

A Novel Approach to QoS Monitoring in the Cloud

2nd Training on Software Services- Cloud computing -
November 11-14

Luigi Sgaglione – EPSILON srl
luigi.sgaglione@epsilononline.com


Intelligence Push in the Enterprise Realm





- Rationale and Approach
- Background and Context
- Research Challenges
- Interfaces
- Architecture and main Components
- Case Study
- Demo
- Contact Info

Rationale and Approach



Intelligence Push in the Enterprise Realm

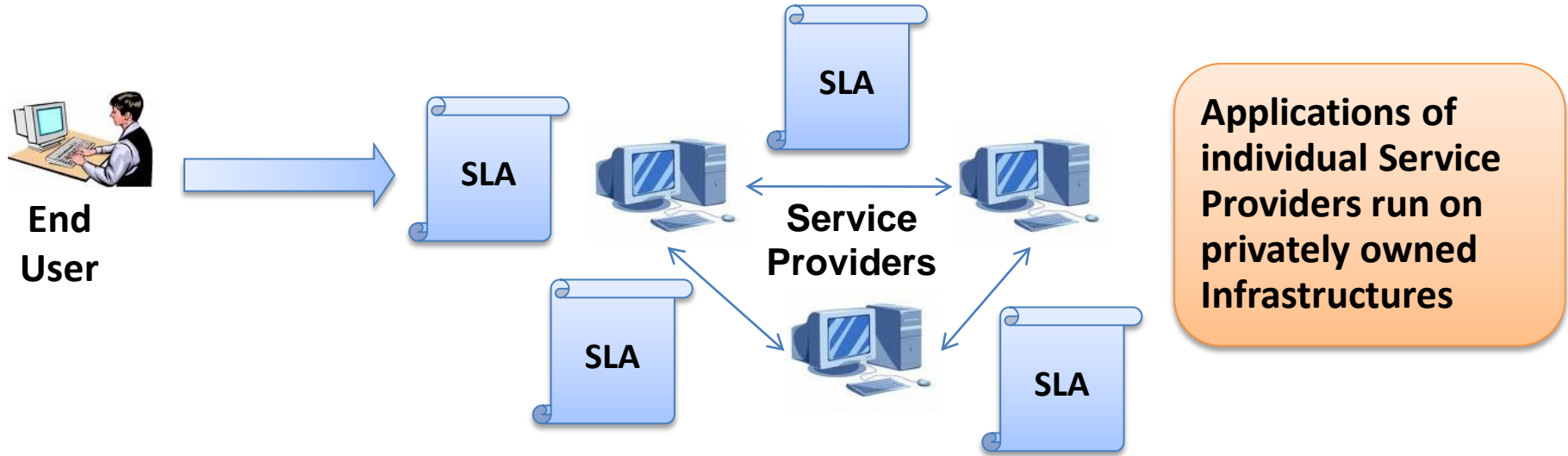


The need for Quality of Service (QoS) monitoring



- Quality of Service (QoS) monitoring is key for a company's success, since assessing the actual quality of what service users are paying for has become a mission-critical business practice requirement, and it will be even more so in the future
- QoS-MONaaS (QoS MONitoring as a Service) implements a dependable QoS monitoring facility, which is made available to all applications running on top of a cloud platform (specifically SRT-15) according to the “as a Service” paradigm
- The availability of a dependable (i.e. reliable and timely) QoS monitoring facility allows each party of a business process to understand if any failure and/or performance issue which they experience is caused by the cloud service provider, the network infrastructure, a specific service provider, or the design of the application itself

QoS monitoring today

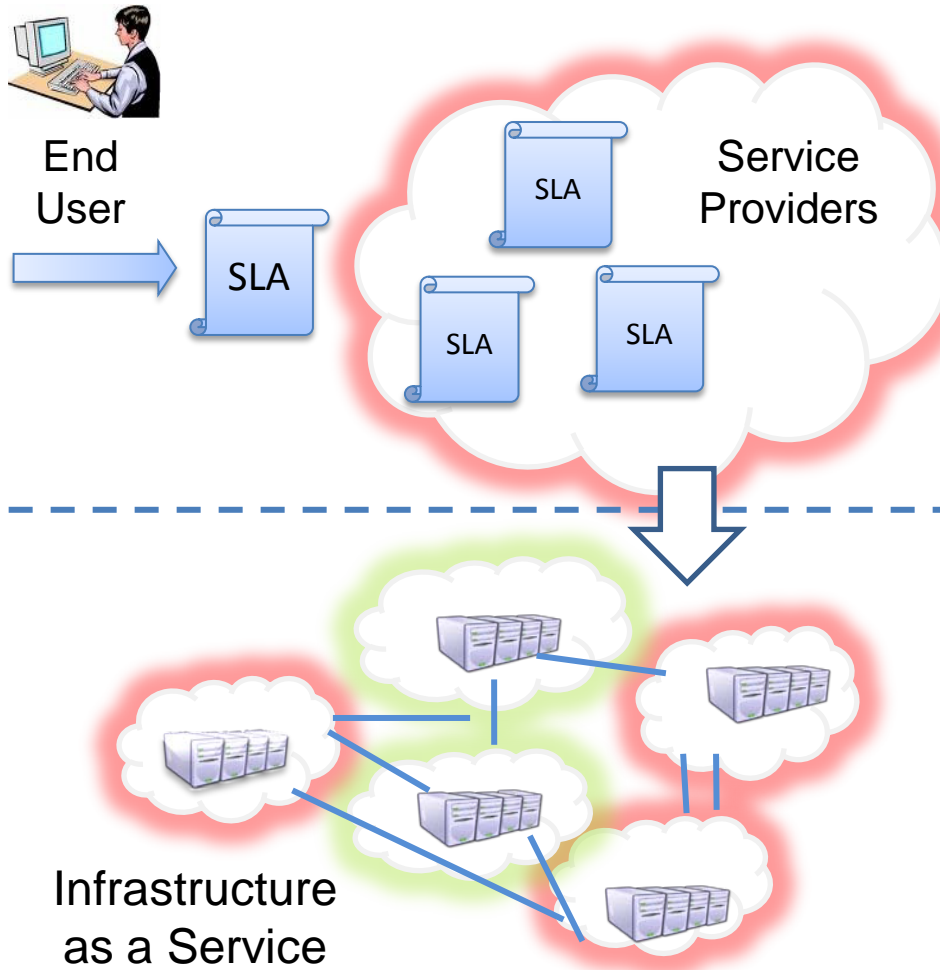


QoS Monitor is an application itself

It also runs on a Privately owned Service Infrastructure

It is a Trusted Third Party (TTP)!

QoS monitoring tomorrow



Issue(s):

Business Applications run on **Service Infrastructures** which are **Shared** and **Virtualized**

Goals:

QoS Monitoring must be provided as a **Service**

It must run on the same **shared infrastructure** (as the applications being monitored)

TTP assumption must be removed

Monitor is a peer
(as opposed to a TTP)

Support for
dynamic
composition of
services

- We claim that the TTP assumption is the main cause of the market failure of applications such as QoS monitoring, PKIs, TSAs, and the like
- Will assume that the monitor is a peer
- Will assume that it runs on top of an untrusted cloud: Trusted “by design”, not “by assumption”
- Must be able to track the evolution of services
- Must be able to provide continuous QoS monitoring (even while changes are taking place)

Background and Context

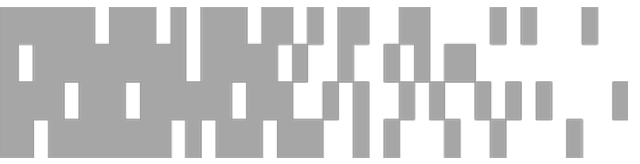
SR15

Intelligence Push in the Enterprise Realm





- Over three quarters of North American and European companies outsource part of their activities → business-related information is spread across different systems and also spread geographically across multiple locations
- The objective of the SRT-15 research project is to bridge the gap between cloud infrastructures and enterprise services by building a distributed service platform.
- The goal of the SRT-15 platform is to enable the integration of dynamic enterprise services on the Internet
- SRT-15 allows for dependable and scalable cloud-based processing of data coming to and from heterogeneous enterprise services spread across multiple distributed locations



Duration

- ▶ 30 months (from October 2010 until March 2013)

Budget

- ▶ Total Cost: 4,40 M€
- ▶ EC funding: 2,82 M€

SRT-15 Consortium as a whole - who is who

Zbigniew Jerzak, Coordinator (#1 - SAP)

Christof Fetzer (# 2 - TUD)

Luigi Romano (# 3 - **EPSILON**)

Flavio Junqueira (# 4 - YAHOO!)

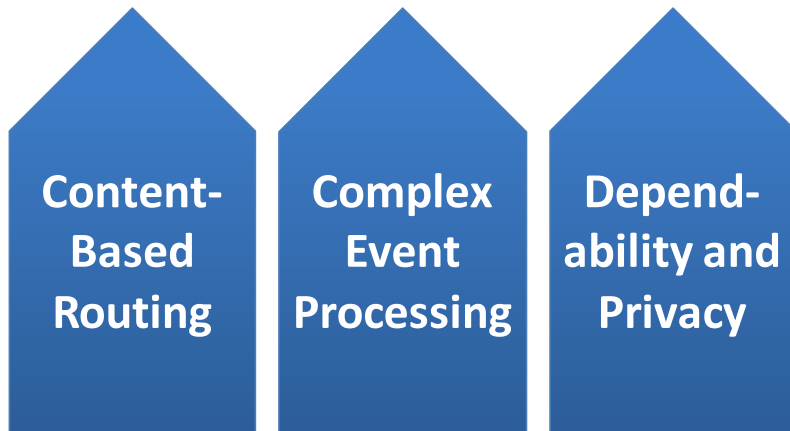
Pascal Felber (# 5 - UNINE)



What is SRT-15

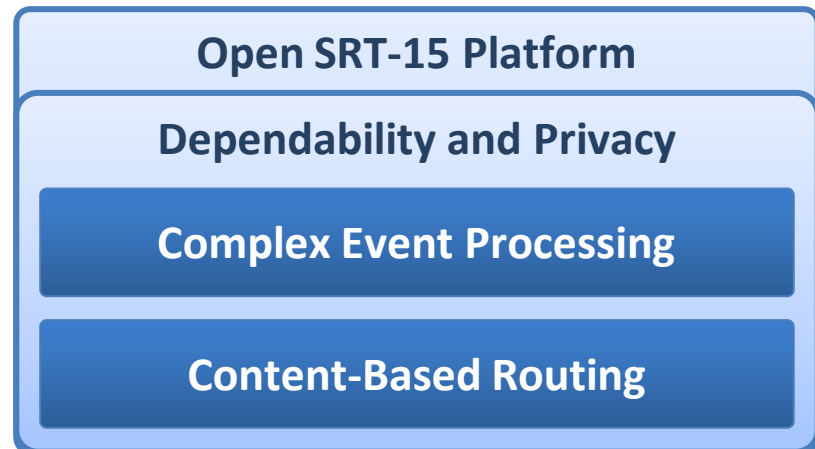
**A distributed service platform
which bridges the gap between cloud infrastructures and enterprise services,
and scales across public and private clouds
allowing for reliable and dynamic interaction among enterprise applications**

Scalable and Open PaaS



Public and Private Infrastructure as a Service

Proof of Concept
(QoS-MONaaS, Business Internet)

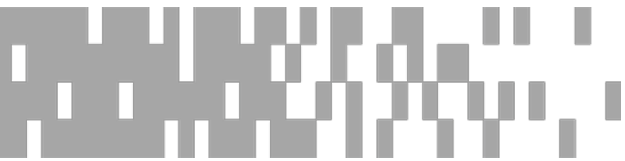


Public and Private Infrastructure as a Service

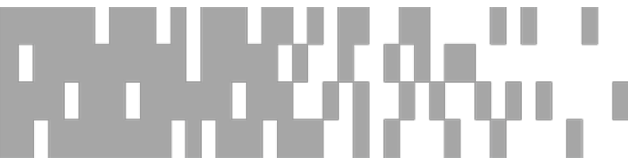


- The content of events is used to route information to the interested applications
- Obsoletes the explicit source and destination addresses on events
- Allows for a loosely coupled communication between the applications and services
- Implements an asynchronous communication model
- Producers and consumers are anonymous to each other and the system as a whole (no explicit source and destination addresses)

Large-Scale Real-Time Complex Event Processing

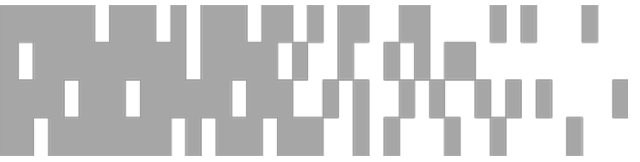


- Complex Event Processing (CEP) allows SRT-15 to optimize and accelerate the processing of data exchanged between applications, services and users
- SRT-15 uses parallel CEP in conjunction with event-based routing, to move computation to the data sources
- The use of parallel CEP allows SRT-15 to process data as it arrives, without the need for expensive and time consuming staging
- SRT-15 will support stateful CEP operators
- As a novel aspect for the domain of CEP, we will control the speed and throughput of operators by controlling the degree of parallelization of the operators
- In SRT-15, we will apply a new model for CEP that is not based on SQL



Aspects of dependability that we intend to address in SRT-15:

- ▶ Exploring alternative replication approaches (i.e. other than Byzantine fault-tolerance through state-machine replication) and reaching more practical solutions to protect systems against faults beyond crash
- ▶ Providing novel mechanisms that can detect data corruptions caused by hardware or software failures in the cloud



QoS-MONaaS

- ▶ Provides a dependable QoS monitoring facility to the applications which will be run on top of the SRT-15 platform

Business Internet Platform

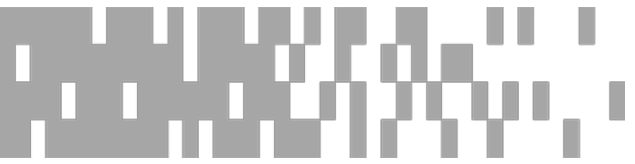
- ▶ For collecting, filtering, managing, aggregating, propagating, and publishing business events in a consistent, contextual way in an open and comprehensive framework, across multiple applications, services, and companies

Research Challenges



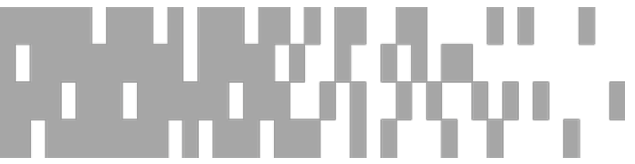
Intelligence Push in the Enterprise Realm





- Cloud computing makes QoS monitoring extremely challenging, for a number of reasons, and in particular:
 - ▶ Business applications currently run (mostly) on privately owned service infrastructures, whose technology is – to a large extent – known. In a cloud computing context, they run on infrastructures which are: (i) shared, and (ii) virtualized. This also applies to the QoS monitor itself.
 - ▶ It has to cope with dynamic composition of services, meaning that it must be able to track the evolution of services and it must be able to provide continuous QoS monitoring, i.e., even while changes are taking place.

What we need from the underlying PaaS



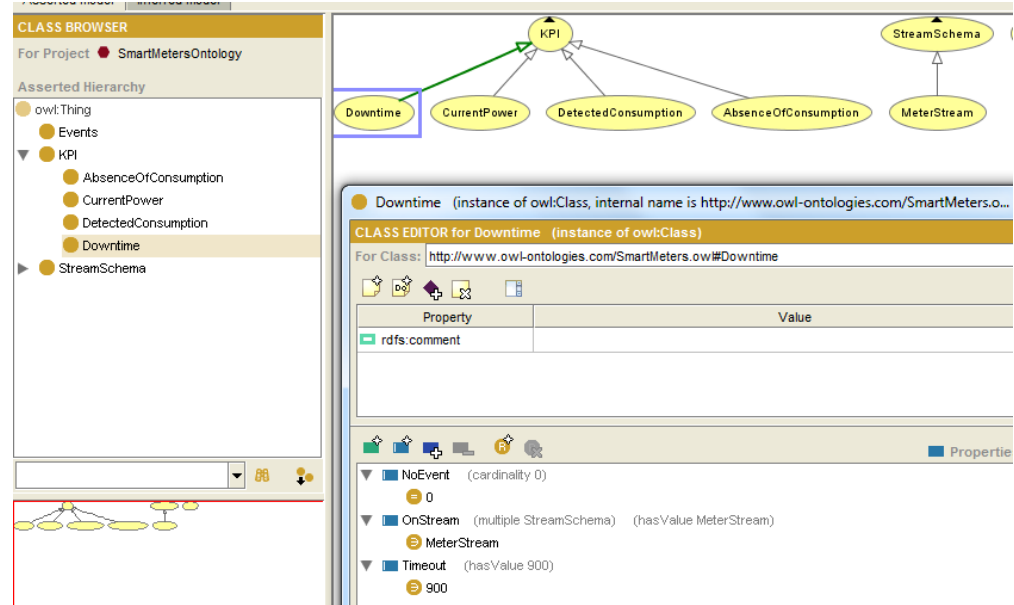
- ▶ That all messages - and related information - which are of interest for QoS monitoring be made available to the QoS-MONaaS application
- ▶ That messages be authentic (i.e the monitor can trust the identity of publisher and the contents of the message)
- ▶ That the real identity of parties being monitored is not revealed to the monitoring service
- ▶ Privacy/secretcy of message contents should also be guaranteed (in most cases)
- ▶ That effective processing capabilities be made available, which allow QoS-MONaaS to process QoS-related data streams in real time

What we need from the provider of the monitored service

- SLA descriptions in WS-agreement language, from which conditions on Key Performance Indicators (KPIs) of interest can be extracted
- Domain-specific information (specified via an ontology), including a formal description of the specific business process

```
AgreementId="Agreement1">  
  
<wsag:Name>Agreement</wsag:Name>  
  
<wsag:Context/>  
<wsag:Terms>  
<wsag:All>  
  
<wsag:ServiceDescriptionTerm wsag:Name="source" wsag:ServiceName="  
  
<wsag:ServiceProperties wsag:ServiceName="Not Defined">  
  
<wsag:GuaranteeTerm Name="DowntimeForADay" Obligated="Provider">  
<wsag:ServiceScope>  
<wsag:ServiceName>Metering</wsag:ServiceName>  
</wsag:ServiceScope>  
<wsag:ServiceLevelObjective>  
<wsag:KPITarget>  
<wsag:KPIName>Downtime</wsag:KPIName>  
<wsag:Target>Count IS_LESS 5 per 1 day</wsag:Target>  
</wsag:KPITarget>  
</wsag:ServiceLevelObjective>  
</wsag:GuaranteeTerm>
```

SLA



Ontology



Web Services Agreement Specification (WS-Agreement)

- Is a Web Services protocol for establishing agreement between two parties, such as between a service provider and consumer,
- Uses an extensible XML language for specifying the nature of the agreement (used by QoS-MONaaS)
- Provides agreement templates to facilitate discovery of compatible agreement parties.



- An ontology is an explicit and formal specification of a conceptualization.
(T.R. Gruber)
- Describes formally a domain
- Finite list of terms and relationships between these

Architecture

SR15

Intelligence Push in the Enterprise Realm



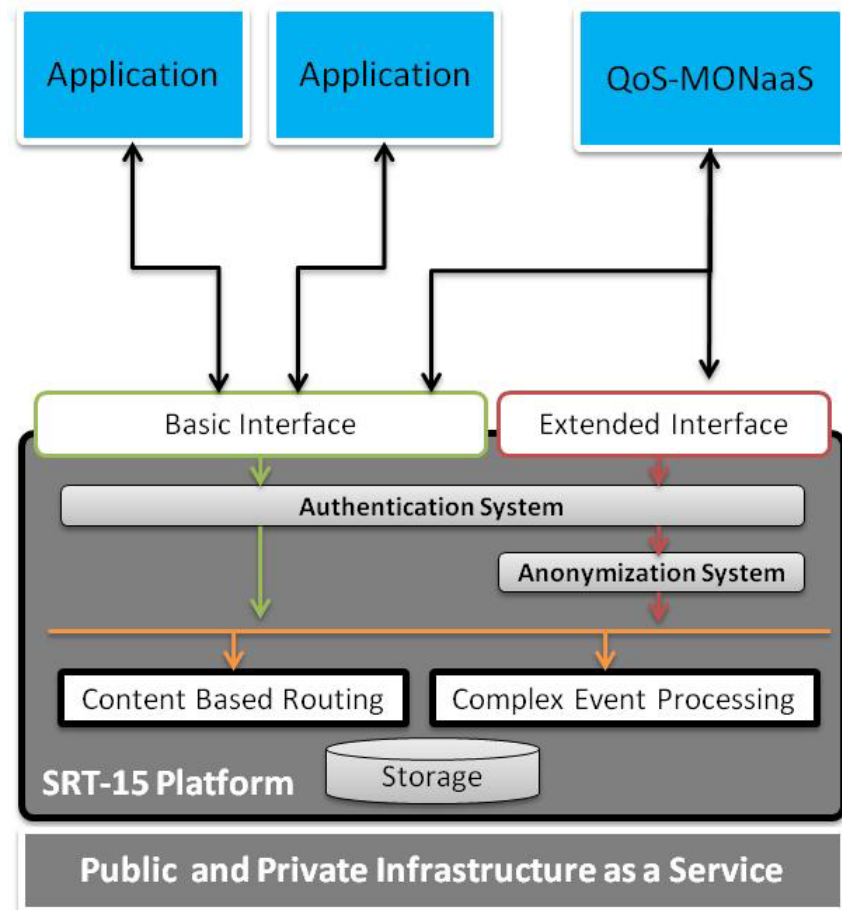
QoS-MONaaS to platform Interfaces

Basic Interface:

- ▶ Makes CEP facilities of the SRT-15 platform available to QoS-MONaaS, which allows QoS-MONaaS to process QoS-related data streams in real time

Extended Interface:

- ▶ All messages - and related information - which are of interest for QoS monitoring are made available to QoS-MONaaS. This is achieved by combining CBR and CEP facilities of the SRT-15 platform.



Anonymization System: guarantees "trusted by design" and "peer among peers" characteristics of QoS-MONaaS. The basic idea is that since the monitor ignores the real identities of parties, it cannot cheat

QoS-MONaaS is exposed as a Web Service (specifically, Axis2)

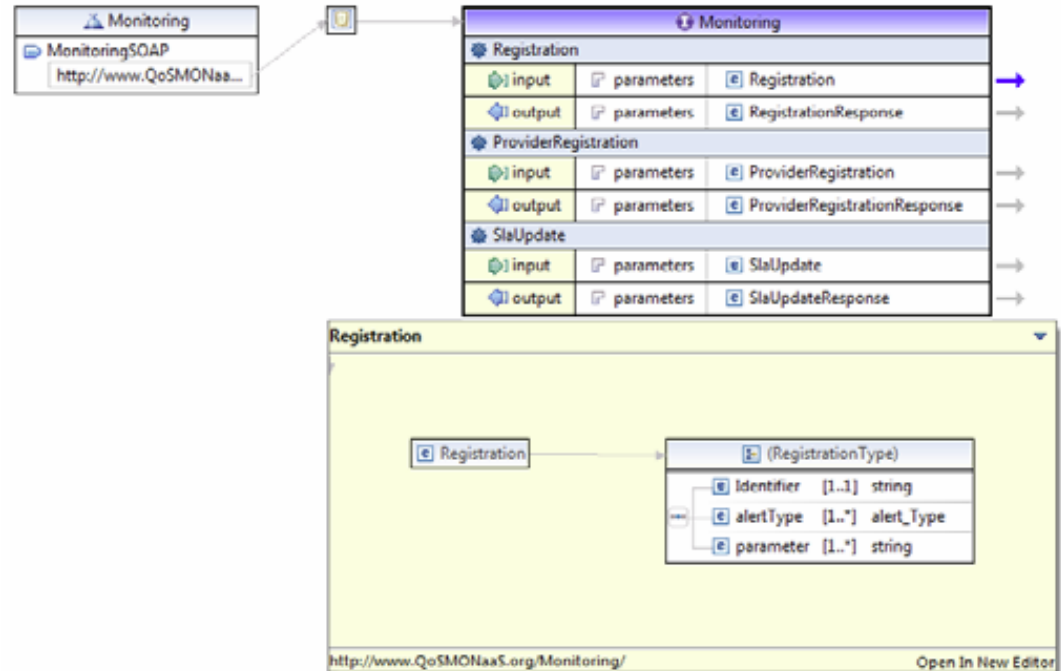
Two sets of operations are available:

▶ Client operations

- stopMonitoring
- violationList
- getProvider
- clientSubscription
- listProviders
- listCategories
-

▶ Provider operations

- domUpdate
- slaUpdate
- providerRegistration
- getReportProvider



QoS-MONaaS: Components

SLA Analyzer:

- ▶ Parses SLAs and extracts: i) KPIs of interest, and ii) conditions which define compliance/non compliance

KPI Meter:

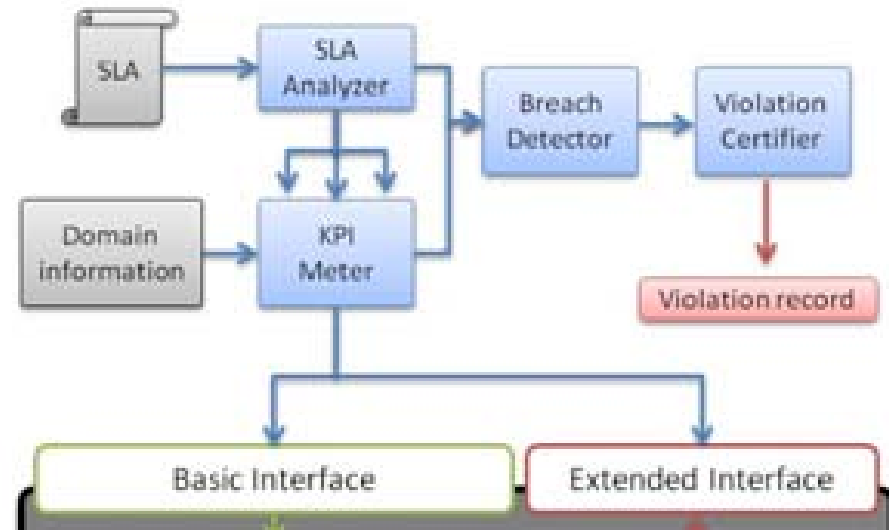
- ▶ Continuously monitors the actual value of KPIs of interest

Breach Detector:

- ▶ Uses the outputs of the KPI monitor and the conditions from the SLA Analyzer to spot contract violations
- ▶ Upon detection of a breach of contract, a violation record is produced

Violation Certifier:

- ▶ Enriches the output of the Breach Detector with a timestamp and a digital signature, so to produce unforgeable evidence of the violation



QoS-MONaaS handles multiple data formats. Data adapters are used, to limit the impact of specific data formats on the application logic

Case Study

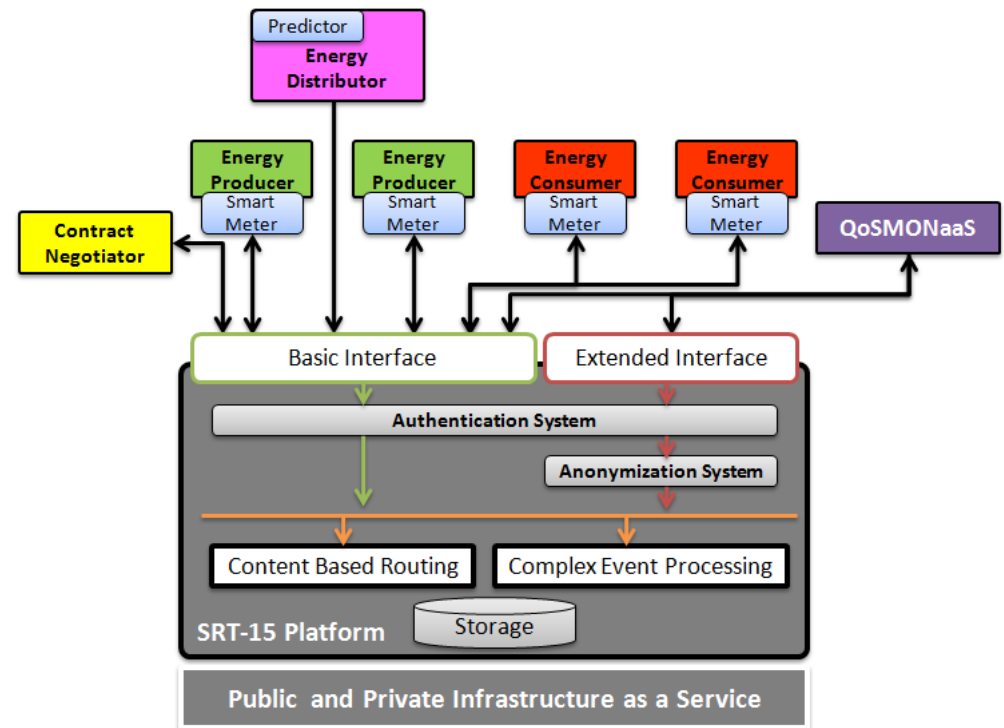
SR-15

Intelligence Push in the Enterprise Realm



QoS monitoring in a Smart Grid

- The implementation of QoS-MONaaS is being validated wrt an Internet of Things (IoT) application developed by SAP
- The application, which is called Smart Meters, implements remote monitoring of power consumption in a Smart Grid environment
- Smart Meters uses the SRT-15 platform for distribution and real-time processing of measurements related to energy consumption and production
- QoS-MONaaS monitors the QoS which is actually delivered, and spots violations



QoS-MONaaS demo:

Web Service

QoS-MONaaS Web Service provides operations which are needed to enable the monitoring of an SLA, and in particular:

▶ Client

- stopMonitoring
- violationList
- getProvider
- getSLA
- slaStatus
- clientSubscription
- startMonitor
- listProviders
- ...

<http://localhost:8080/QoS-MONaaS/services/listServices>

▶ Provider

- domUpdate
- providerRegistration
- getReportProvider
- slaUpdate

Pre-requisite:

- ▶ Service users and service providers must have already authenticated with the SRT-15 platform

QoS-MONaaS demo: client interface

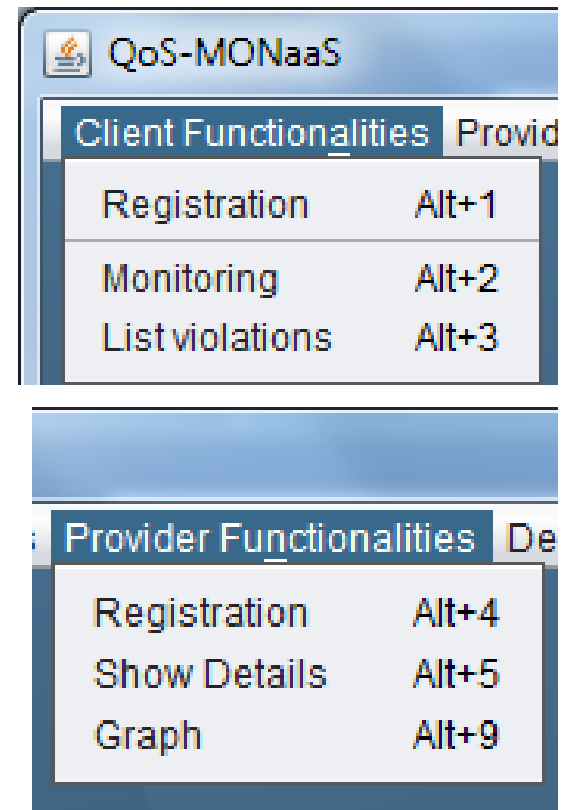
QoS-MONaaS client interface is a possible implementation of an interface for QoS-MONaaS web service.

Client functionalities:

- ▶ Registration
- ▶ Monitoring
- ▶ List violations

Provider functionalities:

- ▶ Registration
- ▶ Show details
- ▶ Graph



QoS-MONaaS demo: provider functionalities

Provider functionalities

▶ Registration

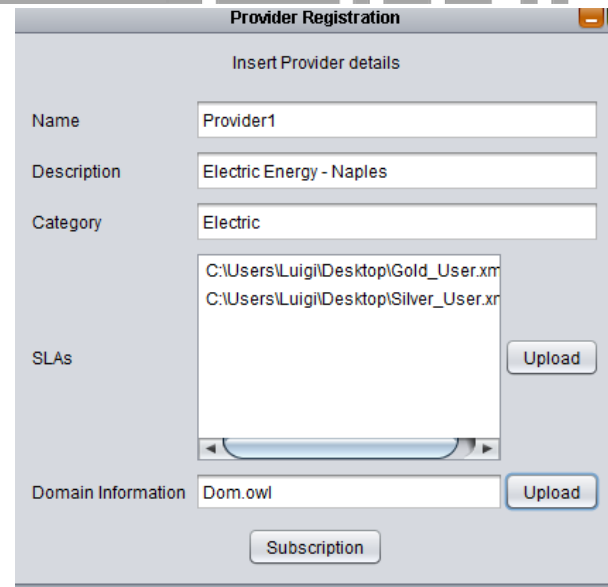
- Allows the provider to register to QoSMONaaS
 - monitoring services will be made available to provider's users
 - SRT-15 allows QoS-MONaaS to access platform data related to the provider (via SRT-15 CEP functions).

▶ Show details

- Allows retrieval of provider's information, and modification of SLAs and domain information files

▶ Graph

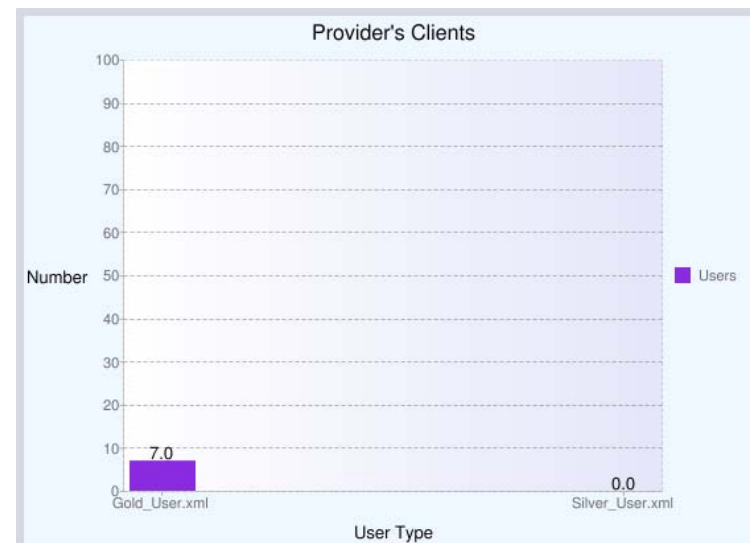
- Shows the number of users who have requested the monitoring service for a given SLA



The screenshot shows a web application window titled "Provider Registration". It contains a form with the following fields and values:

- Name: Provider1
- Description: Electric Energy - Naples
- Category: Electric
- SLAs: A list box containing two entries: "C:\Users\Luigi\Desktop\Gold_User.xml" and "C:\Users\Luigi\Desktop\Silver_User.xml". There is an "Upload" button to the right of this list.
- Domain Information: Dom.owl. There is an "Upload" button to the right of this field.

At the bottom of the form is a "Subscription" button.



QoS-MONaaS demo: client functionalities (1)

Client functionalities

▶ Registration

- Allows the client to register to the QoS-MONaaS monitoring service.

The image displays three sequential screenshots of a web application interface for QoS-MONaaS registration:

- Category List:** A window titled "Category List" with the instruction "Select category of your Provider". It contains a list with "Electric" and "Water".
- Provider List:** A window titled "Provider List" with the instruction "Select your Provider and SLA to monitor". It features a dropdown menu with "Provider1" selected. To the right, there are input fields for "Name" (Provider1), "Description" (Electric Energy - Naples), and "Category" (Electric). Below these, a list of "SLAs" includes "Gold_User.xml" (highlighted) and "Silver_User.xml". A "Subscription" button is at the bottom.
- Set Parameters:** A window titled "Set Parameters" with the instruction "Provider and SLA to monitor". It has input fields for "Provider" (Provider1) and "SLA" (Gold_User.xml). Below, under "Select alert type and set relative parameter", there is a dropdown menu with "SMS" selected and a text input field. At the bottom, there is a "Client Id" field containing "smartmeter" and a "Subscription" button.

QoS-MONaaS demo: client functionalities (2)

Client functionalities

► Monitoring

- Allows starting and stopping the monitoring services of an SLA

The screenshot shows a web-based interface titled "Monitor". At the top, there is a dropdown menu for "SLA to monitor" set to "Provider1-Gold_User.xml", a "Status" field showing "Not Monitored", and a "Download SLA" button. Below this, there are two calendar widgets for "Starting date" and "Ending date", both set to August 2011. The "Starting date" calendar has the 2nd of August highlighted in red. Below the calendars are two time selection fields for "Starting Time" and "Ending Time", both set to "11:04". At the bottom center, there is a large "Start Monitoring" button.

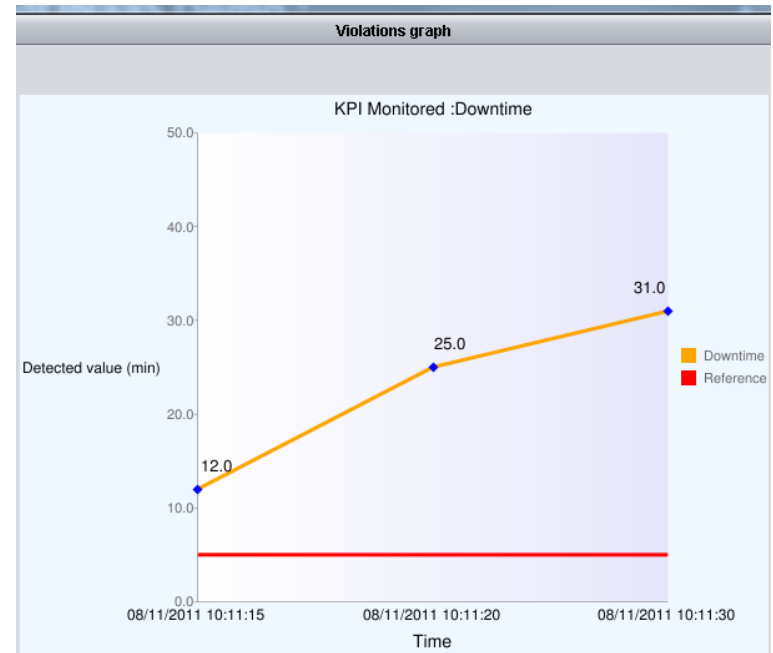
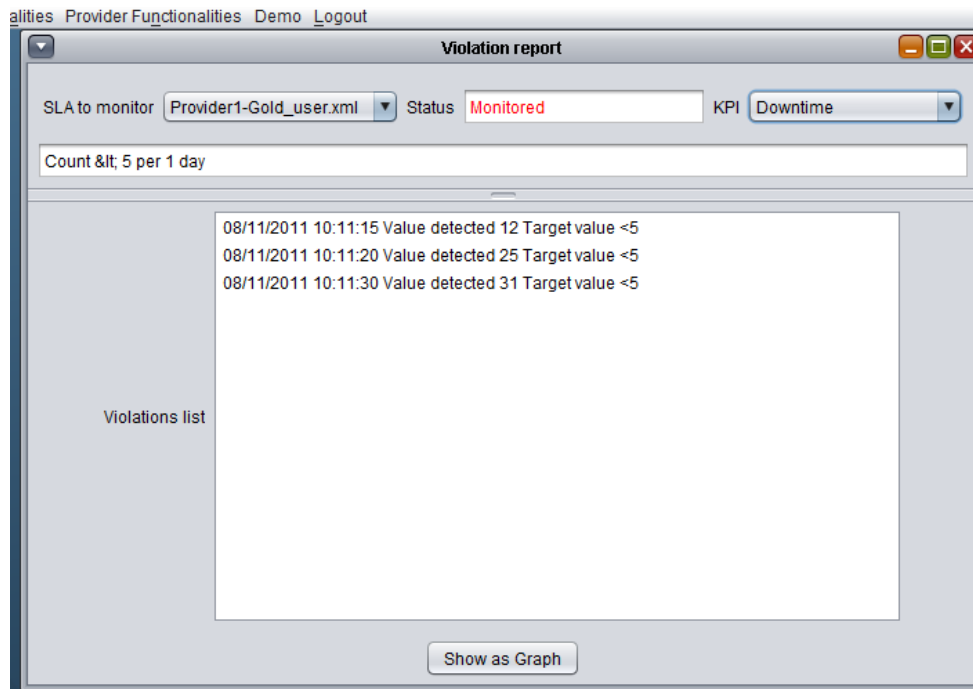
QoS-MONaaS demo: client functionalities (3)



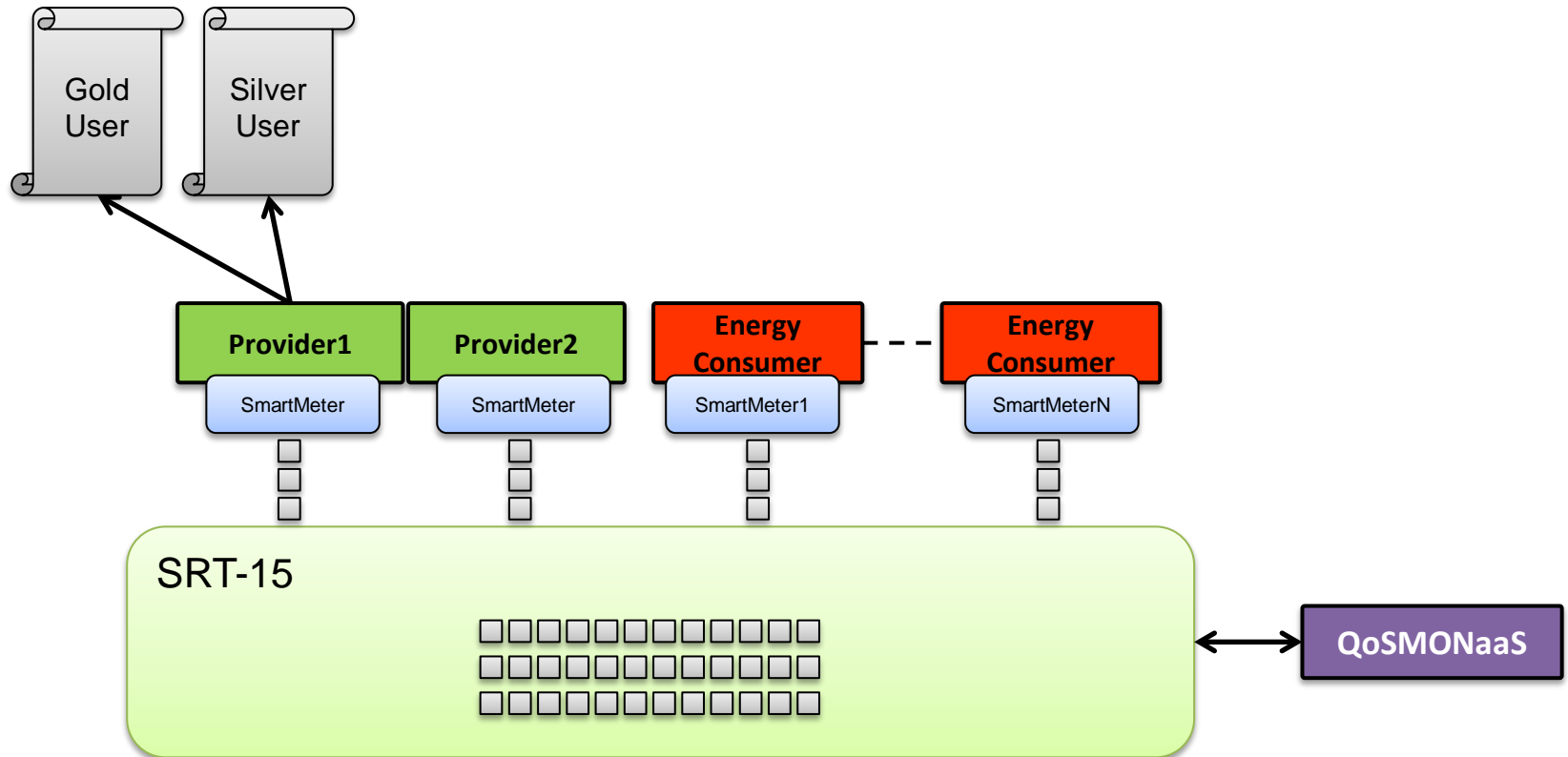
Client functionalities

► List violations

- Allows to show violations of a selected SLA in text and graphic mode

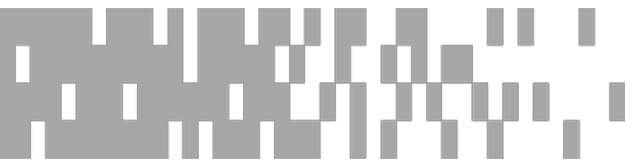


QoS Demo : Scenario



QoS-MONaaS Demo :

Step 1



Login as Demo

- ▶ This allows to use automatic registration functionalities of providers and clients

Login

User Id: Demo

User Type: Demo

Login

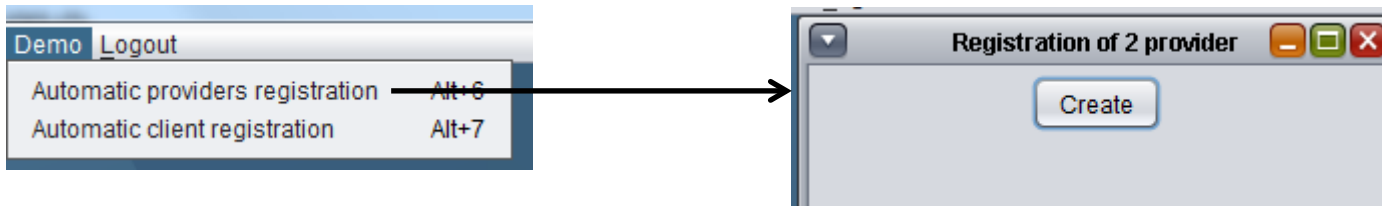
QoS-MONaaS

Client Functionalities Provider Functionalities Demo Logout

Automatic providers registration	Alt+6
Automatic client registration	Alt+7
List providers	Alt+A
List clients	Alt+B

QoS-MONaaS Demo :

Step 2-1



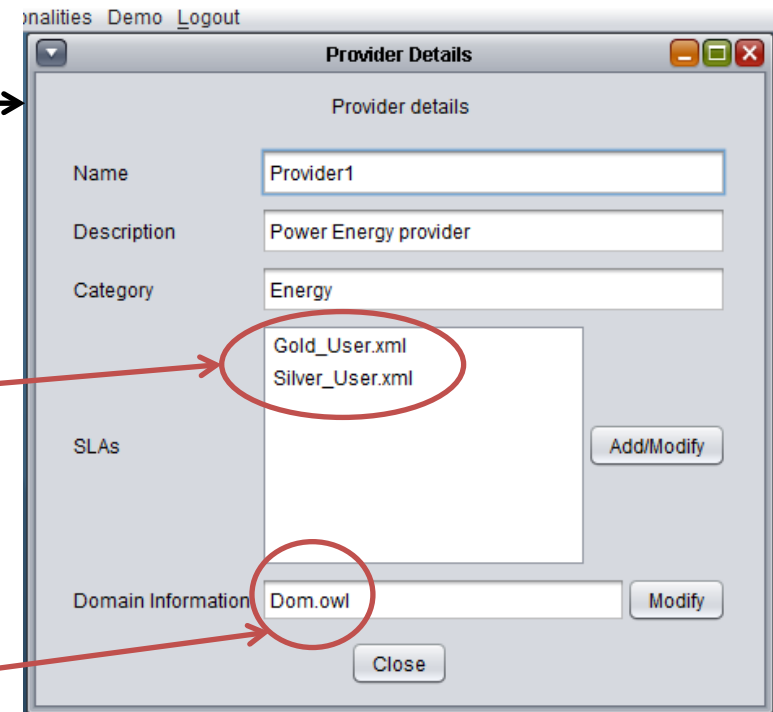
Automatic Provider Registration

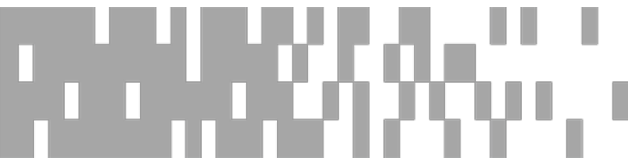
► It registers two providers

- Provider1
- Provider2

SLAs be expose
to the Clients

Domain Information
(Ontology)





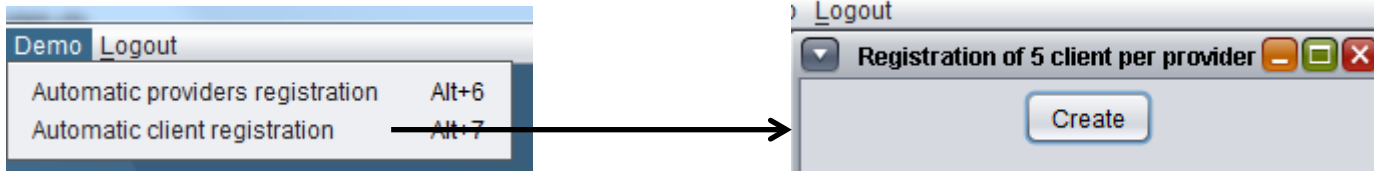
List of created Providers

The screenshot shows the QoS-MONaaS application window. The menu bar includes 'Client Functionalities', 'Provider Functionalities', 'Demo', and 'Logout'. The 'Demo' menu is open, showing options: 'Automatic providers registration' (Alt+6), 'Automatic client registration' (Alt+7), 'List providers' (Alt+A), and 'List clients' (Alt+B). A 'List Providers' dialog box is open in the foreground, displaying a table with the following data:

Name	Category	Description
Provider1	Energy	Power Energy provider
Provider2	Water	Water provider

QoS-MONaaS Demo :

Step 3-1



Automatic Client Registration

► It registers 5 Clients per provider

■ Clients of Provider1

- SmartMeter0
- **SmartMeter1**
- SmartMeter2
- SmartMeter3
- SmartMeter4

It's a client of Provider1

It's a Gold User

It wants to receive violations in List mode

■ Clients of Provider2

- SmartMeter1
- SmartMeter5
- SmartMeter6
- SmartMeter7
- SmartMeter8

QoS-MONaaS Demo :

Step 3-2



List of created Clients

The screenshot shows the QoS-MONaaS application window. The menu bar includes 'Client Functionalities', 'Provider Functionalities', 'Demo', and 'Logout'. The 'Demo' menu is open, showing options: 'Automatic providers registration' (Alt+6), 'Automatic client registration' (Alt+7), 'List providers' (Alt+A), and 'List clients' (Alt+B). The 'List clients' option is selected, and a dialog box titled 'List Clients' is displayed. The dialog box contains a table with the following data:

Provider	Client	SLA
Provider1	SmartMeter0	Gold_user.xml
Provider1	SmartMeter1	Gold_user.xml
Provider1	SmartMeter2	Gold_user.xml
Provider1	SmartMeter3	Gold_user.xml
Provider1	SmartMeter4	Gold_user.xml
Provider2	SmartMeter1	Silver_user.xml
Provider2	SmartMeter5	Gold_user.xml
Provider2	SmartMeter6	Silver_user.xml
Provider2	SmartMeter7	Gold_user.xml
Provider2	SmartMeter8	Gold_user.xml
Provider2	SmartMeter9	Silver_user.xml



What means Gold User ?

```
<wsag:GuaranteeTerm Name="DowntimeForADay" Obligated="Provider">
  <wsag:ServiceScope>
    <wsag:ServiceName>Metering</wsag:ServiceName>
  </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>
    <wsag:KPITarget>
      <wsag:KPIName>Downtime</wsag:KPIName>
      <wsag:Target>Count IS_LESS 5 per 1 day </wsag:Target>
    </wsag:KPITarget>
  </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```



The count of downtime in one day must be less than 5

```
<wsag:GuaranteeTerm Name="AgreementSmartMeter2" Obligated="Provider">
  <wsag:ServiceScope>
    <wsag:ServiceName>Metering</wsag:ServiceName>
  </wsag:ServiceScope>
  <wsag:ServiceLevelObjective>
    <wsag:KPITarget>
      <wsag:KPIName>DetectedConsumption</wsag:KPIName>
      <wsag:Target>Max IS_LESS_INCLUSIVE 2 per 1 day</wsag:Target>
    </wsag:KPITarget>
  </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```



Max consumption in one day must be less than 2

Example of generated state machine: (graphic representation)



KPI1 -> count(Downtime) <5 per 1 day

timeout 15 min

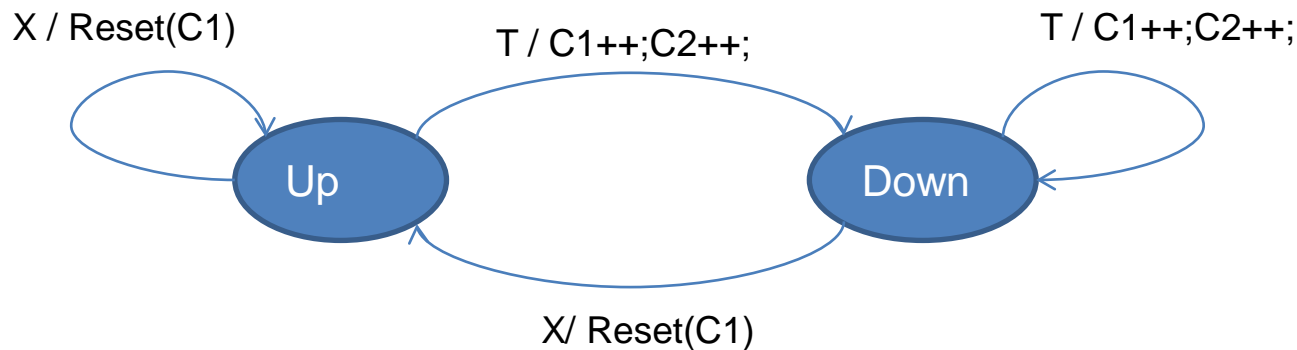
X: measure

T: timing (30 sec)

C1 counts clock "ticks" until timeout expires

C2 counts clock "ticks" until time window expires

C3 counts the number of downtime



Transition guard for T event in Down state

$[(C1==(30-1))\&\&(C2==(2*60*24)-1)] \rightarrow \{C1++;C2++;C3++;forward(C3);reset(C1,C2,C3)\}$

$[(C1==(30-1))\&\&(C2!=(2*60*24)-1)] \rightarrow \{C1++;C2++;C3++;reset(C1)\}$

$[(C1!=(30-1))\&\&(C2!=(2*60*24)-1)] \rightarrow \{C1++;C2++;\}$

$[(C1!=(30-1))\&\&(C2==(2*60*24)-1)] \rightarrow \{C1++;C2++;forward(C3);reset(C2,C3)\}$

Excerpt of state machine code: (automatically generated by QoS-MONaaS)

```
Down
{
    %{New event on data stream}%
    ReceiveEvent(event: CEPEvent*) Up          {
        %{Restart time for timeout}%          resetCounter(1);
    }
    %{Receive Timing}%
    ReceiveTiming() [ctxt.getCounter(1)==29 && ctxt.getCounter(2) == 86399] Down {
        %{count time for timeout}%            incCounter(1);
        %{count time for time window}%        incCounter(2);
        %{count KPI}%                          incCounter(3);
        %{Forward count of KPI}%               forwardEvent(ctxt.getCount(3));
        %{Reset all time}%                      resetCounter(1);      resetCounter(2);
        %{Reset KPI count}%                     resetCounter(3);
    }
    .....
}
```


QoS-MONaaS Demo :

Step 5

- Select Provider1-Gold_user.xml in the “SLA to monitor” box
 - ▶ to monitor the SLA between SmartMeter1 and Provider1
- Set monitoring interval
- Start monitoring

Monitor

SLA to monitor: Provider1-Gold_user.xml | Status: Not Monitored | Download SLA

Starting date: settembre 2011 | Ending date: settembre 2011

Starting Time: 11:02 | Ending Time: 11:02

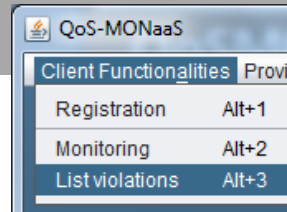
Start Monitoring

Monitoring Response

```
Request of smartmeter1
to monitor Gold_user.xml of Provider1
From Fri Sep 23 11:02:00 CEST 2011
Until Sat Sep 24 11:02:00 CEST 2011
Success
```

QoS-MONaaS Demo :

Step 6



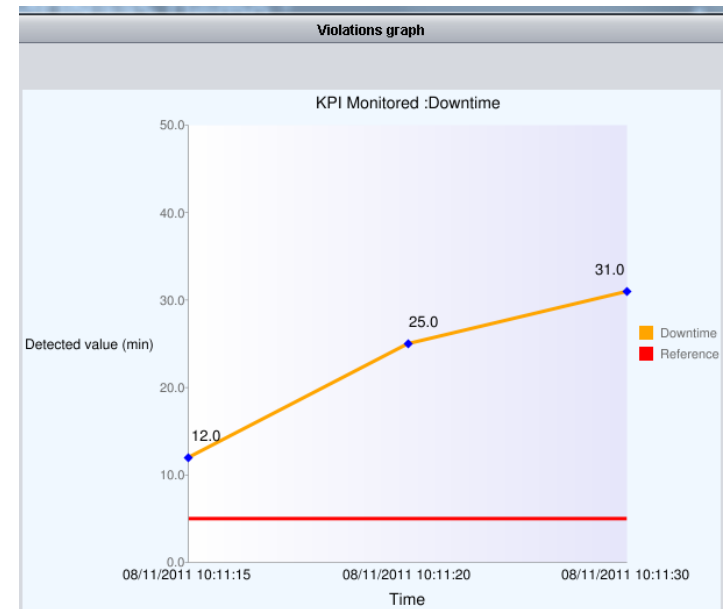
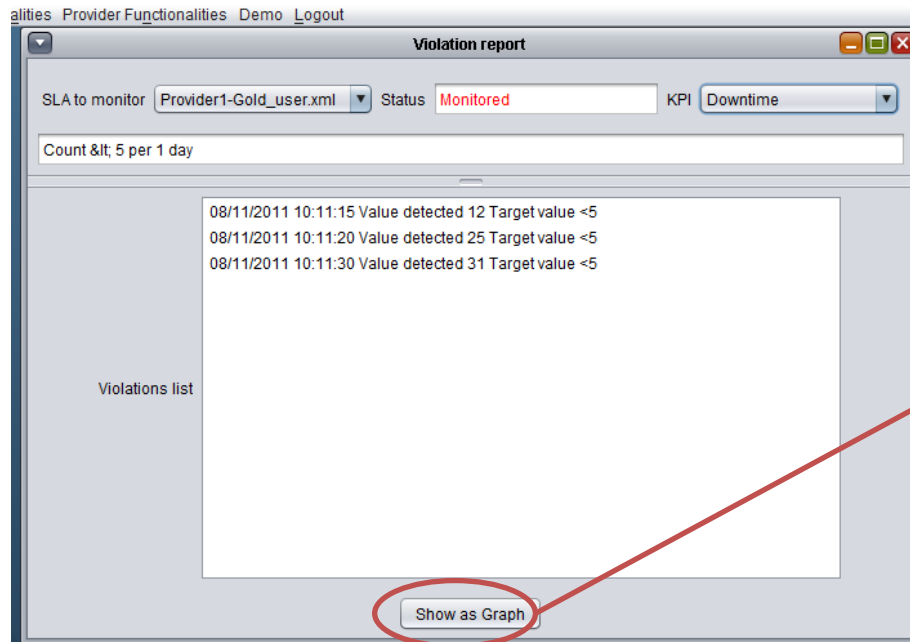
Select List violation

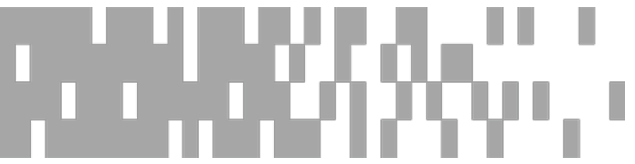
Select Provider1-Gold_user.xml in “SLA to monitor” box

▶ to show violations of the SLA between SmartMeter1 and Provider1

Select Downtime in the “KPI” box

▶ To show violations of KPI Downtime





■ Simple run.....

THANK YOU!

<http://www.srt-15.eu/>

Luigi Sgaglione

luigi.sgaglione@epsilononline.com



TECHNISCHE
UNIVERSITÄT
DRESDEN



YAHOO!

unine
UNIVERSITÉ DE
NEUCHÂTEL



SRT-15 is supported by the European Commission's Seventh Framework Programme (FP7)

