



Target Outcome: b) Sustainable technologies for a Smarter Society

(FP7 614100)

D7.4.1 Initial Design and implementation of the IoT Event Debugging Tool

Published by the IMPRESS Consortium

Dissemination Level: Public



Project co-funded by the European Commission within the 7th Framework Programme and the Conselho Nacional de Desenvolvimento Científico e Tecnológico Objective ICT-2013.10.2 EU-Brazil research and development Cooperation

Document control page

Document file:

D7_4_1_Initial_Design_and_implementation_of_the_IoT_EventDebugging_Tool.doc

Document version: 1.0

Document owner: Peeter Kool (CNET)

Work package: WP7 – IDE Framework for Model-driven development

Task: T7.5 – IoT Event Debugging Tools

Deliverable type: P

Document status: approved by the document owner for internal review

 approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Peeter Kool (CNET)	2014-05-01	Initial ToC
0.5	Peeter Kool, Peter Rosengren (CNET)	2014-06-27	Initial Content
0.8	Peter Rosengren (CNET)	2014-08-10	Additional content
0.9	Peeter Kool (CNET)	2014-08-15	Additional content, editing and spell check, versions for internal review
1.0	Peter Rosengren, Peeter Kool (CNET)	2014-08-31	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Ferry Pramudianto	2014-08-27	Accepted with comments

Legal Notice

The information in this document is subject to change without notice.

The Members of the Impress Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Impress Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Possible inaccuracies of information are under the responsibility of the project. This report reflects solely the views of its authors. The European Commission is not liable for any use that may be made of the information contained therein.

Index:

1. Executive summary	5
2. Introduction	6
3. Developer Tools for event based LinkSmart systems	7
3.1 Windows LinkSmart Event Manager	7
3.2 Event Trace and Debug Tool (ETDT)	8
3.2.1 Event Manager Eavesdrop	9
3.2.2 Event Network Browser	13
4. Future Work	17
5. Class Documentation	18
5.1 EventQuery::EventQuery Class Reference	18
Public Member Functions	18
Detailed Description	18
Constructor & Destructor Documentation	18
Member Function Documentation	19
5.2 IoTWCFSERVICELibrary::EventQueryServiceWS Class Reference	21
Public Member Functions	21
Private Attributes	21
Constructor & Destructor Documentation	21
Member Function Documentation	21
Member Data Documentation	22
5.3 IoTWCFSERVICELibrary::IIoTEventQuery_EventQueryServiceWSService	
Interface Reference	23
Public Member Functions	23
Member Function Documentation	23
5.4 EventTraceAndDebugTool::DataBaseReader Class Reference	23
Public Member Functions	23
Properties	24
Private Member Functions	24
Private Attributes	24
Detailed Description	25
Member Function Documentation	25
Member Data Documentation	26
Property Documentation	26
5.5 EventDB::EventDB Class Reference	27
Public Member Functions	27
Public Attributes	27
Private Attributes	27
Detailed Description	27
Constructor & Destructor Documentation	28
Member Function Documentation	28
Member Data Documentation	28
Member Data Documentation	29
5.6 EventReceiver::EventReceiver Class Reference	30
Public Member Functions	30
Public Attributes	30
Private Member Functions	30
Private Attributes	31
Member Function Documentation	31

Member Data Documentation	33
5.7 EventReceiver::EventReceiverService Class Reference	34
Public Member Functions	34
Public Attributes	34
Private Member Functions.....	34
Private Attributes	34
Member Function Documentation.....	35
Member Data Documentation	36
5.8 EventSubscriberClient Class Reference	37
Public Member Functions	37
Constructor & Destructor Documentation.....	37
Member Function Documentation.....	38
5.9 EventQueue::MSMQ Class Reference.....	39
Public Member Functions	39
Properties.....	39
Private Member Functions.....	39
Private Attributes	39
Detailed Description.....	40
Constructor & Destructor Documentation.....	40
Member Function Documentation.....	40
Member Data Documentation	41
Property Documentation	41
5.10 EventQueue::Queue Interface Reference	42
Public Member Functions	42
Properties.....	42
Detailed Description.....	42
Constructor & Destructor Documentation.....	42
Member Function Documentation.....	43
Property Documentation	44
5.11 EventTraceAndDebugTool::RESTServer Class Reference	45
Public Member Functions	45
Static Public Member Functions	45
Static Private Member Functions.....	45
Private Attributes	45
Member Function Documentation.....	46
Member Data Documentation	48
6. References	49

1. Executive summary

This deliverable describes the initial prototype deliverable developed within task T7.5 IoT Event Debugging Tools.

The first section is a short introduction. The second section gives an overview of the Event Debugging Tools and their functionality. The third section outlines the functionality to be included in the final deliverable D7.4.2 Final Design and implementation of the IoT Event Debugging Tool. Finally there is a section with class documentation of the most important classes.

2. Introduction

The aim of the task T7.5 IoT Event Debugging Tools is to develop a high-level debugging tool which allows developers to trace events and interactions between distributed components. Event Management is a crucial function in the IoT ecosystem in general enabling both loosely coupled communications and data management. In highly distributed systems involving large numbers of devices and actors, the possibility of doing event traceability and debugging are important. For this reason the IMPReSS platform researches and designs mechanisms and tools that support the developers tracing and debugging event patterns and event history, using the LinkSmart event processing architecture.

In this first initial prototype we have been designing and building the initial infrastructure for the IoT Event Debugging Tools. This document describes this initial prototype and also gives an overview of the architecture. Finally we look at the future work that we foresee to be included in to the final prototype.

3. Developer Tools for event based LinkSmart systems

In the first sub section the existing Windows LinkSmart Event Manager is described briefly with regards to the current debugging and tracing possibilities. The second sub section describes the current state of the Event Trace and Debug Tool developed within IMPReSS including the architecture of the tool.

3.1 Windows LinkSmart Event Manager

The windows based LinkSmart event manager provides a visual tool that controls the LinkSmart event manager and also shows it status, see Figure 1. The main purpose of the tool is to provide information of what’s currently going on with the event manager if there are any communication errors or that a subscriber is not responding properly.

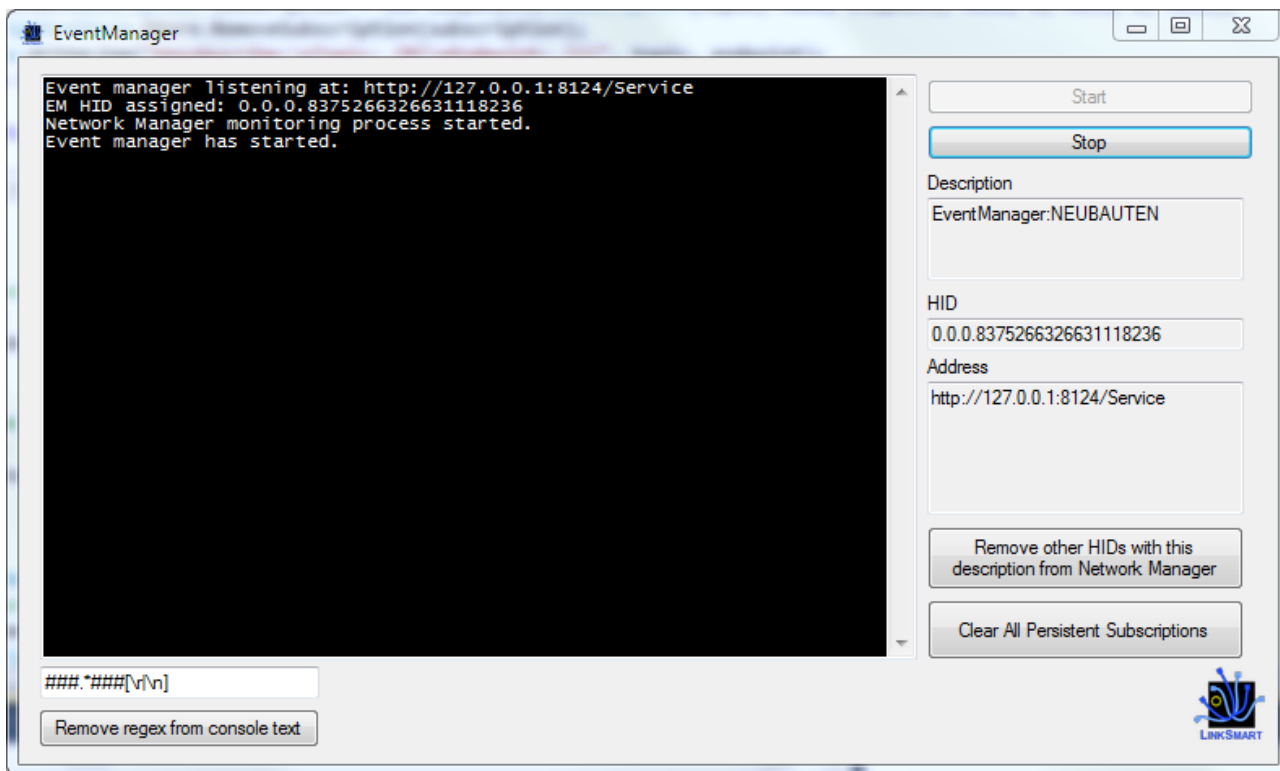


Figure 1: Windows LinkSmart event manager

In the tool it is possible to stop and start the event manager, remove all subscriptions and to clear the all other HID's that match the description of the event manager. The last function is very useful when debugging to avoid problems with dead HID's. The tool window it also shows the status and current settings of the Event Manager:

- **Description:** The description the Event Manager uses when it registers at the Network Manager.
- **HID:** The HID for the Event Manager.

- **Address:** The endpoint where the Event Manager Web Service is published

When a subscriber subscribes to events the information will be shown in the console window in the tool, see Figure 2 below.

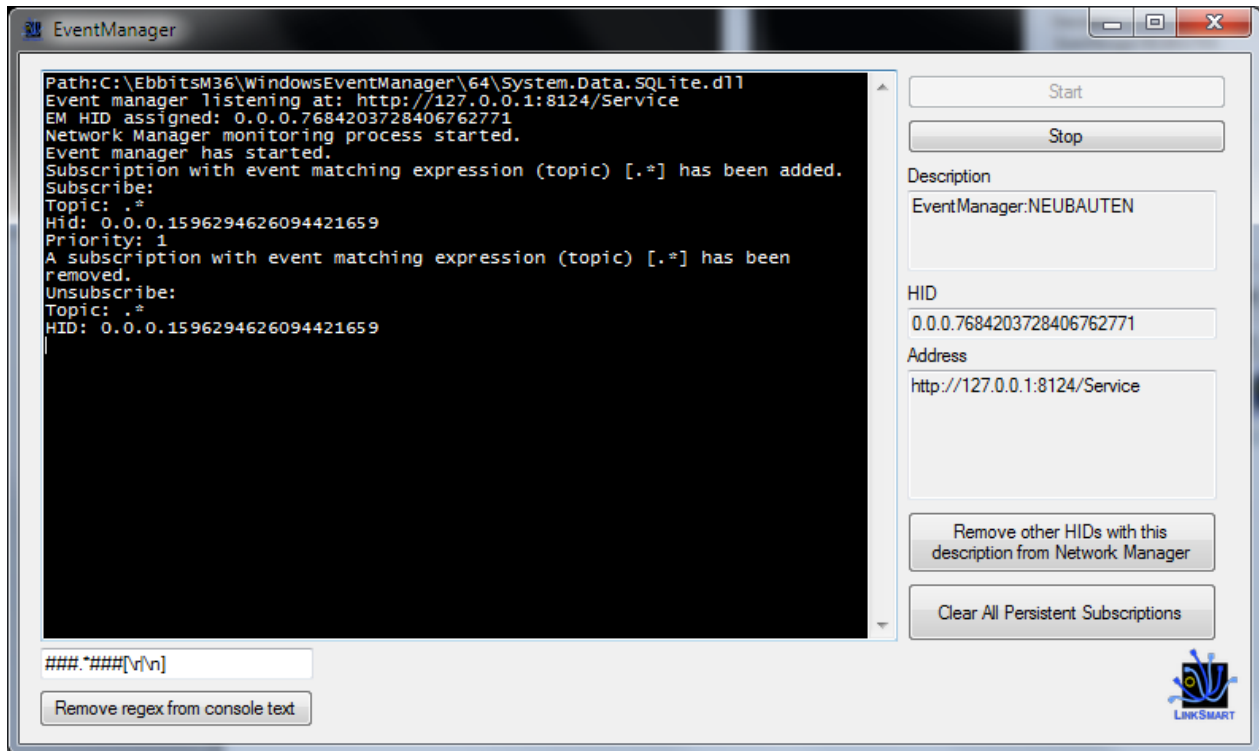


Figure 2: Subscription created and removed

When a subscription is removed it will also be displayed in the console window inside the tool. The information shown when a subscription is created or removed are:

- The HID of the subscriber, i.e. which client it is.
- The Topic of the subscription, i.e. what events it listens to

Errors that occur when the Event Manager distributes events will also be shown in the console. In this case the actual exception message will be shown.

3.2 Event Trace and Debug Tool (ETDT)

There are two basic parts of the Event Trace Debug Tool:

- Event Manager Eavesdrop: Provides the functionality to listen to and process all events that pass through an Event Manager. This also contains a simple browser with query capabilities.
- Event Network Browser: Provides functionality to query individual event consumers and producers which events they have produced and consumed.

3.2.1 Event Manager Eavesdrop

The Event Trace and debug tool provides the capabilities of eavesdropping on all event communication at an event manager. This tool is not part of the event manager like the Windows Event Manager described in the previous section, instead it is run side by side with the LinkSmart Event Manager and can be turned off/on completely independently, see Figure 3.

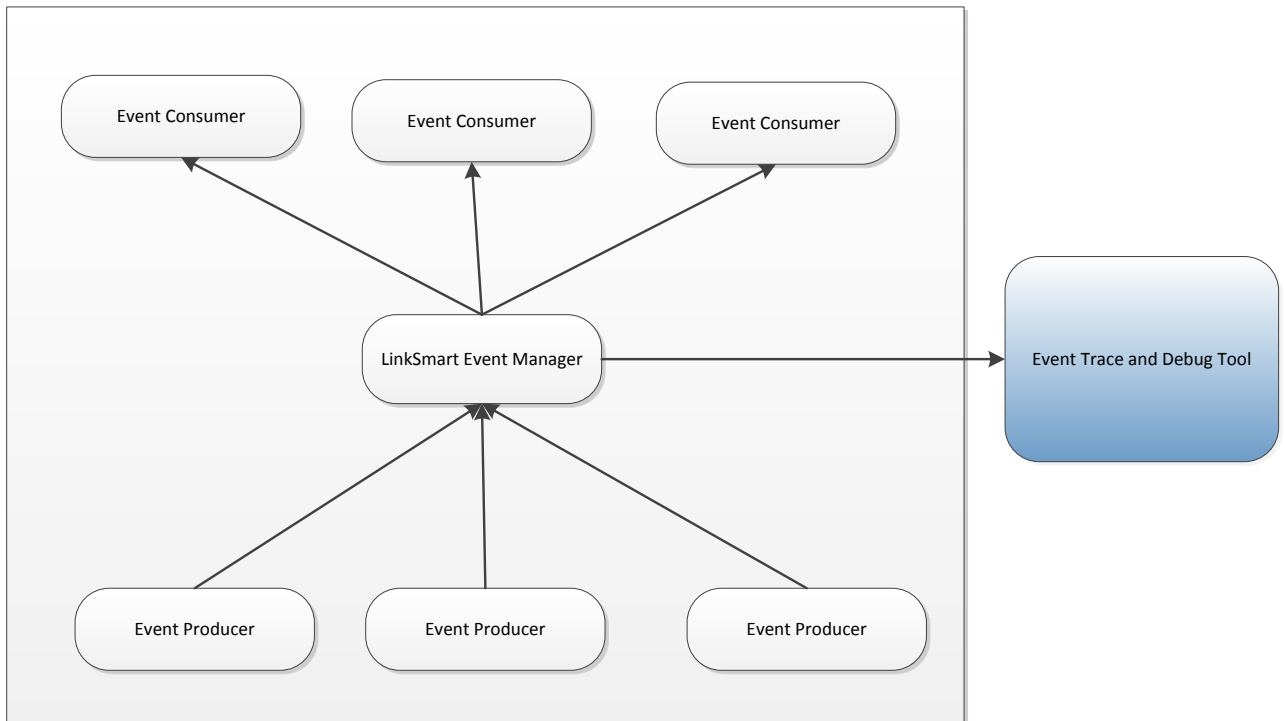


Figure 3: LinkSmart and the Event Trace and Debug Tool

The Event Trace and Debug Tool act as a normal event consumer from the LinkSmart Event Manager but it listens to all events that pass through the Event Manager without any filtering. Because of this it is essential that the Event Trace and Debug Tool is well behaved and does not introduce problems by listening, i.e. it should be transparent for the system and the system behavior should not change when the ETDT is used.

The inner architecture of ETDT reflects this by trying to be as efficient as possible. LinkSmart Event Manager can be handling a large number of events per seconds and therefore the ETDT must also be fast in its processing of events in order not to unnecessarily load the Event Manager, see Figure 4.

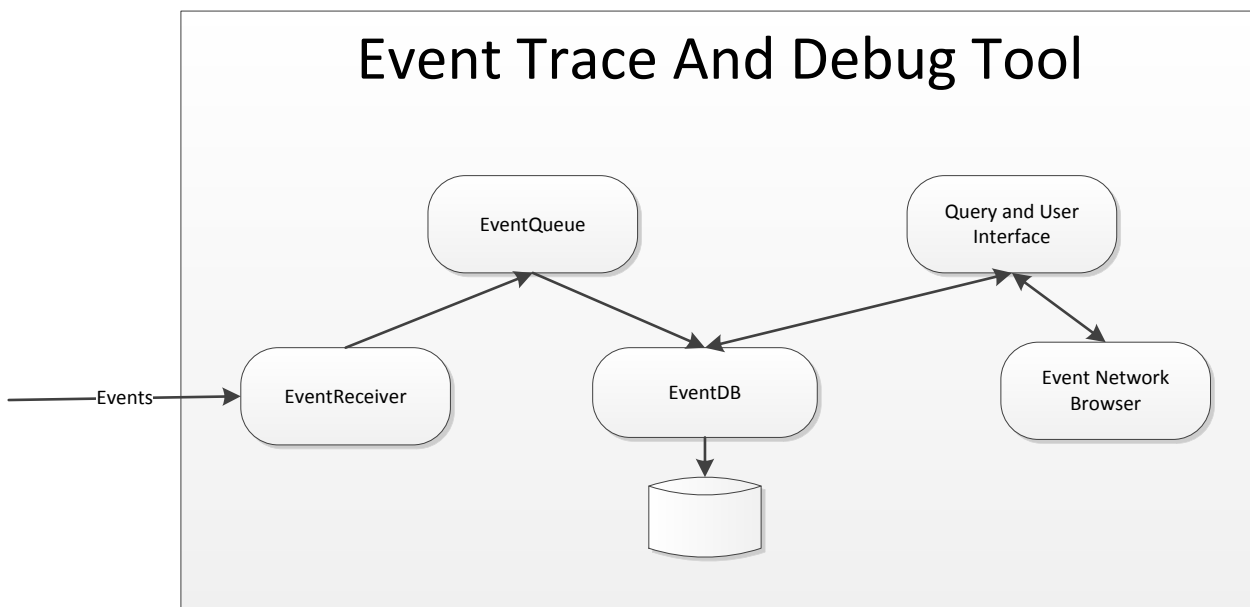


Figure 4: Inner architecture of the ETDT

The main components in the ETDT are:

- **Event Receiver:** Listens to the events. As soon as an event arrives it will queue in the EventQueue without any other processing.
- **EventQueue:** Acts as a buffer in between the persistent store and incoming events. It uses MSMQ as mechanism guarantying that no events are lost.
- **EventDB:** Manages the persistent store of events and provides interfaces to query the stored events database. The default implementation uses SQLite for storage, but this can be changed to in memory databases et c.
- **Query and User Interface:** Is the actual consumer of the stored event data.
- **Event Network Browser:** This part is explained in 3.2.2 Event Network Browser

The ETDT provides a web based user interface that can be used with any ordinary web browser, see Figure 5.

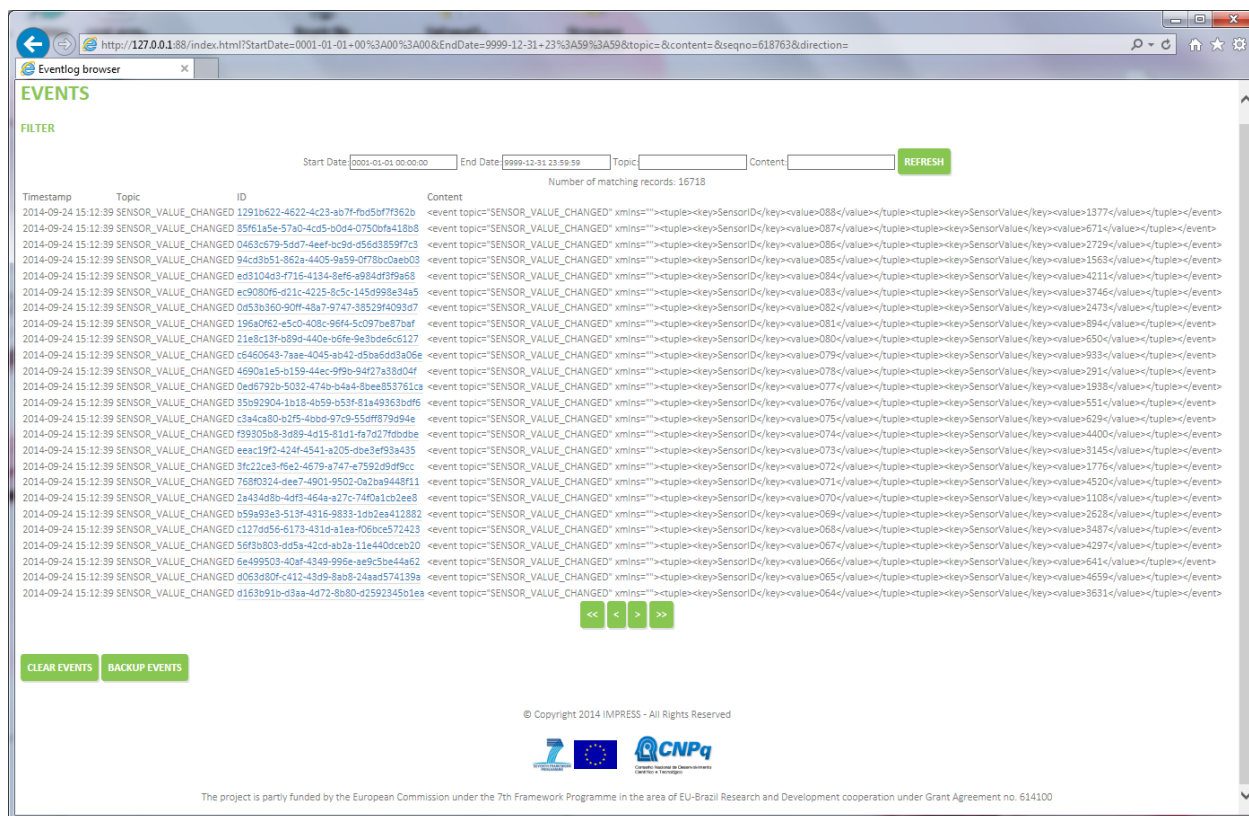


Figure 5: ETDT web based user interface

The tool presents the events always with the newest events first and always in the exact order they have arrived to the tool. The page also contains filters that one can apply for selecting events. An example of this is shown in Figure 6.

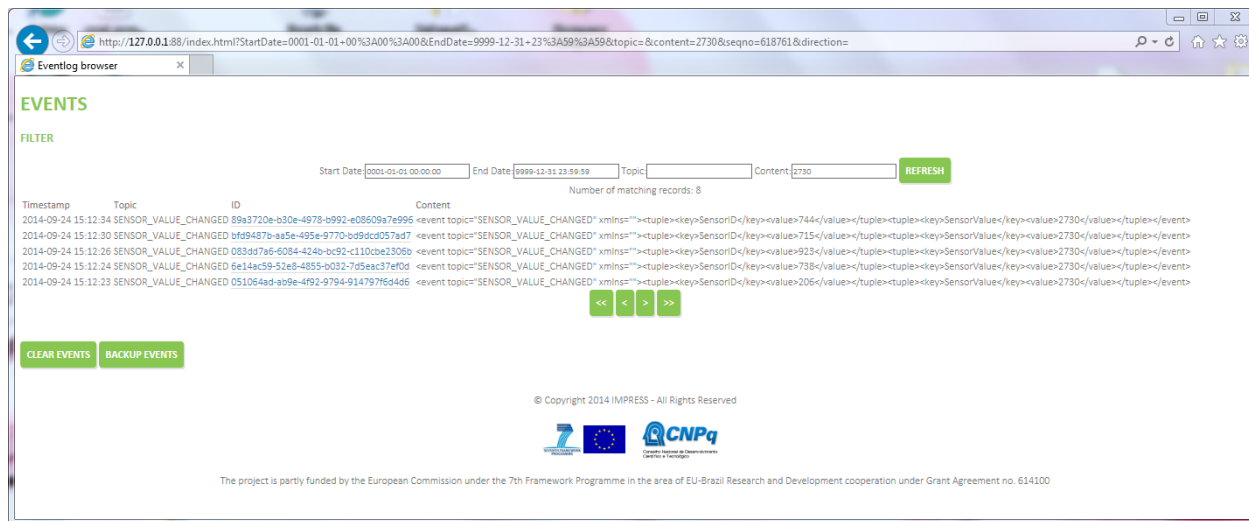


Figure 6: Example of a filter

Here the user has selected to filter on the contents of the events by filling in "12". The user interface will reflect this by only showing events that contain this information. The properties that can be filtered currently are:

- **StartDate:** The earliest date and time of the event.
- **EndDate:** The earliest date and time of the event.
- **Topic:** Any part of the Topic should contain the entered string.
- **Content:** Any part of the Event content should contain the entered string.

There are two more buttons available in the user interface which can be useful when dealing with events:

- **CLEAR EVENTS:** Will clear the database of all previous events making it empty.
- **BACKUP EVENTS:** This will create a backup of the database currently in use. This backup can be used for further analysis with other tools. An interesting option is that this backup can be used for replaying the events in the exact order that they occurred which can be useful for simulation. The database backup is in SQLite 3 format which can easily be exported to other formats.

Finally there is a link on the ID for each event. Clicking this link will gather the full information about the event which will be returned in an XML structure, see Listing 1.

```
<?xml version="1.0" encoding="utf-8"?>
<EventStructure xmlns="http://events.linksmart.org/EpaEvent"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <EventMeta>
    <EventType modelRef="IMPRESSEvent">CHANGE_SENSOR_VALUE</EventType>
    <EventID>c2562eb6-4b70-440d-83a4-fc24919b00c6</EventID>
    <Topic modelRef="IMPRESS">CHANGE_SENSOR_VALUE</Topic>
    <Timestamp>2014-02-24T13:10:12.6299322+01:00</Timestamp>
    <Comment />
    <Source>
      <Location />
      <ObjectID modelRef="IMPRESS">SENSOR_ID_21</ObjectID>
      <ProcessID />
    </Source>
  </EventMeta>
  <Content modelRef="">
    <event topic="CHANGE_SENSOR_VALUE" xmlns="">
```

```
<tuple>
  <key>HEAT</key>
  <value>12</value>
</tuple>
</event>
</Content>
</EventStructure>
```

Listing 1: The full event structure in XML

As can be seen all elements do not have value, these are optional and depends on the original event producer if they will have values. In the example above we do not have any information regarding the actual Location of the event because it is not supplied by the event producer.

3.2.2 Event Network Browser

The Event Network Browser part of the Event Trace and Debugging Tool provides functionality to look inside individual event producers and consumers to see which events they have created and/or consumed. This is very useful when debugging complex event problems where it is not clear who created the event and who consumed it. Typically this functionality can be used to pin point which component is not behaving as expected, i.e. not consuming the correct events or not creating the correct events.

The Event Network Browser part uses functionality in LinkSmart to find and connect to the different event consumers and producers that are part of the LinkSmart network, see Figure 7 below.

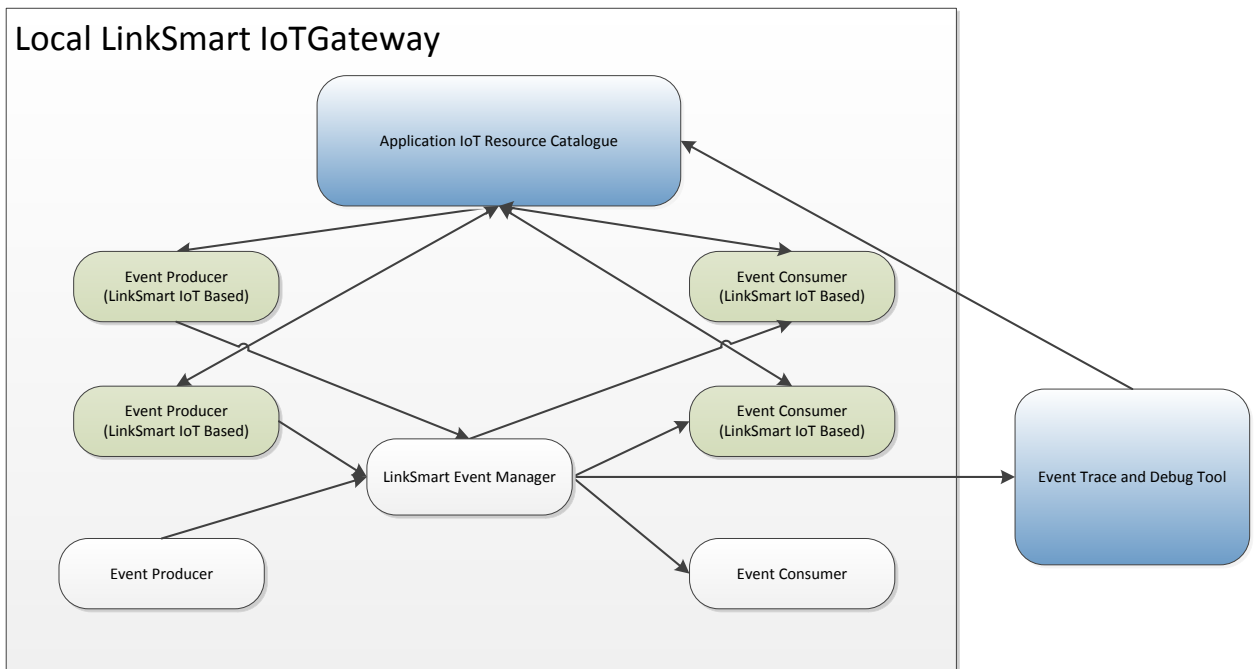


Figure 7: Overview of the architecture of the Event Network Browser architecture

The basic principle used by the Event Network Browser is that it uses the Application IoT Resource Catalogue, which is part of LinkSmart, to find all the LinkSmart based event producers and consumers currently on the network. Using the information returned from the catalogue the Event Network Browser is able to contact the individual producers and consumers.

In order for the Event Network Browser to be able to query the producers and consumers a new service has been created within the LinkSmart IoTResource class library which exposes a Webservice and REST interface where queries can be made with regards to which events it has produced or consumed, see Figure 8 below.

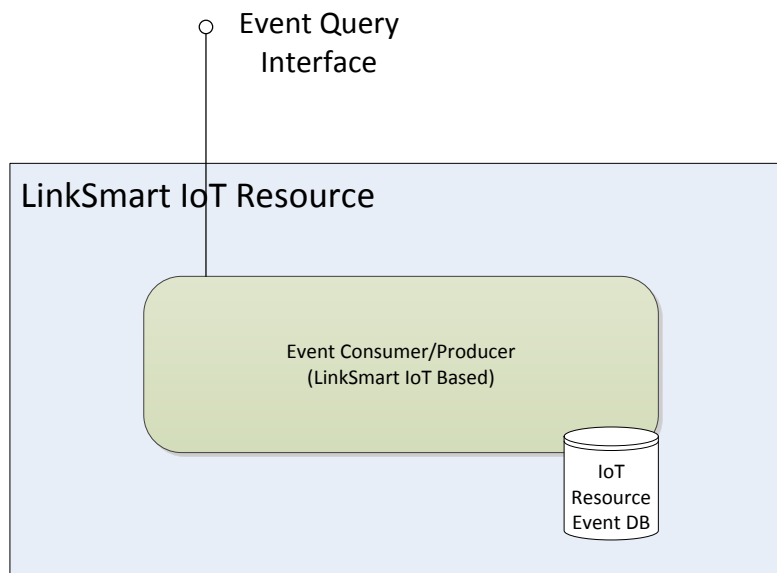


Figure 8: LinkSmart IoT resource Event Query Interface extension

All LinkSmart IoT Resource based components will automatically get this added functionality requiring no change to the original IoT Resource. It only requires that the developer updates the LinkSmart libraries used. Note also that this functionality not only provides the interface but also provides the actual storage of events produced or consumed.

The Event Debugging and Trace Tool provide a simple web based interface for browsing for IoT Resources and to look at the events produced or consumed by the resource. The interface for browsing and selecting IoT Resources is shown in Figure 9 below.



Figure 9: IoT Resource browser in the Event Debug and Trace Tool

The information shown in the tool contains the resources name, the resource type and a link to show the events that have been processed by the resource. Clicking Show Events will display basically the same event browsing tool used by the Event Manager Eavesdrop but with the exception that one can select to filter on received or sent events, see Figure 10 below.

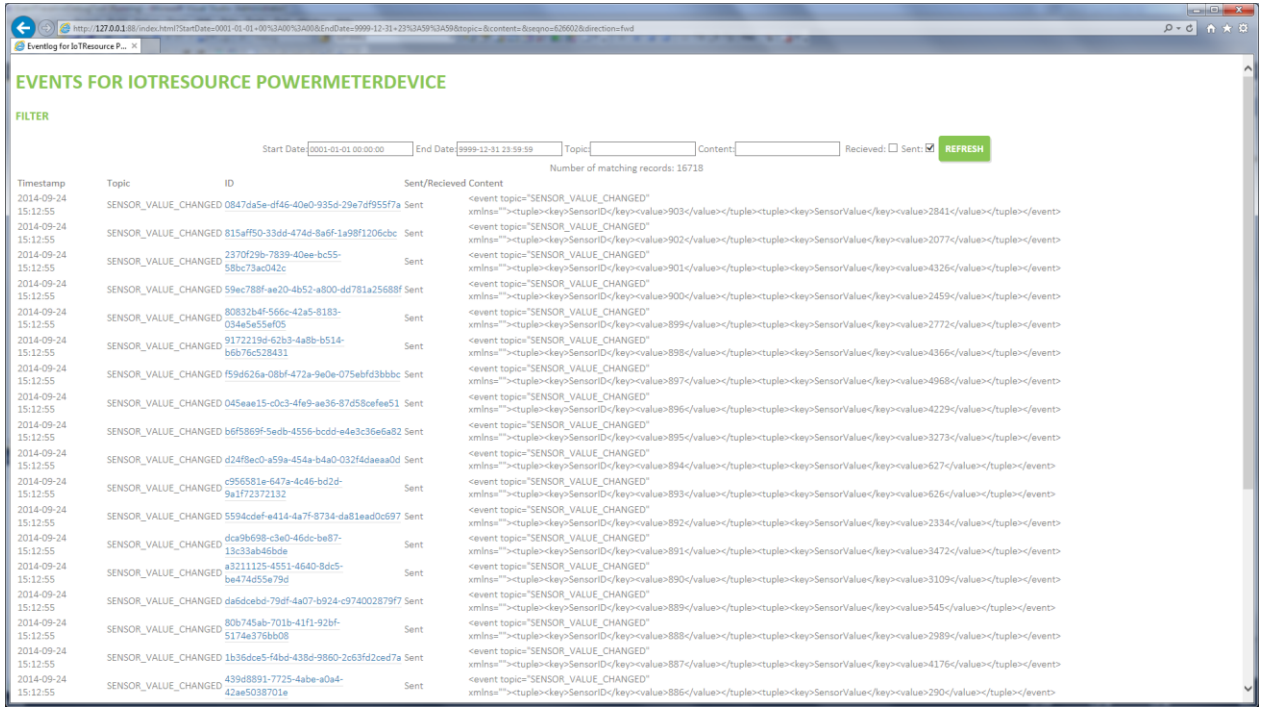


Figure 10: Event Browser for an individual IoT Resource

The information displayed is retrieved directly from the IoT Resource itself using the Event Query Interface.

4. Future Work

The initial implementation and design of the Iot Event debugging tool has provided the infrastructure for eaves dropping the event manager and querying the different event consumers and producers for their event information. This provides the foundation for the new functionality to be included in the Event Trace And Debug tool.

The main focus for the new functionality will focus on the following areas:

- Improve the users query interface including the usage of the event ontology for helping formulating queries.
- Make it possible to visualize the flow of an event as it propagates from producers to the different consumers on historical events.
- Trace functionality where the user can select some parameters to specify which events they want to track in "real time".
- "Real Time" visualization of events that are selected for tracking.
- Possibility to freeze the status at a certain time, i.e. that the tool will persist all IoT Resource event histories for a timespan so that the information can be further analyzed.

5. Class Documentation

5.1 EventQuery::EventQuery Class Reference

Summary description forEventQuery.

Public Member Functions

- **EventQuery** (string IoTID, string name, string vendor, string deviceURN)

Initializes a new instance of the **EventQuery** class.

- override void **Start** ()
- override void **Stop** ()
- override System.String **CreateWS** ()

Creates the Webservice.

- System.String **EventQueryService_SearchForEvent** (System.DateTime StartTime, System.DateTime EndTime, System.Boolean SentEvents, System.Boolean RecievedEvents, System.String EventContentQuery)

Search for event.

- System.String **EventQueryService_SearchForEventWithID** (System.String EventID, System.Boolean RecievedEvents, System.Boolean SentEvents)

Search for event with identifier.

Detailed Description

Summary description forEventQuery.

Constructor & Destructor Documentation

EventQuery::EventQuery::EventQuery (string IoTID, string name, string vendor, string deviceURN)
[inline]

Initializes a new instance of the **EventQuery** class.

Parameters:

IoTID The IoT id.

name The device name.

vendor The vendor name.

deviceURN The device URN.

Member Function Documentation

override void EventQuery::EventQuery::Start () [inline]

override void EventQuery::EventQuery::Stop () [inline]

override System.String EventQuery::EventQuery::CreateWS () [inline]

Creates the Webservice.

Returns:

System.String EventQuery::EventQuery::EventQueryService_SearchForEvent (System.DateTime *StartTime*, System.DateTime *EndTime*, System.Boolean *SentEvents*, System.Boolean *RecievedEvents*, System.String *EventContentQuery*) [inline]

Search for event.

Parameters:

StartTime The start time.

EndTime The end time.

SentEvents if set to true [sent events].

RecievedEvents if set to true [recieved events].

EventContentQuery The event content query.

Returns:

System.String EventQuery::EventQuery::EventQueryService_SearchForEventWithID (System.String *EventID*, System.Boolean *RecievedEvents*, System.Boolean *SentEvents*) [inline]

Search for event with identifier.

Parameters:

EventID The event identifier.

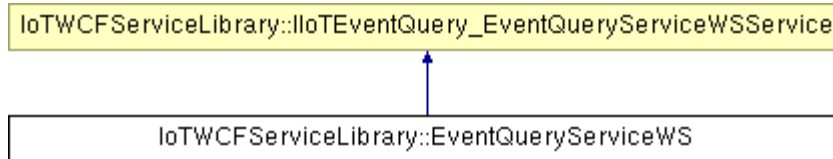
RecievedEvents if set to true [recieved events].

SentEvents if set to true [sent events].

Returns:

5.2 IoTWCFServiceLibrary::EventQueryServiceWS Class Reference

Inheritance diagram for IoTWCFServiceLibrary::EventQueryServiceWS:



Public Member Functions

- `EventQueryServiceWS (EventQuery.EventQuery theDevice)`

Initializes a new instance of the **EventQueryServiceWS** class.

- `System.String SearchForEvent (System.DateTime StartTime, System.DateTime EndTime, System.Boolean SentEvents, System.Boolean RecievedEvents, System.String EventContentQuery)`

Searches for event.

- `System.String SearchForEventWithID (System.String EventID, System.Boolean RecievedEvents, System.Boolean SentEvents)`

Searches for event with identifier.

Private Attributes

- `EventQuery.EventQuery m_eventquery`

Constructor & Destructor Documentation

IoTWCFServiceLibrary::EventQueryServiceWS::EventQueryServiceWS (EventQuery.EventQuery *theDevice*) [`inline`]

Initializes a new instance of the **EventQueryServiceWS** class.

Parameters:

theDevice The device.

Member Function Documentation

System.String IoTWCFServiceLibrary::EventQueryServiceWS::SearchForEvent (System.DateTime *StartTime*, System.DateTime *EndTime*, System.Boolean *SentEvents*, System.Boolean *RecievedEvents*, System.String *EventContentQuery*) [`inline`]

Searches for event.

Parameters:

StartTime The start time.

EndTime The end time.

SentEvents if set to `true` [sent events].

RecievedEvents if set to `true` [recieved events].

EventContentQuery The event content query.

Returns:

Implements `IoTWCFSserviceLibrary::IIoTEventQuery_EventQueryServiceWSService` (p.23).

System.String IoTWCFSserviceLibrary::EventQueryServiceWS::SearchForEventWithID (System.String *EventID*, System.Boolean *RecievedEvents*, System.Boolean *SentEvents*) [inline]

Searches for event with identifier.

Parameters:

EventID The event identifier.

RecievedEvents if set to `true` [recieved events].

SentEvents if set to `true` [sent events].

Returns:

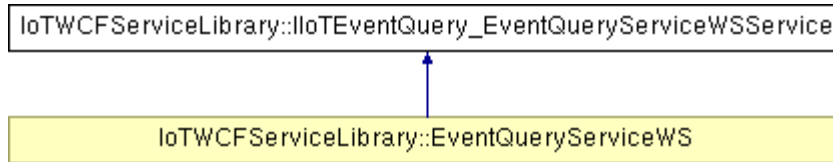
Implements `IoTWCFSserviceLibrary::IIoTEventQuery_EventQueryServiceWSService`

Member Data Documentation

EventQuery.EventQuery IoTWCFSserviceLibrary::EventQueryServiceWS::m_eventquery [private]

5.3 IoTWCFServiceLibrary::IIoTEventQuery_EventQueryServiceWSService Interface Reference

Inheritance diagram for IoTWCFServiceLibrary::IIoTEventQuery_EventQueryServiceWSService:



Public Member Functions

- System.String **SearchForEvent** (System.DateTime StartTime, System.DateTime EndTime, System.Boolean SentEvents, System.Boolean RecievedEvents, System.String EventContentQuery)
- System.String **SearchForEventWithID** (System.String EventID, System.Boolean RecievedEvents, System.Boolean SentEvents)

Member Function Documentation

System.String IoTWCFServiceLibrary::IIoTEventQuery_EventQueryServiceWSService::SearchForEvent (System.DateTime StartTime, System.DateTime EndTime, System.Boolean SentEvents, System.Boolean RecievedEvents, System.String EventContentQuery)

Implemented in IoTWCFServiceLibrary::EventQueryServiceWS (p.30).

System.String IoTWCFServiceLibrary::IIoTEventQuery_EventQueryServiceWSService::SearchForEventWithID (System.String EventID, System.Boolean RecievedEvents, System.Boolean SentEvents)

Implemented in IoTWCFServiceLibrary::EventQueryServiceWS

5.4 EventTraceAndDebugTool::DataBaseReader Class Reference

Manages all interactions with the SQLite database.

Public Member Functions

- void **Init** ()
- XmlDocument **ExecuteQuery** ()

Executes the query.

- XmlDocument **GetEvent** (string id)

Gets the event.

- void ClearDatabase ()

Clears the database.

- Int64 GetMinSeqNo ()

Gets the minimum seq no.

- void CreateDBCopY ()

Creates the database copy.

Properties

- DateTime **StartDateTime** [get, set]

start interval for retrieval or NULL if no start date time

- DateTime **EndDateTime** [get, set]

end interval for retrieval or NULL if no end date time

- Int64 **SeqNo** [get, set]

SeqNumber where to start to retrieve data.

- string **TopicQuery** [get, set]

Query string for topic.

- string **ContentQuery** [get, set]

QueryString for content.

Private Member Functions

- string BuildSQLForSearch ()

Builds the SQL for search.

- string BuildSQLForCount ()

Builds the SQL for counting instances.

Private Attributes

- DbConnection **eventDB** = null

Database Connection.

- DateTime **mStartDateTime** = DateTime.MinValue
- DateTime **mEndDateTime** = DateTime.MaxValue
- Int64 **mSeqNo** = Int64.MaxValue
- string **mTopicQuery** = ""

- `string mContentQuery = ""`

Detailed Description

Manages all interactions with the SQLite database.

Member Function Documentation

void EventTraceAndDebugTool::DataBaseReader::Init () [inline]

string EventTraceAndDebugTool::DataBaseReader::BuildSQLForSearch () [inline, private]

Builds the SQL for search.

Returns:

string EventTraceAndDebugTool::DataBaseReader::BuildSQLForCount () [inline, private]

Builds the SQL for counting instances.

Returns:

XmlDocument EventTraceAndDebugTool::DataBaseReader::ExecuteQuery () [inline]

Executes the query.

Returns:

XmlDocument EventTraceAndDebugTool::DataBaseReader::GetEvent (string *id*) [inline]

Gets the event.

Parameters:

id The identifier.

Returns:

void EventTraceAndDebugTool::DataBaseReader::ClearDatabase () [inline]

Clears the database.

Int64 EventTraceAndDebugTool::DataBaseReader::GetMinSeqNo () [inline]

Gets the minimum seq no.

Returns:

void EventTraceAndDebugTool::DataBaseReader::CreateDBCOPY () [inline]

Creates the database copy.

Member Data Documentation

DbConnection EventTraceAndDebugTool::DataBaseReader::eventDB = null [private]

Database Connection.

DateTime EventTraceAndDebugTool::DataBaseReader::mStartDateTime = DateTime.MinValue [private]

DateTime EventTraceAndDebugTool::DataBaseReader::mEndDateTime = DateTime.MaxValue [private]

Int64 EventTraceAndDebugTool::DataBaseReader::mSeqNo = Int64.MaxValue [private]

string EventTraceAndDebugTool::DataBaseReader::mTopicQuery = "" [private]

string EventTraceAndDebugTool::DataBaseReader::mContentQuery = "" [private]

Property Documentation

DateTime EventTraceAndDebugTool::DataBaseReader::StartDateTime [get, set]

start interval for retrieval or NULL if no start date time

DateTime EventTraceAndDebugTool::DataBaseReader::EndDateTime [get, set]

end interval for retrieval or NULL if no end date time

Int64 EventTraceAndDebugTool::DataBaseReader::SeqNo [get, set]

SeqNumber where to start to retrieve data.

string EventTraceAndDebugTool::DataBaseReader::TopicQuery [get, set]

Query string for topic.

string EventTraceAndDebugTool::DataBaseReader::ContentQuery [get, set]

QueryString for content.

5.5 EventDB::EventDB Class Reference

Manages the storing of events to the database. The events are read from the DBStore queue.

Public Member Functions

- EventDB (EventQueue.Queue DBStoreQueue)

Constructor for event router.

- void **Start** ()

Starts the service by spawning a new thread that immediately calls the Recieve function.

- void RecieveEvents ()

Recieve Events from the queue and route them.

- void **store** (Event.EventStructure receivedEvent)

Store an event in a database.

Public Attributes

- bool **stop** = false

Flag to stop the thread.

Private Attributes

- EventDBServices edbs = new EventDBServices()

The event Database service.

- EventQueue.Queue DBStoreQueue = null

The queue for the events to be stored.

- Thread **workThread** = null

The working thread.

Detailed Description

Manages the storing of events to the database. The events are read from the DBStore queue.

Constructor & Destructor Documentation

EventDB::EventDB::EventDB (EventQueue.Queue *DBStoreQueue*) [inline]

Constructor for event router.

Parameters:

DBStoreQueue Queue for the events to be stored

Member Function Documentation

void EventDB::EventDB::Start () [inline]

Starts the service by spawning a new thread that immediately calls the Recieve function.

void EventDB::EventDB::RecieveEvents () [inline]

Recieve Events from the queue and route them.

Will always run until serviceStopped = True and then sets safelyStopped=True

void EventDB::EventDB::store (Event.EventStructure *receivedEvent*) [inline]

Store an event in a database.

Parameters:

receivedEvent Received event

Member Data Documentation

bool EventDB::EventDB::stop = false

Flag to stop the thread.

EventDBServices EventDB::EventDB::edbs = new EventDBServices() [private]

The event Database service.

EventQueue.Queue EventDB::EventDB::DBStoreQueue = null [private]

The queue for the events to be stored.

Thread EventDB::EventDB::workThread = null [private]

The working thread.

void EventDB::EventDBServices::AddParameter (ref DbCommand *dbcommand*, string *parameterName*, DbType *dbType*, object *value*) [*inline*, *protected*]

Adds a parameter to supplied dbcommand.

Parameters:

dbcommand DbCommand object that the parameter will be added to

parameterName Name of the parameter to be added

dbType The database datatype of the parameter

value The actual value

Member Data Documentation

DbConnection EventDB::EventDBServices::eventDB = null [*private*]

Instance of connection to database.

5.6 EventReceiver::EventReceiver Class Reference

Inheritance diagram for EventReceiver::EventReceiver:



Public Member Functions

- void Init (EventQueue.Queue RuleQueue, EventQueue.Queue DBQueue, string EPAName, string ProjectName)

Method to instantiate the Event Receiver.

- void Init (EventQueue.Queue DBQueue, string EPAName, string ProjectName)

Method to instantiate the Event Receiver.

- void **Stop** ()
- void **Subscribe** (string my_Subscription, int priority)

Method for adding a subscription to the EventManager.

Public Attributes

- string **address** = ""

Callback address.

- string EventListenerHID = ""

Private Member Functions

- void SetUpEventManager ()

Setting up the connection to the EventManager.

- bool EventSubscriber. **notify** (string topic, **eventmanager.linksmart.eu.Part[]** parts)

Receiver of LinkSmart Part based events Converts it to enhanced events with more metadata fields and puts it in the rule engine queue and in Event DB queue (if event DB != null).

- void ParseEventData (string topic, eventmanager.linksmart.eu.Part[] parts)
- DateTime **ConvertDateTime** (string timestamp)
- bool EventSubscriber. **notifyEnhancedXmlDocument** (string receivedEvent)

Receives the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

- bool EventSubscriber. **notifyEnhanced** (Event.EventStructure **receivedEvent**)

Receives the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

- Event.EventStructure **InstantiateEvent** ()

Instantiates all elements of an enhanced Event.

Private Attributes

- Event.EventStructure **receivedEvent**

Received and enhanced event.

- EventQueue.Queue RuleQueue

Queue for processing by the rule engine.

- EventQueue.Queue DBQueue

Queue for storing by the Event DB.

- NetworkManager20.Registration **rid** = null
- NetworkManager.NetworkManagerApplicationService **m_networkmanager** = null
- NetworkManager20.NetworkManager **m_networkmanager20** = null
- EventManager.EventManagerImplementation **m_eventmanagerlistener** = null
- EventManager.EventManagerImplementation **m_eventmanager** = null

Instance of the EventManager.

Member Function Documentation

void EventReceiver::EventReceiver::Init (EventQueue.Queue RuleQueue, EventQueue.Queue DBQueue) [inline]

Method to instantiate the Event Receiver.

Parameters:

RuleQueue Queue for rule engine processing, must be set.

DBQueue Queue for event DB storage. Can be null if event storage is not used.

void EventReceiver::EventReceiver::Init (EventQueue.Queue DBQueue) [inline]

Method to instantiate the Event Receiver.

Parameters:

DBQueue Queue for event DB storage. Can be null if event storage is not used.

void EventReceiver::EventReceiver::Stop () [inline]

void EventReceiver::EventReceiver::SetUpEventManager () [inline, private]

Setting up the connection to the EventManager.

void EventReceiver::EventReceiver::Subscribe (string *my_Subscription*, int *priority*) [inline]

Method for adding a subscription to the EventManager.

Parameters:

my_Subscription Topic/Name of the event to subscribe to

priority Priority of the event

bool EventSubscriber. EventReceiver::EventReceiver::notify (string *topic*, eventmanager.linksmart.eu.Part[] *parts*) [inline, private]

Receiver of LinkSmart Part based events Converts it to enhanced events with more metadata fields and puts it in the rule engine queue and in Event DB queue (if event DB != null).

Parameters:

topic Topic for the even

parts Payload of the event

void EventReceiver::EventReceiver::ParseEventData (string *topic*, eventmanager.linksmart.eu.Part[] *parts*) [inline, private]

DateTime EventReceiver::EventReceiver::ConvertDateTime (string *timestamp*) [inline, private]

bool EventSubscriber. EventReceiver::EventReceiver::notifyEnhancedXmlDocument (string *receivedEvent*) [inline, private]

Recieves the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

Parameters:

outgoingEvent Received event

bool EventSubscriber. EventReceiver::EventReceiver::notifyEnhanced (Event.EventStructure *receivedEvent*) [inline, private]

Recieves the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

Parameters:

outgoingEvent Received event

Event.EventStructure EventReceiver::EventReceiver::InstantiateEvent () [inline, private]

Instantiates all elements of an enhanced Event.

Member Data Documentation

Event.EventStructure EventReceiver::EventReceiver::receivedEvent [private]

Received and enhanced event.

EventQueue.Queue EventReceiver::EventReceiver::RuleQueue [private]

Queue for processing by the rule engine.

EventQueue.Queue EventReceiver::EventReceiver::DBQueue [private]

Queue for storing by the Event DB.

string EventReceiver::EventReceiver::EPAName = "" [private]

The EPAs name.

string EventReceiver::EventReceiver::ProjectName = "" [private]

The project name.

string EventReceiver::EventReceiver::address = ""

Callback address.

string EventReceiver::EventReceiver::EventListenerHID = ""

NetworkManager20.Registration EventReceiver::EventReceiver::rid = null [private]

NetworkManager.NetworkManagerApplicationService

EventReceiver::EventReceiver::m_networkmanager = null [private]

NetworkManager20.NetworkManager EventReceiver::EventReceiver::m_networkmanager20 = null
[private]

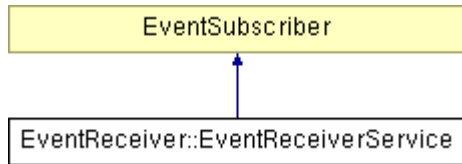
EventManager.EventManagerImplementation EventReceiver::EventReceiver::m_eventmanagerlistener = null [private]

EventManager.EventManagerImplementation EventReceiver::EventReceiver::m_eventmanager = null
[private]

Instance of the EventManager.

5.7 EventReceiver::EventReceiverService Class Reference

Inheritance diagram for EventReceiver::EventReceiverService:



Public Member Functions

- void **Init** ()

Method to instantiate the Event Receiver.

- void **Subscribe** (string my_Subscription, int priority)

Method for adding a subscription to the EventManager.

Public Attributes

- string **address** = ""

Instance of the Event Router class.

Private Member Functions

- void **SetUpEventManager** ()

Setting up the connection to the EventManager.

- bool EventSubscriber. **notify** (string topic, **eventmanager.linksmart.eu.Part[]** parts)

Receiver of LinkSmart Part based events.

- bool EventSubscriber. **notifyEnhanced** (Event.EventStructure **receivedEvent**)

Receiver of enhanced events.

- bool EventSubscriber. **notifyEnhancedXmlDocument** (string receivedEvent)

Receives the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

- Event.EventStructure **InstantiateEvent** ()

Instantiates all elements of an Event.

Private Attributes

- Event.EventStructure **receivedEvent**

Received and enhanced event.

- `EventQueue.MSMQ eventQueue = new EventQueue.MSMQ()`

Instance of the Event Database class.

- `EventManager.EventManagerImplementation m_eventmanagerlistener = null`
- `EventManager.EventManagerImplementation m_eventmanager = null`

Member Function Documentation

void EventReceiver::EventReceiverService::Init () [inline]

Method to instantiate the Event Receiver.

void EventReceiver::EventReceiverService::SetUpEventManager () [inline, private]

Setting up the connection to the EventManager.

void EventReceiver::EventReceiverService::Subscribe (string *my_Subscription*, int *priority*) [inline]

Method for adding a subscription to the EventManager.

bool EventSubscriber. EventReceiver::EventReceiverService::notify (string *topic*, eventmanager.linksmart.eu.Part[] *parts*) [inline, private]

Receiver of LinkSmart Part based events.

bool EventSubscriber. EventReceiver::EventReceiverService::notifyEnhanced (Event.EventStructure *receivedEvent*) [inline, private]

Receiver of enhanced events.

bool EventSubscriber. EventReceiver::EventReceiverService::notifyEnhancedXmlDocument (string *receivedEvent*) [inline, private]

Receives the event and puts it in the rule engine queue and in Event DB queue (if event DB != null).

Parameters:

outgoingEvent Received event

Event.EventStructure EventReceiver::EventReceiverService::InstantiateEvent () [inline, private]

Instantiates all elements of an Event.

Member Data Documentation

Event.EventStructure EventReceiver::EventReceiverService::receivedEvent [private]

Received and enhanced event.

EventQueue.MSMQ EventReceiver::EventReceiverService::eventQueue = **new EventQueue.MSMQ()**
[private]

Instance of the Event Database class.

Instance of the Event Queue class

string EventReceiver::EventReceiverService::address = ""

Instance of the Event Router class.

Callback address

EventManager.EventManagerImplementation

EventReceiver::EventReceiverService::m_eventmanagerlistener = **null** [private]

EventManager.EventManagerImplementation EventReceiver::EventReceiverService::m_eventmanager = **null** [private]

5.8 EventSubscriberClient Class Reference

Inheritance diagram for EventSubscriberClient:



Public Member Functions

- `EventSubscriberClient ()`
- `EventSubscriberClient (string endpointConfigurationName)`
- `EventSubscriberClient (string endpointConfigurationName, string remoteAddress)`
- `EventSubscriberClient (string endpointConfigurationName, System.ServiceModel.EndpointAddress remoteAddress)`
- `EventSubscriberClient (System.ServiceModel.Channels.Binding binding, System.ServiceModel.EndpointAddress remoteAddress)`
- `bool notify (string topic, eventmanager.linksmart.eu.Part[] parts)`

Callback for listening to Events.

- `bool notifyEnhanced (Event.EventStructure receivedEvent)`
- `bool notifyEnhancedXmlDocument (string eventXml)`

Constructor & Destructor Documentation

`EventSubscriberClient::EventSubscriberClient ()` [inline]

`EventSubscriberClient::EventSubscriberClient (string endpointConfigurationName)` [inline]

`EventSubscriberClient::EventSubscriberClient (string endpointConfigurationName, string remoteAddress)` [inline]

`EventSubscriberClient::EventSubscriberClient (string endpointConfigurationName, System.ServiceModel.EndpointAddress remoteAddress)` [inline]

`EventSubscriberClient::EventSubscriberClient (System.ServiceModel.Channels.Binding binding, System.ServiceModel.EndpointAddress remoteAddress)` [inline]

Member Function Documentation

bool EventSubscriberClient::notify (string *topic*, eventmanager.linksmart.eu.Part[] *parts*) [inline]

Callback for listening to Events.

Parameters:

topic The event Topic

parts List of key value pairs

Returns:

true if all is OK, false otherwise

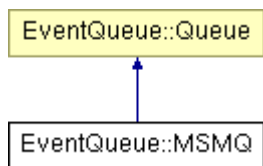
bool EventSubscriberClient::notifyEnhanced (Event.EventStructure *receivedEvent*) [inline]

bool EventSubscriberClient::notifyEnhancedXmlDocument (string *eventXml*) [inline]

5.9 EventQueue::MSMQ Class Reference

Implements a queue using **MSMQ**.

Inheritance diagram for EventQueue::MSMQ:



Public Member Functions

- bool **Init** (string Name)

Creates the **Queue** in **MSMQ** if it does not already exist.

- bool **Queue** (string item)

Adds Item to **Queue**.

- string **DeQueue** ()

Reads Item from **Queue**.

- bool **QueueEvent** (Event.EventStructure eventItem)

Adds Event to queue.

- Event.EventStructure **DeQueueEvent** ()

Reads event from **Queue**.

Properties

- string **name** [get]

return the queue name

Private Member Functions

- ~MSMQ ()

Destructor that closes the queue if it is open.

Private Attributes

- string **m_name**

Instance of the Event Database class.

- string **m_MSMQname**

The **MSMQ** queue name.

- MessageQueue **jobQueue** = null

The MSMQ queue.

- `object _lock = new object()`

TheLock for sending.

Detailed Description

Implements a queue using MSMQ.

MSMQ needs to be installed.

Constructor & Destructor Documentation

EventQueue::MSMQ::~~MSMQ () [*inline, private*]

Destructor that closes the queue if it is open.

Member Function Documentation

bool EventQueue::MSMQ::Init (string Name) [*inline*]

Creates the Queue in MSMQ if it does not already exist.

Returns:

True if all went well, False otherwise.

bool EventQueue::MSMQ::Queue (string item) [*inline*]

Adds Item to Queue.

Parameters:

item Item to be queued

Returns:

True if the Item was Queued. False otherwise.

string EventQueue::MSMQ::DeQueue () [*inline*]

Reads Item from Queue.

Returns:

Item or NULL when failed

bool EventQueue::MSMQ::QueueEvent (Event.EventStructure *eventItem*) [inline]

Adds Event to queue.

Returns:

True if event managed to be sent, False otherwise

Event.EventStructure EventQueue::MSMQ::DeQueueEvent () [inline]

Reads event from **Queue**.

Returns:

The event if successfull, null otherwise.

Member Data Documentation

string EventQueue::MSMQ::m_name [private]

Instance of the Event Database class.

The queue name

string EventQueue::MSMQ::m_MSMQname [private]

The MSMQ queue name.

MessageQueue EventQueue::MSMQ::jobQueue = null [private]

The MSMQ queue.

object EventQueue::MSMQ::_lock = new object() [private]

TheLock for sending.

Property Documentation

string EventQueue::MSMQ::name [get]

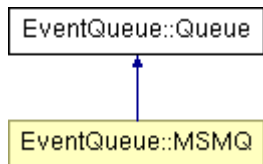
return the queue name

Read Only

5.10 EventQueue::Queue Interface Reference

Creates a **Queue** Interface to be used for communicating within an Event Processing Agent (EPA) NOTE! Implementation must be thread safe.

Inheritance diagram for EventQueue::Queue:



Public Member Functions

- bool **Init** (string Name)

Initializes the **Queue**.

- bool **Queue** (string item)

Adds Item to **Queue**.

- string **DeQueue** ()

Reads Item from **Queue**.

- bool **QueueEvent** (Event.EventStructure eventItem)

Adds Event to queue.

- Event.EventStructure **DeQueueEvent** ()

Reads event from **Queue**.

Properties

- string **name** [get]

return the queue name

Detailed Description

Creates a **Queue** Interface to be used for communicating within an Event Processing Agent (EPA) NOTE! Implementation must be thread safe.

Constructor & Destructor Documentation

bool EventQueue::Queue::Queue (string item)

Adds Item to **Queue**.

Parameters:

item Item to be queued

Returns:

True if the Item was Queued. False otherwise.

Implemented in `EventQueue::MSMQ` (p.40).

Member Function Documentation**`bool EventQueue::Queue::Init (string Name)`**

Initializes the `Queue`.

Parameters:

Name Name of `Queue`

Returns:

True if all went OK, otherwise false

Implemented in `EventQueue::MSMQ` (p.40).

`string EventQueue::Queue::DeQueue ()`

Reads Item from `Queue`.

Returns:

Item or NULL when failed

Implemented in `EventQueue::MSMQ` (p.40).

`bool EventQueue::Queue::QueueEvent (Event.EventStructure eventItem)`

Adds Event to queue.

Returns:

True if event managed to be sent, False otherwise

Implemented in `EventQueue::MSMQ` (p.41).

`Event.EventStructure EventQueue::Queue::DeQueueEvent ()`

Reads event from `Queue`.

Returns:

The event if successfull, null otherwise.

Implemented in `EventQueue::MSMQ` (p.41).

Property Documentation

string EventQueue::Queue::name [get]

return the queue name

Read Only

Implemented in `EventQueue::MSMQ` (p.41).

5.11 EventTraceAndDebugTool::RESTServer Class Reference

Public Member Functions

- void **RunHttpServer** ()

Runs the web server forever.

- void **Stop** ()

Stops the web server.

- void **WebServer_IncomingRequest** (object sender, HttpRequestEventArgs e)

Callback Event from WebServer when a HTTP request is made.

- string **ProcessRequest** (XmlDocument xDoc, HttpListenerRequest request, ref string contentType)

Processes a JSON webservice request.

- string **formatErrorMessage** (string mess)

Formats an error message in JSON.

- string **parseMethod** (HttpListenerRequest request)

Extracts the method from the request.

- string **FormatFile** (string filename, bool isList)

Formats the JSON result correctly.

- bool **FileHttpCall** (string url)

Determines if it just a filebased call or a REST call.

- string **FileMimeType** (string url)

Determines the MIME type.

Static Public Member Functions

- static void **WriteLogFile** (string logMessage)

Static method for adding an entry to a log file including a time stamp.

Static Private Member Functions

- static int **RandomNumber** (int min, int max)

Creates a random number.

Private Attributes

- string **m_endPoint**

Stores the HTTP callback server address.

- WebServer **m_webServer** = null

Stores the mote address.

Member Function Documentation

void EventTraceAndDebugTool::RESTServer::RunHttpServer () [inline]

Runs the web server forever.

void EventTraceAndDebugTool::RESTServer::Stop () [inline]

Stops the web server.

void EventTraceAndDebugTool::RESTServer::WebServer_IncomingRequest (object sender, HttpRequestEventArgs e) [inline]

Callback Event from WebServer when a HTTP request is made.

Parameters:

sender Sender

e Arguments

string EventTraceAndDebugTool::RESTServer::ProcessRequest (XmlDocument xDoc, HttpListenerRequest request, ref string contentType) [inline]

Processes a JSON webservice request.

Parameters:

xDoc The POST part of JSON converted to XML

request The complete request object

Returns:

The result of the request as a string

string EventTraceAndDebugTool::RESTServer::formatErrorMessage (string mess) [inline]

Formats an error message in JSON.

Parameters:

mess The error message

Returns:

The error message in JSON format.

string EventTraceAndDebugTool::RESTServer::parseMethod (HttpListenerRequest request) [inline]

Extracts the method from the request.

Parameters:

request The request object

Returns:

The method as string

string EventTraceAndDebugTool::RETSerVer::FormatFile (string *filename*, bool *isList*) [inline]

Formats the JSON result correctly.

NOT USED any longer

Parameters:

filename The name of the XML file to be loaded and converted to JSON

isList True if it is a JSON list

Returns:

The formatted result

static void EventTraceAndDebugTool::RETSerVer::WriteLogFile (string *logMessage*) [inline, static]

Static method for adding an entry to a log file including a time stamp.

Parameters:

logMessage The message to be added to the logfile

bool EventTraceAndDebugTool::RETSerVer::FileHttpCall (string *url*) [inline]

Determines if it just a filebased call or a REST call.

Parameters:

url The call url

Returns:

True if its a file call, false otherwise

string EventTraceAndDebugTool::RETSerVer::FileMimeType (string *url*) [inline]

Determines the MIME type.

Parameters:

url The call url

Returns:

The MIME

static int EventTraceAndDebugTool::RETSerVer::RandomNumber (int *min*, int *max*) [inline, static, private]

Creates a random number.

Parameters:

min The minimum.

max The maximum.

Returns:

A random number

Member Data Documentation

string EventTraceAndDebugTool::RESTServer::m_endPoint [private]

Stores the HTTP callback server address.

WebServer EventTraceAndDebugTool::RESTServer::m_webServer = null [private]

Stores the mote address.

6. References

(LINKSMART, 2014) <http://www.hydramiddleware.eu/news.php>, visited 2014-07-15.

(LINKSMART2,2014) <http://sourceforge.net/projects/linksmart/>, visited 2014-07-15.

(LINKSMART3,2014) [D12.9 Final External Developers Workshops Teaching Materials.pdf](#),visited 2014-07-15.