Using CORE+ Cognitive Engine

Project name:	Cognitive radio trial environment +
Project's short name:	CORE+
Project number:	81837 (VTT)

Version history:

Version	Date	Status	Author(s)	Remarks
0.1	22.2.13	Draft	Marko Palola	First draft created.
0.2	18.3.2013	Draft	Marko Palola	And operator, new event names for ASA case
0.3	12.4.2013	Draft	Marko Palola	Limiting IF requirements, list of CORE client names
0.4	30.4.2013	Draft	Marko Palola	New decision rule options
0.5	23.9.2013	Draft	Marko Palola	Updated
0.6	18.11.2013	Draft	Marko Palola	Decision Rule state machine
0.7	20.12.2013	Draft	Marko Palola	Updated for v.0.54
0.8	8.10.2014	Draft	Marko Palola	Installing, SA Trial 3 updates, simple decision with HelloWorld application
0.9	15.12.2014	Draft	Marko Palola	Version 0.80 CORE+ trial system

Contents

1 CORE+ Trial environment

Trial environment consist of CORE+ tools and existing systems which can be monitored and changed by the outcome of the decision making.

CORE+ Trial environment consists of

1. External systems: The systems you wish to control and integrate using the CORE+ client and cognitive engine tools.

   For example, the trial environment has been used to interface hardware such LTE base stations, Wi-Fi base stations and various software components such as databases, LTE network management software, and a Mobile IP client application. Only limiting factor is the interfaces the external system can provide.

2. CORE+ clients, which connect to the interface of the external systems and gather information from them and notify the CORE+ cognitive engine about changes in the gathered information. If external systems are possible to configure, the CORE+ client is also able to define a configuration command which can be activated as a result of CORE+ cognitive engine decision making.

   There are CORE+ client source codes available for minimal implementations such as Hello World and ExampleCoreClient.

3. CORE+ Cognitive Engine, in which decision rules can be defined to control the external systems based on the gathered information.

4. CORE+ cognitive manager, which manages a group of CORE+ clients and connects with a CORE+ cognitive engine

This document explains how CORE+ trial system can be set up and how it can be used with a few examples.

2 CORE+ Client

2.1 Hello World

The minimal Hello World –type CORE+ Client, which is sending a hello message to the CORE+ Cognitive Engine is the following:

```
package cognitivemanager;

import java.io.IOException;

import trialsystem.server.decisionengine.Event;
import trialsystem.shared.Names;

public class HelloWorld extends CoreClient {

 HelloWorld() throws IOException {
 }

    @Override
 public void configurationChanged() {
 }

 @Override
 public boolean deliverCurrentStatus() {
    try {
      sendEvent(Event.createEvent(getClientName(),
                Names.EVENT_TEST,          //event name
                Names.EVENT_TYPE_TEST,     //event type
                10,                        //Time-to-live
                "message.content=hello world;"));
    }
    catch(IOException e) {
       return false;
    }
    return true;
 }

 public static void main(String args[]) throws IOException {
    new HelloWorld();
 }
}
```

The above sample code is able to connect with the cognitive engine and deliver “hello world” as new information for decision making. The message type is a test event and its name is also test. The message information is valid duration of 10 seconds for decision making in the CORE+ cognitive engine. The event names and types are used for routing the events to the interested core clients and cognitive engines.

Executing the HelloWorld CORE+ client requires the following configuration file in place in the current directory. File name is `coreclient.txt`.

```
#Core trial core client v0.1 config file
#Wed Feb 08 10:04:58 EET 2012
client_name=TestProducer
register_event_1=TestEvent
register_event_2=no
register_event_3=no
subscribe_event_1=TestEvent
capability_1=no
capability_name_1=handover
capability_valueName_1=away
capability_parameter_1=network.namesubscribe_event_2=no
capability_2=no
capability_name_2=handover
capability_valueName_2=to
capability_parameter_2=network.name
subscribe_event_3=no
use_server=no
server_port=8080
status_interval=5000
debug=no
gui=true
gui_name=Hello World
manager_url=http\://core.willab.fi\:8081/cognitivemanager
```

The most important definitions are

- manager_url, referencing the address of the CORE+ cognitive manager.
- client_name: each CoreClient should be identified with a unique name, name, currently, must be known by the cognitive engine, too. For testing purposes TestProducer can be a default name.
    - Also usable names for testing are Example1, Example2, to Example19 and TestCC1 to TestCC100
- register_event_1 : This declares the type of event that this CORE+ client is producing.
- subscribe_event_1: This declares the type of event that CORE+ client is interested on. CORE+ cognitive engine will supply this type of data as soon as new data is available.

Download this example from here:
http://core.willab.fi/sites/default/files/CORE-HelloWorldExample.zip

You can edit the HelloWorld.java file and compile it

```
javac -d ./ -classpath HelloWorld.jar HelloWorld.java
```

and execute the compiled version by using it before the version in the jar-file.

```
java -classpath ./;HelloWorld.jar cognitivemanager.HelloWorld
```

See also the readme.txt file attached in the zip archive.

2.2 ExampleCoreClient

More advanced CORE+ client is managing a resource – in this example – a timer, which can be started, stopped and reset by the CORE+ cognitive engine.

Download this example from here:
http://core.willab.fi/sites/default/files/CORE-ExampleCoreClient.zip.

3 CORE+ Cognitive Manager

Purpose: Isolates the customer environment and running CORE+ clients from the CORE+ cognitive engine. Serves a group of CORE+ clients.

Current version: 201411 / 0.80
Delivery type: ZIP archive
Maturity: high quality demonstrator
Tested platforms: Java version 7 64bit (Linux/Windows)
Other supported platforms: none
Requirements:

- Java 7 32/64bit (Windows) Java 7 64bit (OpenSDK in Linux)
- An open HTTP incoming port for CORE+ clients (default 8081)

CORE+ cognitive manager is by default run by the side of CORE+ cognitive engine. It is recommended to run a local CORE+ cognitive manager to improve the network data transfer performance.

CORE+ cognitive manager requires a following configuration file: **cognitivemanager.txt**:

```
manager_url=http\://localhost\:8080/core/core/decisionengine
#manager_url=http\://core.willab.fi\:8080/core/core/decisionengine
server_port=8081
```

The file declares the address of the CORE+ cognitive engine we are connectected to.

Server port defines the port we are listening for incoming HTTP connections from the CORE+ cognitive engine.

Download the CORE+ cognitive manager from here:
http://core.willab.fi/sites/default/files/CORE-CognitiveManager.zip

4 CORE+ Cognitive Engine

Purpose: CORE+ cognitive engine collects information from CORE+ clients for decision making, produces new information and commands CORE+ clients to initiate adjustments on external system.

Current version: 201211 / 0.80
Delivery type: ZIP archive
Maturity: demonstrator
Tested platforms: Java version 7 64bit (Linux/Windows)
Other supported platforms: none
Requirements:

- Apache Tomcat7
- Java 7 32/64bit (Windows) Java 7 64bit (OpenSDK in Linux)
- FireFox, Iron or Chrome WWW browser for user interface
  The GWT plugin is asked to be installed the first time.

4.1 Take-up options

There are three ways you can use CORE+ trial environment:

1) Try out of CORE+ trial environment and the CORE+ cognitive engine remotely using examples in this document. You are able to develop and run locally CORE+ clients and edit decision rules remotely.

2) Download and install CORE+ cognitive engine locally or on a cloud service. You are able to develop your own scenario (with CORE+ clients and cognitive engine) without Internet delays.

3) Use Docker to launch the CORE+ services in a Linux system. (upcoming)

4.2 Installing CORE+ cognitive engine

Install required programs:

- Apache Tomcat6/7
- Java 7

Use the default port 8080 for Tomcat (http://host:8080/)

CORE+ cognitive engine can be installed from a ZIP file by extracting it to a Tomcat's default webapps dir `/var/lib/tomcat6/webapps/core/` directory. It uses port 8081 by default.

Restart Tomcat (`/etc/init.d/tomcat6 restart` OR `service tomcat6 restart`)

4.2.1 Verification of the CORE+ server tools installation

To make sure the cognitive engine is running by Tomcat use the following URLs in the WWW browser.

http://127.0.0.1:8080/core/eventsService : The correct answer from this link is “HTTP method GET is not supported by this URL”.

http://127.0.0.1:8080/core/core/decisionengine : The correct answer from this link is “CORE Trial decision engine” with some version info.

The CORE+ Trial user interface answers from http://127.0.0.1:8080/core/ or http://127.0.0.1:8080/core/index.html

The cognitive manager answers from http://127.0.0.1:8081/cognitivemanager and displays “CORE project – Cognitive Manager “ with some version info.

4.2.2 Problem solving

The catalina.out file is the log for the tomcat, but it will contain also messages generated from the CORE+ Trial system. In the log a line “Deploying web application directory **core**” indicates the core directory is found and will be processed.

In case the core section of the log contains errors related to eventService or cognitiveEngine such as **Invalid <url-pattern>** definitions. Possible solution is to define Servlet paths as follows (with additional ‘/’ in the beginning and removing core from the decisionengine path).

The servlet paths are defined in the Tomcat’s `webapps/core/WEB-INF/web.xml` file.

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
              http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         version="2.5"
         xmlns="http://java.sun.com/xml/ns/javaee">
  <listener>
    <listener-class>
      trialsystem.server.decisionengine.servlet.DecisionEngineTimer
    </listener-class>
  </listener>

        <servlet>
          <servlet-name>eventsService</servlet-name>
          <servlet-class>
                trialsystem.server.EventsServiceImpl
          </servlet-class>
        </servlet>

        <servlet>
          <servlet-name>decisionengine</servlet-name>
          <servlet-class>
            trialsystem.server.decisionengine.DecisionEngineServlet
          </servlet-class>
              <!-- Load this servlet at server startup time -->
          <load-on-startup>5</load-on-startup>
        </servlet>

        <servlet-mapping>
          <servlet-name>eventsService</servlet-name>
          <url-pattern>/core/eventsService</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
          <servlet-name>decisionengine</servlet-name>
          <url-pattern>/core/core/decisionengine/*</url-pattern>
        </servlet-mapping>

</web-app>
```

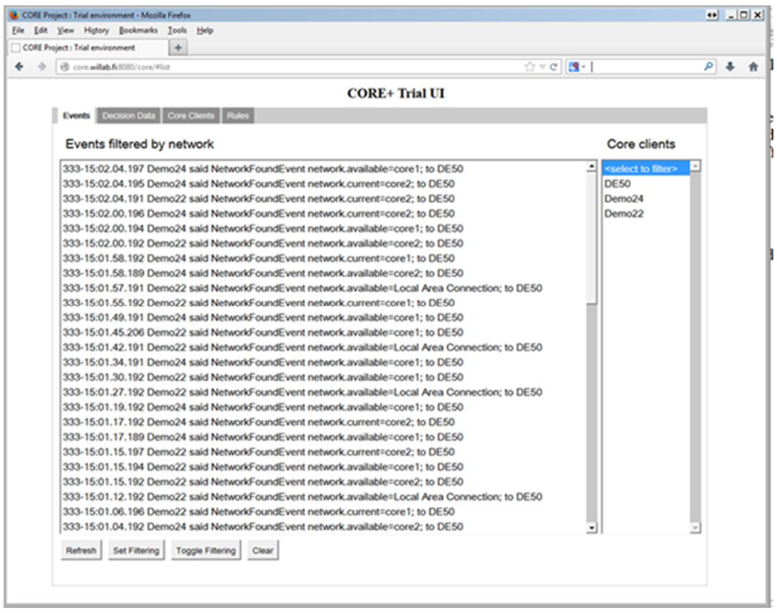
The log messages related to NullPointer in AppClassLoader.loadClass indicate the CORE+ Trial class files are probably compiled using different version of JDK. This can be corrected by installing a JDK 1.7 or better.

4.3 Using the CORE+ Trial environment

The user interface shows information received by the cognitive engine and outgoing information based on routing and decisions.

4.3.1 Display incoming events

The incoming and outgoing events are show by the Trial UI. Each row contains one event with gathered information. The latest event information is at top.


CORE Project : Trial environment - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CORE Project : Trial environment
core.willab.fi:8080/core/#list
CORE+ Trial UI
Events
Decision Data
Core Clients
Rules
Events filtered by network
Core clients
<select to filter>
DE50
Demo24
Demo22
Refresh
Set Filtering
Toggle Filtering
Clear

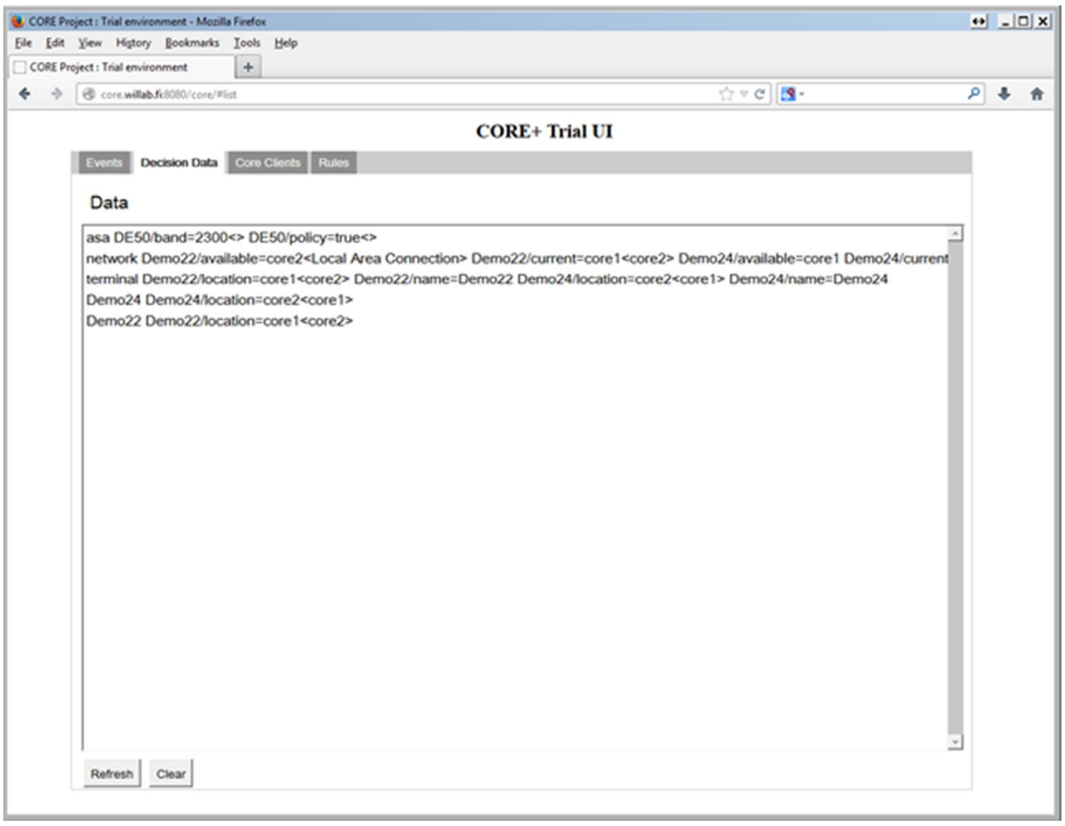
```
333-15:02.04.197 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:02.04.195 Demo24 said NetworkFoundEvent network.current=core2; to DE50
333-15:02.04.191 Demo22 said NetworkFoundEvent network.current=core2; to DE50
333-15:02.00.196 Demo24 said NetworkFoundEvent network.current=core2; to DE50
333-15:02.00.194 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:02.00.192 Demo22 said NetworkFoundEvent network.available=core2; to DE50
333-15:01.58.192 Demo24 said NetworkFoundEvent network.current=core1; to DE50
333-15:01.58.189 Demo24 said NetworkFoundEvent network.available=core2; to DE50
333-15:01.57.191 Demo22 said NetworkFoundEvent network.available=Local Area Connection; to DE50
333-15:01.55.192 Demo22 said NetworkFoundEvent network.current=core1; to DE50
333-15:01.49.191 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.45.206 Demo22 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.42.191 Demo22 said NetworkFoundEvent network.available=Local Area Connection; to DE50
333-15:01.34.191 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.30.192 Demo22 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.27.192 Demo22 said NetworkFoundEvent network.available=Local Area Connection; to DE50
333-15:01.19.192 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.17.192 Demo24 said NetworkFoundEvent network.current=core2; to DE50
333-15:01.17.189 Demo24 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.15.197 Demo22 said NetworkFoundEvent network.current=core2; to DE50
333-15:01.15.194 Demo22 said NetworkFoundEvent network.available=core1; to DE50
333-15:01.15.192 Demo22 said NetworkFoundEvent network.available=core2; to DE50
333-15:01.12.192 Demo22 said NetworkFoundEvent network.available=Local Area Connection; to DE50
333-15:01.06.196 Demo22 said NetworkFoundEvent network.current=core1; to DE50
333-15:01.04.192 Demo24 said NetworkFoundEvent network.available=core2; to DE50
```

The line “333-15:02.04.195 Demo24 said NetworkFoundEvent network.current=core1; to DE50” shows an incoming event details. The receiver is DE50, which is the name of CORE+ cognitive engine. Demo24 is the name of the CORE+ client that sent the event. Information “network.current=core2;” is a collected data and can be used for decision making.

Events can be filtered by clicking one of the producer’s name in the Core client-list or by using the set filtering button below.

4.3.2 Decision data

Decision data shows what CORE+ cognitive engine is learned from the external systems


CORE Project : Trial environment - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CORE Project : Trial environment
core.willab.fi:8080/core/#list
CORE+ Trial UI
Events
Decision Data
Core Clients
Rules
Data
Refresh
Clear

```
asa DE50/band=2300<> DE50/policy=true<>
network Demo22/available=core2<Local Area Connection> Demo22/current=core1<core2> Demo24/available=core1 Demo24/current
terminal Demo22/location=core1<core2> Demo22/name=Demo22 Demo24/location=core2<core1> Demo24/name=Demo24
Demo24 Demo24/location=core2<core1>
Demo22 Demo22/location=core1<core2>
```

The displayed data is currently used for the decision making. The collected data is shown in separate lines which each show the parameters and values related to a component and the CORE+ client that produced the data. For example in a line containing:

```
network Demo22/available=core2<Local Area Connection> Demo22/current =
core1<core2>
```

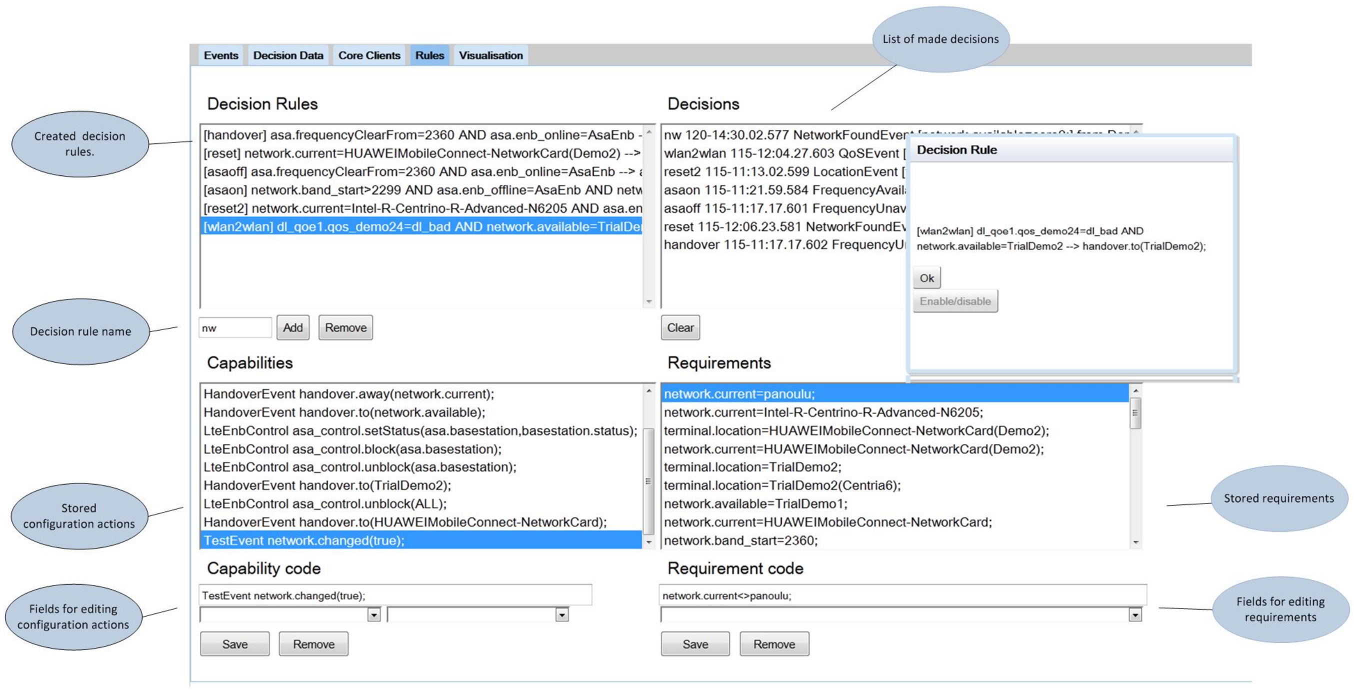
It shows, the Demo22 CORE+ client has delivered values for parameters available and current information for the network component. The previous values of parameters are between the <> characters.

4.3.3 Making simple decisions

The Rules view is used to edit the decision making.

In the figure, user has edited a new IF requirement code for the current network value "network.current<>panoulu;". New IF rule is stored to be available for decision rules.

The IF requirement can be combined with the decision rule name and selected configuration action as a new decision rule, which is then processed by the cognitive engine as long as the rule is visible in the Decision Rules list.



Decision Rules

```
[handover] asa.frequencyClearFrom=2360 AND asa.enb_online=AsaEnb -
[reset] network.current=HUAWEIMobileConnect-NetworkCard(Demo2) -->
[asaoff] asa.frequencyClearFrom=2360 AND asa.enb_online=AsaEnb --> a
[asaon] network.band_start>2299 AND asa.enb_offline=AsaEnb AND netw
[reset2] network.current=Intel-R-Centrino-R-Advanced-N6205 AND asa.en
[wlan2wlan] dl_qoe1.qos_demo24=dl_bad AND network.available=TrialDe
```

Decision rule name: nw

Decisions

```
nw 120-14:30.02.577 NetworkFoundEven
wlan2wlan 115-12:04.27.603 QoSEvent [
reset2 115-11:13.02.599 LocationEvent [
asaon 115-11:21.59.584 FrequencyAvail
asaoff 115-11:17.17.601 FrequencyUnav
reset 115-12:06.23.581 NetworkFoundEv
handover 115-11:17.17.602 FrequencyU
```

Decision Rule

```
[wlan2wlan] dl_qoe1.qos_demo24=dl_bad AND
network.available=TrialDemo2 --> handover.to(TrialDemo2);
```

Capabilities

```
HandoverEvent handover.away(network.current);
HandoverEvent handover.to(network.available);
LteEnbControl asa_control.setStatus(asa.basestation,basestation.status);
LteEnbControl asa_control.block(asa.basestation);
LteEnbControl asa_control.unblock(asa.basestation);
HandoverEvent handover.to(TrialDemo2);
LteEnbControl asa_control.unblock(ALL);
HandoverEvent handover.to(HUAWEIMobileConnect-NetworkCard);
TestEvent network.changed(true);
```

Requirements

```
network.current=panoulu;
network.current=Intel-R-Centrino-R-Advanced-N6205;
terminal.location=HUAWEIMobileConnect-NetworkCard(Demo2);
network.current=HUAWEIMobileConnect-NetworkCard(Demo2);
terminal.location=TrialDemo2;
terminal.location=TrialDemo2(Centria6);
network.available=TrialDemo1;
network.current=HUAWEIMobileConnect-NetworkCard;
network.band_start=2360;
```

Capability code

```
TestEvent network.changed(true);
```

Requirement code

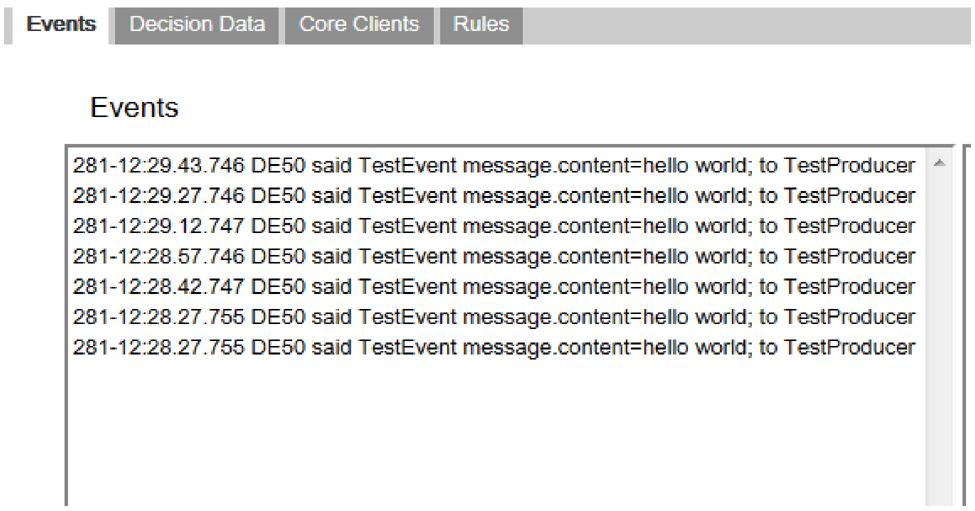
```
network.current<>panoulu;
```

Other operators are

< : smaller than : for example "throughput.value<4000;"
= equals : for example "throughput.value=4000;"
<> or != not equal

The operators work also for textual strings.

Simple decision with HelloWorld example

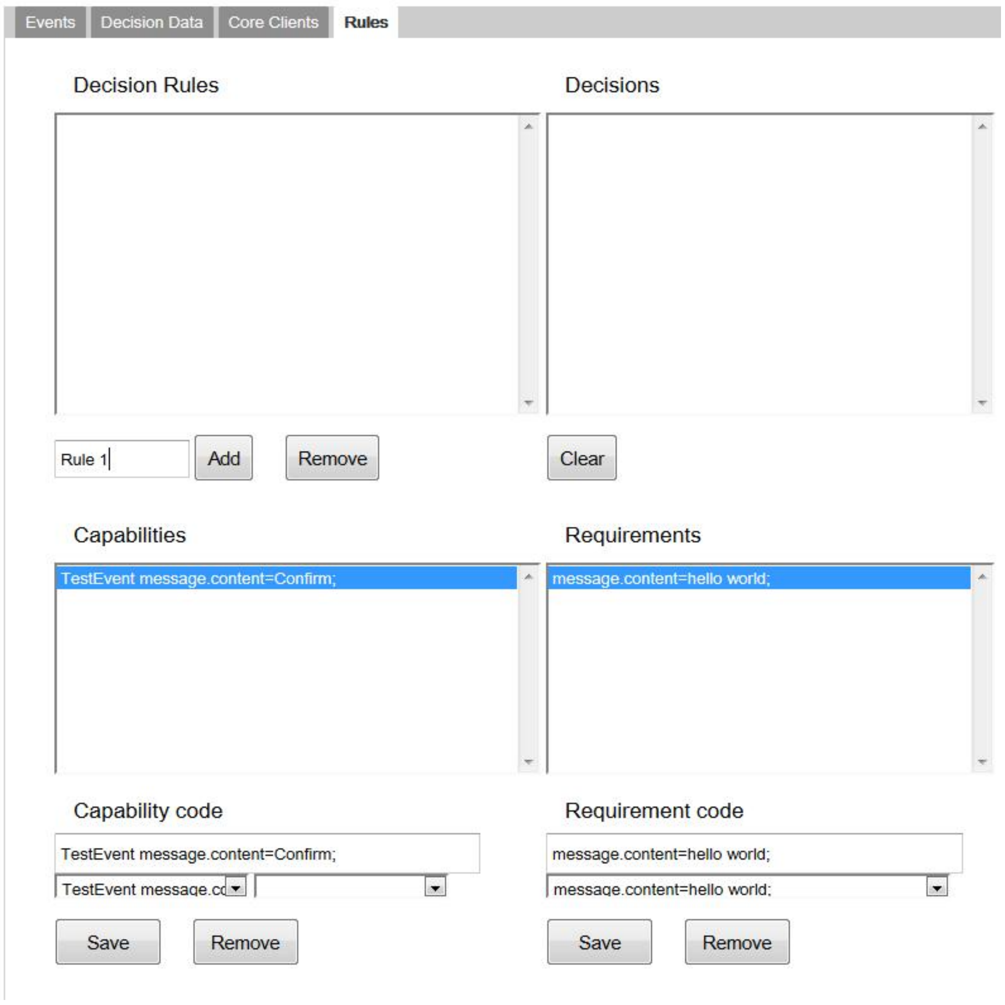


Events

```
281-12:29.43.746 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:29.27.746 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:29.12.747 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:28.57.746 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:28.42.747 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:28.27.755 DE50 said TestEvent message.content=hello world; to TestProducer
281-12:28.27.755 DE50 said TestEvent message.content=hello world; to TestProducer
```

In the event view. The HelloWorld application sis delivering the message.content periodically.

To set up a simple rule switch to Rules tab



Rule 1

TestEvent message.content=Confirm;

message.content=hello world;

TestEvent message.content=Confirm;

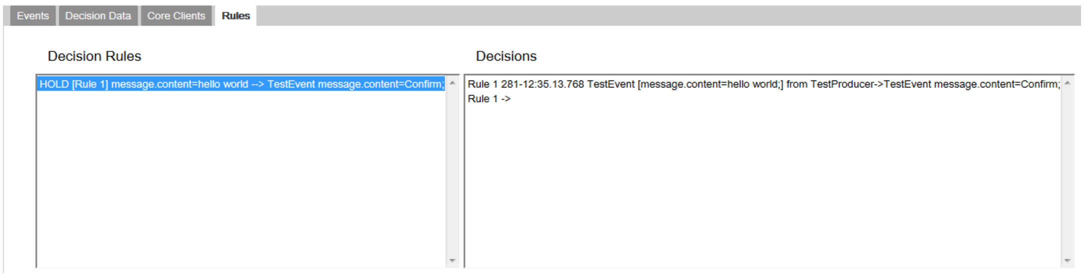
TestEvent message.co

message.content=hello world;

message.content=hello world;

Create a new capability code, save it. Similarly create a comparison requirement for message.content value. In this case, the requirement expects to see “Hello world” text as content.

Give a rule name and add a new decision rule. It should appear in the Decision Rules tab. After a while, as soon as the rule activates, it will trigger and the Confirm message is sent to TestEvent listeners.



HOLD [Rule 1] message.content=hello world --> TestEvent message.content=Confirm;

Rule 1 281-12:35.13.768 TestEvent [message.content=hello world;] from TestProducer->TestEvent message.content=Confirm;
Rule 1 ->

Since the HelloWorld application is also a listener, it should receive the Confim content and call the startCapability method.

```
281-12:36.43.364 CC 1   Begin capability TestEvent message.content=Confirm;
281-12:36.43.364 CC 1   Finishing capability TestEvent message.content=Confirm;
```

Adding multiple requirements into a single decision rule: AND

AND is now supported to make multiple comparison before the decision rules is activated.

Adding a new if sentence to an existing rule requires that the rule name is set to the same and the same capability is selected than is used by an existing decision rule. In multiple requirement rules with AND operator – all the requirements must be true before the capability is activated.

If the decision rule name is different upon storing a rule, the rule is created as a new rule.

In case the decision rule name is the same and the capability is different, the existing rule is replaced.

Rules can be removed by selecting it and pressing "Remove rule" button and both decision rules and decisions can be opened in a separate window.

Making decisions based on data from a specific CORE+ client

Sometimes it is necessary to make decisions based on data sent by a certain CORE+ client. For example in cases you have the same CORE+ client running in different places, they deliver similar data, but they will use different CORE+ client names.

The IF requirement can be limited to a single CORE+ client by using the following notation:

```
network.current=core1(Demo23);
```

In the example, the network information send by a CORE+ client called Demo23 is used in the decision making. Without specification any CORE+ client sending network.current=core1; can be used to activate a decision rule.

4.3.4 Enabling adjustments in CORE+ clients

Before the decision rule can be stored a capability must be defined.

The capabilities / configuration actions can be modified similarly online and these can be defined by CORE+ clients in their configuration.

The CORE+ client will receive the capability/adjustment command by the following method

```
/**
   * Starts capability for the coreclient.
    *
   * @param command the capability command
   */
 public void startCapability(String command) {
 }
```

It is left to the CORE+ client developer to implement the startCapability-method for their needs. A good practise is to call method completeCapability() as soon as the startCapability or a worker thread is finished.

Lets assume the CORE+ client subscribes a TestEvent and ties a simple command handover.start() to it.

The configuration file should have the following:

```
subscribe_event_1=HandoverEvent
capability_1=yes
capability_name_1=handover
capability_valueName_1=start
```

The CORE+ client communicates the capability automatically to the cognitive engine and it should be selectable in the rules view. You should see "`handover.start();`" there.

You can define a similar capability in the rules view by entering a capability code without parameters:

```
HandoverEvent handover.start();
```

Note that the semicolon is required.

4.3.5 Configuration action that uses the collected data for decision making

Cognitive engine is able to use previously gathered data for decision making as long as data has not expired.

A capability can be defined so that it needs a parameter. The purpose of the parameter is to identify which collected data will replace the parameter at the moment of decision making.

The capability code format for declaring a parameter is

```
HandoverEvent/network.current;handover.away(network.current);
```

Lets assume some CORE+ client has send data containing `network.current=wlan;` to the CORE+ cognitive engine. A decision made with above capability will result the command `HandoverEvent handover.away(wlan);` will be entered to the `startCapability()` method of all subscribed CORE+ clients.

The easy way of declaring capabilities with parameters are via the CORE+ client configuration file.

In the previous example will will add a new row, which defines the parameter:

```
subscribe_event_1=HandoverEvent
capability_1=yes
capability_name_1=handover
capability_valueName_1=start
capability_parameter_1=network.name
```

This will result a capability command with a parameter

```
HandoverEvent/network.name;handover.start(network.name);
```

The network.name will be replaced by the actual value found from the cognitive engine.

4.3.6 Configuration action that creates new data

A simple capability command is also supported, for declaring new data as a result of the decision. New data will be routed to CORE+ clients and other CORE+ cognitive engines based on their subscriptions.

Format for a capability that creates a new data is

```
EventName ComponentName.ValueName=value;
```

For example:
```
NetworkEvent network.load=high;
NetworkEvent network.load=14;
```

The created data also is available for decision making.

4.4 Delaying decision making by using requirement time information

Upcoming feature

4.5 Using a state machine in decisions

Sometimes it is necessary to decisions that are valid over a long period of time and usually some other decisions are made only during that time. Cognitive engine uses an internal state machine that can be used organise multiple decisions easily. There is an associated state machine for each decision rule that has four states which can be used in the decision making.

For example, a decision rule that is named as “Asa evacuation” is presented a data parameter called “state.asaevacuation”, which can have the following states and transitions:

```
1) Initial state:        state.asaevacuation=false
2) Decision state:       state.asaevacuation=started
3) Active state:         state.asaevacuation=true
4) Ending state          state.asaevacuation=completed
```

The state is changed from the initial state to the decision state in two cases:
a) Automatically at the moment the decision rule is processed to be true.
b) Capability command `ControlEvent state.start(asaevacuation);` is used.

The state is changed from Decision state to Active state automatically after the decision state is processed by the cognitive engine.

The active state is held until
a) Capability command ControlEvent `state.stop(asaevacuation);` is used.
b) A long time has passed and "Asa evacuation" decision rule is true again which results entering the decision state again.

After the `state.stop` command is used the Ending state is held until the cognitive engine processes the decision rules with "`state.asaevacuation=completed;`"

The purpose of the autonomous transition from the decision state and the ending state is to make possible only one decision based on the state change. This can be used in cases the decision is preferred to happen only once for example to issue command to shutdown a basestation, while during the initial and active states the decisions are repeated on a regular interval if the conditions are not changing.

In the decision rules, you can use the state data of all decision rules in the requirements using all comparison operators.

5 Available names for CORE+ Clients

Each CORE+ Client should have a unique name that is used for routing events correctly between CORE+ Clients and CORE+ Cognitive Engines based on their event subscriptions.

Usable names for testing are

- o Example1, Example2 to Example19 and TestCC1 to TestCC100

Reserved CORE+ client names are:

- CWC1 to CWC10 are reserved for University of Oulu
- Centria1 to Centria10 are reserved for Centria.
- PC1 to PC20 are reserved for VTT
- Demo1-Demo30 and Qosmet1-Qosmet30 are reserved for common demonstrators.
- AsaController1-5, AsaRepository1-5 and AsaEnbController1-5 are reserved for the ASA trials.
- Visualiser1-Visualiser10 are available for visualisation purposes

Adding new names into the system, requires, at the moment, recompilation of the Cognitive Engine and clients.

6 Available event names

The following list contains the data event types and event names that are available in the Names.class. These are needed by CORE+ client when creating new Event objects to be delivered to cognitive engine.

```
//event types
  public static final String EVENT_TYPE_TEST = "TestEvents";
  public static final String EVENT_TYPE_HANDOVER = "Handover";
  public static final String EVENT_TYPE_CONTROL = "Control";
  public static final String EVENT_TYPE_QOS = "QoS";
  public static final String EVENT_TYPE_LOCATION = "Location";
  public static final String EVENT_TYPE_NETWORK_ACCESS = "Access";
  public static final String EVENT_TYPE_WARP = "Warp";
  public static final String EVENT_TYPE_RESOURCE = "Resource";
  public static final String EVENT_TYPE_LOAD = "Load";

  //Event names
  public static final String EVENT_TEST = "TestEvent";

  public static final String CAPABILITY_NOTIFICATION = "Capability";
  public static final String REQUIREMENT_NOTIFICATION = "Requirement";

  public static final String EVENT_QOS = "QoSEvent";
  public static final String EVENT_LOCATION = "LocationEvent";

  public static final String EVENT_NETWORK_FOUND = "NetworkFoundEvent";

  public static final String EVENT_HANDOVER = "HandoverEvent";
  public static final String EVENT_HANDOVER_LTE = "HandoverLteEvent";
  public static final String EVENT_HANDOVER_MOBILE = "HandoverMobileEvent";
  public static final String EVENT_HANDOVER_FIXED = "HandoverFixedEvent";
  public static final String EVENT_HANDOVER_WLAN = "HandoverWlanEvent";
  public static final String EVENT_HANDOVER_BRONZE_USER = "HandoverBronzeEvent";
  public static final String EVENT_HANDOVER_GOLD_USER = "HandoverGoldEvent";
  public static final String EVENT_RESERVE_CHANNEL = "ReserveEvent";
  public static final String EVENT_OCCUPIED_CHANNEL = "OccupiedEvent";
  public static final String EVENT_CHANNEL_OCCUPIER = "OccupierEvent";
  public static final String EVENT_CLIENT_COUNT = "ClientCountEvent";
  public static final String EVENT_TYPE_RADIO = "Radio";
  public static final String EVENT_RADIO_DATA = "RadioData";
  public static final String EVENT_RADIO_SET  = "RadioSet";
  public static final String EVENT_LTE_CONTROL  = "LteControl";
  public static final String EVENT_TYPE_ASA = "ASA";
  public static final String EVENT_POLICY = "PolicyEvent";
  public static final String EVENT_INCUMBENT = "IncumbentEvent";
  public static final String EVENT_EXCEPTION = "ExceptionEvent";
  public static final String EVENT_LTE_DONGLE_CONTROL  = "LteDongleControl";
  public static final String EVENT_LTE_ENB_CONTROL  = "LteEnbControl";
  public static final String EVENT_LTE_ENB_STATUS = "LteEnbStatus";
  public static final String EVENT_EVACUATION  = "EvacuationEvent";
  public static final String EVENT_ACTIVATION  = "ActivationEvent";
  public static final String EVENT_ASA_CONTROL  = "AsaControlEvent";
  public static final String EVENT_ASA_STATUS  = "AsaStatusEvent";
  public static final String EVENT_USER_CONTROL  = "UserControlEvent";
  public static final String EVENT_NETWORK_PRIORITY = "NetworkPriorityEvent";
  public static final String EVENT_BASESTATION = "BasestationEvent";
  public static final String EVENT_BASESTATION_FOUND = "BasestationFoundEvent";
  public static final String EVENT_SECTOR = "SectorEvent";
  public static final String EVENT_TYPE_TIMER = "Timer";
  public static final String EVENT_TIMER_CONTROL = "TimerControl";
  public static final String EVENT_TIMER_DATA  = "TimerData";
  public static final String FREQUENCY_AVAILABLE = "FrequencyAvailable";
  public static final String FREQUENCY_UNAVAILABLE = "FrequencyUnavailable";
  public static final String FREQUENCY_CONTROL = "FrequencyControl";
  public static final String FREQUENCY_STATUS = "FrequencyStatus";
  public static final String EVENT_FREQUENCY = "FrequencyEvent";
  public static final String EVENT_VISUALISATION = "VisualisationEvent";
  public static final String EVENT_STATE = "StateEvent";
  public static final String EVENT_LOAD = "LoadEvent";
```

New data types and names can be added for different purposes. Currently, it requires building a new version of the CORE+ Cognitive Engine.

7 More information

- See http://core.willab.fi site
- These instructions how to begin:

How to use CORE+ cognitive engine:
http://core.willab.fi/sites/default/files/CORE-using_cognitive_engine.pdf

CORE+ Cognitive Engine for Apache Tomcat
http://core.willab.fi/sites/default/files/CORE-CE.zip