors shall be logged	Logging actors activity
Audit Trail Event	Get Measured Values	Each communication between two actors shall be logged	Logging actors activity
Audit Trail Event	Send Asset Configurations	Each communication between two actors shall be logged	Logging actors activity

3.1 Actors/Transactions

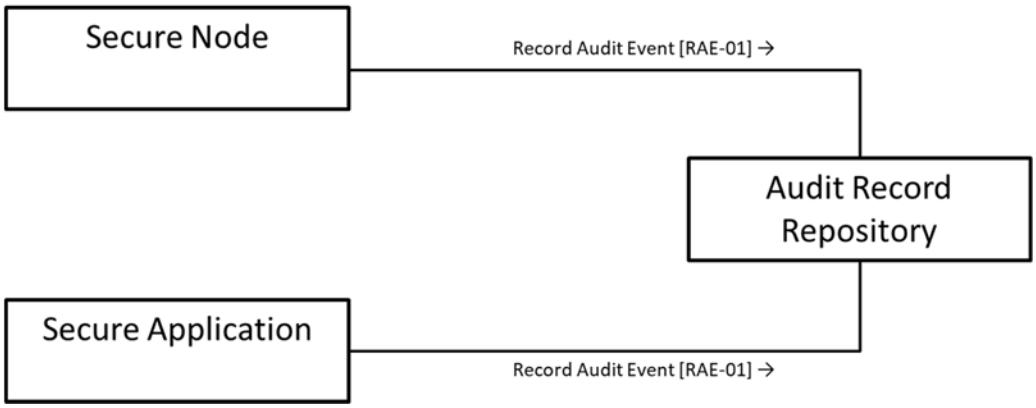

Figure 2: Actors-Transactions Diagram for Record Audit Event

Table 2: Transactions for Check server status

Actors	Transaction	Optionality	Section
Secure Node	Record Audit Event	R	4.1
Secure Application	Record Audit Event	R	4.1
Audit Record Repository	Record Audit Event	R	4.1

3.1.1 Actor Descriptions and Actor Profile Requirements

3.1.1.1 Secure Node

A Secure Node is an actor that provides security and privacy services for all software and services on that system[1]. It is an actor that can be appended to every meta-actor to enable security services in the Smart Energy domain. For this profile, the actor is responsible for the audit record.

3.1.1.2 Secure Application

A Secure Application is an actor that provides security and privacy services for both grouped actors and for functionality provided by related software and services within control of the Secure Application. It is not responsible for the security of its environment, e.g. the operating system and the database.[1] For this profile, the actor is responsible for the audit record and can be appended to every meta-actor.

3.1.1.3 Audit Record Repository

An Audit Record Repository receives audit events and stores them. The capacity of an Audit Record Repository is not specified in this profile, because the variety of deployment needs makes it impractical to set requirements for the report volume and repository capacity needed; so the vendor has to make its own a design decision.[1] For this profile, the actor stores audit records and can be appended to every meta-actor.

3.1.2 Transactions

3.1.2.1 Record Audit Event

The Secure Node/Secure Application creates an audit record and transmits it to an Audit Record Repository. The Audit Record Repository stores the audit record. The connection between both actors has to be a secure connection via TCP/IP. The transmission of the record audit event is specified in Section 4.

3.1.2.2 Messages for Record Audit Event

The audit records are used for different application scenarios; so the content of records can vary, e.g. Audit Message for Query, Application Activity Message or User Authentication; however, the implementation for transmitting the audit record is always the same that is implemented with the transaction “Record Audit Event”. So, the SCD file with different message content is specified in Section 4.

[1] More information can be found at: http://www.ihe.net/uploadedFiles/Documents/ITI/IHE_ITI_TF_Vol1.pdf#nameddest=9__Audit_Trail_and_Node_Authent S.74

3.2 Actor Options

Table 3: Actor Options for Audit Trail Event

Actor	Role	Option	Vol. & Section
Secure Node	Client	Smart Energy Audit Trail	
Secure Application	Client	Smart Energy Audit Trail	
Audit Record Repository	Server	none	

3.2.1 Smart Energy Audit Trail

The Audit Trail Event is specified in the healthcare domain by the DICOM Supplement 95; in this profile, the event is adapted to the energy sector, so the actor option is changed to “Smart Energy Audit Trail”. The Smart Energy Audit Trail is an exemplary specification how audit trails can look like in the energy sector, it enables the record of actors’ actions to provide security and privacy mechanism. It defines audit events on the communication between actors in the energy domain. A schema is specified for encoding a reported event, so that all actions related to the transactions can be logged in an Audit Event Repository.

3.3 Information Flow Process

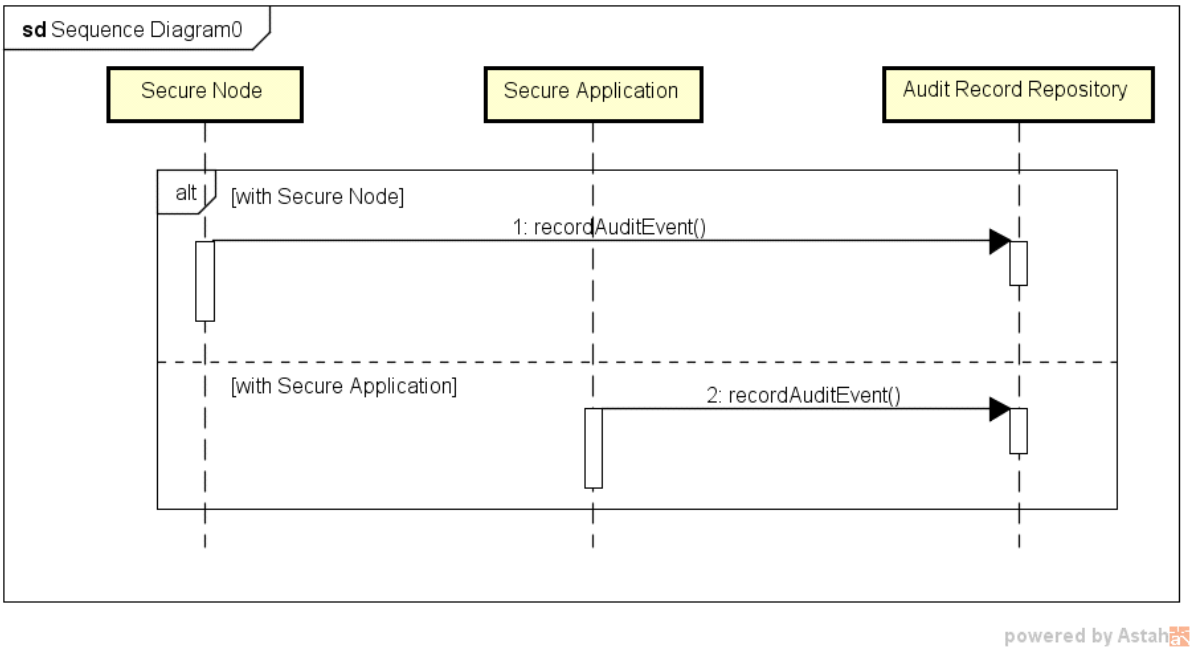

Figure 3: Sequence Diagram for "Audit Trail Event"

3.4 Implementation Strategies

The IHE ATNA specification does not have a concrete implementation strategy for the event logging. Audit events has to be created with the specified XSD schema in Section 4.2 and the restrictions for the attributes has to be considered.

3.5 Communication Requirements

For the event logging a TLS connections between the Secure Node/Secure Application is needed. TLS provides mutual authentication, reliable message transport and private communication through data encryption. ATNA does not restrict implementations and deployments to only use the ATNA

specified methods. For interoperability reasons, TLS must be implemented and available to be configured.[2] Otherwise, the communication requirements specified in the Operational Integration Profile shall be considered.

3.6 Security Considerations

Only a TLS connection has to be implemented and operational, technical and administrative aspects have to be considered. These are often specified by law or regulations, e.g. data protection laws and regulations for in-house back-ups have to be respected by implementing the Audit Trail Event.

[2] IHE IT Infrastructure Technical Framework, Vol. 1: Integration Profiles, S. http://www.ihe.net/uploadedFiles/Documents/ITI/IHE_ITI_TF_Vol1.pdf#nameddest=9__Audit_Trail_and_Node_Authent

4 Transactions

4.1 Transaction: Record Audit Event

The transaction “Record Audit Event” is defined in IHE IT Infrastructure TF Vol. 2a. At this point, a short overview of the transaction is given.

4.1.1 Scope

The Secure Node/Secure Application creates an audit event with the data structure defined in Section 4.2 and stores it in the Audit Record Repository.

4.1.2 Actor Roles

Table 4: Actor Roles for "Record Audit Event"

Role	Description	Actor
Creator	The creator creates an audit record and transmits this record to the receiver.	The following actors may play the role of the client: Secure Node, Secure Application
Receiver	The receiver gets an audit record from the creator and this for audit purposes.	The following actors may play the role of the server: Audit Record Repository

4.1.3 Referenced Standards

The referenced standards are listed in the IHE transaction “Record Audit Event” at the “IHE IT Infrastructure TF, Vol. 2a”. For the Smart Energy area, the following standards are necessary:

- IETF RFC 5424 The Syslog Protocol
- IETF RFC 5425 Transmission of Syslog Messages over TLS
- IETF RFC 5426 Transmission of Syslog Messages over UDP
- NIST SP 800-92 Guide to Computer Security Log Management
- W3C Recommendation: Extensible Markup Language (XML)
- DICOM Supplement 95: Audit Trail Messages

4.1.4 Interaction Diagrams

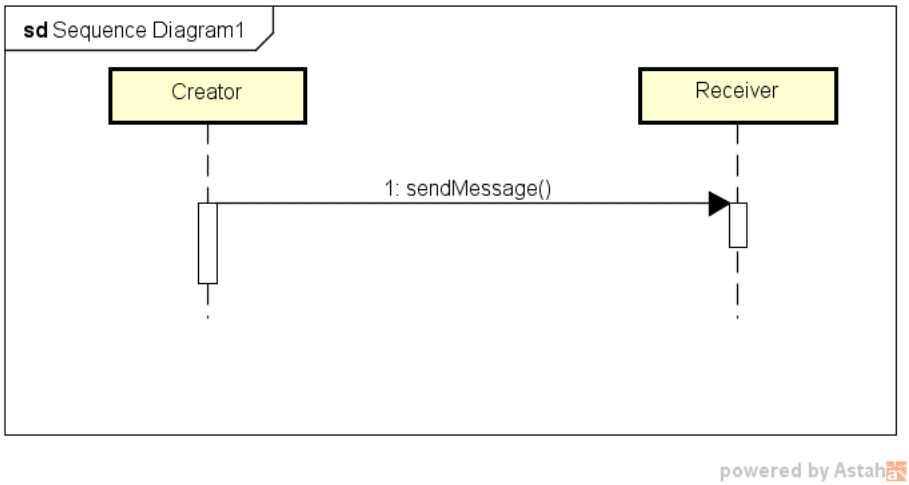

Figure 4: Sequence Diagram for "Record Audit Event"

4.2 Transaction: Messages for Record Audit Event

In this transaction, different kind of data structures for the Record Audit Event are specified. The message and field names are adopted from the Digital Imaging and Communication in Medicine (DICOM) Supplement 95: Audit Trail Messages.[3] If necessary, fields were added, the optionality was changed and enumerations in the appendix were adapted for the Smart Energy area.
The XML schema for the record audit event is as follows:

```
<?xml version='1.0'?>

<!DOCTYPE xs:schema PUBLIC "-//W3C//DTD XSD 1.1//EN" "XMLSchema.dtd" [

<!-- provide ID type information even for parsers which only read the
     internal subset -->
<!ATTLIST xs:schema          id  ID  #IMPLIED>
<!ATTLIST xs:complexType     id  ID  #IMPLIED>
<!ATTLIST xs:complexContent  id  ID  #IMPLIED>
<!ATTLIST xs:simpleContent   id  ID  #IMPLIED>
<!ATTLIST xs:extension       id  ID  #IMPLIED>
<!ATTLIST xs:element         id  ID  #IMPLIED>
<!ATTLIST xs:group           id  ID  #IMPLIED>
<!ATTLIST xs:all             id  ID  #IMPLIED>
<!ATTLIST xs:choice          id  ID  #IMPLIED>
<!ATTLIST xs:sequence        id  ID  #IMPLIED>
<!ATTLIST xs:any             id  ID  #IMPLIED>
<!ATTLIST xs:anyAttribute    id  ID  #IMPLIED>
<!ATTLIST xs:attribute       id  ID  #IMPLIED>
<!ATTLIST xs:attributeGroup  id  ID  #IMPLIED>
<!ATTLIST xs:unique          id  ID  #IMPLIED>
<!ATTLIST xs:key             id  ID  #IMPLIED>
<!ATTLIST xs:keyref          id  ID  #IMPLIED>
<!ATTLIST xs:selector        id  ID  #IMPLIED>
<!ATTLIST xs:field           id  ID  #IMPLIED>
<!ATTLIST xs:assert          id  ID  #IMPLIED>
<!ATTLIST xs:include         id  ID  #IMPLIED>
<!ATTLIST xs:import          id  ID  #IMPLIED>
<!ATTLIST xs:redefine        id  ID  #IMPLIED>
<!ATTLIST xs:override        id  ID  #IMPLIED>
<!ATTLIST xs:notation        id  ID  #IMPLIED>
]>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified" xml:lang="EN"
           targetNamespace="http://www.w3.org/2001/XMLSchema"
           version="structures.xsd (rec-20120405)">
       <complexType name="Event" minOccurs="1" maxOccurs="1">
              <xs:element name="EventID" minOccurs="1" maxOccurs="1" type="String"/>
              <xs:attribute name="EventActionCode" default="string" use="required">
                     <xs:simpleType>
                            <xs:restriction base="xs:string">
                                   <xs:enumeration value="C"/> <!-- Create -->
                                   <xs:enumeration value="R"/> <!-- Read -->
                                   <xs:enumeration value="U"/> <!-- Update -->
                                   <xs:enumeration value="D"/> <!-- Delete -->
                                   <xs:enumeration value="E"/> <!-- Execute -->
                            </xs:restriction>
                     </xs:simpleType>
              </xs:attribute>
              <xs:element name="EventDescription" minOccurs="0" maxOccurs="1" type="String"/>
              <xs:element name="EventDateTime" minOccurs="1" maxOccurs="1" type="xs:date"/>
              <xs:attribute name="EventOutcomeIndicator" default="string" use="required">
                     <xs:simpleType>
                            <xs:restriction base="xs:string">
                                   <xs:enumeration value="0"/> <!-- Nominal success (use if
status otherwise unknown or ambiguous) -->
                                   <xs:enumeration value="4"/> <!-- Minor failure (per
reporting Meta-Actor definition) -->
                                   <xs:enumeration value="8"/> <!-- Serious failure (per
reporting Meta-Actor definition) -->
                                   <xs:enumeration value="12"/> <!-- Major failure
(reporting Meta-Actor now unavailable) -->
```

[3] ftp://medical.nema.org/medical/dicom/final/sup95_ft.pdf

```
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="EventTypeCode" default="string" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="schedule"/> <!-- Schedule -->
                            <xs:enumeration value="measurements"/> <!-- Measurement -
->
                            <xs:enumeration value="configuration"/> <!-- Asset
Configurations -->
                            <xs:enumeration value="REP"/> <!-- Report -->
                            <xs:enumeration value="ERR"/> <!-- Error -->
                            <xs:enumeration value="LI"/> <!-- Login -->
                            <xs:enumeration value="LO"/> <!-- Logout -->
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:complexType name="Actor" minOccurs="1" maxOccurs="2">
                <xs:element name="MetaActorID" minOccurs="1" maxOccurs="1" type="String"/>
                <xs:element name="MetaActorName" minOccurs="0" maxOccurs="1" type="String"/>
                <xs:element name="MetaActorIsRequestor" minOccurs="1" maxOccurs="1"
type="Boolean"/>
                <xs:attribute name="RoleIDCode" default="string" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Client"/> <!-- The client is the
transaction initiator that starts and sends or requests the values. -->
                            <xs:enumeration value="Server"/> <!-- The server is the
transaction responder that receives the values. -->
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:element name="NetworkAccessPointTypeCode" minOccurs="0" maxOccurs="1"
type="String"/>
                <xs:element name="NetworkAccessPointID" minOccurs="1" maxOccurs="1"
type="String"/>
        </xs:complexType>
</xs:schema>
```

4.2.1 Audit Message for Query

This message describes the event of a query being issued or received. It does not record the response to the query; it only records the fact that a query was issued.

Table 5: Semantic for "Audit Message for Query"

Real World Entities	Field Name	Opt.	Value Constraints
Event	EventID	M	Identity of the event, String 255
	EventActionCode	M	Enumerated Value (have a look at Table 8): E
	EventDescription	C	not specialized, Lob type
	EventDateTime	M	not specialized (consider requirements from the IHE consistence time profile)
	EventOutcomeIndicator	M	Enumerated Value (have a look at Table 9)
	EventTypeCode	M	Enumerated Value (have a look at Table 10): *measurements* or *configuration*
Meta-Actor issuing the query	MetaActorID	M	Identity of the Meta-Actor that starts the event, String 255
	MetaActorName	C	not specialized, String 255
	MetaActorIsRequestor	M	Boolean
	RoleIDCode	M	Enumeration (have a look at Table 11)
	NetworkAccessPointTypeCode	C	not specialized, String 255
	NetworkAccessPointID	M	IP address of the Meta-Actor, String 255

Meta-Actor responding to the query	MetaActorID	M	Identity of the Meta-Actor that starts the event, String 255
	MetaActorName	C	not specialized, String 255
	MetaActorIsRequestor	M	Boolean
	RoleIDCode	M	Enumeration (have a look at Table 11)
	NetworkAccessPointTypeCode	C	not specialized, String 255
	NetworkAccessPointID	M	IP address of the Meta-Actor, String 255

4.2.2 Application Activity Message

This message describes an event for tracking any kind of application entity e.g. starting or stopping applications, reports, or failures.

Table 6: Semantic for "Application Activity Message"

Real World Entities	Field Name	Opt.	Value Constraints
Event	EventID	M	Identity of the event, String 255
	EventActionCode	M	Enumerated Value (have a look at Table 8)
	EventDescription	C	not specialized, Lob type
	EventDateTime	M	not specialized (consider requirements from the IHE consistence time profile)
	EventOutcomeIndicator	M	Enumerated Value (have a look at Table 9)
	EventTypeCode	M	Enumerated Value (have a look at Table 10): *schedule*, *measurements*, *configuration*, REP, or ERR
Meta-Actor started application	MetaActorID	M	Identity of the Meta-Actor that starts the event, String 255
	MetaActorName	C	not specialized, String 255
	MetaActorIsRequestor	M	Boolean
	RoleIDCode	M	Enumeration (have a look at Table 11)
	NetworkAccessPointTypeCode	C	not specialized, String 255
	NetworkAccessPointID	M	IP address of the Meta-Actor, String 255

4.2.3 User Authentication

Actors have attempted to log in or log off to a system before data can be exchanged; the message log is independent of the success of the event.

Table 7: Semantic for "User Authentication"

Real World Entities	Field Name	Opt.	Value Constraints
Event	EventID	M	Identity of the event, String 255
	EventActionCode	M	Enumerated Value (have a look at Table 8): E
	EventDescription	C	not specialized, Lob type
	EventDateTime	M	not specialized (consider requirements from the IHE consistence time profile)
	EventOutcomeIndicator	M	Enumerated Value (have a look at Table 9)
	EventTypeCode	M	Enumerated Value (have a look at Table 10): LI or LO
Meta-Actor issuing the authentica-tion	MetaActorID	M	Identity of the Meta-Actor that starts the event, String 255
	MetaActorName	C	not specialized, String 255
	MetaActorIsRequestor	M	Boolean

	RoleIDCode	M	Enumeration (have a look at Table 11)
	NetworkAccessPointTypeCode	C	not specialized, String 255
	NetworkAccessPointID	M	IP address of the Meta-Actor, String 255
Meta-Actor performing the authentica-tion	MetaActorID	M	Identity of the Meta-Actor that starts the event, String 255
	MetaActorName	C	not specialized, String 255
	MetaActorIsRequestor	M	Boolean
	RoleIDCode	M	Enumeration (have a look at Table 11)
	NetworkAccessPointTypeCode	C	not specialized, String 255
	NetworkAccessPointID	M	IP address of the Meta-Actor, String 255

5 Appendix

5.1 Enumeration for Event Action Code

Table 8: Event Action Code

Event Action Code	Description
C	Create
R	Read
U	Update
D	Delete
E	Execute

5.2 Enumeration for Event Outcome Indicator

Table 9: Event Outcome Indicator

Event Outcome Indicator	Description
0	Nominal success (use if status otherwise unknown or ambiguous)
4	Minor failure (per reporting Meta-Actor definition)
8	Serious failure (per reporting Meta-Actor definition)
12	Major failure (reporting Meta-Actor now unavailable)

5.3 Enumeration for Event Type Code

Table 10: Event Type Code

Event Type Code	Description
schedule	Schedule
measurements	Measurement
Configuration	Asset Configurations
REP	Report
ERR	Error
LI	Login
LO	Logout

5.4 Enumeration for Role ID Code

Table 11: Role ID Code

Role ID Code	Description
Client	The client is the transaction initiator that starts and sends or requests the values.
Server	The server is the transaction responder that receives the values.