Engineering Virtual Domain-Specific Service Platforms

Specific Targeted Research Project: FP7-ICT-2009-5 / 257483

Service Platform Infrastructure Repository Concept & Realization (Interim)

Abstract

The INDENICA project provides infrastructure components and tools to support the effective creation of domain-specific Virtual Service Platforms (VSP). In this report, accompanying the submitted code deliverable, we present a concept and first prototype for a Service Platform Infrastructure repository to support the creation of VSPs. The repository encapsulates design time, deployment time, as well as runtime aspects of VSPs. At design time, the repository allows for the creation of infrastructure component libraries to foster reuse and protect investments in integration actions. Furthermore, it enables the retention of relevant design-time information for the deployment and runtime environment of VSPs. At runtime, the repository acts as the central entity responsible for storing and forwarding data to all VSP components using a messaging infrastructure.

Document ID:	INDENICA - D2.3.1
Deliverable Number:	D2.3.1
Work Package:	WP2
Type:	Deliverable
Dissemination Level:	CO
Status:	final
Version:	1.0
Author(s):	*TUV, SUH*

Project Start Date: October 1st 2010, Duration: 36 months

Version History

0.1	02. Nov 2011	Initial version
0.2	06. Dec 2011	Update Overview, Repository sections
0.3	12. Jan 2012	Prototype Implementation Notes, Usage Guide
0.4	14. Jan 2012	Update Prototype Usage Guide
0.5	27. Jan 2012	Incorporate review comments
1.0	27. Jan 2012	Final version for submission.

Document Properties

The spell checking language for this document is set to UK English.

Table of Contents

Table of Contents 3
Table of Figures 5
1 Introduction 6
2 Overview 7
3 Repository 8
4 Prototype Implementation 9
4.1 Usage Guide 9
5 Conclusion 27

Table of Figures

Figure 1: INDENICA Runtime Infrastructure Architecture Overview 7
Figure 2: Repository Architecture Overview 8
Figure 3: Configuration Launch Dialog 10
Figure 4: Connect to Repository Database 10
Figure 5: Establish Database Connection 10
Figure 6: Environment Configuration Dialog 11
Figure 7: Create new Environment Configuration 11
Figure 8: New Environment Configuration Dialog 12
Figure 9: Environment Configuration Defaults and Usage Hints 12
Figure 10: Sample Infrastructure Instance Environment Configuration 13
Figure 11: Launch Incoming Events Dialog 14
Figure 12: Load previously created Configuration 14
Figure 13: Incoming Events Dialog 15
Figure 14: An exemplary Monitoring Event Type 16
Figure 15: ServiceInvocationFailure Event Type 17
Figure 16: Launch the outgoing Adaptation Events Dialog 18
Figure 17: Exemplary Adaptation Interface Event 19
Figure 18: Exemplary Adaptation Interface Event 20
Figure 19: Launch Monitoring Engine Configuration 21
Figure 20: Exemplary Monitoring Rule 22
Figure 21: Launch Adaptation Engine Configuration 23
Figure 22: Exemplary Adaptation Rule 24
Figure 23: Exemplary Adaptation Rule 25
Figure 24: Exemplary Tuscany Runtime Configuration 26

1 Introduction

The INDENICA project aims at providing infrastructure components and tools to support the efficient and effective creation of domain-specific Virtual Service Platforms (VSP). The Service Platform Infrastructure Repository represents a central component to achieve these goals at design time, deployment time, and runtime. At design time, the repository acts as infrastructure component library to foster reuse, decrease development and integration time, as well as protect investments in the INDENICA platform. At deployment time, the repository holds deployment descriptors and environment configuration information enabling easy (re-) deployment of VSPs and their components. Furthermore, the repository allows for the transfer of relevant design and deployment time information to the runtime environment. This allows runtime components, such as monitoring and adaptation facilities, to take additional information about the deployed VSPs and their infrastructure into account, facilitating greater control over the runtime characteristics of VSPs and their environment.

In this prototype deliverable document we will briefly discuss the concept and architecture for the Indenica Service Platform Infrastructure Repository, and provide an overview of the current state of the prototype implementation, along with a detailed usage guide for the provided tools.

2 Overview

The INDENICA service platform infrastructure consists of a number of components that help to build and run virtual service platforms. It uses a messaging infrastructure, providing a unified communication interface for all infrastructure components. As shown in Figure 1, control interfaces provide for a unified mechanism for a bidirectional communication with domain-specific application services and components. The platform provides a publish/subscribe mechanism, enabling complex communication pattern, and allows components to receive information they are interested in. The infrastructure enables efficient handling of large amounts of streaming data (messaging infrastructure), persistent storage of relevant information and its retrieval (repository), as well as aggregation of online and offline data.

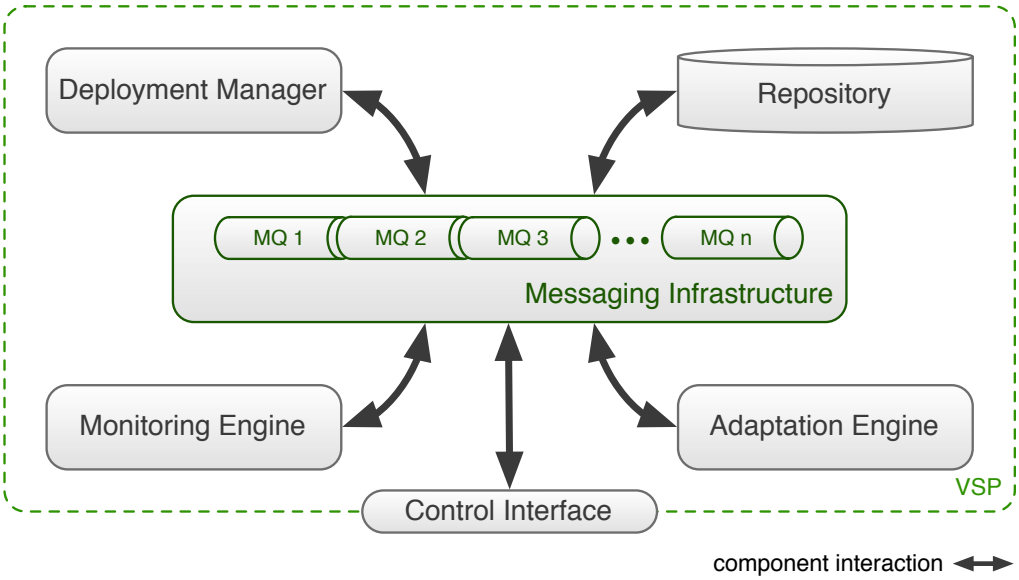

Figure 1: INDENICA Runtime Infrastructure Architecture Overview

The repository contains data about deployment configurations needed by the deployment manager, which is responsible for instantiating and assembling virtual service platforms. The monitoring engine combines online data about the domain-specific runtime components with historical data to draw conclusions about the system's status. Based on this information, the adaption engine reasons about and triggers reconfigurations. The rules and policies determining the monitoring and adaptation engines are also stored in the repository. The repository will allow storing the following artefacts, and will provide specialized storage and retrieval methods that are tailored to the specifics of each of the artefact types:

- Deployment descriptors
- Monitoring rules
- Adaptation rules
- Platform instance configuration
- View-based modelling framework models (WP3)
- Variability engineering information (WP2)

3 Repository

The repository is the entity in INDENICA responsible for retention and retrieval of persistent data. It manages data that needs to be persisted or be available for retrieval using query mechanisms. The key design goals for the repository are availability (provide a scalable architecture that is able to cope with high volumes of data and frequent requests), data store abstraction (technological flexibility with respect to the technology used for storing data in the backend), and domain-specific data retrieval modes (different query methods tailored to the specifics of various types of information managed by the repository).

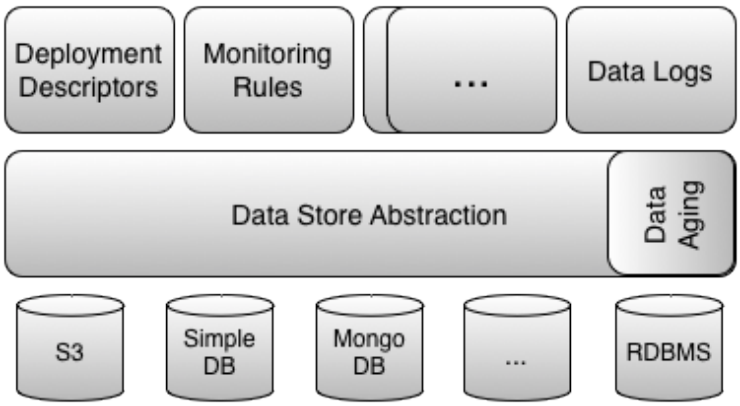

Figure 2: Repository Architecture Overview

An overview of the repository architecture is shown in Figure 2. It is designed to be data store agnostic, accommodating for different requirements of service platform installations. The repository furthermore allows for the retrieval of stored data via custom interfaces for different types of information (deployment descriptors, monitoring rules, log data, etc.). The repository's capabilities will range from storing simple key/value entries for global configuration values to providing structured and interconnected data formats for platform-specific variability models to performing complex continuous event-based queries in reaction to monitoring events.

Such tailored interfaces in the form of domain-specific query languages, APIs, or GUIs allow for efficient interaction with the repository. The data store abstraction allows for the transparent use of multiple storage back ends, ideally suited for the stored data and the particular application environment.

The data ageing component will be integrated into the next version of the repository. It will allow annotating data with descriptions about their importance and relevance over time, as well as how the data can be aggregated to provide continuous operation. Based on that information, the data aging component will schedule aggregation or even removal of less important data. This will be especially useful for data logs.

4 Prototype Implementation

In this section, we present the current status of the prototype implementation of the Service Platform Infrastructure Repository, along with a step-by-step guide to creating a basic INDENICA runtime instance using the repository and provided tools.

The basic prototype infrastructure is realized using Apache Tuscany[1], a Service Component Architecture[2] (SCA) container, RabbitMQ[3], an Advanced Message Queuing Protocol[4] (AMQP) messaging system, Esper[5], a Complex Event Processing (CEP) engine, and MongoDB[6], a document-oriented database. These technologies have been chosen based on a careful evaluation of the repository's requirements and the current state of the art in database and data processing technology. As mentioned in Section 3, the repository is designed for scalability, data store abstraction and domain-specific data retrieval. In the current prototype implementation, we have integrated a MongoDB database backend into the data store abstraction for illustrative purposes. MongoDB's scalable and agile design is well-suited for deployment in Cloud environments and provides ideal characteristics for storing and querying high volumes of persistent data (e.g., event logs). Non-persistent data with high volatility and time-critical online queries (e.g., required by the INDENICA Monitoring Engine) are handled by Esper. The following usage guide is based on an end-to-end use case that stores and executes monitoring and adaptation rules for an exemplary service platform.

4.1 Usage Guide

In this section we provide a step-by-step guide for creating a basic instance of the INDENICA runtime infrastructure using the Service Platform Infrastructure Repository.

In order to create a new instance of the INDENICA runtime infrastructure, running instances of RabbitMQ and MongoDB are required.

To guide the user through the steps necessary to create a runtime infrastructure instance, a simple launch dialog can be invoked using:

```
sh launchConfiguration.sh
```

[1] http://tuscany.apache.org/

[2] http://oasis-opencsa.org/sca

[3] http://www.rabbitmq.com/

[4] http://www.amqp.org/confluence/display/AMQP/AMQP+Specification

[5] http://esper.codehaus.org/

[6] http://www.mongodb.org/

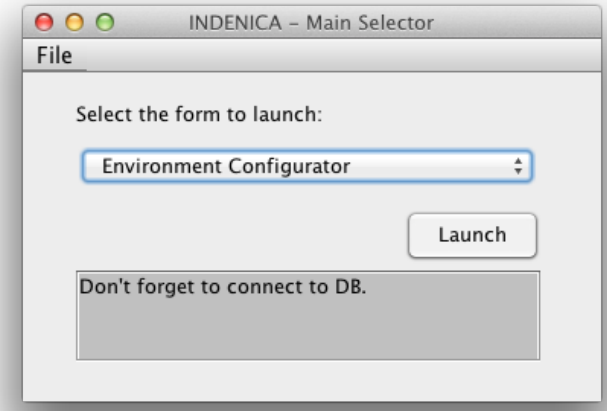

Figure 3: Configuration Launch Dialog

Next, a connection to the repository database must be established by invoking "File, Connect to DB."

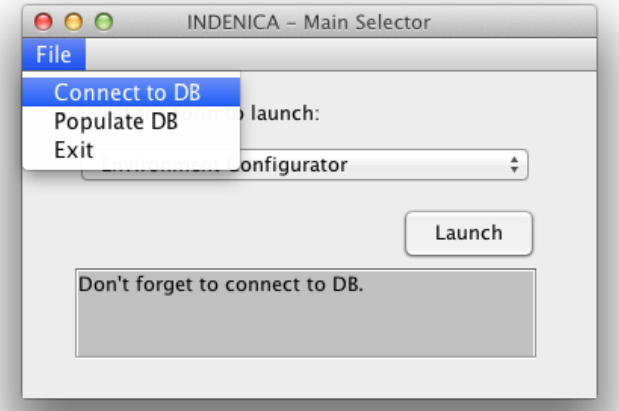

Figure 4: Connect to Repository Database

The data entered in the connection dialog must point to a running MongoDB instance in order to complete correctly.

Figure 5: Establish Database Connection

After successfully establishing a connection to the database, an environment configuration for a new infrastructure instance can be created by selecting "Environment Configurator" in the dropdown list and clicking "Launch."

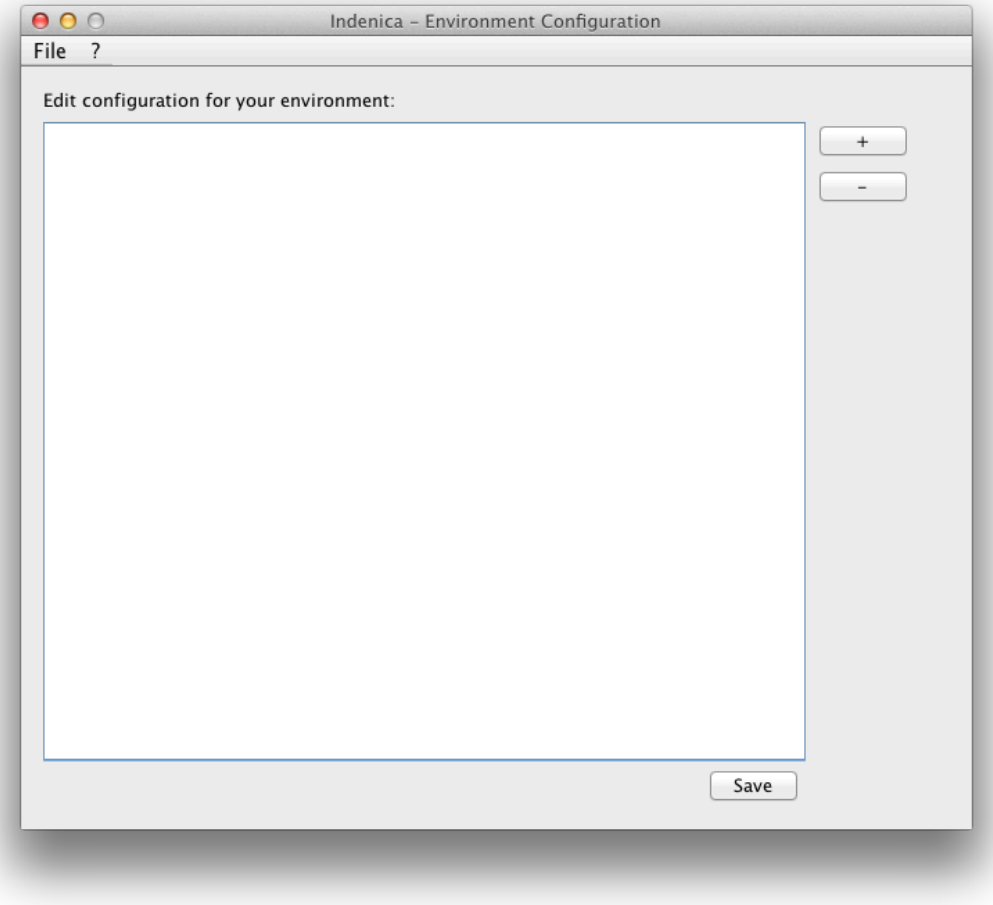

Figure 6: Environment Configuration Dialog

To create a new environment configuration, select “File, New Configuration.”

Figure 7: Create new Environment Configuration

In the following dialog, a name for the infrastructure instance to be created must be supplied (in our case, “SampleInstance”).

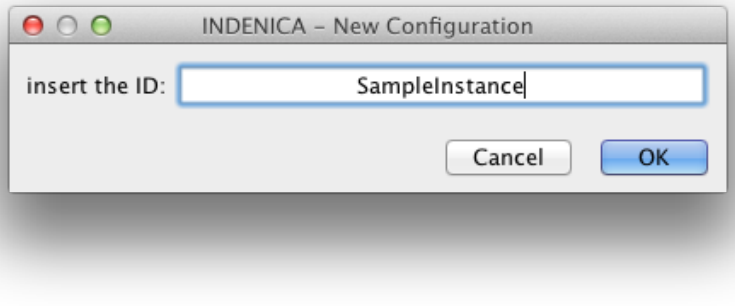

Figure 8: New Environment Configuration Dialog

After confirming the creation of the new instance configuration, the environment configuration is pre-populated with relevant configuration properties that need to be provided by the user. Where appropriate, the values are set to factory default settings, or contain usage information.

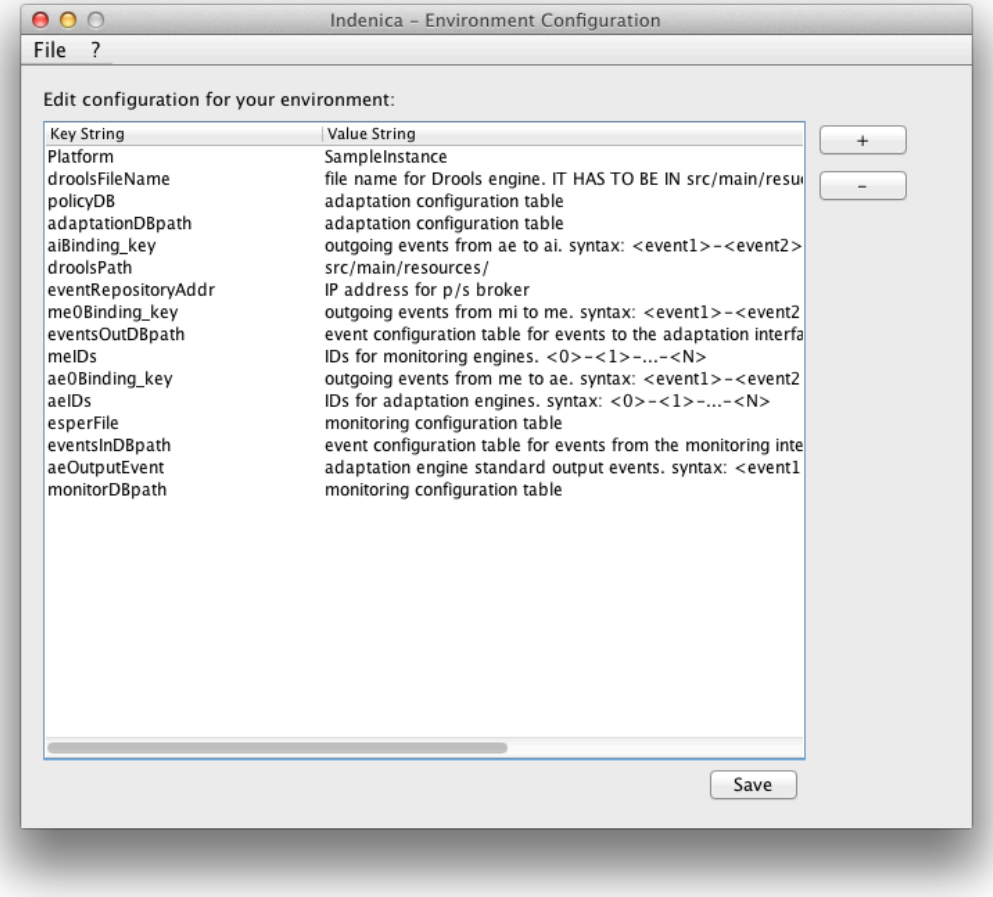

Figure 9: Environment Configuration Defaults and Usage Hints

Figure 10 shows an exemplary configuration for an infrastructure instance. This configuration creates one Monitoring Engine (ME) instance, as well as one Adaptation Engine (AE) instance and defines events that these components receive.

In a future version, we aim at providing a more visually appealing interface to defining components and their interactions using a drag-and-drop approach.

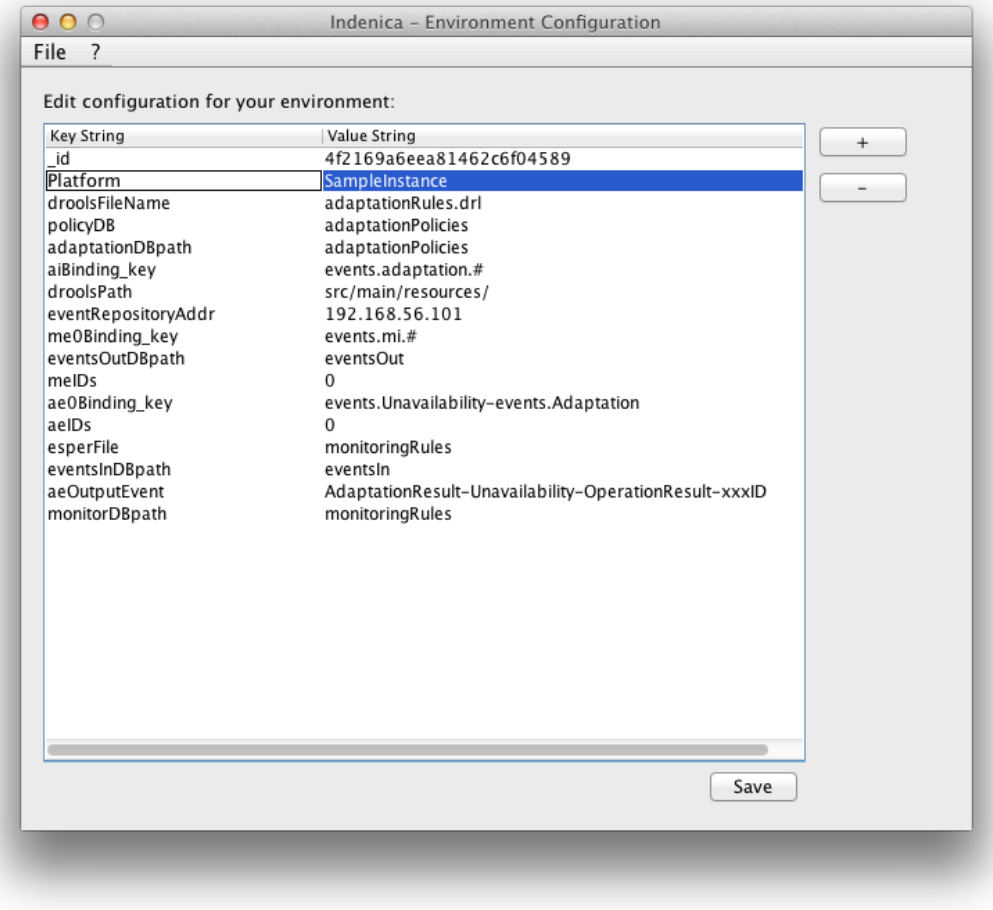

Figure 10: Sample Infrastructure Instance Environment Configuration

The configuration can then be saved using the “Save” button on the lower right, and the “Environment Configuration” dialog can be closed.

The next step in setting up an infrastructure instance involves defining concrete events that the monitoring interfaces emit for consumption by the previously defined monitoring engine.

In the launcher dialog, select “Events Incoming” in the dropdown box and click “Launch.”

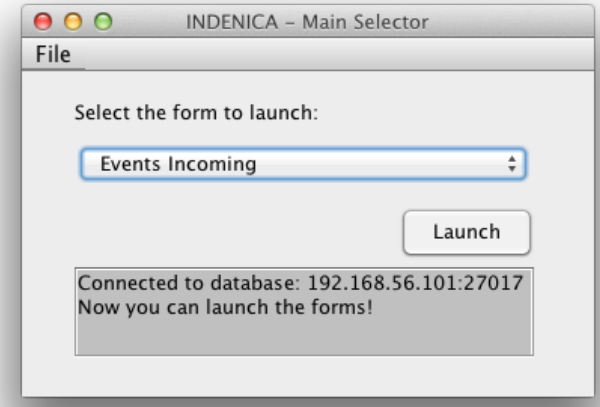

Figure 11: Launch Incoming Events Dialog

The previously created environment configuration must be loaded in the incoming events dialog.

To load an environment configuration, select "File, Load Configuration," click "Explore" and select "SimpleInstance."

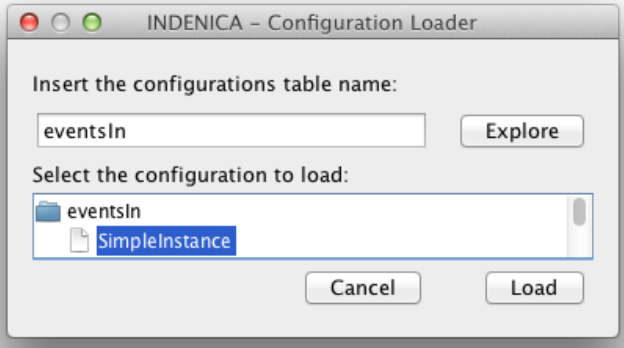

Figure 12: Load previously created Configuration

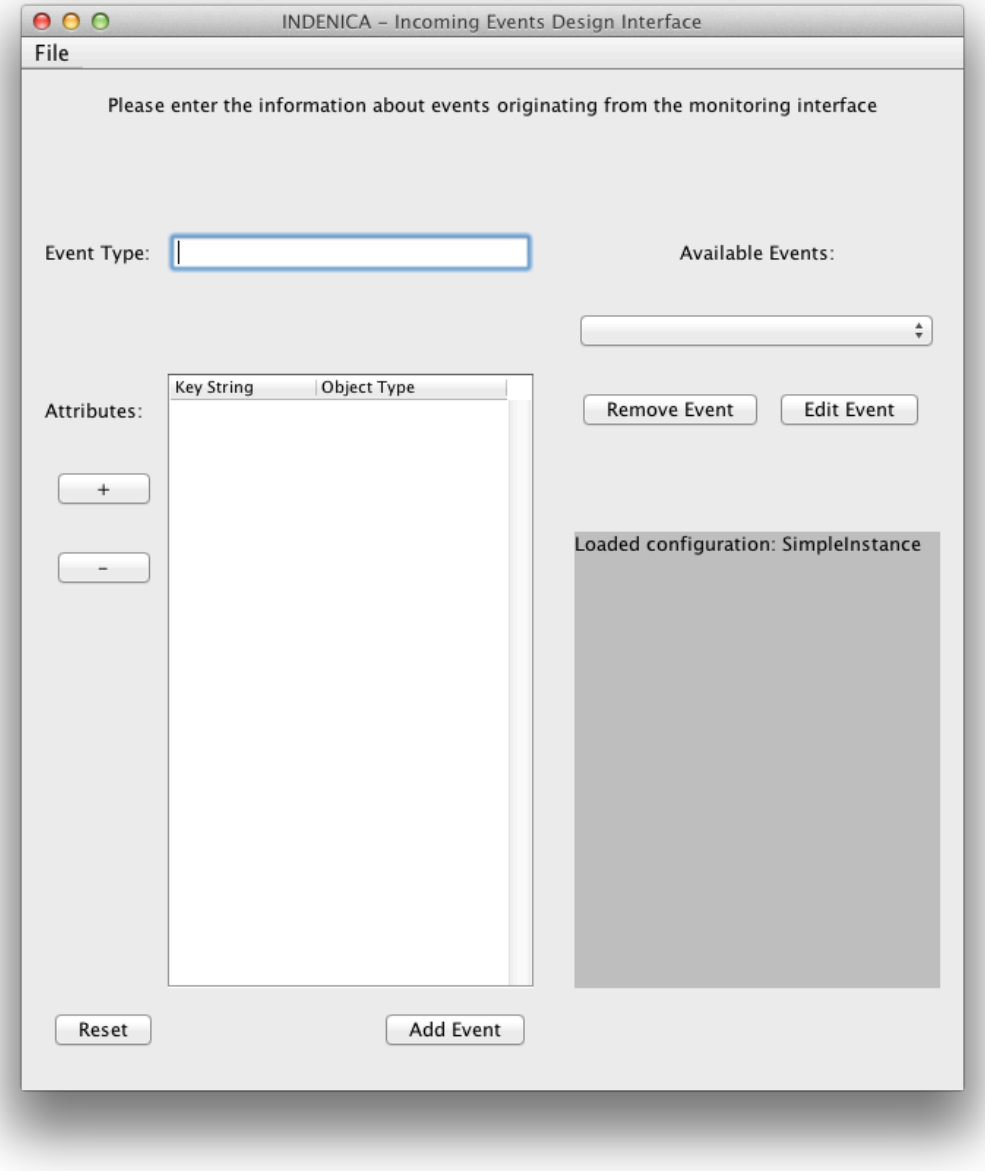

Figure 13: Incoming Events Dialog

Now, events originating from the monitoring interface can be created. Figure 14 shows an exemplary event type, 'ServiceInvocationEvent', and according properties.

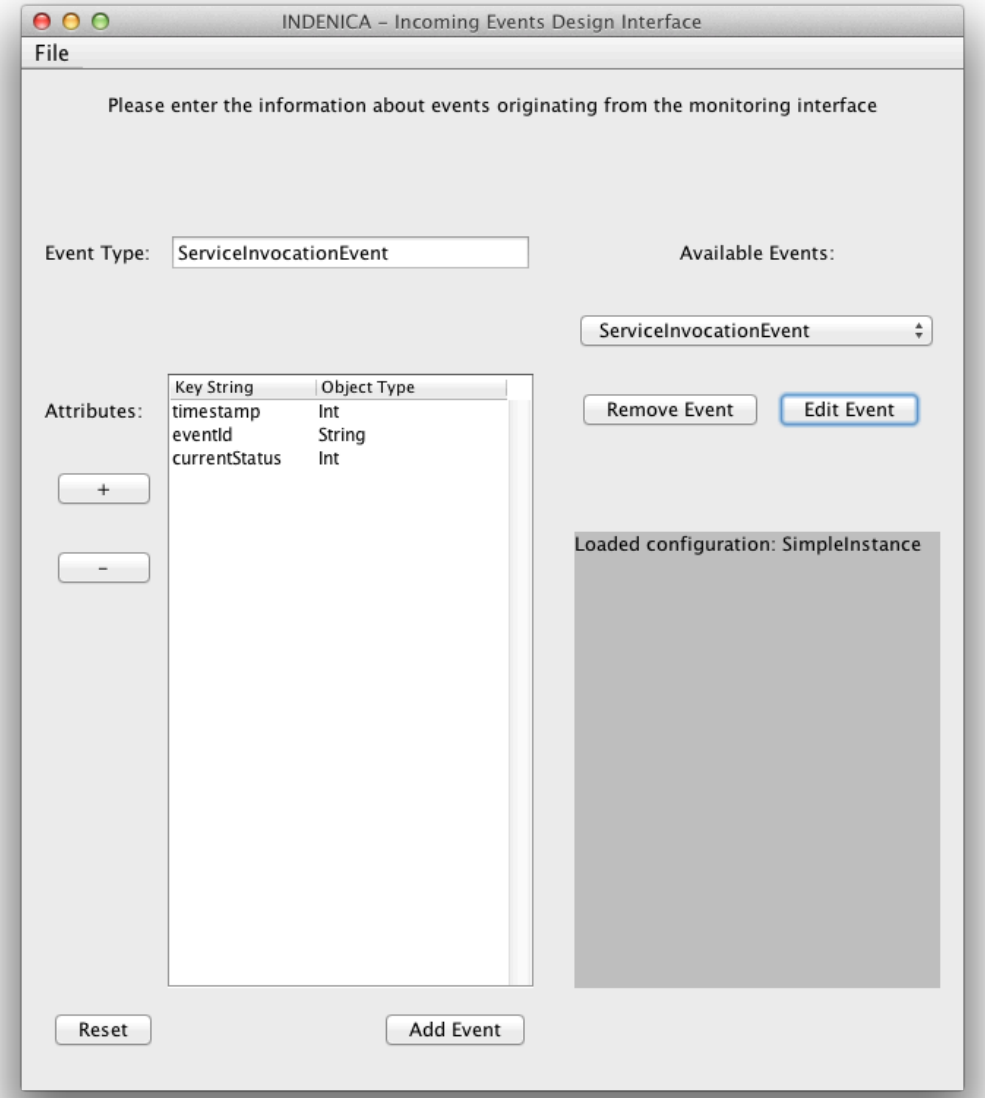

Figure 14: An exemplary Monitoring Event Type

Event types and their settings are saved using the "Add Event" button on the bottom of the dialog. For the sample instance, we will add a second event type, 'ServiceInvocationFailureEvent,' as depicted in Figure 15.

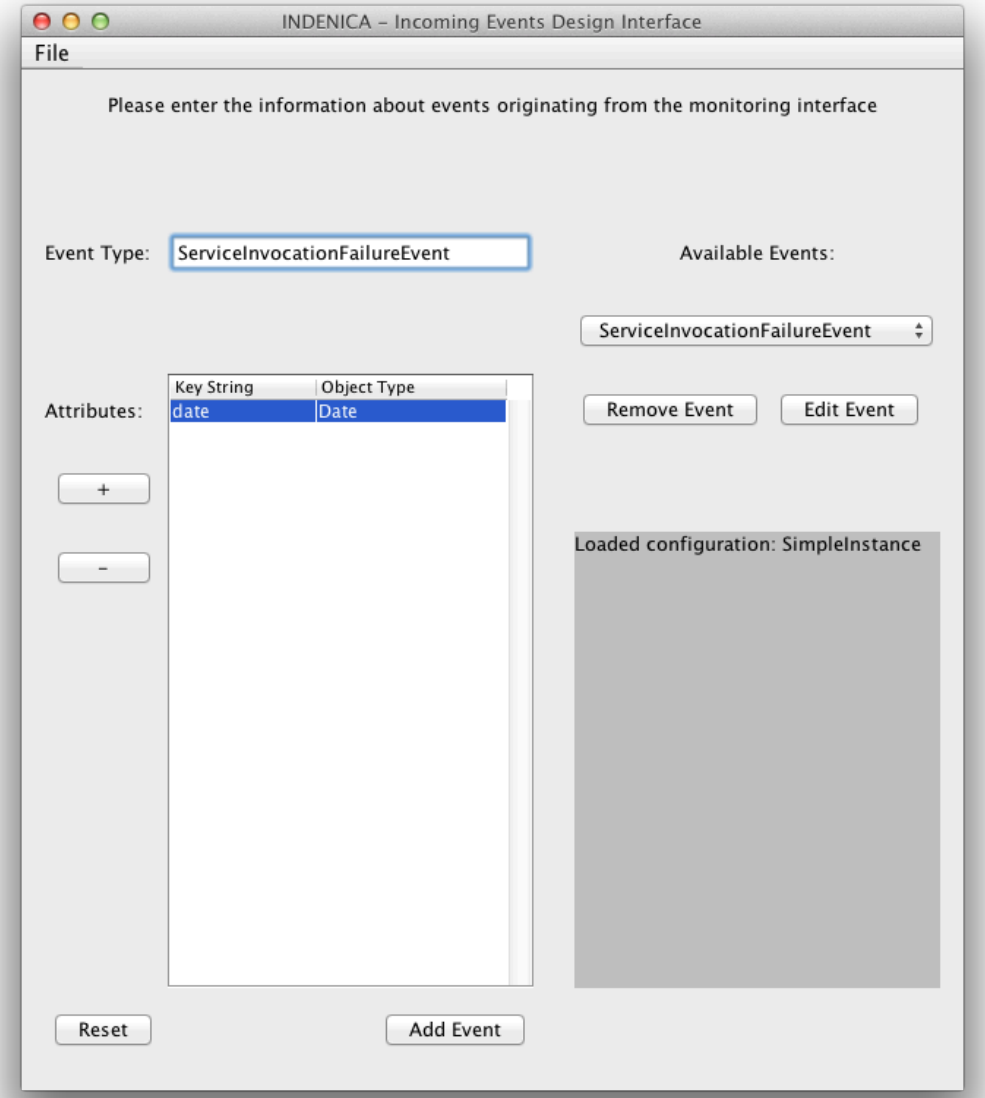

Figure 15: ServiceInvocationFailure Event Type

The next step involves specifying adaptation events sent to the adaptation interface in order to influence integrated service platforms. Selecting “Events Outgoing” in the launch dialog opens the according dialog.

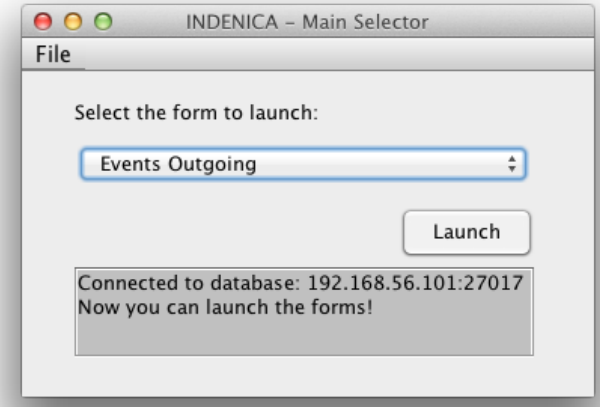

Figure 16: Launch the outgoing Adaptation Events Dialog

Again, the environment configuration must be loaded by invoking “File, Load Configuration,” clicking “Explore,” and selecting “SampleInstance.”

Events sent to the adaptation interface can be created similar to the incoming events. For our sample instance, we will create two events representing a notification about a healthy or unhealthy system state (“adaptation.SystemOK” and “adaptation.SystemKO”), as shown in the following two figures.

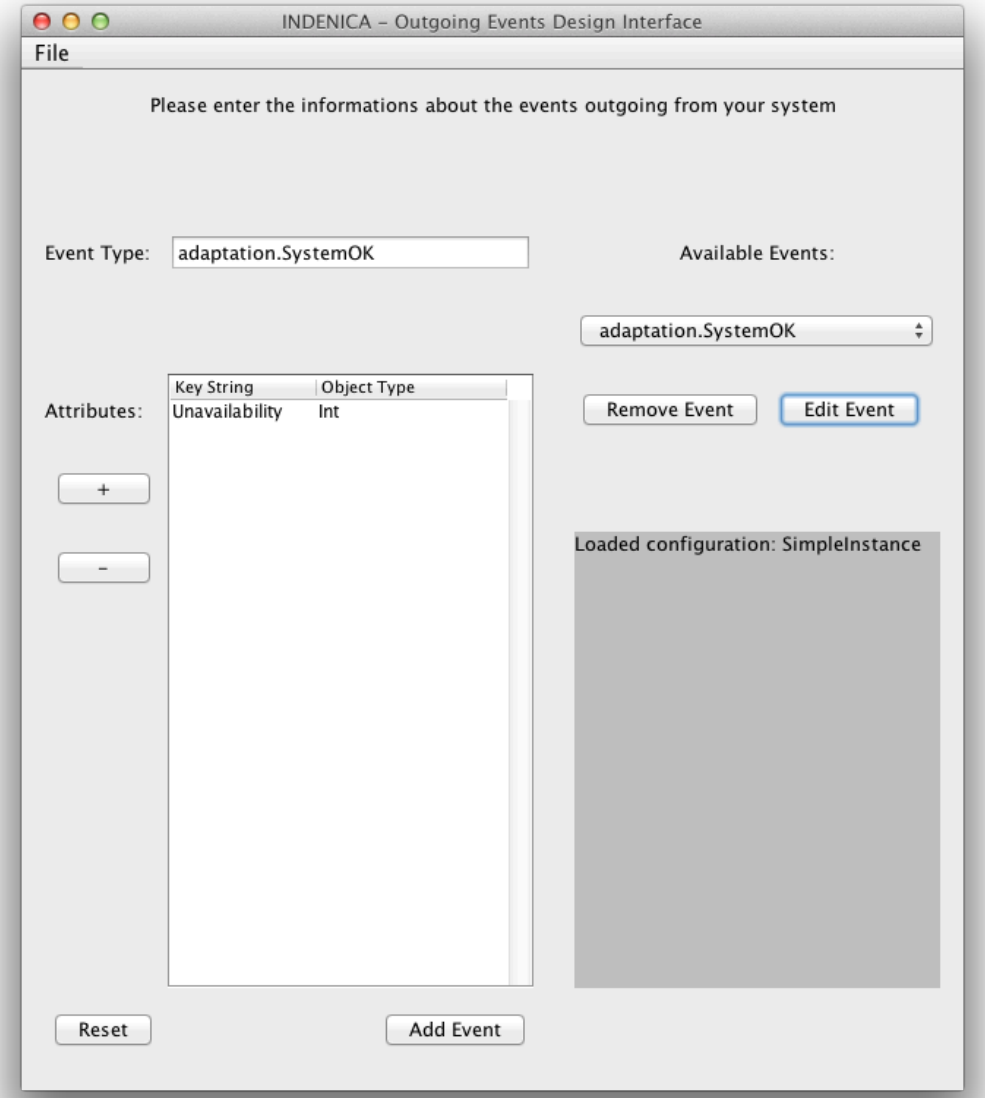

Figure 17: Exemplary Adaptation Interface Event

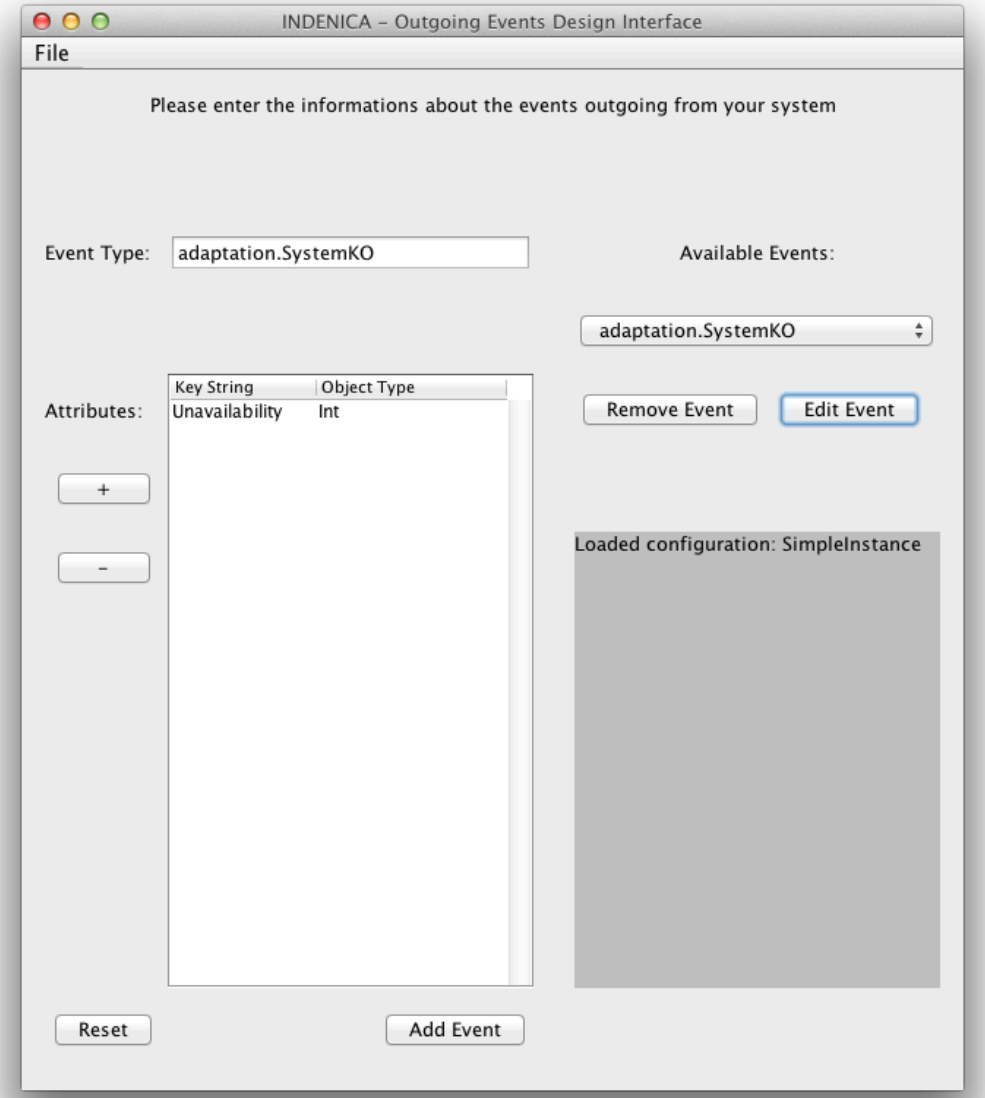

Figure 18: Exemplary Adaptation Interface Event

After saving the outgoing adaptation interface events, we can now configure the monitoring engine by selecting the according option in the dropdown menu in the launch dialog.

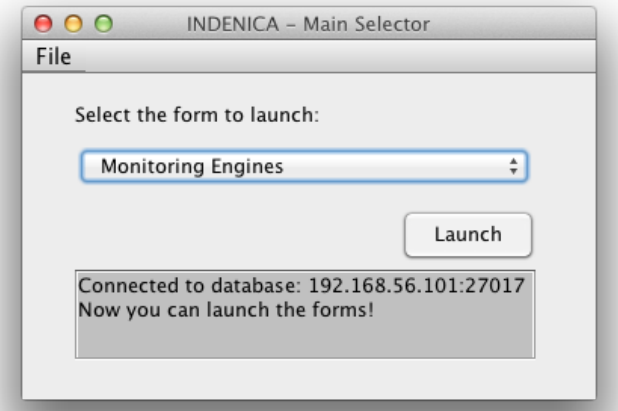

Figure 19: Launch Monitoring Engine Configuration

After loading the environment configuration by invoking "File, Load Configuration," clicking "Explore," and selecting "SimpleInstance," the specified Monitoring Engines can be configured.

To configure Monitoring Engine 0, we select it in the "Engine ID" dropdown and choose "Load." Now, monitoring rules can be added, using the rules specified in the "Events Incoming" dialog. The current prototype implementation supports rules in the Esper Query Language[7] (EQL). Currently, the monitoring rules are provided manually, but the aim for a future version of the repository is to integrate this part with the view-based modelling approach of WP3. Integration with WP3 will enable to automatically generate monitoring rules from UML models and to build up a library of reusable and composable monitoring rules.

Figure 20 shows an exemplary monitoring rule evaluating the availability ratio of a monitored service by analysing successful and failed invocations over a period of one day.

[7] http://esper.sourceforge.net/esper-0.7.5/doc/reference/en/html/EQL.html

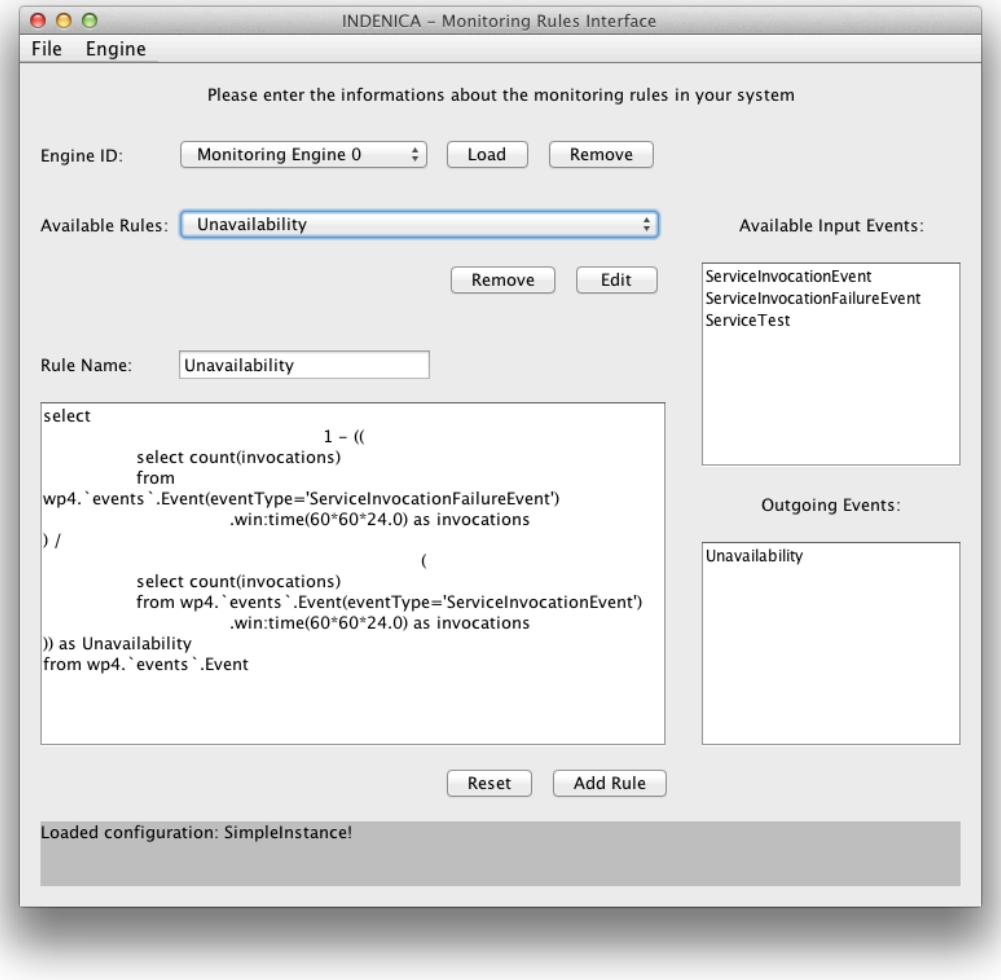

Figure 20: Exemplary Monitoring Rule

After saving the monitoring rule (using “Add Rule” on the lower right), we can configure actions the AE should take in response to certain monitoring events by invoking the “Adaptation Engines” configuration in the launch dialog.

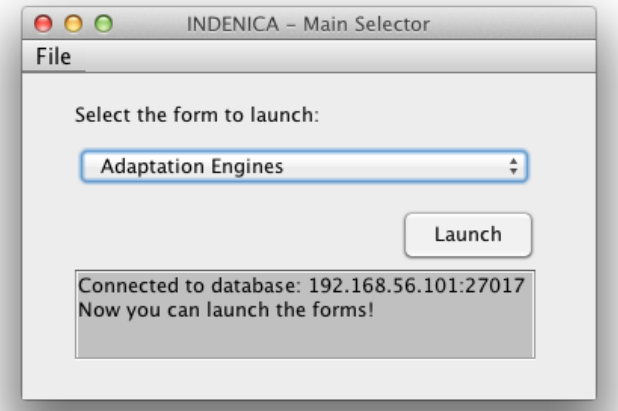

Figure 21: Launch Adaptation Engine Configuration

After loading the environment configuration by invoking "File, Load Configuration," clicking "Explore," and selecting "SimpleInstance," the specified AEs can be configured.

To configure Adaptation Engine 0, we select it in the "Engine ID" dropdown and choose "Load." Now, adaptation rules can be added, using the event specified in the ME configuration dialog. The current prototype implementation supports adaptation rules in the Drools[8] rule language.

For the sample instance, we will create two simple adaptation rules, as shown in Figure 22 and Figure 23.

[8] http://docs.jboss.org/drools/release/5.4.0.Beta1/drools-expert-docs/html_single/index.html#d0e2777

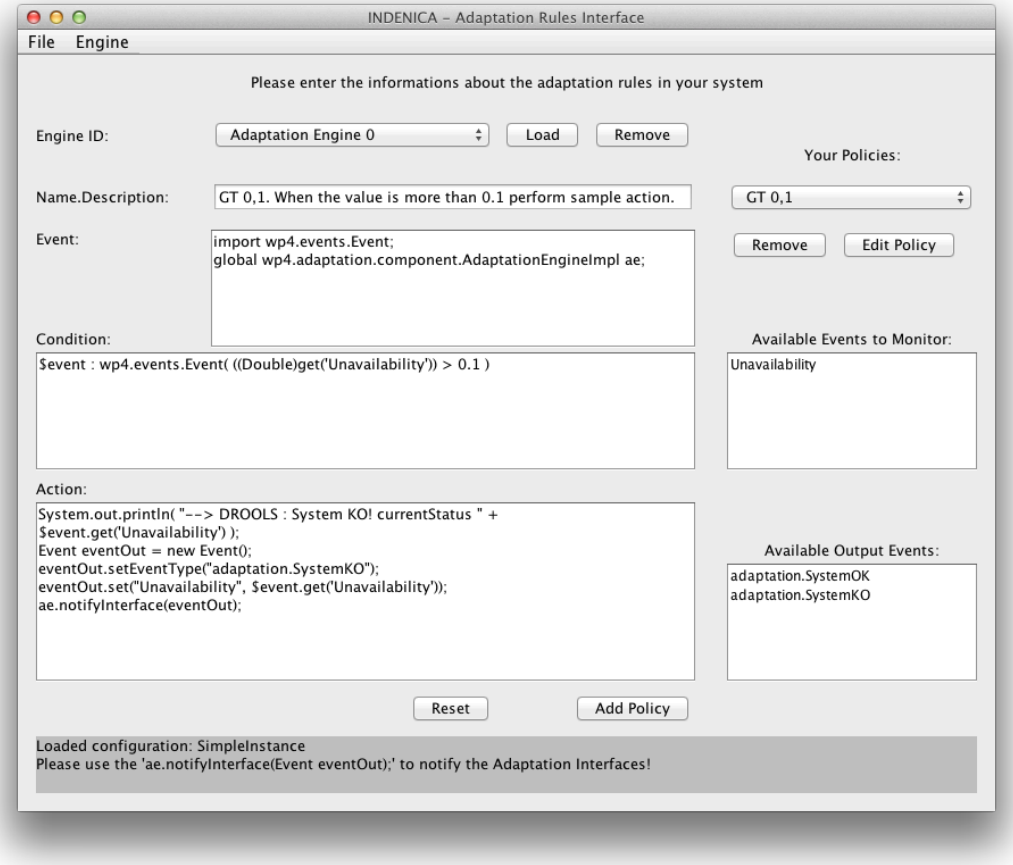

Figure 22: Exemplary Adaptation Rule

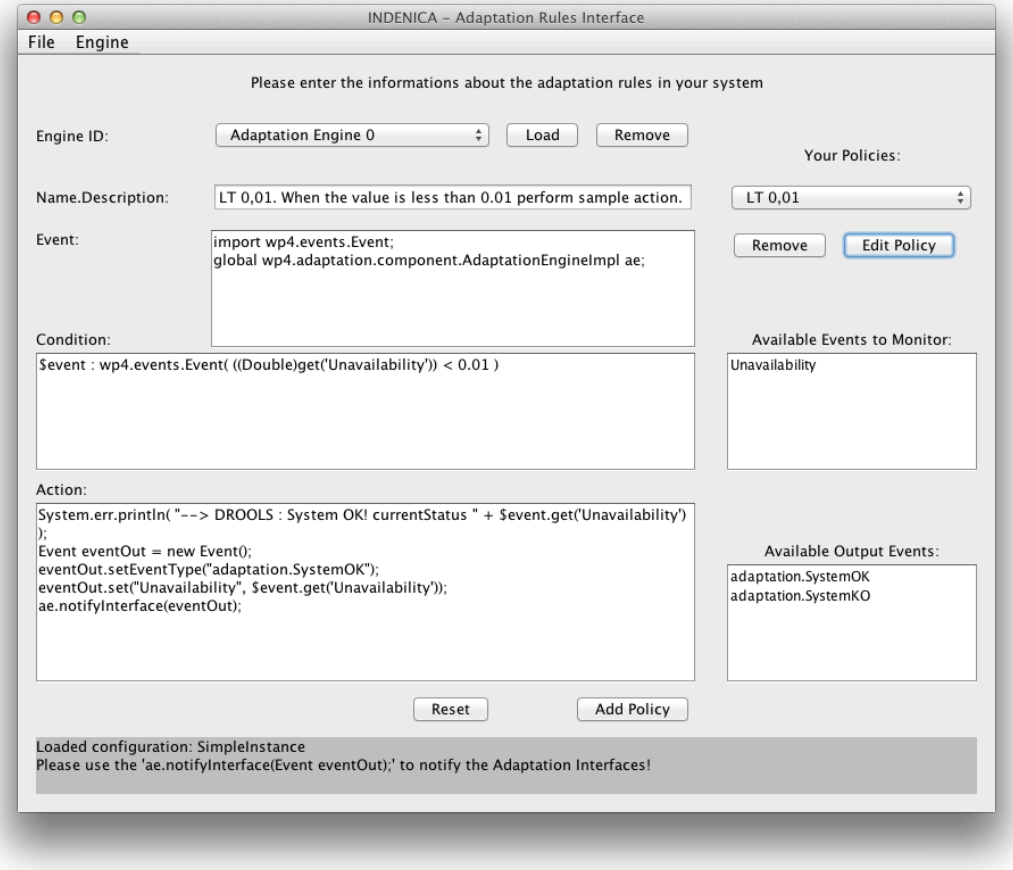

Figure 23: Exemplary Adaptation Rule

The adaptation rules complete the environment configuration. A platform integrator is now required to create the appropriate Monitoring Interface (MI) and Adaptation Interface (AI) for the integrated platforms to interact with the INDENICA platform. To ease this step, we will provide for an infrastructure integration library, that allows for the retention and reuse of the created integration interfaces. In the current prototype implementation the infrastructure library is represented by java packages `indenica.[monitoring|adaptation].component`. In a future version, the infrastructure library will be tightly integrated into the development and configuration workflow.

The prototype deliverable contains exemplary MI and AI implementations, that allow for the integration of different kinds of platforms.

To invoke the created platform, a Tuscany configuration composite file is needed. This configuration artefact will be generated by the View-based Modelling Framework. In the current prototype implementation, this file has to be supplied manually.

```
<composite xmlns="http://www.osoa.org/xmlns/sca/1.0"
       targetNamespace="http://indenica.eu/repository"
       xmlns:hw=" http://indenica.eu/repository"
    name="IndenicaRuntime">

    <component name="ComponentInitializerComponent">
        <implementation.java
class="indenica.deployment.component.ComponentInitializerImpl" />
```

```
        <reference name="adaptationInterface"
target="AdaptationInterfaceComponent" />
        <reference name="repository" target="RepositoryComponent" />
        <reference name="platform" target="SimplePlatformComponent" />
    </component>

    <component name="MonitoringInterfaceComponent">
        <implementation.java
class="indenica.monitoring.component.SimpleMonitoringInterfaceImpl" />
    </component>

    <component name="AdaptationInterfaceComponent">
        <implementation.java
class="indenica.adaptation.component.SimpleAdaptationInterfaceImpl" />
        <reference name="platform" target="SimplePlatformComponent" />
        <reference name="repository" target="RepositoryComponent" />
    </component>

       <component name="RepositoryComponent">
        <implementation.java
class="indenica.repository.component.RepositoryImpl" />
        <property name="dbAddress">192.168.56.101</property>
        <property name="dbPort">27017</property>
        <property name="platform">SimplePlatform</property>
        <property name="adminDB">adminDB</property>
    </component>

       <component name="SimplePlatformComponent">
              <implementation.java
class="indenica.sample.SimplePlatformImpl" />
              <reference name="monitoringInterface"
target="MonitoringInterfaceComponent" />
       </component>
</composite>
```

Figure 24: Exemplary Tuscany Runtime Configuration

After saving the composite file in the '`src/main/resources`' directory, the completed infrastructure instance can now be started using:

```
java –jar indenicaInfrastructure.jar \
  indenica.deployment.Launcher <composite_file_name>
```

5 Conclusion

The Service Platform Infrastructure Repository takes a key role in the Virtual Service Platform, providing global access to various types of information with diverse requirements concerning storage and retrieval. In this document we describe the interim version of the INDENICA Service Platform Infrastructure Repository prototype, including documentation on its usage. The key design goals of the repository are scalability, data store abstraction and domain-specific data retrieval possibilities. We present the repository architecture, its main components, along with a set of tools developed to support platform infrastructure configuration using the Service Platform Infrastructure Repository. The demonstrated use case illustrates in detail how the tools are utilized to administer configurations and rules for some core components of the platform, the Monitoring Engine, Monitoring Interface, Adaptation Engine, and Adaptation Interface. The current implementation builds on state-of-the-art technologies (e.g., Tuscany, Esper, MongoDB) that have been carefully evaluated and chosen to fulfil their particular purpose. The prototype in its current form provides a solid basis for extensibility and further integration with other parts of the platform that rely on the storage capabilities of the repository. Future versions of the Service Platform Infrastructure Repository will be more tightly integrated with View-Based Modelling (WP3) and Variability Engineering (WP2) to provide support for their specific needs concerning data formats, consistency checks, data ageing, notifications, and update mechanisms.