Executive Order Spy System

ARDUINO TRON

AI–IoT Internet of Things Drools-BPM

The AI-IoT Drools-BPM Design Architecture

Arduino Tron ESP8266 MQTT Telemetry Transport IoT / (M2M) Machine-to-Machine

Version: 1.30
Published: 01/09/2019

FOR MORE INFORMATION CONTACT:

Steven Woodward
Telephone: (770) 998-3900
Email: info@iotbpm.com
Arduino Tron AI-IoTBPM – Artificial Intelligent Internet of Things http://www.iotbpm.com

Custom Software Development – Executive Order Corporation

www.executiveordercorp.com

Executive Order Corporation - We make Things Smart

Executive Order Corp is a leading provider of technology that helps global companies design, develop, deploy, and integrate software applications

Media Contact
Company Name: Executive Order Corporation
Contact Person: Steven Woodward
Email: info@iotbpm.com
Phone: (770) 998-3900
Website: www.iotbpm.com

JBoss Rules

Arduino Tron AI-IoT-Internet of Things Drools-jBPM Development

DOCUMENT REVISION HISTORY

List changes between each template release.
Increment Release Number by 1 is only between published versions.
(Update the Page Heading when published)

Version #	Date	Revised By	Description of Changes
0.5	3/04/2016	Steven Woodward	First Draft of AI-IoTBPM Internet of Things.
1.02	7/07/2016	Steven Woodward	Add Rules modifications for Use Case requirements.
1.20	9/03/2018	Steven Woodward	Arduino Tron ESP8266 MQTT Telemetry Transport (M2M) Machine-to-Machine.
1.30	01/09/2019	Steven Woodward	AI-IoTBPM Applications using Drools-jBPM Expert System Engine Applications.

SUPPORTING DOCUMENTS

Document Number	Document Name Description
1	Business Requirements Document – Executive Order Document
2	SRS (System Requirements & Use-Cases) Document – EO Document
3	High-level Design Document – Executive Order Document
4	IF-SPECS Documents – Executive Order Document
5	Low Level Design – Application Development – Executive Order Document

Executive Order Corporation © 2011, 2013, 2014-2019 All rights reserved.

This documentation is the proprietary intellectual property of Executive Order Corporation. Any unauthorized use, reproduction, preparation of derivative works, performance, or display of this document, or software represented by this document is strictly prohibited.

Executive Order Corporation and the Executive Order logo design are trademarks and/or service marks of Executive Order Corporation. All other trademarks, service marks, and trade names are owned by their respective companies.

About Executive Order Corporation

We make Things Smart

Executive Order Corp provides custom software built by software professionals. We specialize in IoTBPM (Internet of Things), desktop and web enabled IT solutions for small and large business enterprises. Our professional offerings span business and technology consulting, business application development, mobile messaging solutions, custom web design, e-commerce development, web maintenance, website re-engineering, website optimization for search engine submission, internet marketing hosting solutions for enterprises, GPS, IoTBPM (Internet of Things),remote sensing services and development program architecture of AI-Drools and jBPM (Business Process Management).

TABLE OF CONTENTS

1 Arduino Tron AI-IoT Artificial Intelligent Internet of Things

1.1 Internet of Things Artificial Intelligent Architecture

1.1.1 Internet of Things Artificial Intelligence Conclusion

- An inference model provides a conclusion reached on the basis of evidence and reasoning.

1.1.2 When IoT meets Artificial Intelligence

In the Internet of Things (IoT), as more and more devices and pieces of software interconnect, a great necessity arises for the systems that allow complex situations to be detected in a simple collaborative way by people and devices and be able to react quickly upon detection of these situations.

Internet of Things (IoT) provides lots of telemetry and sensor data; however, the data points by themselves do not provide value unless they can be annualized and turned into actionable, contextualized information. Big data and data visualization techniques allow us to gain new insights by batch-processing and off-line analysis. Real-time sensor data analysis and decision-making are often done manually but to make it scalable, it is preferably automated. Artificial Intelligence (AI) provides us with the framework and tools to go beyond trivial real-time decision and automation use cases for IoT.

With AI-IoTBPM (Artificial Intelligence – Internet of Things) it is important to understand the difference and relationship between big data and real-time event reasoning, known as temporal reasoning. Big data analysis of sensor data retrieved from many IoTBPM devices provides statistical information on particular components and data points. Decision making will allow deciding whether there is a need for maintenance of one particular component. With temporal reasoning, IoT sensors provide information that Drools AI is acted on immediately. For example; in Drools AI-IoT, judging impact avoidance of a vehicle and making course adjustments is an example of AI temporal reasoning or a rational agent.

The AI-IoTBPM rational agent is a central concept in artificial intelligence. An agent is something that perceives its environment through sensors and acts upon that environment via actuators, servos or motors. For example, a robot may rely on cameras as sensors and act on its environment via motors.

A rational agent is an agent that acts, and that does 'the right thing.' The right thing depends on the performance criterion defined for an agent, but also on an agent's prior knowledge of the environment, the sequence of observations the agent has made in the past and the choice of actions that an agent can perform. The AI BRMS Drools itself is the heart of the agent that computes, and reasons based on the available data and its knowledge of the IoT sensors on the environment.

1.1.3 AI Patterns in STREAM or CLOUD Mode

Drools AI-IoTBPM patterns behave differently in STREAM mode when compared to CLOUD mode. In CLOUD mode, the engine assumes that all facts and events are known in advance (there is no concept of the flow of time) therefore, AI patterns are evaluated immediately.

When running in STREAM mode, patterns with temporal constraints require the engine to wait for an event to occur before activating the rule. The time period is automatically calculated by the engine and events are considered immutable state changes, the results of which will fire a rule.

Real-time sensor data analysis and decision-making are often done manually but to make it scalable, it is preferably automated. AI provides us with the framework and tools to go beyond trivial real-time decision and automation use cases for IoT.

Executive Order Corp has developed both a CLOUD-based and a STREAM-based architecture that observes its environment via IoT defined sensors and acts on its environment through AI BRMS Drools-jBPM software, arriving at conclusions reached on the basis of evidence and reasoning.

Executive Order Corp has developed an IoTBPM platform that uses concepts of AI and applied those to the use case of smarter decision making in IoT.

"If a machine thinks, then a machine can do." – Steven Woodward

"It's not you interacting with the machine; it's the machine interacting with you." – Steven Woodward

2 Arduino Tron AI-IoT Internet of Things Tiered Design Architecture

2.1 AI-Artificial Intelligent IoT-Internet of Things

2.1.1 Advantages of an IoTBPM Artificial Intelligent Rules Engine

The Internet of Things (IoT) is just the flowing of data between devices. It is inert and not as powerful as data flowing between intelligent devices (things) that can decide for themselves. As we have said in the opening "If a machine thinks, then a machine can do."

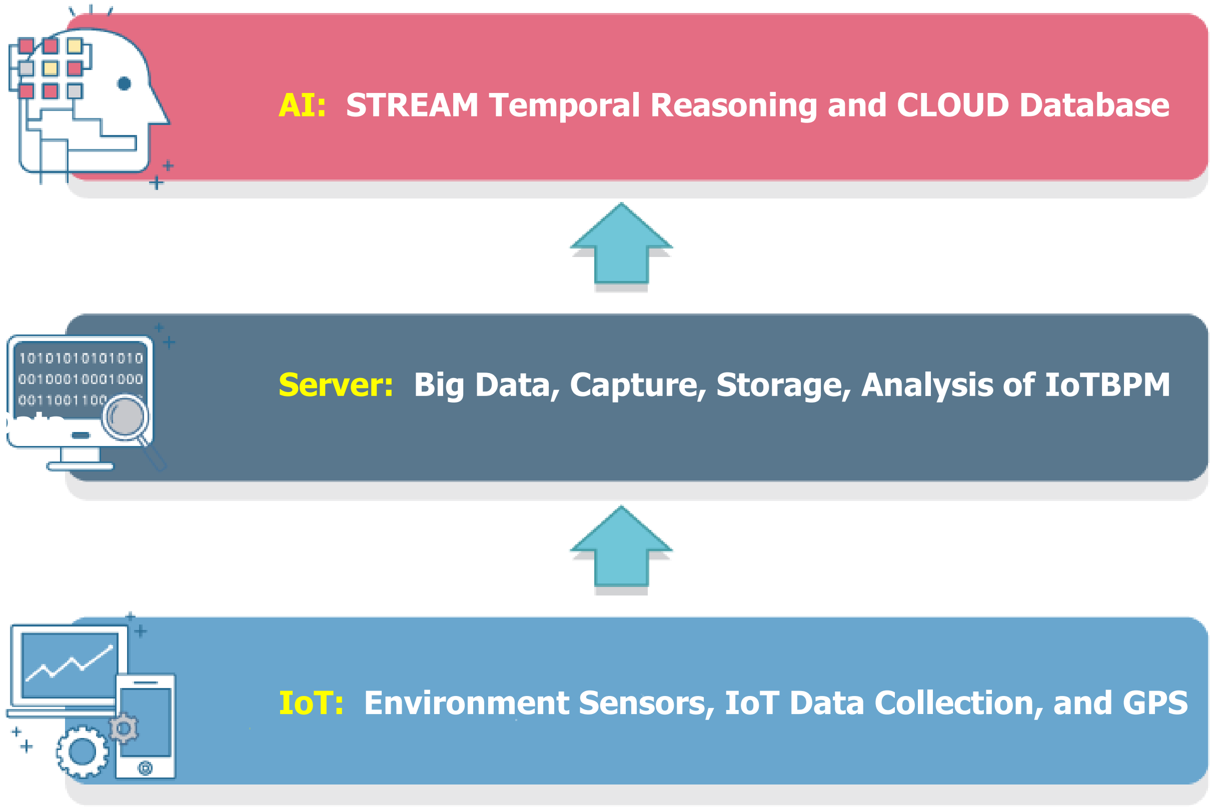
The billions of things that fall under the domain of IoTBPM produce massive volumes of data, and this is where the greatest potential for AI-IoTBPM (Artificial Intelligence - Internet of Things) lies. Tapping into those data streams allows businesses, government and industry to improve manageability and ease of use for consumers, consumer safety and awareness, and enable quantum leaps in the level and quality of products and services in our lives.

The definition of Artificial Intelligence (AI) is the intelligence exhibited by machines or software. Arduino Tron AI-IoT is enabling what have previously been "dumb" devices to add new layers of functionality and access creating the basis for smart homes, smart cars, and smart manufacturing. It's not just about automatically turning on an air conditioner. The data collected, combined with Arduino Tron AI-IoTBPM makes life easier with intelligent automation, predictive analytics, and proactive intervention.

In an IoTBPM scenario, AI can help companies in finding the groups of highly meaningful data and analyze trends/patterns for better decision making. Think about a car that never breaks down because each component is monitored and replaced before the MTBF (mean time between failures). The specific component failure is analyzed, and each replacement component is an improvement on the last.

These are just a few promising applications of AI in IoT. The potential for highly individualized services are endless and dramatically changes the way people live. Executive Order Arduino Tron provides us the three-tiered architecture to provide a complete AI-IoTBPM system.

- **AI (Drools-JBPM):** AI-IoTBPM the Internet of Things Drools-jBPM Expert System.
- **Server (EOSpy):** Sensor Processor live map GPS Tracking, Analysis of IoT Data
- **IoT (Arduino Tron):** Android App / Arduino Sensors, IoT Data Collection, and GPS


AI: STREAM Temporal Reasoning and CLOUD Database
Server: Big Data, Capture, Storage, Analysis of IoTBPM
IoT: Environment Sensors, IoT Data Collection, and GPS

2.2 Arduino Tron IoT-Internet of Things Tiered Design Overview

2.2.1 AI (Drools-jBPM) Tier

Executive Order Corp has developed a three-tiered approach to AI-IoT. The Arduino Tron AI tier is both a CLOUD-based and a STREAM-based architecture that observes its environment via IoTBPM defined sensors and acts on its environment through AI BRMS Drools software, arriving at conclusions reached on the basis of evidence and reasoning.

Executive Order Corporation has developed an IoTBPM platform that uses concepts of AI and applied those to the use case of smarter decision making in IoT.

AI-IoTBPM patterns behave differently in STREAM mode when compared to CLOUD mode. In CLOUD mode, the engine assumes that all facts and events are known in advance (there is no concept of the flow of time) therefore, AI patterns are evaluated immediately.

When running in STREAM mode, patterns with temporal constraints require the engine to wait for an event to occur before activating the rule. The time period is automatically calculated by the engine and events are considered immutable state changes, the results of which will fire a rule(s).

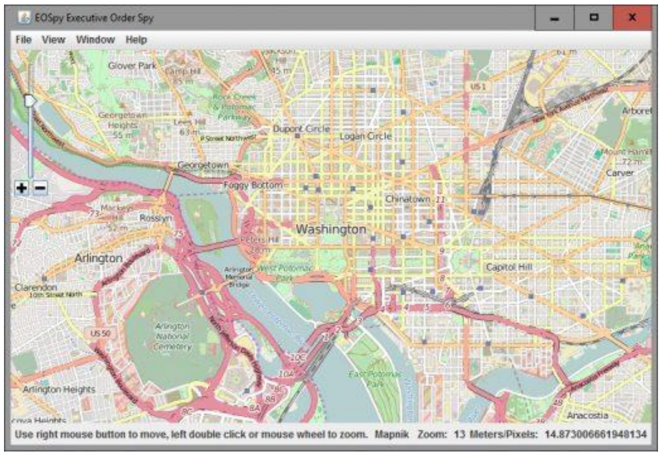
Real-time sensor data analysis and decision-making are often done manually but to make it scalable, it is preferably automated. AI provides us with the framework and tools to go beyond trivial real-time decision and automation use cases for IoT.

This is the Cognizant tier where the AI-IoTBPM "Thinking" happens. Consider this: You have an intelligent refrigerator. It knows your milk is bad. It tells you when you open the door. Who cares? It notifies you in a meeting at work. So, what? It knows that you are walking into a supermarket and reminds you inside to pick up milk. Now that's AI-IoTBPM magic. As we said in the opening "It's not you interacting with the machine; it's the machine interacting with you."

2.2.2 EOSpy (Server) Tier

Executive Order Corp provides two middle-tier server solutions. **EOSpy Windows,** a desktop application that can be installed on a windows computer. **EOSpy Server,** a web application (Web app) server application that delivers EOSpy information over the internet through a web browser. **EOSpy Mobile** is an Android mobile application that allows you to access EOSpy Server information remotely.

- **EOSpy-Executive Order Sensor Processor System for Windows**



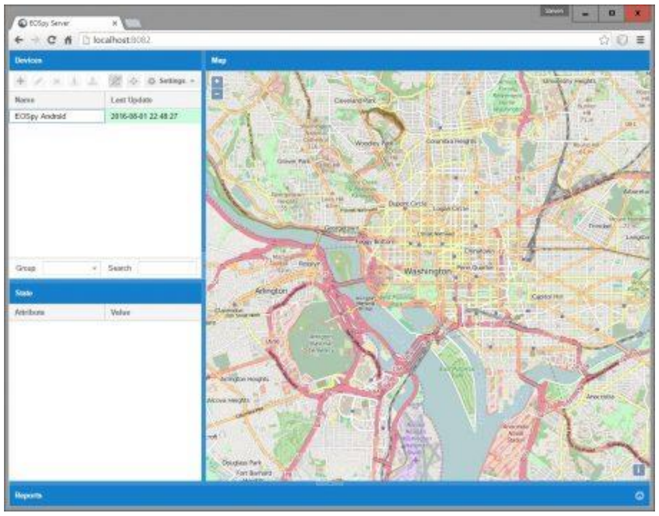
Use right mouse button to move, left double click or mouse wheel to zoom. Mapnik Zoom: 13 Meters/Pixels: 14.873006661948134

EOSpy windows desktop application main control window ties all location and environment monitoring information on one GPS map screen. With live map GPS tracking system that supports more than 90 GPS communication protocols and more than 800 models of GPS tracking devices from popular GPS vendors, EOSpy is designed to support as many tracking devices as possible. Check the EOSpy device list for supported GPS tracking devices. With EOSpy software, setup is a breeze. Just install the EOSpy software on your computer, enter the new GPS device unique identifier and you're ready to go. It is that easy to setup and use.

Executive Order Spy windows desktop application is designed for viewing "Real Time" live GPS tracking information over the internet/mobile cell network that does not require a monthly service subscription.

The EOSpy system provides information about any GPS tracking device directly on your computer. This unique product design allows live GPS tracking and surveillance without a costly monthly, third-party subscription service.

- **EOSpy-Executive Order Sensor Processor System for Server**

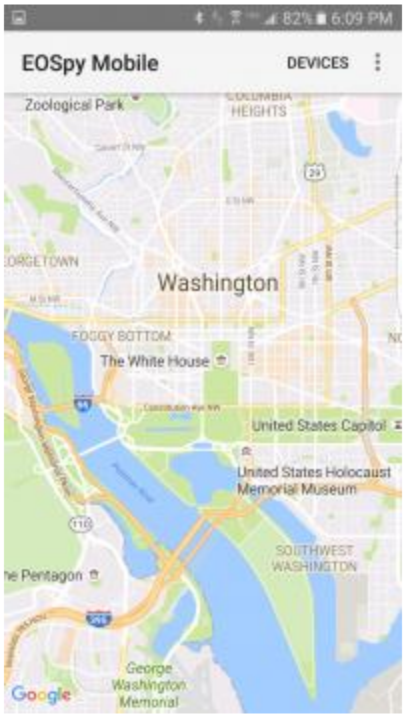


EOSpy server web application (Web app) is a server application that delivers EOSpy GPS information over the internet through a web browser interface. The main control window ties all location and environment monitoring information on one GPS web browser map screen. EOSpy server is designed to support as many tracking devices as possible from popular GPS vendors. Check the EOSpy device list for supported GPS tracking devices. Setup is a breeze with EOSpy server software. Just install the EOSpy server software on your server computer or AWS, enter the new GPS device unique identifier and use any browser to connect to your web server.

EOSpy GPS information via a web browser.

Executive Order Spy server is a web application for viewing "Real Time" live GPS tracking information over the internet/mobile cell network that does not require a monthly service subscription or fee. The EOSpy server provides information about any GPS tracking device directly in your web browser. This unique product design allows live GPS tracking and surveillance without a costly monthly, third-party subscription service.

- **EOSpy-Executive Order Sensor Processor System for Mobile**



EOSpy mobile GPS tracking system – An Android mobile version of EOSpy server. The mobile Android application allows you to use your mobile Android phone to monitor your GPS tracking device remotely.

EOSpy mobile is easy-to-use and helps you stay connected in "Real-Time" with your EOSpy GPS devices and IoT telemetry information. This provides seamless integration between your EOSpy server live-map GPS tracking devices and your mobile phone.

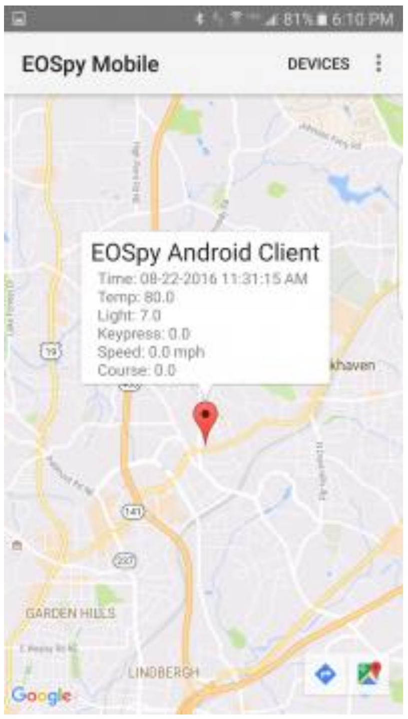
Connect to your EOSpy server – web application (Web app) is a server application that delivers EOSpy GPS information over the mobile interface. The main control window ties all location and environment monitoring information on one mobile GPS map screen. It is designed to support as many tracking devices as possible from popular GPS vendors.

Using EOSpy mobile app for viewing "Real Time" live GPS tracking information over the internet/mobile cell network does not require a monthly service subscription or fee. EOSpy mobile provides information about any GPS GSM tracking device directly on your Android Smart Phone.

EOSpy mobile monitors buildings, vehicles and people from anywhere in the world.

EOSpy mobile shows your GPS tracking devices on your Android phone map and receives ambient temperature, ambient light, and simple button press information on your GPS tracking device.

With the EOSpy mobile GPS tracking system, you can stay connected and informed to what's important to you from your IoT devices. Additionally, the Arduino Tron ESP8266 provides MQTT telemetry transport for (M2M) Machine-to-Machine and IoT.



EOSpy mobile can also receive IR object temperature, humidity sensor, pressure sensor, accelerometer, gyroscope, magnetometer, digital microphone, magnetic sensor, and EOSpy Mobile has Advanced Reverse Geocoding. Geocoding is the transformation process of addresses and places to coordinates and is sometimes called forward geocoding; whereas reverse geocoding uses geographic coordinates to find a description of the location, most typically a postal address or place name. EOSpy mobile will provide you with the address of each GPS tracking device.

EOSpy mobile – The GPS tracking automation and remote monitoring system is a complete package for business or office. Its wireless GPS tracking allows you to monitor your office, systems, personal property, and business from anywhere in the world. Receive remote information from any number of events like when an employee arrives on-site or where a vehicle is located.

EOSpy mobile advanced reverse geocoding – Geocoding is the transformation process of addresses and places to coordinates and is sometimes called forward geocoding; whereas reverse geocoding uses geographic coordinates to find a description of the location, most typically a postal address or place name. EOSpy provides you with the address of each GPS tracking device.

2.2.3 IoT (Arduino Tron / EOSpy) Tier

EOSpy supports over 100 different communications protocols from popular vendors and more than 800 different models and devices. Additionally, EOSpy supports Android application development giving you an IoTBPM platform to develop custom IoTBPM Android applications and devices. EOSpy supports a very extensive IoTBPM Tier layer providing both custom and off-the-shelf solutions. All of which is extensible to build your own custom IoTBPM solution.

- **EOSpy Client**

Executive Order Spy Android Client app allows you to use your mobile phone as a GPS tracking device. It reports location and additional information to EOSpy at selected time intervals. The app also sends remote ambient light intensity, temperature, and humidity information to the EOSpy live map server. Using an internet-connected or mobile cell network connected Android phone or device, streaming location and environment information is at your fingertips.

Remote streaming of additional information is possible like the following: accelerometer, magnetometer, gyroscope, IR temperature, barometer, and equipment status and condition. Monitor buildings, vehicles and people from anywhere in the world. Stay connected and informed to what's important to your company and business.

Executive Order Spy Client - The GPS tracking automation and remote monitoring system is a complete package for home or office. Its wireless GPS tracking allows you to monitor your office, systems, personal property and employees from anywhere in the world. Receive information from any number of events like when an employee arrives on-site, where a vehicle is located, and even receive remote ambient light intensity, temperature, humidity, and additional information.

With remote live GPS map tracking information, you can monitor real-time data anywhere, anytime using your internet-connected computer or tablet. EOSpy supports more than 100 GPS communication protocols and more than 800 models of GPS tracking devices from popular GPS vendors.

- ***Custom Android Apps and Dedicated Devices***

For many commercial business applications, the desire is to use a device for a specific function without the distraction and security threats of an open ecosystem. Custom Android apps allow your business to harness the powerful capabilities of the Android operating system and tailor it to your specific needs. These dedicated device solutions can be custom mounted products or off-the-shelf handheld devices or purpose-built devices fitting your needs.

The EOSpy Client Executive Order Spy Android Client app is an excellent, proven, pre-built platform app that allows you to use a mobile Android device as a GPS tracking device. The EOSpy Android Client app provides a solid platform that your company can build its own custom dedicated app on using an internet-connected or mobile cell network connected Android device for location and environment information. The EOSpy Client provides a stable, proven, customizable application for your business to send telemetry information to your corporate or cloud computers.

- **EOSpy-TI Client**

EOSpy-TI wireless GPS tracking allows you to monitor your office, systems, personal property, and fleet from anywhere in the world. Receive remote information from any number of events like when an employee arrives on-site to where a vehicle is located. The SensorTag Reader reads all sensors from the TI-SensorTag Bluetooth LE device. EOSpy-TI sends GPS position and remote data for the following: ambient temperature, IR object temperature, humidity sensor, pressure sensor, ambient light, gyroscope, magnetometer, digital microphone, magnetic sensor, accelerometer, and simple button press, magnetometer, and additional conditional information.

Executive Order Spy Android Client app allows you to use your mobile phone as a GPS tracking and TI BLE SensorTag sensor device.

- **EOSpy GPS Tracking Devices**

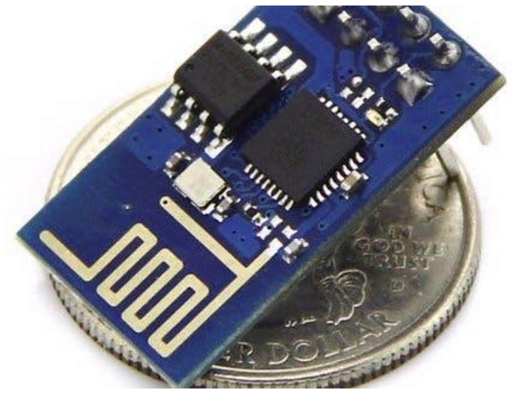
EOSpy is compatible with most popular GPS tracker devices, and many companies make various off-the-shelf GPS tracking devices that can be purchased from a number of vendors.

These GPS tracker devices report location and additional information to the EOSpy server at selected time intervals. Using an internet-connected or mobile cell network, constant information is at your fingertips.

EOSpy supports more than 100 GPS communication protocols and more than 800 models of GPS Tracking Devices from popular GPS vendors. It is designed to support as many tracking devices as possible. Devices that can be installed in your vehicle, miniature devices that can be carried, GPS watches, GPS device collars and property tracking/locators, these GPS Tracking Devices from a huge number of manufacturers are compatible with the EOSpy GPS map system. An extensive list of GPS tracking devices and applications supported by EOSpy Server is available on the compatible devices EOSpy website. This list is the mini real-time GSM / GPRS / GPS trackers Executive Order Spy devices most commonly supported by the EOSpy GPS map system Server.

2.3 Arduino Tron – Micro ESP8266 WiFi NodeMCU MQTT

- **Arduino Tron – Smart Micro-Miniature IoTBPM MQTT Devices**



Executive Order Corporation provides EOSpy (Executive Order Sensor Processor System) Arduino Tron for AI-IoTBPM Internet of Things with Artificial Intelligence, jBPM and Drools Rules Inference AI Architecture for AT-ES Systems. At Executive Order, we make **"Things Smart."**

Arduino Tron – A miniature smart Arduino ESP8266 MQTT telemetry transport device. The Arduino Tron smart micro device is about the size of a quarter and can fit into the smallest spaces in your equipment. The Arduino Tron micro can send alerts on equipment failures, faults or service conditions. This keeps you in constant contact with your equipment, employees, assets & field equipment status providing you with instant alerts and status conditions.

The Arduino Tron is built on the Arduino NodeMCU ESP8266 WiFi MQTT Telemetry Transport Machine-to-Machine (M2M) Internet of Things (IoT) device for IoT. The Arduino Tron AI-IoTBPM Client transmits IoT data using WiFi to the Arduino Tron AI-IoT for Drools-jBPM engine MQTT for alerts and processing.

Arduino Tron AI-IoT: AI-IoT Application using Drools-jBPM Expert System Engine for Arduino Tron and EOSpy AI-IoT Applications designed Arduino ESP8266 MQTT Telemetry Transport WiFi NodeMCU.

- **Custom Hardware, Prototype to Fabrication and Manufacturing**

A major driver in the **IoT** space is the "new hardware movement" where hardware development is becoming an agile process and looking much more like software development. New tools allow for hardware to be developed and shipped with much greater flexibility and with shorter timelines. Executive Order Corp can help your company with complete Electronic Manufacturing Services (EMS).

  - **Printed Circuit Board Design**
  - **Rapid Prototyping**
  - **Industrial Design**
  - **Manufacturing Specifications**
  - **Q/A Testing**
  - **Arduino MQTT (IoT) Software**

Executive Order Corp can custom design and build/program your **Arduino IoT** devices. Executive Order Corp Electronic Manufacturing Services (EMS) are provided by companies that design, assemble, produce, and test electronic components and printed circuit board (PCB) assemblies for Original Equipment Manufacturers (OEMs).

Executive Order Corp EMS service provides a variety of manufacturing services including design, assembly, and testing. Our EMS service is complete from planning, design, develop, to production and board testing. We source the components from our trusted distributors, assemble and test the products.

- **The Internet of Things (IoT)**

At Executive Order Corp,. we are uniquely positioned with the hardware and software experience to help your business capitalize on the **IoT** revolution. We help your company with **IoT.**

  - **Physical Engineering**: Electrical, Component, and Mechanical
  - **Design**: Prototyping, Industrial design, UI / UX
  - **Manufacturing**: Design for Manufacturing, Supply Chain Management (SCM)
  - **Security**: Device, Software, Cloud, and Protocol Specific such as BLE 4.2
  - **Software**: Networking and Infrastructure, Embedded-Systems Programming, Big Data, Machine Learning, Servers, Cloud Computing, Arduino Apps and Web

"**Internet of Things**" is a set of technology that will gradually and sometimes almost imperceptibly begin to affect us in the coming years. Any specific device or application might be small, but the combination of sensor and devices creates significant long-term changes that can both make our lives easier and more informed. The Arduino ESP8266 ESP-01S WiFi Serial Transceiver Module with 4MB Flash for Arduino provides a very inexpensive **IoT MQTT** Telemetry Transport platform.

IoT promises to provide "**smart**" environments (homes, cities, hospitals, schools, stores, offices, etc.) and smart products (cars, trucks, airplanes, trains, buildings, devices, etc.). However, the task of moving beyond "connected" to "smart" IoT devices is daunting. Moving beyond just collecting IoT data and transitioning to leveraging the new wealth of IoTBPM data to improving the smart decision making is the key. Executive Order EOSpy/Arduino Tron AI-IoTBPM helps these IoTBPM devices, environments, and products to self-monitor, self-diagnose and eventually, self-direct.

I tell people the key with AI-IoTBPM is, "If a machine thinks, then a machine can do."
Also, "It's not you interacting with the machine, it's the machine interacting with you."

- **Arduino WiFi MQTT Sensors and IoT Devices**

Sensors provide the data you need to automate processes and gain visibility into your business operations. At Executive Order Corp,. we have extensive expertise in the Industrial Internet of Things (IIoT) and Machine to Machine (M2M) technology that can be what your company needs to increase efficiencies or distributive business models. Additionally, you can add jBPM and Drools Rules AI-ES reasoning to your business models for "**Smart**" IoT device reasoning.

Arduino Tron WiFi provides remote streaming of the following additional information directly to the EOSpy live map server: ambient temperature, IR object temperature, humidity sensor, pressure sensor, ambient light, accelerometer, gyroscope, magnetometer, digital microphone, magnetic sensor, and simple button press, and circuit board-equipment status conditions. Monitor buildings, vehicles, and people from anywhere in the world. Stay connected and informed to what's important to your business.

- **Developing Embedded Systems**

Developing embedded systems and devices firmware requires the ability to work with both hardware and software, observe real-time constraints, account for new and custom designs, and configure or create new operating systems. Executive Order Corp offers a wide range of embedded design services that can transform an idea into a complete full product. This includes the devising of system architecture, board and firmware design, application software development, mechanical design, proto-typing, validation, regulatory certification, and pilot production. Let our experienced professionals support your project and instill the highest level of confidence in your embedded system design.

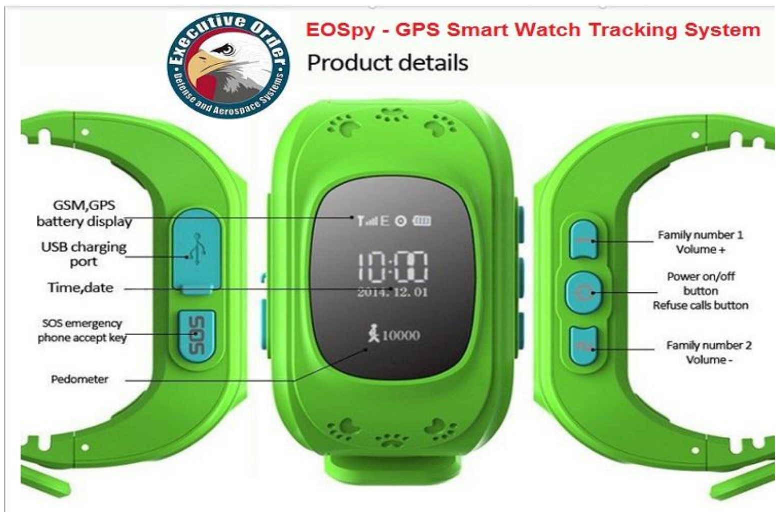
- **Custom Arduino Applications and Dedicated Devices**

For many commercial business applications, the desire is to use a device for a specific function without the distraction and security threats of an open ecosystem. Custom Arduino applications allow your business to harness the powerful capabilities of the Arduino system and tailor it to your specific needs. These dedicated device solutions can be custom mounted products or off-the-shelf handheld devices.

The Arduino Tron is an excellent, proven, pre-build, platform application that allows you to use a mobile Arduino MQTT telemetry transport device. It can report location and additional information to the EOSpy server at selected time intervals. The Arduino Tron application provides a solid platform that your company can build its own custom dedicated application on using an internet-connected or mobile cell phone network connected Arduino device for GPS location, environment, and condition or reporting device (board) error and status information remotely.

In addition to the Arduino NodeMCU sensors: light sensor, pressure sensor, temperature and humidity sensor, the Arduino NodeMCU can receive action commands from EOSpy via HTTP for 7 seg display, relay, buzzer, infrared emitter, LED, motor actuators, gate access, auditable speech, alarms, etc.

- **EOSpy Wearables**



EOSpy Workforce tracking - manage and protect your employees with personal GPS trackers.

Receive information from any number of events like when an employee arrives on-site, where an employee is located and even receive remote alert information.

EOSpy is ready to track employee time and location, from the time an employee arrives on-site until the time they leave.

EOSpy Workforce GPS time tracking adds great value to any organization with benefits for employers and managers. Locate employees in real-time and invoice with confidence with data to support client invoicing and confirmation on employee on-site location.

Wearable devices provide a new range of capabilities related to IoT. Some new functions that wearable IoT devices provide are related to identification, security, and GPS location. Consider an advanced IoT badge that includes biometric capabilities such as fingerprint activation. IoT badges can also include capabilities for location sensing, useful in emergencies to make sure everyone has successfully evacuated the building. An IoT wearable bracelet provides a more reliable indication of location since it is less likely to be left behind.

IoT wearable devices could automatically connect to devices around the home. Perhaps you have a preferred lighting level when watching TV. You could turn on the TV, and your wearable device could help adjust the lighting level from the connected LED lights within the room. An intelligent house could support all these interactions automatically from your IoT wearable device.

AI-IoT Arduino Tron Sensor

The EOSPY AI-IoTBPM Arduino Tron Sensor software allows you to interface and sends MQTT telemetry transport information from your external connected Arduino devices to the EOSpy server. The AI-IoT Arduino Tron Agent software uses a WiFi wireless transceiver interface to stream telemetry information to the EOSpy server for any control module sensors or remote control connected Arduino Tron device.

AI-IoTBPM Arduino Tron Agent

The EOSPY AI-IoTBPM Arduino Tron Agent software interface allows you to send commands with the EOSPY AI-IoTBPM software to control external Arduino connected devices. The AI-IoTBPM Arduino Tron Agent software uses a WiFi wireless transceiver interface to control and interact with module sensors and remote control devices. You can control any device from the EOSpy AI-IoTBPM Arduino Tron Agent software or stream any interface over the WiFi or internet. With the EOSpy AI-IoTBPM Arduino Tron Agent software you can automatically turn on lights, appliances, cameras, and lock/unlock doors from the Drools-jBPM expert system processing model.

Executive Order Corporation provides custom software and hardware development for Arduino MQTT, NodeMCU, (ESP8266 WiFi microcontroller), low cost, smart IoT, WI-FI enabled, Lua 5.1.4 devices.

The Executive Order Corp Arduino Tron ESP8266 MQTT telemetry transport Machine-to-Machine (M2M) Internet of Things software and Arduino Tron MQTT AI-IoTBPM Client using EOSpy AI-IoTBPM Drools-jBPM latest software can be download from the Github website.

Download Arduino Tron IoT BPM from https://github.com/eodas/IoTBPM

3 Arduino Tron AI-IoT Internet of Things Drools-jBPM Architecture

3.1 AI-IoTBPM Internet of Things Rules Design Overview

3.1.1 Why use a Rule Engine and jBPM for IoT Internet of Things?

Some frequently asked questions:

- When should you use jBPM or Rule engine?
- What advantage does jBPM and Rule engine have over hand-coded "if...then" approaches?
- Why should you use jBPM / Rules engine instead of an imperative programming language?

3.1.2 Advantages of a Rule Engine

Below summarizes the key business benefits for using BRMS - BPM, a comprehensive platform for business rules management, business resource optimization and Complex Event Processing (CEP).

• Declarative Programming

Rule engines allow you to say "What to do," not "How to do it."

The key advantage of this point is that using rules can make it easy to express solutions to difficult problems and consequently have those solutions verified. Rules are much easier to read than code.

Rule systems are capable of solving very hard problems providing an explanation of how the solution was arrived at and why each "decision" along the way was made, not so easy with other Artificial Intelligent (AI) Systems like neural networks or the human brain.

• Logic and Data Separation

Your data is in your domain objects; the logic is in the rules. This is fundamentally breaking the OO coupling of data and logic, which can be an advantage or a disadvantage depending on your point of view. The upshot is that the logic can be much easier to maintain when there are changes in the future, as the logic is all laid out in rules. This can be especially true if the logic is cross-domain or multi-domain logic. Instead of the logic being spread across many domain objects or controllers, it can all be organized in one or more discrete rules files.

• Speed and Scalability

The Rete algorithm, the Leaps algorithm, and their descendants such as Drools' ReteOO provide very efficient ways of matching rule patterns to your domain object data. These are especially efficient when you have datasets that change in small portions as the rule engine can remember past matches. These algorithms are battle-proven.

• Centralization of Knowledge

By using rules, you create a repository of knowledge (a knowledge base) which is executable. This means it's a single point of truth, for business policy, for instance. Ideally, rules are so readable that they can also serve as documentation.

• Tool Integration

Tools such as Eclipse provide ways to edit and manage rules and get immediate feedback, validation and content assistance. Auditing and debugging tools are also available.

• Explanation Facility

Rule systems effectively provide an "explanation facility" by being able to log the decisions made by the rule engine along with why the decisions were made.

• Understandable Rules

By creating object models and, optionally, domain-specific languages that model your problem domain you can set yourself up to write rules that are very close to natural language. They lend themselves to logic that is understandable to possibly nontechnical and domain experts.

3.2 IoT Internet of Things Solution Design Considerations

3.2.1 IoTBPM Design Methodology

The following key design methodologies are utilized in the Executive Order Corp AI-IoTBPM application.

Starting in Drools 7, it introduces the **Business Logic Integration Platform** which provides a unified and integrated platform for **Rules, Workflow,** and **Event Processing**. **Drools –** BRMS system with a forward-chaining and backward-chaining inference-based rules engine, allowing fast and reliable evaluation of business rules and complex event processing.

- **Drools –** Allows fast and reliable evaluation of business rules and complex event processing.
  - A rule engine is a fundamental building block to create an expert system which, in artificial intelligence; drools emulates the decision-making ability of a human expert.
  - Drools can reason to a conclusion (infer) beyond what we currently know.
  - An inference model provides a conclusion reached on the basis of evidence and reasoning.
    - We say that Drools emulates the decision-making ability of a human expert.

- **Declarative Programming**

Rule engines allow you to say "What to do," not "How to do it."

The key advantage of this point is that using rules can make it easy to express solutions to very difficult problems and consequently have those solutions verified. Rules are much easier to read than traditional programming code.

- **Logic and Data Separation**

Your data is in your domain objects; the logic is in the rules. This is fundamentally breaking the OO coupling of data and logic, which can be an advantage/disadvantage depending on your point of view. The upshot is that the logic can be much easier to maintain as there are changes in the future, as the logic is all laid out in rules.

This can be especially true if the logic is cross-domain or multi-domain logic. Instead of the logic being spread across many domain objects or controllers, it can all be organized in one or more very distinct rules files co-located across IT infrastructure. With IoT devices by default, they are separated in a Hyper Computing environment. This given by architecture design creates the separation of logic and data.

jBPM is a flexible Business Process Management (BPM) Suite. A business process allows you to model your business goals by describing the steps that need to be executed to achieve those goals, and the order of those goals are depicted using a BPM flow chart. **jBPM -** Flexible Business Process Management suite allowing you to model your business goals by describing the steps that need to be executed to achieve those goals.

- **jBPM –** Allows you to model business goals by describing the steps that need to be executed to achieve those goals and the order of those goals are depicted using a flow chart diagram.

- jBPM is a flexible Business Process Management (BPM) Suite. It allows you to model, execute, and monitor business processes throughout their life cycle through GUI.
- Business Process Models are expressed in a given process modeling language. OMG commissioned the development of the BPMN2.

The BRMS is making the Enterprise Business decisions, and the BPM is executing the Business Goal.

- **Business Rules Engine Methodology**

Provides a mechanism to define, deploy, execute, monitor and maintain the variety and complexity of decision logic that is used by operational systems within an organization. There are two methods of execution for a rule system: Forward-Chaining and Backward-Chaining; systems that implement both are called Hybrid Chaining Systems. Drools provide seamless Hybrid chaining for our IoT systems:

- Forward Chaining is "data-driven" and thus reactionary, with facts being asserted into working memory, which results in one or more rules being concurrently true and scheduled for execution by the Agenda Engine. In short, we start with a fact, it propagates, and we end in a conclusion.
- In Forward Chaining, the system starts from a set of facts, and applies a set of rules, and tries to find a way of using those rules and facts to deduce a conclusion or come up with a suitable course of action. Usually, when a rule is triggered, it is then fired, which means its conclusion adds to the facts database. If the conclusion of the rule that has fired is an action or a recommendation, then the system may cause that action to take place.
- Backward Chaining is "goal-driven," meaning that we start with a conclusion which the engine tries to satisfy. It searches for conclusions that it can satisfy; as sub-goals; these then help satisfy some unknown part of the current goal. It continues this process until either the initial conclusion is proven or there are no more sub-goals.
- In backward chaining, we start from a conclusion, which is the hypothesis we wish to prove, and we aim to show how that conclusion can be reached from the rules and facts in the database. The conclusion we are aiming to prove is called a goal, and so reasoning in this way is known as a derivation query or goal-driven reasoning. Prolog is AI-ES, an example of a Backward Chaining engine.

It is hard to talk specifics without details about each use case, but the general idea is that the more rules and the more complex the rules are, the more benefits the Drools Rules engine gives you. In other words, when the number of rules increases, then the rules engine is orders of magnitude faster.

This happens because increasing the number of rules usually has little to negligible impact on the response time; it has Big O(1) scalability for an increasing number of rules.

- **The IoT STREAM Rules System**

Most of the rules used by an IoT STREAM System are event declaration. To declare a FACT type as an “event” all that is required is to assign the @role metadata tag to the fact type.

The @role metadata tag accepts two possible values:

- **Fact**: (this is the default) Declares that the type is to be handled as a regular fact. This would be a typical CLOUD type of rule.
- **Event**: Declares that the type is to be handled as an event. Every event has an associated timestamp assigned to it.

By default, the timestamp for a given event is read from the session clock and assigned to the event at the time the event is inserted into the working memory. All facts are static and stored in the IoTBPM system production memory as rules or events. The facts that the inference engine matches against are kept in the working memory. Using the CEP pattern of processing the event-driven architecture from the transaction database to the JBoss Drools/Fusion Engine (CEP) the facts are inserted into the IoTBPM working memory when the data collection/transaction occurs.

- **Note:** Defining terms is not the goal of this guide and as so, let's adopt a loose definition that, although not formal, allows us to proceed with a common understanding. So, in the scope of this guide: Event is a record of a significant change of state in the application domain at a given point in time. Events are immutable and can be embellished. A transaction is an event.
- **Event-Driven Architectures**

An Event-Driven Architecture (EDA) is a software architecture pattern promoting the production, detection, consumption of and reaction to events. Building applications and systems around an event-driven architecture allow these applications and systems to be constructed in a manner that facilitates more responsiveness because event-driven systems are, by design, more normalized to unpredictable and asynchronous environments.

Event processing is a method of tracking and analyzing (processing) streams of information (data) about things that happen (events) and deriving a conclusion from them. Complex event processing, or CEP, is event processing that combines data from multiple sources to infer events or patterns that suggest more complicated circumstances. The goal of complex event processing is to identify meaningful events (such as opportunities or threats) and respond to them as quickly as possible.

The core benefits of this architecture approach are that it provides loose coupling of the components, IoTBPM transaction database, and JBoss Drools/Fusion IoT-AI system. A component publishes events about actions that it is executing and transaction database subscribes/listens to these events.

The transaction database subscriber listens for events, stores and inserts the facts into the IoT-AI working memory. An orchestration layer then handles the actual inserting into working memory of only events we are interested in analyzing by the rules inference engine.

These events are notices happening across the various layers of the organization. An event may also be defined as a "change of state," when a measurement exceeds a predefined threshold of time, response, or other value. IoT complex event processing analysts gives the organizations a new way to analyze patterns in real-time and help the business side communicate better with IT and service departments.

After inserted, new events into Drools Working Memory from the data collection ingestion the Event Stream Processing, or ESP passes-off to our complex event processing (CEP), which takes precedence.

ESP-Event Stream Processing is a set of technologies which include event visualization, event databases, event-driven middleware, and our event processing language (Drools).

CEP - Complex Event Processing

CEP is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the event cloud. CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.

CEP deals with the task of processing streams of event data with the goal of identifying the meaningful pattern within those streams, employing techniques such as detection of relationships between multiple events, event correlation, event hierarchies, and other aspects such as causality, consequence, and timing. These are inserted in the rules engine and synthetic facts or events that are conclusions of facts. This is the sequencing of synthetic events from actual events/faces and the conclusions drawn.

CEP allows patterns of simple and ordinary events to be considered to infer that a complex event has occurred. Complex event processing evaluates a confluence of events and then takes action. The events (notable or ordinary) may cross event types and occur over a long period of time. Event across Drools BRMS engine sessions. The event correlation may be causal, temporal, or spatial.

CEP requires the employment of sophisticated event interpreters, event pattern definition and matching, and correlation techniques from the Drools BRMS engine. CEP is commonly used to detect and respond to business anomalies, threats, locations, and opportunities and is well suited for our IoTBPM domain.

3.3 Drools Fusion Engine Solution Design

3.3.1 Design Use Case Methodology

Event processing use cases share several requirements, and goals with business rules use cases.

These overlaps happen both on the business side and the technical side.

On the Business side:

- Business rules are frequently defined based on the occurrence of scenarios triggered by events. Examples could be:
    - On an algorithmic trading application: Take action if the security price increases X% compared to the day opening price where price increases are usually denoted by events on a stock trade application.
    - On a monitoring application: Take action if the temperature on the server room increases X degrees in Y minutes where sensor readings are usually denoted by events.
- Both business rules and event processing queries change frequently and require immediate response for the business to adapt itself to new market conditions, new regulations, and new enterprise policies.

From a technical perspective:

- Both require seamless integration with the enterprise infrastructure and applications, especially on autonomous governance, including, but not limited to, lifecycle management, auditing, security, etc.
- Both have functional requirements like pattern matching and non-functional requirements like response time and query/rule explanation.

In this context, Drools Fusion is the module responsible for adding event processing capabilities into the drools rules platform.

Supporting complex event processing, though, is much more than simply understanding what an event is. CEP scenarios share several common and distinguishing characteristics:

- Usually required to process huge volumes of events, but only a small percentage of the events are of real interest.
- Events are usually immutable since they are a record of state change.
- Usually, the rules and queries on events must run in reactive modes, i.e., react to the detection of event patterns.
- Usually, there are strong temporal relationships between related events.
- Individual events are usually not important. The system is concerned about patterns of related events and their relationships.
- Usually, the system is required to perform composition and aggregation of events.

Based on this general common characteristic, Drools Fusion defined a set of goals to be achieved to support complex event processing appropriately:

- Support events, with their proper semantics, as first-class citizens.
- Allow detection, correlation, aggregation, and composition of events.
- Support processing of streams of events.
- Support temporal constraints in order to model the temporal relationships between events.
- Support sliding windows of interesting events.
- Support a session scoped unified clock.
- Support the required volumes of events for CEP use cases.
- Support to (re)active rules.
- Support adapters for event input into the engine (pipeline).

This list of goals is based on the requirements not covered by Drools expert itself since in a unified platform all features of one module are leveraged by the other modules. Drools Fusion is born with enterprise-grade features like pattern matching that is paramount to a CEP product, but that is already provided by Drools expert. In the same way, all features provided by Drools Fusion are leveraged by Drools flow (and vice-versa) making process management aware of event processing and vice-versa.

Additionally, in some scenarios, you have to discard equal objects (objects of the same type and values) when they are inserted into the working memory to avoid data inconsistency and unnecessary activations. The preferred method of discarding duplicated facts on insertion is an insertion – use of a custom classLoader in a knowledgeAgenda method.

We can have a system of streams where the events are transmitted, and these events can be of the same type but have to be processed in different ways without mixing them in their processing. Drools Fusion can handle this scenario by creating entry points that can be used to integrate these streams with the rules patterns to process the events that are going to arrive from these streams.

3.4 Event Semantics

An *event* is a fact that presents a few distinguishing characteristics:

- **Usually immutable:** Since, by the previously discussed definition, events are a record of a state change in the application domain, i.e., a record of something that already happened, and the past cannot be "changed," events are immutable. This constraint is an important requirement for the development of several optimizations and the specification of the event lifecycle. This does not mean that the Java object representing the event object must be immutable. Quite the contrary. The engine does not enforce immutability of the object model because one of the most common use cases for rules is event data enrichment.
    - **Note**: As a best practice, the application is allowed to populate un-populated event attributes (to enrich the event with inferred data), but already populated attributes should never be changed.
- **Strong temporal constraints:** Rules involving events usually require the correlation of multiple events, especially temporal correlations where events are said to happen at some point in time relative to other events.
- **Managed lifecycle:** Due to their immutable nature and the temporal constraints, events usually will only match other events and facts during a limited window of time, making it possible for the engine to manage the lifecycle of the events automatically. In other words, once an event is inserted into the working memory, it is possible for the engine to find out when an event can no longer match other facts and automatically delete it, releasing its associated memory and resources.
- **Use of sliding windows:** Since all events have timestamps associated to them it is possible to define and use sliding windows over them, allowing the creation of rules on aggregations of values over a period of time. Example: Average of an event value over 60 minutes.

Drools supports the declaration and usage of events with both semantics: **point-in-time** events and **interval-based** events.

- **Note:** A simplistic way to understand the unification of the semantics is to consider a *point-in-time* event as an *interval-based* event whose *duration is zero*.

3.5 Event Processing Modes

Rules engines, in general, have a well-known way of processing data and rules and provide the application with the results. Also, there are not many requirements on how facts should be presented to the rules engine especially because, in general, the processing itself is time independent. That is a good assumption for most scenarios but not for all of them. When the requirements include the processing of real time or near real time events, time becomes an important variable of the reasoning process.

The following sections explain the impact of time on rules reasoning and the two modes provided by Drools for the reasoning process.

3.5.1 Cloud Mode

The CLOUD processing mode is the default processing mode. Users of rules engine are familiar with this mode because it behaves in the same way as any pure forward chaining rules engine, including previous versions of Drools.

When running in CLOUD mode the engine sees all facts in the working memory, does not matter if they are regular facts or events as a whole. There is no notion of flow of time, although events have a timestamp as usual. In other words, although the engine knows that a given event was created, for instance, on January 1st 2018, at 09:35:40.767, it is not possible for the engine to determine how "old" the event is because there is no concept of "now."

In this mode, the engine applies its usual many-to-many pattern matching algorithm using the rules constraints to find the matching tuples, activate, and fire rules as usual.

This mode does not impose any kind of additional requirements on facts. So, for instance:

- There is no notion of time — no requirements clock synchronization.
- There is no requirement on event ordering. The engine looks at the events as an unordered Cloud against which the engine tries to match rules.

On the other hand, since there are no requirements, some benefits are not available either. For instance, in CLOUD mode it is not possible to use sliding windows because sliding windows are based on the concept of "now" and there is no concept of "now" in CLOUD mode.

Since there is no ordering requirement for events, it is not possible for the engine to determine when events can no longer match and as so, there is no automatic life-cycle management for events. i.e., the application must explicitly delete events when they are no longer necessary, in the same way, the application does with regular facts.

Cloud mode is the default execution mode for Drools but, in any case, like any other configuration in Drools, it is possible to change this behavior either by setting a system property using configuration property files or using the API. The corresponding property is:

```
KieBaseConfiguration config = KieServices.Factory.get().newKieBaseConfiguration();
config.setOption( EventProcessingOption.CLOUD );
```

The equivalent property is:

```
drools.eventProcessingMode = cloud
```

3.5.2 Stream Mode

The STREAM processing mode is the mode of choice when the application needs to process streams of events. It adds a few common requirements to the regular processing but enables a whole lot of features that make stream event processing a lot simpler.

The main requirements to use STREAM mode are:

- Events in each stream must be time-ordered. i.e., inside a given stream event that happened first must be inserted first into the engine.
- The engine forces synchronization between streams through the use of the session clock, so, although the application does not need to enforce time ordering between streams, the use of non-time-synchronized streams may result in some unexpected results.

Given that the above requirements are met, the application may enable the STREAM mode using the following API:

```
KieBaseConfiguration config = KieServices.Factory.get().newKieBaseConfiguration();
config.setOption( EventProcessingOption.STREAM );
```

Alternatively, the equivalent property:

drools.eventProcessingMode = stream

When using the STREAM, the engine knows the concept of the flow of time and the concept of "now," i.e., the engine understands how old events are based on the current timestamp read from the session clock.

This characteristic allows the engine to provide the following additional features to the application:

- Sliding window support
- Automatic event lifecycle management
- Automatic rule delaying when using negative patterns

With temporal reasoning, IoT sensors provide information that Drools AI is acted on immediately.

All these features are explained in the following sections:

3.5.3 Role of Session Clock in Stream mode

When running the engine in CLOUD mode, the session clock is used only to timestamp the arriving events that don't have a previously defined timestamp attribute. Although in STREAM mode, the session clock assumes an even more important role.

In STREAM mode, the session clock is responsible for keeping the current timestamp and based on it, the engine does all the temporal calculations on event's aging, synchronizes streams from multiple sources, schedules future tasks and so on.

The documentation on the session clock, in this section, shows you how to configure and use different session clock implementations.

3.5.4 Negative Patterns in Stream Mode

Negative patterns behave differently in STREAM mode when compared to CLOUD mode. In CLOUD mode, the engine assumes that all facts and events are known in advance (there is no concept of the flow of time) and so, negative patterns are evaluated immediately.

When running in STREAM mode, negative patterns with temporal constraints may require the engine to wait for a time period before activating a rule. The time period is automatically calculated by the engine in a way that the user does not need to use any tricks to achieve the desired result.

For instance:

Example 3.5.4.1. A rule that activates immediately upon matching

```
rule "Sound the alarm"
when
    $f : FireDetected( )
    not( SprinklerActivated( ) )
then
    // sound the alarm
end
```

The above rule has no temporal constraints that would require delaying the rule. Therefore, the rule activates immediately. The following rule, on the other hand, must wait for 10 seconds before activating since it may take up to 10 seconds for the sprinklers to activate:

Example 3.5.4.2. A rule that automatically delays activation due to temporal constraints

```
rule "Sound the alarm"
when
    $f : FireDetected( )
    not( SprinklerActivated( this after[0s,10s] $f ) )
then
    // sound the alarm
end
```

This behavior allows the engine to keep consistency when dealing with negative patterns and temporal constraints at the same time. The above would be the same as writing the rule as below, but does not burden the user to calculate and explicitly write the appropriate duration parameter:

Example 3.5.4.3. Same rule with explicit duration parameter

```
rule "Sound the alarm"
    duration( 10s )
when
    $f : FireDetected( )
    not( SprinklerActivated( this after[0s,10s] $f ) )
then
    // sound the alarm
end
```

The following rule expects every 10 seconds at least one "Heartbeat" event, if not the rule fires. The special case in this rule is that we use the same type of the object in the first pattern and the negative pattern. The negative pattern has the temporal constraint to wait between 0 to 10 seconds before firing, and it excludes the heartbeat bound to $h. Excluding the bound heartbeat is important since the temporal constraint [0s, ...] does not exclude by itself the bound event $h from being matched again, thus preventing the rule to fire.

Example 3.5.4.4. Excluding bound events in negative patterns

```
rule "Sound the alarm"
when
    $h: Heartbeat( ) from entry-point "MonitoringStream"
    not( Heartbeat( this != $h, this after[0s,10s] $h ) from entry-point
"MonitoringStream" )
then
    // Sound the alarm
end
```

3.6 Session Clock

Reasoning over time requires a reference clock. To mention one example, if a rule reasons over the average price of a given stock over the last 60 minutes, how the engine knows what stock price changes happened over the last 60 minutes in order to calculate the average? The obvious response is by comparing the timestamp of the events with the "current time." How does the engine know what **time it is now**? Again, obviously, by querying the session clock.

The session clock implements a strategy pattern allowing different types of clocks to be plugged and used by the engine. This is very important because the engine may be running in an element of different scenarios that may require different clock implementations. Just to mention a few:

- **Rules testing:** Testing always requires a controlled environment, and when the tests include rules with temporal constraints it is necessary to not only control the input rules and facts but also the flow of time.
- **Regular execution:** Usually when running rules in production the application requires a real-time clock that allows the rules engine to react immediately to the time progression.
- **Special environments:** Specific environments may have specific requirements on time control. Cluster environments may require clock synchronization through heartbeats, or JEE environments may require the use of an AppServer provided clock, etc.
- **Rules replay or simulation:** To replay scenarios or simulate scenarios it is necessary that the application also controls the flow of time.

3.6.1 Available Clock Implementations

Drools provides 2 clock implementations out of the box — the default real-time clock, based on the system clock and an optional pseudo clock, controlled by the application.

3.6.2 Real Time Clock

By default, Drools uses a real-time clock implementation that internally uses the system clock to determine the current timestamp.

To explicitly configure the engine to use the real time clock, just set the session configuration parameter to real time:

```
KieSessionConfiguration config = KieServices.Factory.get().newKieSessionConfiguration();
config.setOption( ClockTypeOption.get("realtime") );
```

3.6.3 Pseudo Clock

Drools also offers out of the box implementation of a clock that is controlled by the application that is called Pseudo Clock. This clock is especially useful for unit testing temporal rules since it can be controlled by the application and so the results become deterministic.

To configure the pseudo session clock, do:

```
KieSessionConfiguration config = KieServices.Factory.get().newKieSessionConfiguration();
config.setOption( ClockTypeOption.get("pseudo") );
```

As an example of how to control the pseudo session clock:

```
KieSessionConfiguration config = KieServices.Factory.get().newKieSessionConfiguration();
conf.setOption( ClockTypeOption.get( "pseudo" ) );
KieSession session = kbase.newKieSession( conf, null );

SessionPseudoClock clock = session.getSessionClock();

// then, while inserting facts, advance the clock as necessary:

FactHandle handle1 = session.insert( tick1 );
clock.advanceTime( 10, TimeUnit.SECONDS );
FactHandle handle2 = session.insert( tick2 );
clock.advanceTime( 30, TimeUnit.SECONDS );
FactHandle handle3 = session.insert( tick3 );
```

3.7 Sliding Windows

Sliding windows are a way to scope the events of interest by defining a window that is constantly moving. The two most common types of sliding window implementations are time-based windows and length based windows.

The next sections detail each of them.

- **Important** Sliding Windows are only available when running the engine in STREAM mode. Check the event processing mode section for details on how the STREAM mode works.
- **Important** Sliding windows start to match immediately and defining a sliding window does not imply that the rule has to wait for the sliding window to be "full" in order to match.

  For instance, a rule that calculates the average of an event property on a window: length(10) will start calculating the average immediately, and it will start at 0 (zero) for no-events, and will update the average as events arrive one by one.

3.7.1 Sliding Time Windows

Sliding time windows allows you to write rules that will match events occurring in the last X time units.

For instance, if the user wants to consider only the stock ticks that happened in the last 2 minutes, the pattern would look like this:

```
StockTick() over window:time( 2m )
```

Drools uses the "**over**" keyword to associate windows to patterns.

On a more elaborate example, if the user wants to sound an alarm in case the average temperature over the last 10 minutes read from a sensor is above the threshold value, the rule would look like this:

Example 3.7.1.1. Aggregating values over time windows

```
rule "Sound the alarm in case temperature rises above threshold"
when
    TemperatureThreshold( $max : max )
    Number( doubleValue > $max ) from accumulate(
        SensorReading( $temp : temperature ) over window:time( 10m ),
        average( $temp ) )
then
    // sound the alarm
End
```

The engine automatically disregards any SensorReading older than 10 minutes and keep the calculated average consistent.

- **Important** Please note that time-based windows are considered when calculating the interval an event remains in the working memory before being expired but an event falling off a sliding window does not mean by itself that the event will be discarded from the working memory, as there might be other rules that depend on that event. The engine will discard events only when no other rules depend on that event, and the expiration policy for that event type is fulfilled.

3.7.2 Sliding Length Windows

Sliding length windows work the same way as time windows but consider events based on order of their insertion into the session instead of flow of time. For instance, if the user wants to consider only the last 10 RHT company stock ticks independent of how old they are, the pattern would look like this:

```
StockTick( company == "RHT" ) over window:length( 10 )
```

As you can see, the pattern is similar to the one presented in the previous section, but instead of using window:time to define the sliding window, it uses window:length.

Using a similar example to the one in the previous section, if the user wants to sound an alarm in case the average temperature over the last 100 readings from a sensor is above the threshold value, the rule would look like:

Example 3.7.2.1. Aggregating values over length windows

```
rule "Sound the alarm in case temperature rises above threshold"
when
    TemperatureThreshold( $max : max )
    Number( doubleValue > $max ) from accumulate(
        SensorReading( $temp : temperature ) over window:length( 100 ),
        average( $temp ) )
then
    // sound the alarm
End
```

The engine only considers the last 100 readings to calculate the average temperature.

- **Important** Please note that falling off a length based window is not criteria for event expiration in the session. The engine disregards events that fall off a window when calculating that window but does not remove the event from the session based on that condition alone as there might be other rules that depend on that event.
- **Important** Please note that length based windows do not define temporal constraints for event expiration from the session, and the engine will not consider them. If events have no other rules defining temporal constraints and no explicit expiration policy, the engine will keep them in the session indefinitely.

3.8 Streams Support

Most CEP use cases have to deal with streams of events. The streams can be provided to the application in various forms, from JMS queues to flat text files, from database tables to raw sockets or even through web service calls. In any case, the streams share a common set of characteristics:

- **Events** in the stream are ordered by a timestamp. The timestamp may have different semantics for different streams, but they are always ordered internally.
- **Volumes** of events are usually high.
- **Atomic events** are rarely useful by themselves. Usually, meaning is extracted from the correlation between multiple events from the stream and also from other sources.
- **Streams** may be homogeneous, i.e., contain a single type of events, or heterogeneous, i.e., contain multiple types of events.

Drools generalized the concept of a stream as an "entry-point" into the engine. An entry point for drools is a gate from which facts come. The facts may be regular facts or special facts like events.

In Drools, facts from one entry point (stream) may join with facts from any other entry point or event with facts from the working memory. Although, they never mix, i.e., they never lose the reference to the entry point through which they entered the engine. This is important because one may have the same type of facts coming into the engine through several entry points, but one fact that is inserted into the engine through entry point A will never match a pattern from an entry point B, for example.

3.8.1 Declaring and Using Entry Points

Entry points are declared implicitly in Drools by directly making use of them in the rule. i.e., referencing an entry point in a rule will make the engine, at compile time, identify and create the proper internal structures to support that entry point for this rule.

So, for instance, let's imagine a banking application where transactions are fed into the system coming from streams. One of the streams contains all the transactions executed in ATMs. So, if one of the rules says a withdrawal is authorized if and only if the account balance is over the requested withdrawal amount, the rule would look like:

Example 3.8.1.1. Example of Stream Usage

```
rule "authorize withdrawal"
when
    WithdrawRequest( $ai : accountId, $am : amount ) from entry-point "ATM
Stream"
    CheckingAccount( accountId == $ai, balance > $am )
then
    // authorize withdrawal
End
```

In the previous example, the engine compiler identifies that the pattern is tied to the entry point "ATM Stream" and will both create all the necessary structures for the rule base to support the "ATM Stream" and only matches WithdrawalRequests coming from the "ATM Stream." In the previous example, the rule is also joining the event from the stream with a fact from the main working memory (CheckingAccount).

Now, let's imagine a second rule that states that a fee of $2 must be applied to any account for which a withdrawall request is placed at a bank branch:

Example 3.8.1.2. Using a different Stream

```
rule "apply fee on withdrawal on branches"
when
    WithdrawRequest( $ai : accountId, processed == true ) from entry-point
"Branch Stream"
```

```
        CheckingAccount( accountId == $ai )
    then
        // apply a $2 fee on the account
    End
```

The previous rule will match events of the exact same type as the first rule (WithdrawalRequest) but from two different streams. So, an event inserted into "ATM Stream" will never be evaluated against the pattern on the second rule because the rule states that it is only interested in patterns coming from the "Branch Stream."

So, entry points besides being a proper abstraction for streams are also a way to scope facts in the working memory and a valuable tool for reducing cross products explosions. However, that is a subject for another time.

Inserting events into an entry point is equally simple. Instead of inserting events directly into the working memory, insert them into the entry point as shown in the example below:

Example 3.8.1.3. Inserting facts into an entry point

```
// create your rulebase and your session as usual
KieSession session = ...
// get a reference to the entry point
EntryPoint atmStream = session.getEntryPoint("ATM Stream" );
// and start inserting your facts into the entry point
atmStream.insert( aWithdrawRequest );
```

The previous example shows how to manually insert facts into a given entry point. Although, usually the application will use one of the many adapters to plug a stream end point like a JMS queue directly into the engine entry point without coding the inserts manually. The Drools pipeline API has several adapters and helpers to do that as well as examples on how to do it.

3.9 Memory Management for Events

- **Important** The automatic memory management for events is only performed when running the engine in STREAM mode. Check the event processing mode section for details on how the STREAM mode works.

One of the benefits of running the engine in STREAM mode is that the engine can detect when an event can no longer match any rule due to its temporal constraints. When that happens, the engine can safely delete the event from the session without side effects and release any resources used by that event.

There are basically 2 ways for the engine to calculate the matching window for a given event:

- Explicitly, using the expiration policy
- Implicitly, analyzing the temporal constraints on events

3.9.1 Explicit expiration offset

The first way of allowing the engine to calculate the window of interest for a given event type is by explicitly setting it. To do that, use the declare statement and define an expiration for the fact type:

Example 3.9.1. Explicitly defining an expiration offset of 30 minutes for StockTick events

```
declare StockTick
    @expires( 30m )
End
```

The above example declares an expiration offset of 30 minutes for StockTick events. After that time, assuming no rule still needs the event, the engine will expire and remove the event from the session.

- **Important** An explicit expiration policy for a given event type overrides any inferred expiration offset for that same type.

3.9.2 Inferred expiration offset

Another way for the engine to calculate the expiration offset for a given event is implicit, by analyzing the temporal constraints in the rules. For instance, given the following rule:

Example 3.9.2.1. Example rule with temporal constraints

```
rule "correlate orders"
when
    $bo : BuyOrderEvent( $id : id )
    $ae : AckEvent( id == $id, this after[0,10s] $bo )
then
    // do something
End
```

Analyzing the above rule the engine automatically calculates that whenever a BuyOrderEvent matches, it needs to store it for up to 10 seconds to wait for matching AckEvent's. So, the implicit expiration offset for BuyOrderEvent will be 10 seconds. AckEvent, on the other hand, can only match existing BuyOrderEvent's and so its expiration offset will be zero seconds.

The engine will make this analysis for the whole rulebase and find the offset for every event type.

- **Important** An explicit expiration policy for a given event type overrides any inferred expiration offset for that same type.

3.10 Temporal Reasoning

Temporal reasoning is another requirement of any CEP system. As discussed previously, one of the distinguishing characteristics of events is their strong temporal relationships in **IoT** (Internet of Things) device computing.

With temporal reasoning IoT device sensors provide information that **AI-BPM** can act on immediately and conditionally, determined by situational awareness. In Drools AI-IoT, judging the impact avoidances of a vehicle and making course adjustments, is an example of Drools AI-IoT temporal reasoning. The distinction is acting on events and not reasoning over data, as in Cloud.

Temporal reasoning is a type of stream reasoning and is an extensive field of research from its roots on temporal modal logic to its more practical applications in business systems. There are hundreds of papers and thesis written, and approaches are described for several applications. Drools takes a pragmatic and simple approach based on several sources about maintaining knowledge about temporal intervals which make it an elegant solution for **IoT**.

Drools implements the interval-based time event semantics described by Allen and represents point-in-time events as interval-based events with duration 0 (zero). Events or alerts are raised by the Arduino Tron IoT devices and indicate a change in condition, again elegant for a Drools rules AI-IoT solution.

3.10.1 Temporal Situational Awareness (SA)

As distributed IoT sensors and applications become larger and more complex, the simple processing of raw sensor and actuation data streams becomes impractical. Instead, data streams must be fused into tangible facts, information that is combined with the knowledge to perform meaningful operations.

In current IoT systems, sensing and actuation are mostly done at the bare bones data level, whereas many IoT applications demand higher level situation awareness of - and reasoning about - the systems' states and the physical environment where they operate to perform intelligent decisions.

Situational Awareness (SA) is the perception of environmental elements by our IoT devices and events with respect to time or space, the comprehension of their meaning, and the projection of their status after some variable has changed, such as time, or some other variable, such as a predetermined event.

This is what we have accomplished with our AI-IoT Drools-BPM implementation. Our Arduino Tron IoT devices are judging and making decisions after cognitive situational reasoning.

3.10.2 Complex Event Processing

Event Stream Processing, or ESP, is a set of technologies which include event visualization, event databases, event-driven middleware, and our event processing language (Drools). After inserting events, the ESP passes-off to our complex event processing (CEP), which takes precedence.

The manipulations of events are described by CEP rules, which are event-condition-actions that combine continuous query primitives with context operators (e.g., temporal, logical, quantifiers) on received events, checking for correlations among these events, and generating complex (or composite) events that summarize the correlation of the input events. Most CEP systems have the concept of Event Processing Agents (EPAs), which are modules that implement event processing workflows.

3.10.3 IoTBPM - Internet of Things Reasoning

The IoT paradigm aims at connecting billions of IoTBPM devices to the internet. This requires suitable architecture and technologies capable of bridging the vast heterogeneity of the devices to provide meaningful services. To enable this seamless integration of devices to provide sophisticated services, the concept of AI reasoning has to address a number of issues with the growing number of devices e.g., scalability, heterogeneity, and reliability.

3.10.4 Temporal Operators

Drools implements all 13 operators defined by Allen and also their logical complement (negation). This section details some of the operators and their parameters used in IoT.

3.10.5 After

The after evaluator correlates two events and matches when the temporal distance from the current event to the event being correlated belongs to the distance range declared for the operator.

Let's look at an example:

```
$eventA : EventA( this after[ 3m30s, 4m ] $eventB )
```

The previous pattern will match if and only if the temporal distance between the time when $eventB finished and the time when $eventA started is between (3 minutes and 30 seconds) and (4 minutes).

3.10.6 Before

The before evaluator correlates two events and matches when the temporal distance from the event being correlated to the current correlated belongs to the distance range declared for the operator.

Let's look at an example:

```
$eventA : EventA( this before[ 3m30s, 4m ] $eventB )
```

The previous pattern will match if and only if the temporal distance between the time when $eventA finished and the time when $eventB started is between (3 minutes and 30 seconds) and (4 minutes).

3.10.7 Coincides

The coincides evaluator correlates two events and matches when both happen at the same time. Optionally, the evaluator accepts thresholds for the distance between events start and finish timestamps.

Let's look at an example:

```
$eventA : EventA( this coincides $eventB )
```

The previous pattern matches if and only if the start timestamps of both $eventA and $eventB are the same AND the end timestamp of both $eventA and $eventB also are the same.

Optionally, this operator accepts one or two parameters. These parameters are the thresholds for the distance between matching timestamps.

- If only one parameter is given, it is used for both start and end timestamps.

- If two parameters are given, then the first is used as a threshold for the start timestamp and the second one is used as a threshold for the end timestamp.

3.10.8 During

The during evaluator correlates two events and matches when the current event happens during the occurrence of the event being correlated.

Let’s look at an example:

```
$eventA : EventA( this during $eventB )
```

The previous pattern will match if and only if the $eventA starts after $eventB starts and finishes before $eventB finishes.

3.10.9 Finishes

The finishes evaluator correlates two events and matches when the current events start timestamp happens after the correlated events start timestamp, but both end timestamps occur at the same time.

Let’s look at an example:

```
$eventA : EventA( this finishes $eventB )
```

The previous pattern will match if and only if the $eventA starts after $eventB starts and finishes at the same time $eventB finishes.

3.10.10 Includes

The includes evaluator correlates two events and matches when the event being correlated happens during the current event. It is the symmetrical opposite of during evaluator.

Let’s look at an example:

```
$eventA : EventA( this includes $eventB )
```

The previous pattern will match if and only if the $eventB starts after $eventA starts and finishes before $eventA finishes.

3.10.11 Starts

The starts evaluator correlates two events and matches when the current event end timestamp happens before the correlated event's end timestamp, but both start timestamps occur at the same time.

Let’s look at an example:

```
$eventA : EventA( this starts $eventB )
```

The previous pattern will match if and only if the $eventA finishes before $eventB finishes and starts at the same time $eventB starts.

3.10.12 Started By

The started by evaluator correlates two events and matches when the correlating events end timestamp happens before the current events end timestamp, but both start timestamps occur at the same time. Let’s look at an example:

```
$eventA : EventA( this startedby $eventB )
```

The previous pattern matches if and only if the $eventB finishes before $eventA finishes and starts at the same time $eventB starts.

The startedby evaluator accepts one optional parameter. If it is defined, it determines the maximum distance between the start timestamp of both events in order for the operator to match. Example:

```
$eventA : EventA( this starts[ 5s ] $eventB )
```

4 Rules Writing Performance Memory and Testing

4.1 Drools Writing Rules Best Practice

Example 4.1.1. Drools Rules Practices

Drools performance is based on how Rete trees and nodes are created, how Drools indexes them, and why an increasing number of objects in Drools hardly affects the total time taken to execute it. Rules written intelligently can drastically reduce the number of nodes in the Rete tree, thus, further increasing memory and impacting the performance.

Some Drools rules writing best practice, and executing the rules as fast as possible:

- Put the most restricting condition on the top
- The conditions that you feel should be on the highest priority - put them on the top
- The then conditions that you use should be diligently prepared
- Use the same order of conditions across your rules
- Do not use eval unless you have to
- Put evals at the bottom of your conditions
- Do not use if statements inside consequences
- Using shortcuts for Booleans cause JIT errors on Drools 5.4 so do use them as House (windowOpen == true) not House (windowOpen)
- Avoid using salience. In most cases, it leads to maintenance issues
- Plan using an Eclipse-Drool UI to create good rules
- Never attempt using if-statements inside the then part
- Use shortcuts for Boolean because they often cause errors
- Always follow the pattern of RWTE - i.e., 1. RULE 2. WHEN 3. THEN 4. END
- Avoid calling heavy java objects. It causes troubles in most cases
- Try to integrate the rules with custom classes
  rather than predefined sets to be used for your operations
- The condition that you are using when the part should be interlinked and not null
  (i.e., the condition should be linked to some values which have existence)
- Always use the Drools generalized concept of a stream as an "entry point" into the engine
  An entry point is for drools a gate from which facts come, the source where there were inserted
  The facts may be regular facts or special facts like events
- Use the import statements properly. Import statements work like import statements in Java. You need to specify the fully qualified paths and type names for any objects you want to use in the rules. Drools automatically imports classes from the Java package of the same name, and also from the package java.lang

- **Package**: Every rule starts with a package name. The package acts as a namespace for rules. Rule names within a package must be unique. Packages in rules are similar to packages in Java. They serve the same purpose.

- **Import statement**: Whatever facts you want to apply the rule on, those facts need to be imported into your application.

- **Rule definition**: Consists of the rule name, the condition, and the consequence. Drools keywords are **rule, when, then,** and **end**. The **when** part is the condition in both the rules and the **then** part is the consequence.

- **Load the rules once; batch inserts all facts and fire once:** Letting the engine work out the most optimal way to execute is more efficient and easier to maintain from the development perspective. It also prevents the rules being loaded/parsed multiple times before getting the desired outcome.

- **Don't overload rules:** Each rule should describe one and only one scenario. The engine will optimize shared conditions: i.e., rules that share conditions of a fact (in the same order) share their Rete nodes.

4.2 Drools Performance and Memory Internals

Drools Rules "lots of objects" and "complex logic in sequence"

JBoss Drools - The keywords for performance impact are "lots of objects" and "complex logic in sequence". JBoss Drools uses Rete's algorithm to execute rules. Rete's algorithm is an efficient pattern matching algorithm. In the later versions of Drools, the Phreak engine optimization has been added, and this has greatly improved performance, making the engine lazy evaluation instead of eggier evaluation.

JBoss Drools has its own implementation of Rete's algorithm. A rule in Drools is represented by a Rete tree. A Rete tree consists of nodes. These nodes are mostly conditional evaluations. Everything in a Drools rule is represented by a Rete tree node. Apart from conditional nodes, the Rete tree also consists of AND nodes, OR nodes, and start/end nodes (and a few other types of nodes as well).

The main part of the algorithm is the creation of the Rete tree. Whenever a fact (object) is inserted in the Drools engine, a Rete tree is created. The creation of this Rete tree is done by some intelligent algorithm. The Rete tree, once created, executes in almost no time over the facts (objects) and gives the result. Most of the time is spent creating the Rete tree rather than executing it.

This is something that you can also experience during debugging. Whenever a fact is inserted into the Drools session, it takes a bit of time. However, the firing of rules is almost instantaneous.

So, the way this Rete algorithm has been implemented inside Drools accounts for its efficiency. However, what will happen to the JBoss Drools Efficient Rete algorithm when "LOTS OF OBJECTS" will come into the picture?

The answer lies in two techniques that Drools uses to store nodes: Node Sharing and Node Indexing.

Drools caches nodes while building Rete's tree which is known as node sharing. Whenever a new node is created, it's checked whether there is an equivalent node already present in the cache. If an equivalent node is found in the cache, the cached node is used instead, and the new node is discarded. This makes Drools more memory efficient.

Drools keeps a hash table of the object properties and Rete tree nodes. It is known as node indexing and is used to avoid evaluating the same conditions multiple times. Node indexing speeds up the propagation of nodes in the Rete network which accounts for high performance of the Drools engine.

The in-depth analysis of Rete tree creation and node propagation in the Rete network tells that the way rules are written might also affect the performance as it directly impacts the propagation in Rete tree. Rules written intelligently can drastically reduce the number of nodes in the Rete tree, thus, further increasing the performance.

So, these techniques help Drools to process a large number of objects efficiently. The conclusion is - increasing the number of objects in Drools hardly affects the total time taken to execute it.

4.3 Drools Testing Rules Methodology

Drools Rules tested as code or data?

Should business rules be embedded as a part of the application code that doesn't change very often or are rules more like your application data (for example, pricing lists), which you expect to change on almost a daily basis?

This comes up often. Depending on our answer, we will deploy our rules very differently.

The answer, somewhat confusing, is that rules are both.

- Rules are as powerful as the normal code and should be treated in the same way. (For example, before deployment, any changes should be thoroughly tested.)
- Rules are as easy to change as data because first, they live outside the "normal" application and second, rule engines are expressly designed to easily allow changes to the business rules.

So, we are really saying "**both,**" depending on impact.

4.3.1 Testing of the Rules

Testing of the rules is achieved by means of unit testing each rule or a couple of rules representing a certain scenario. Each such test case is covered by several unit tests attempting to evaluate a different situation. The tests comprising these unit tests are grouped in compliance with the separation of the rules into .drl files. In other words, a test may add into knowledge builder only the .drl file holding the declarations and the .drl file representing the tested logic.

Each test is composed of unit tests and a before and after method. The before method is responsible for composing the session before each unit test and the after method for correct disposal of it. The constructing of the session is performed in a similar way as in the init phase of the application with the exception of resources, channels, and clock.

Only the resources necessary for the scenario are used for creating knowledge packages. The mock objects are substituted for actual channels as they require the application server to operate.

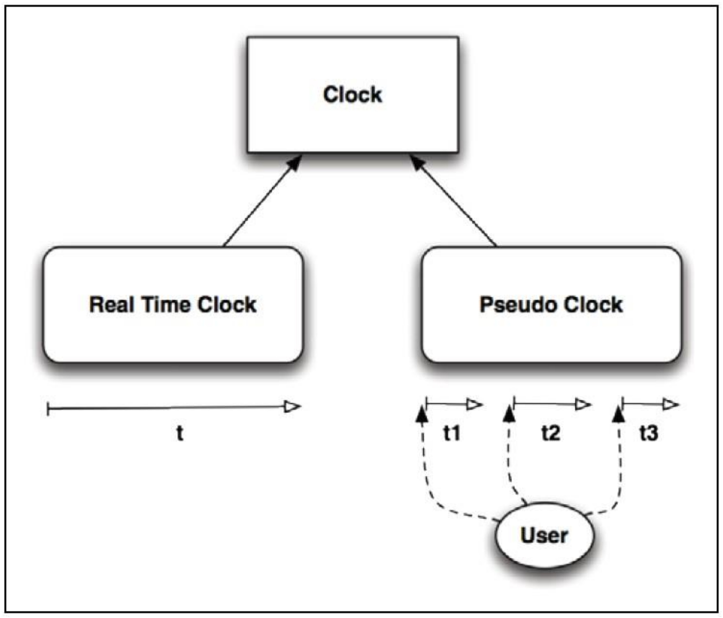


Figure 4.3.1.1: The two Drools clock implementations.

Since many of the rules employ the temporal reasoning aspects, the clock implementation in the tests must be different. Drools provides two clock implementations demonstrated in Figure 13.1, where the Real Time Clock applies the JVM clock, and Pseudo Clock enables the application to control the flow of time in all different ways. In order for the real-time clock to be the default option, the setting of the pseudo-clock has to be configured while creating the session.

4.3.2 Unit Testing Rules

An important point to note is that you should carry out unit testing in the rules that you write. It is manual unit testing, but we still checked that your blocks of rules produced the outcome that you expected. All we're talking about here is automating the process.

Unit testing also has the advantage of documenting the code because it gives a working example of how to call the rules. It also makes your rules and codes more reusable. You've just proved (in your unit test) that you can call your code on a standalone basis, which is an important first step for somebody else to be able to use it again in the future.

Run tests with a small amount of data on all of your rules, i.e., with a minimal number of facts in the rule session and test the results that particular rule was fired as expected. For the sample data, use static data and define minimal test data for each rule test.

5 Arduino Tron AI-IoT Drools-jBPM Intelligent Things

5.1 AI-IoTBPM Artificial Intelligent Reasoning

5.1.1 IoTBPM AI-Artificial Intelligent Smart Things Automation

The Internet of Things (IoT) refers to a network of connected devices collecting and exchanging data and processes over the internet. IoT promises to provide "smart" environments (homes, cities, hospitals, schools, stores, offices, etc.) and smart products (cars, trucks, airplanes, trains, buildings, devices, etc.). While this data is useful, there is still "a disconnect" in integrating these IoT devices with mission-critical business processes and corporate cloud data awareness.

The task of moving IoT devices beyond "connected" to "smart" is daunting. Moving beyond collecting IoT data and transitioning, to leveraging this new wealth of IoT data, to improving the smart decision-making process is the key to automation. Artificial Intelligence (**AI**) will help these IoT devices, environments and products to self-monitor, self-diagnose and eventually, self-direct. This is what we said in the opening of this book, "If a machine thinks, then a machine can do."

However, one of the key concepts in enabling this transition from connected to smart is the ability to perform **AI Analytics**. The traditional analytic models of pulling all data into a centralized source such as a data warehouse or analytic sandbox is going to be less useful. We are not trying just to analyze complex IoT data, we are trying to make "smart decisions" and take actions base on our AI Analytics of IoT devices.

IoT Definition *– IoT is the integration of computer-based systems into our physical-world.*

Our world is increasingly linked through the number of already connected IoT devices. IoT components are equipped with sensors and actuators that enable sensing, acting, collecting and exchange data via various communication networks including the internet. These IoT devices such as wearable, GPS, smartphones, connected cars, vending machines, smart homes, and automated offices are used in areas such as supply chain management, intelligent transport systems, robotics, and remote healthcare. Businesses can rapidly gain a competitive edge by using the information and functionalities of IoT devices (sensors and actuators). So, business processes uses IoT information to incorporate real-world data, to make informed decisions, optimize their execution, and adapt itself to context changes.

Also, the increase in processing power of IoT devices enables them to take part in the execution of the business logic. This way IoT devices can aggregate and filter data and make decisions locally by executing parts of the business logic whenever central control is not required, reducing both the amount of exchanged data and of central processing involvement.

The power of the IoT device increases greatly when business process (jBPM) can use them to provide information about our real-world as well as execute IoT device actions as part of our business process. The jBPM-BPMN modular allow us to define both the business processes and IoT devices behavior at the same time using one (BPM) diagram. In our examples, we will be adding Drools-jBPM to IoT devices. Making **"Things Smart"** is the application of AI to IoT platform via DroolsRules Inference Reasoning, jBPM, and ES-Expert Systems Architecture.

With the use of AI Drools-jBPM analysis and reasoning in IoT devices, we can orchestrate dissimilar devices that normally have no awareness of each other. This creates opportunities for direct integration of computer-based into the physical-world that has never been available before. This results in improved efficiency, accuracy, and economic benefits by increased automation - reduced intervention. This IoT orchestration of IoT devices gives us the ability for action after our AI decision.

5.1.2 IoT Human Interface or Human-Task Node

Another important aspect of IoT AI Drools-jBPM business processes management is the human interface and system interaction. While some of the work performed in an IoT jBPM process can be executed automatically, some tasks may need to be executed by human actors. jBPM supports a special human task node inside processes for modeling this interaction with human users. This human task node allows process designers to define the properties related to the task that the human actor needs to execute.

Consider that our IoT device may raise an alarm, or alert to a device or process fault condition or action. The jBPM supports the use of human tasks inside processes using a user task node. A user task node represents an atomic task that needs to be executed by a human actor (in this example maybe answer the alarm). As far as the jBPM engine is concerned, human tasks are similar to any other external service that needs to be invoked and are implemented as a domain-specific service.

Because a human task is an example of such a domain-specific service, the process itself contains a high-level, abstract description of the human task that needs to be executed, and a work item handler is responsible for binding this abstract tasks to a specific implementation of the human task.

The IoT AI Drools-jBPM project provides a default implementation of a human task service based on the WS-Human-Task specification. If you have a requirement to integrate an existing human task service you can use this service. It manages the life cycle of the tasks (creation, claiming, completion, etc.) and stores the state of the tasks. To have human actors participate in your IoT processes:

1. Include human task nodes inside your process to model the interaction with human actors
2. Integrate a task management component (e.g., the WS-Human-Task provided by jBPM)
3. Have the end-users interact with a human task IoTBPM client interface.

How can IoT benefit from jBPM? Let us consider a complex system with multiple components interacting within a smart environment being aware of the components' locations, movements, and interactions. Such a system can be a smart factory with autonomous robots, a retirement home with connected residents, or, at a larger scale, a smart city. While the parties in the system can track the movements of each component and also relate multiple components' behaviors to each other, they do not know the components' agendas. Often their interactions are based on habits, i.e., routine low-level processes, which represent recurring tasks. Some of these routines are more time and cost critical than others, some may be dangerous or endanger others, and some may just be inefficient or superfluous. Knowing their agendas, their goals, and their procedures can enable a better basis for planning, execution, and safety.

How can jBPM benefit from IoT? Let us consider a complex process with multiple parties interacting in the context of a business transaction. Such a process can be, for example, a procurement process, where goods are ordered, delivered, stored, and paid for. While the system can track each automatically executed activity on its own, it relies on messages from other parties and manually entered data in the case of manual activities. If this data is not entered or entered incorrectly, discrepancies between the digital (i.e., computerized representation) process and the real-world execution of the process occur. Similar concerns hold, if the process participants do not obey the digital process under certain circumstances (e.g., an emergency in healthcare) or have not entered the data, though in the real-world process the respective activity was already executed.

Such scenarios might be better manageable when closely linking the digital process with the physical-world as enabled by the integration of IoT and jBPM; e.g., the completion of manual activities can be made observable through the usage of appropriate sensors. IoT can complete jBPM with continuous data sensing and physical actuation for improved decision making. Decisions in processes require relevant information as a basis for making meaningful decisions. Data from IoT, such as events provided through in-memory databases or **CEP** (Complex Event Processing) can be useful in this context.

5.1.3 AI-IoT Drools-jBPM Artificial Intelligent Reasoning Makes IoT Smart

AI-IoT is a mix of Business Processes (BPM) with Business Rules Drools (Reasoning), to define advanced and complex scenarios. Also, Drools Rules Engine adds the ability of temporal reasoning, allowing business processes to be monitored, improved, and cover business scenarios that require temporal inferences. Event stream processing focused on the capabilities of processing streams of events in (near) real time, while the main focus of CEP (Complex Event Processing) was on the correlation and composition of atomic events into complex (compound) events. IoTBPM for CEP is primarily an event processing concept that deals with the task of processing multiple events with the goal of identifying the meaningful events within the IoT event cloud. CEP in IoTBPM employs techniques for detection of complex patterns of many events, event correlation and abstraction, and event hierarchies.

5.2 Arduino Tron IoT AI Architecture

The power of the IoT device increases greatly when business process (jBPM) can use them to provide information about our real-world as well as execute IoT Agent (devices) as part of our business process. Arduino Tron adds Drools-jBPM to these IoT "smart" devices. At the beginning of Chapter Two, we introduced the EOSpy/Arduino Tron technology stack diagram, and now we will review each component. Executive Order EOSpy provides us the three-tiered architecture to provide a complete AI-IoT system.

- **AI (Drools-JBPM):** AI-IoTBPM the Internet of Things Drools-jBPM Expert System.
- **Server (EOSpy):** Sensor Processor live map GPS Tracking, Analysis of IoT Data
- **IoT (Arduino Tron):** Android Environment Sensors, IoT Data Collection, and GPS

5.2.1 AI (Drools-jBPM) - IoT Internet of Things Expert System

The Executive Order Corp Arduino Tron ESP8266 MQTT Telemetry Transport Machine-to-Machine (M2M) Internet of Things (IoT) software and Arduino Tron MQTT AI-IoT Client using Arduino Tron AI-IoT Drools-jBPM latest software can be download from the Github website.

Download Arduino Tron from https://github.com/eodas/IoTBPM

GIT the EOSPY AI-IoTBPM from the source code repository and Import Existing Maven project.

Drools - Allows fast and reliable evaluation of business rules and complex event processing.

• A rule engine is a fundamental building block to create an expert system; an AI/ES System
• Drools can reason to a conclusion (infer) beyond what we currently know
• An inference model provides a conclusion reached on the basis of evidence and reasoning
• We say that; Drools emulates the decision-making ability of a human expert

Drools Engine Type Definitions

Forward Chaining

• Forward Chaining is "data-driven" Drools method of deriving a particular goal from a given knowledge base and set of inference rules.

The application of inference rules results in new knowledge (from the consequents of the relations matched), which is then added to the knowledge base. In forward chaining, the system starts from a set of facts, and a set of rules, and tries to find a way of using those rules and facts to deduce a conclusion or come up with a suitable course of action.

Usually, when a rule is triggered, it is then fired, which means its conclusion is added to the rules facts database. If the conclusion of the rule that has fired is an action or a recommendation, then the system may cause that action to take place or the recommendation to be made.

Backwards Chaining

• Backward chaining is a goal-driven method of deriving a particular goal from a given knowledge base and set of inference rules. Inference rules are applied by matching the goal of the search to the consequents of the relations stored in the knowledge base.

In backward chaining, we start from a conclusion, which is the hypothesis we wish to prove, and we aim to show how that conclusion can be reached from the rules and facts in the database. The conclusion we are aiming to prove is called a goal, and so reasoning in this way is known as goal-driven reasoning.

Backward chaining starts with the goal state, which is the set of conditions the agent wishes to achieve in carrying out its plan. It now examines this state and sees what actions could lead to it. Prolog is backwards chaining AI-ES engine.

jBPM Business Process Management

jBPM is a flexible Business Process Management (BPMN 2.0) that allows you to model, execute, and monitor business processes throughout their life cycle. Business Process Management (jBPM) was established to analyze, discover, design, implement, execute, monitor and evolve collaborative business processes within and across organizations. BPM implements a strategic **Goal** of an organization.

A business process allows you to model business goals by describing the steps that need to be executed to achieve those goals, and the order of those goals are depicted using a (BPM) flow chart. Executable business processes bridge the gap between business, users, developers and IoT devices as they are higher-level and use domain-specific concepts that are understood by business users but can also be executed directly by developers and IoT devices.

Additionally, jBPM is an extremely time-sensitive and responsive technology that allows time-critical, dynamic business processes to be changed quickly and while processes are still in progress. This means that jBPM systems can take advantage of the real-time nature of data coming from and going to our IoT devices. This is an ideal fit for our IoT business needs. jBPM supports human-centric, system-centric and hybrid scenarios. In a human-centric scenario, the jBPM system factors in the human element in the business process, putting the person in the center of decision-making and action. This makes it a great match for IoT medical and wearable technology advances.

Where is the human role in this increasingly systemized world? The beauty of IoT and jBPM is that technology becomes an important factor for its users. Systems can guide and advise and leave the most difficult decisions to the experts. Hand in hand, users and their system will be better equipped to provide easier, faster and more optimized service. The integration between IoT devices and jBPM presents a viable solution with a bright future – one that will connect people, things, and systems together as part of business-critical processes as never before.

OptaPlanner – A Constraint resource solver that optimizes use cases such as message routing, employee roistering, vehicle routing, task assignment, and cloud optimization. OptaPlanner is used directly in the EOSpy AI-IoT Drools-jBPM Constraint Solver System. The OptaPlanner is covered extensively in the next chapter, EOSpy AI-IoT OptaPlanner Constraint Solver Reasoning.

5.2.2 Server (EOSpy) - Sensor Processor Map GPS Tracking and Analysis of IoT Data

EOSpy Server Web application (Webapp) is a server application that delivers EOSPY – Executive Order Sensor Processor System GPS information over the internet through a simple Web Browser interface. The EOSPY server main control window ties all location and environment monitoring information on one GPS web browser map screen.

The EOSPY server is designed to support as many tracking devices as possible from popular GPS vendors. EOSPY server also works with many different browsers, including your mobile phone and tablet device browser. EOSPY – Executive Order Sensor Processor System Server mobile application for viewing “Real Time” live GPS tracking information over the internet/mobile cell network.

To install the EOSPY server program on your Windows computer, download the eospy.exe installation program and click on the eospy.exe install program. EOSPY will by default install in the destination location: *C:/Program Files/EOSpy Server* and create a Start Menu folder: EOSPY on your desktop.

To start the EOSPY server, click on the Eagle icon on your desktop. The EOSPY login and map will appear in your browser. The default email and password are both: admin

You can have an unlimited number and combination of EOSPY clients and/or GPS tracking devices in use with the EOSPY Server.

Configure Arduino Tron, EOSpy Client (Android), or GPS Tracking Devices – Many companies make various off-the-shelf GPS tracking devices. Configuring these devices will vary a little from vendors. First, add the new device with a unique identifier into the EOSPY – Executive Order Sensor Processor System Server. Next, configure your device to use the appropriate EOSPY Server IP address and port number. If the device fails to report, check the IP Address and Device ID.

5.2.3 IoT (Arduino Tron) - Android Environment Sensors, IoT Data Collection, and GPS

The EOSPY Client is all of the IoT devices you have in the physical-world that transmit data to the EOSpy Server. These are the Arduino Tron (Sensor and Server versions from Github), Android Client application from the Google Store (standard or TI-SensorTag versions), the GSM/GPRS/GPS tracking devices you can purchase from popular GPS vendors and any other device you buy or build.

EOSpy (Android) Standard and TI-SensorTag Client

The EOSPY-TI SensorTag Client is the GPS tracking automation and the TI BLE SensorTag remote monitoring system is a complete package for business office or personal use. To install the EOSPY Client (Android) application on your phone, download the EOSPY application from the Google App Store. To start the EOSPY Client, click on the Eagle icon on your phone. The EOSPY Client screen will appear. You can also download the EOSPY TI-SensorTag Client version and install on your Android device.

The EOSPY-TI SensorTag Client Android app also sends remote temperature sensor, humidity sensor, ambient light level, SensorTag buttons, and magnetometer information to the EOSPY live map server. EOSPY-TI SensorTag wireless GPS tracking allows you to monitor office systems, personal property, and fleet from anywhere in the world. Receive remote information from any number of events like when an employee arrives on-site to where a vehicle is located. The EOSPY-TI SensorTag reader will read all sensors from the TI-SensorTag Bluetooth LE device. EOSPY-TI will send GPS position and remote sensor TI-SenorTag data for ambient temperature, IR object temperature, humidity sensor, pressure sensor, ambient light, accelerometer, gyroscope, magnetometer, digital microphone, magnetic sensor, and simple button press, magnetometer, and additional information.

To configure a new EOSPY Client, you need to enter the EOSPY server address, domain name, or IP address into the server address. Next, add this device in the EOSPY server by entering the device name and the device identifier. Swipe the service status ON, and YOU'RE DONE. The device will appear on the EOSPY server map the next time the EOSPY Client sends GPS position information.

GSM/GPRS/GPS Tracking Devices

EOSPY supports more than 90 GPS communication protocols and more than 800 models of GPS tracking devices from popular GPS vendors. Many companies make various off-the-shelf GPS tracking devices. Review the list of supported devices for information about your GPS Tracking Device. Configuring these device vary a little from vendors. First, add the new device with a unique identifier into the EOSPY Executive Order Sensor Processor System Server. Next, configure your device to use the appropriate EOSPY server IP address and port number.

EOSpy Arduino Tron (Sensor and Server) Applications

The EOSPY AI-IoTBPM Arduino Tron ESP8266 MQTT telemetry transport Machine-to-Machine (M2M) IoTBPM system is built on two Arduino Tron software components.

- The EOSPY AI-IoT Arduino Tron Sensor software allows you to send MQTT Telemetry Transport information from external WiFi connected Arduino devices to the EOSPY AI-IoTBPM Server.
- The EOSPY AI-IoTBPM Arduino Tron Agent software interface allows you to send commands with the EOSPY AI-IoTBPM Server to control external Arduino connected devices.

To install the EOSPY Arduino Tron application on your Arduino Device, download the EOSPY Arduino Tron Sensor/Server application from GIT.

The Arduino Tron WiFi MQTT was designed as an extremely lightweight pub/sub messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, and in a range of home automation and small device scenarios.

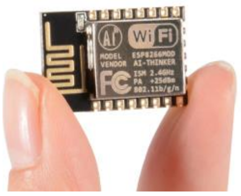
In a decentralized approach, Arduino Tron IoT devices (sensors and actuators) can work together to execute parts of business processes, reducing the number of exchanged messages and promoting central process engine scalability, since information is processed locally and then forwarded.

The Arduino Tron Agent, IoTBPM processes are provided as web services and, in this way, can also be integrated with business processes. The Arduino Tron Sensor, IoTBPM processes and sends information automatically to the EOSpy Server at intervals during the IoTBPM devices loop sequence of operation. Also, the Arduino Tron Sensor can also send information in response to an event or action.

The Arduino Tron (Sensor and Server) software is covered extensively in the next section, EOSpy Arduino Tron ESP8266 MQTT telemetry transport software.

5.3 Arduino Tron ESP8266 MQTT Telemetry Transport

- **Arduino Tron – Smart Micro-Miniature IoT WiFi MQTT Devices**



Arduino Tron – A Miniature Smart Arduino ESP8266 MQTT Telemetry Transport WiFi NodeMCU Device. The Arduino Tron smart microdevice is about the size of a quarter and can fit into the smallest spaces in your equipment cabinets. The Arduino Tron microdevice can send alerts on equipment failures, faults or service conditions. This keeps you in constant contact with your equipment, employees, assets, field equipment, and provides you instant alerts to conditions.

The EOSpy (Executive Order Sensor Processor System) Arduino Tron is an excellent, proven, pre-built, platform application that allows you to use a mobile Arduino MQTT telemetry transport device. It can report location and additional information to EOSpy at selected time intervals. The EOSpy / Arduino Tron application provides a solid platform that your company can build its own custom dedicated application on using an internet-connected or mobile cell phone network connected Arduino device for GPS location, environment, and condition or reporting device (board) error and status information remotely.

In addition to the Arduino NodeMCU sensors such as light sensor, pressure sensor, temperature, and humidity sensor, the Arduino NodeMCU can receive action commands from EOSpy/HTTP for 7 seg display, relay, buzzer, infrared emitter, LED, motor actuators, gate access, auditable speech, alarms.

Executive Order provides custom software and hardware development for Arduino MQTT, NodeMCU, (ESP8266 WiFi microcontroller), low cost, smart IoT, WI-FI Enabled, Lua 5.1.4 devices. We also design, assemble, produce, and test electronic components and printed circuit board (PCB) assemblies for Original Equipment Manufacturers (OEMs). With the EOSpy AI-IoTBPM Arduino Tron system, we have touched only a small fraction of the design possibilities for IoTBPM devices.

5.3.1 Arduino Tron AI-IoTBPM Drools-jBPM Server

The EOSPY / AI-IoTBPM Arduino Tron Sensor software allows you to interface and send MQTT telemetry transport information from your external connected Arduino devices to the AI-IoTBPM Drools-jBPM Server. The AI-IoT Arduino Tron Agent software uses a WiFi wireless transceiver interface to stream telemetry information to the AI-IoTBPM Drools-jBPM Server for any control module sensors or remote control connected Arduino device.

Arduino Tron Agent - To install the Arduino Tron application on your Arduino Device, download the Arduino Tron Agent Sensor application from the Github website. The Arduino Tron Agent AI-IoTBPM software interface allows you to send commands with the Arduino Tron AI-IoTBPM software to control external Arduino connected devices.

Download Arduino Tron from https://github.com/eodas/IoTBPM

GIT the EOSPY AI-IoTBPM from the source code repository, and Import Existing Maven project.

Update the with WiFi network values for network SSID (name) and network password.

```
// Update these with WiFi network values
const char* ssid     = "your-ssid"; // your network SSID (name)
const char* password = "your-password"; // your network password
```

Update the EOSPY Server IP address and unique unit ID values and add in AI-IoTBPM Server.

```
// Update these with Arduino Tron service IP address and unique unit id
byte server[] = { 10, 0, 0, 2 }; // Set EOSpy server IP address as bytes
String id = "100111"; // Arduino Tron Device unique unit id
```

You need to set a different device unique identifier for each one of your Arduino Tron Sensor ESP8266 MQTT devices. Next, match the Arduino Tron sensor device unique identifier to the device unique unit ID in the AI-IoTBPM Server application. If your Arduino Tron sensor device is reporting or you don’t know your device identifier, you can configure your device first and look at the server log file.

Above are all the fields you need to provide/configure values for, the remaining fields are used in the Arduino Tron sensor application. Also, you may use a DHT11 digital temperature and humidity sensor. See the Arduino Tron sensor sketch for more details and information.

```
const readPushButton // Values for the digitalRead value from gpiox button
const readDHT11Temp  // Values for the DHT11 digital temp/humidity sensor
const readLDRLight // Values for the LDR analog photocell light sensor
const readIRSensor // Arduino values for IR sensor connected to GPIO2
```

To use a DHT11 digital temperature and humidity sensor install the SimpleDHT library and uncommit the lines for dht11 in the Arduino Tron sensor sketch for more details and information. Change the readDHT11Temp value to true. Also, you can change the reporting time by modifying the timeCounter.

EOSPY currently supports these data fields in the Server Event data model

```
id=6 &event=allEvents &protocol=osmand &servertime=<date> &timestamp=<date>
&fixtime=<date> &outdated=false &valid=true &lat=38.85 &lon=-84.35
&altitude=27.0 &speed=0.0 &course=0.0 &address=<street address> &accuracy=0.0
&network=null &batteryLevel=78.3 &textMessage=Message_Sent &temp=71.2
&ir_temp=0.0 &humidity=0.0 &mbar=79.9 &accel_x=-0.01 &accel_y=-0.07
&accel_z=9.79 &gyro_x=0.0 &gyro_y=-0.0 &gyro_z=-0.0 &magnet_x=-0.01
&magnet_y=-0.07 &magnet_z=9.81 &light=91.0 &keypress=0.0 &alarm=Temperature
&distance=1.6 &totalDistance=3.79 &motion=false
```

You can add additional fields to the data model and transmit via any device for EOSpy AI-IoT Drools-jBPM processing. All of these values can be evaluated in the Drools Rules engine and the jBPM process.

5.3.2 AI-IoTBPM Arduino Tron Agent

The Arduino Tron Agent interface allows you to send commands with the Arduino Tron AI-IoTBPM software to control external IoT WiFi connected devices. The AI-IoTBPM Arduino Tron Agent software uses a WiFi wireless transceiver interface to control and interact with module sensors and remote control devices. You can control any device from the EOSpy AI-IoTBPM Arduino Tron Agent software or stream any interface over the WiFi internet. With the EOSpy AI-IoTBPM Arduino Tron Agent software you can automatically turn on lights, appliances, cameras, and lock/unlock doors from the Drools-jBPM Expert System processing model.

To configure the Arduino Tron Agent software, update the with WiFi network values for network SSID (name) and network password. Start the Arduino IDE Serial Monitor window to see what EOSpy AI-IoTBPM Arduino Tron Agent code is doing via the serial connection from ESP-01 via 3.3v console cable to your PC. The EOSpy AI-IoT Arduino Tron Agent will use this URL (http://) to connect address.

Set this URL address in the *eospy.properties* file *arduinoAgent=http://10.0.0.2* to configure the connection to the EOSpy AI-IoTBPM Arduino Tron Agent. This is the URL that all EOSpy AI-IoTBPM Arduino Tron Agent commands will be transmitted from the EOSpy AI-IoTBPM Drools-jBPM program.

You can use a command like the following to send actions to the EOSpy AI-IoTBPM Arduino Tron Agent.

```
com.iotbpm.server.AgentConnect.getInstance().sendPost("/LED3=ON");
```

The EOSpy AI-IoTBPM Arduino Tron Agent NodeMCU can receive action commands from EOSpy via HTTP URL post for 7 Seg display, relay, buzzer, infrared emitter, LED, motor actuators, gate access, auditable speech, alarms, or any other device via an HTTP URL post command.

We can use a relay in this Arduino circuit to switch an electric device on and off. A relay is an electromagnetic switch, which is controlled by a small current, and used to switch ON and OFF relatively a larger current. A relay is a good example of controlling the devices, using a smaller Arduino device.

Executive Order Corp. will be releasing additional sketches like the LED and power relay example for use with the EOSpy AI-IoTBPM Arduino Tron Agent application. Be sure to follow us on Github for future Arduino sketch and build ideas.

5.4 EOSpy AI-IoT Drools-jBPM Installation Configure

5.4.1 EOSpy AI-IoT Configure Eclipse, Drools-jBPM and BPMN2

This quick guide provides installation and configuration instructions for the EOSPY AI-IoTBPM program, Eclipse IDE, Eclipse Plugins, Drools-jBPM and BPMN2 on your Windows computer.

- Download and install the "Eclipse IDE for Java Developers"
- Use the Eclipse feature to add new software, which is available on the Eclipse menu "Help -> Install New Software." Select the "Add" option and these two packages:

  Drools + jBPM Update Site 7.15.0
  http://downloads.jboss.org/jbpm/release/7.15.0.Final/updatesite/

  BPMN2-Modeler 1.4.2
  http://download.eclipse.org/bpmn2-modeler/updates/oxygen/1.4.2/

The Executive Order Corp Arduino Tron ESP8266 MQTT telemetry transport Machine-to-Machine (M2M) IoTBPM software and Arduino Tron MQTT AI-IoTBPM Client using EOSpy AI-IoTBPM Drools-jBPM latest software can be download from the Github website.

- Download Arduino Tron from https://github.com/eodas/IoTBPM
- GIT the EOSPY AI-IoTBPM from the source code repository and import existing maven project.

Once installed you can configure the different runtime properties in the *eospy.properties* file. It is easier just to uncomment the runtime environment that you intend to execute. Note: If you have an Arduino Tron Server, update the arduinoAgent IP address; otherwise, just leave this committed out.

In Eclipse, run the class EOSpy AI-IoTBPM as a Java application. This is the main class for EOSpy AI-IoTBPM Drools-jBPM expert system.

EOSPY Client – To install the EOSPY client application on your phone, download the EOSPY application from the Google App Store. To start the EOSPY client, click on the Eagle icon on your phone. The EOSPY client screen will appear. You can also download the EOSPY TI-SensorTag client version.

To configure a new EOSPY client, you will need to enter the AI-IoTBPM server address, domain name, or IP address into the server address. Next, add this device in the AI-IoTBPM server by entering the device name and the device identifier. Swipe the service status **ON,** and YOU'RE DONE. The device will appear on the EOSPY server map the next time the EOSPY client sends GPS position information.

GPS Tracking Devices – Many companies make various off-the-shelf GPS tracking devices. Configuring these devices will vary a little from vendors. First, add the new device with a unique identifier into the EOSPY – Executive Order Sensor Processor System Server. Next, configure your device to use the appropriate EOSPY server IP address and port number. If the device fails to report, check the IP address and device ID.

Device Unique Identifier

If you don't know your device identifier, you can configure your device first and look at the server log file. When the server receives a message from an unknown device it writes a record containing the unique identifier of a new device. Look for records like "Unknown device – 123456789012345"; "Unknown device" 123456789012345 is your new device identifier.

Address and Port

To select the correct port, find your device in the list of supported devices. The port column of the corresponding row contains default port numbers for your device. If you want to use variations from the default ports, you can change them in the configuration file.

EOSPY supports more than 90 GPS communication protocols and more than 800 models of GPS tracking devices from popular GPS vendors. Review the list of supported devices for information about your GPS tracking device on www.eospy.com

5.5 Arduino Tron AI-IoT Drools-jBPM Installation Configure

5.5.1 Arduino Tron IoT AI-Artificial Intelligent Streamline Automation

The Arduino Tron allows you to send IoT sensor data information directly to the AI-IoT Drools-jBPM Expert System from any Arduino device. This provides a very lite streamline IoT to Drools-jBPM (Business Process Management) application process with or without GPS LAT/LON, speed, bearing, and altitude positioning information. This makes for a very efficient IoT Drools-jBPM Expert System.

If you have fixed IoT sensors monitoring conditions/environments or IoT devices that operate equipment, then the Arduino Tron is all that is needed. The IoT devices send their WiFi information directly to the Arduino Tron Agent. The Arduino Tron Agent then processes the IoT data through the Drools Inference Engine and the jBPM process model.

With Arduino Tron jBPM-BPMN modular it allows us to define both the business processes and IoT devices behavior at the same time using one diagram. With Arduino Tron adding just Drools-jBPM to IoT, we make the IoTBPM devices "smart". This technique also helps Drools to process a large number of objects more efficiently and is designed for high volumes of data.

Arduino Tron allows you to send IoT sensor data and information directly to the AI-IoTBPM Drools-jBPM Expert System from the Arduino device. This provides a very lite streamline IoT to Drools-jBPM (Business Process Management) application process with sensor and GPS positioning information.

This quick section helps you install and configure the Arduino Tron – Executive Order Sensor Processor System components.

Executive Order Arduino Tron has several components to provide a complete AI-IoTBPM system:

- **Arduino Tron AI-IoTBPM (Java):** Internet of Things Drools-jBPM Expert System.
- **Arduino Tron Sensor (Arduino):** Application to send sensor MQTT Telemetry Transport.
- **Arduino Tron Agent (Arduino):** Software to control external Arduino connected devices.
- **Arduino Tron Server (Arduino):** Web Server Interface for Controlling all IoT devices.

You can have an unlimited number and combination of Arduino Tron IoTBPM Devices and/or EOSPY GPS Client tracking devices in use with Arduino Tron AI-IoT.

(Optionally, you can download EOSPY server from our website www.eospy.com and Download EOSPY GPS client from the Google Store, standard or TI-SensorTag version.)

5.5.2 Arduino Tron AI-IoT Configure Eclipse, Drools-jBPM and BPMN2

This quick guide provides installation and configuration instructions for the Arduino Tron AI-IoTBPM program, Eclipse IDE, Eclipse Plugins, Drools-jBPM and BPMN2 on your windows computer.

- Download and install the "Eclipse IDE for Java Developers."
- Use the Eclipse feature to add new software, which is available on the Eclipse menu "Help -> Install New Software". Select the "Add" option and these two packages:
  Drools + jBPM Update Site 7.15.0
  http://downloads.jboss.org/jbpm/release/7.15.0.Final/updatesite/
  BPMN2-Modeler 1.5.0
  http://download.eclipse.org/bpmn2-modeler/updates/oxygen/1.5.0/

The Executive Order Corp Arduino Tron ESP8266 MQTT telemetry transport Machine-to-Machine (M2M) IoTBPM software and Arduino Tron MQTT AI-IoTBPM Client using EOSpy AI-IoTBPM Drools-jBPM latest software can be download from the Github website.

- Download Arduino Tron from https://github.com/eodas/IoTBPM
- GIT the Arduino Tron AI-IoTBPM from the source code repository and import existing maven project into Eclipse IDE project.

Once installed you can configure the different runtime properties in the *arduinotron.properties* file. It is easier just to uncomment the runtime environment that you intend to execute. In Eclipse, run the class ArduinoTron as a Java application. This is the main class in ArduinoTron AI-IoT Drools-jBPM system.

5.6 Arduino Tron IoTBPM Sensor Software Suite

5.6.1 Arduino Tron Sensor Send MQTT Telemetry Transport and Web Serer

The power of the IoT (Internet of Things) device increases greatly when business process (jBPM) can use them to provide information about our real-world as well as execute IoT devices as part of our business process. The jBPM-BPMN modular allow us to define both the business processes and IoT devices behavior at the same time using one diagram. With Arduino Tron adding Drools and jBPM to IoT, we make the IoT devices "smart". Moving beyond just collecting IoT data and transitioning, to leveraging the new wealth of IoT data, to improving the SMART decision making is the key. The Executive Order Arduino Tron AI-IoTBPM will help these IoT devices, environments, and products to self-monitor, self-diagnose and eventually, self-direct.

Arduino Tron allows you to send IoT sensor data and information directly to the AI-IoTBPM Drools-jBPM Expert System from the Arduino device. This provides a very lite streamline IoT to Drools-jBPM (Business Process Management) application process with sensor and GPS positioning information.

The Arduino Tron MQTT sensor software allows you to interface and sends MQTT Telemetry Transport Information from your external connected Arduino devices to the Arduino Tron AI-IoT Drools-jBPM server. The AI-IoTBPM Arduino Tron Agent software uses a WiFi wireless transceiver interface to stream telemetry information to the Arduino Tron Agent for any control module sensors or remote control connected Arduino device.

5.6.2 Arduino Tron IoT Sensor

Arduino Tron Sensor – To install the Arduino Tron application on your Arduino device, download the Arduino Tron Sensor application from GIT. Update the with WiFi network values for network SSID (name) and network password. Update the Arduino Tron Agent IP address and unique unit ID values.

Also, you may use a DHT11 digital temperature/humidity sensor and other sensors (see the Arduino Tron Sensor sketch for more details and information).

5.6.3 Arduino Tron IoT Sensor Emulator

Arduino Tron IoT Sensor Emulation - Prototype an Arduino-based low-power WiFi Internet of Things (IoT) device with built-in sensors emulation that could be used to deliver sensor data from any location in the world, and potentially control connected devices such as thermostats, lights, door locks, and other automation products.

Use a serial monitor to emulate sensor input values to Arduino Tron. This will allow you to prototype the final IoT device before custom-designing the PCB (printed circuit board).

5.6.4 Arduino Tron Agent Control External Arduino Connected Devices

The Arduino Tron Agent AI-IoTBPM software interface allows you to send commands with the Arduino Tron AI-IoTBPM Server software to control external Arduino connected devices. The IoTBPM Arduino Agent software uses a WiFi wireless transceiver interface to control and interact with module sensors and remote control devices.

You can control any device from the Arduino Tron Agent software or stream any interface over the WiFi internet. With the Arduino Tron Agent software, you can automatically turn on lights, appliances, cameras, lock/unlock doors, and open doors from the Drools-jBPM expert system processing.

5.6.5 Arduino Tron Web Server Interface for Controlling IoT Devices

Arduino Tron Web Server provides an IoT dashboard, management, and control for remote management of your Internet-Enabled IoT devices. The Arduino Tron Web Server operates completely on an Arduino ESP-01, integrated low power 32-bit CPU RISC processor, on-chip memory and integrated WiFi 802.11 b/g/n. The ESP-01 Arduino Tron smart microdevice is about the size of a quarter.

The beauty of the Arduino Tron Web Server is that the IoT device technology becomes interactive with humans through a simple Web Server. With the Arduino Tron Web Server getting your IoT project working in the cloud is a fast-easy solution. We will discuss the Arduino Tron Web Server in Chapter 6.

5.7 Arduino Tron AI-IoT Drools-jBPM Application Examples

5.7.1 Movement AI-IoT Drools-jBPM Artificial Intelligent Smart Automation

In the movement example, we will demonstrate how to invoke **business rules** from within our application and how to execute our jBPM processes and how to handle the interactions between **process** and **rules** using an IoTBPM sensor device.

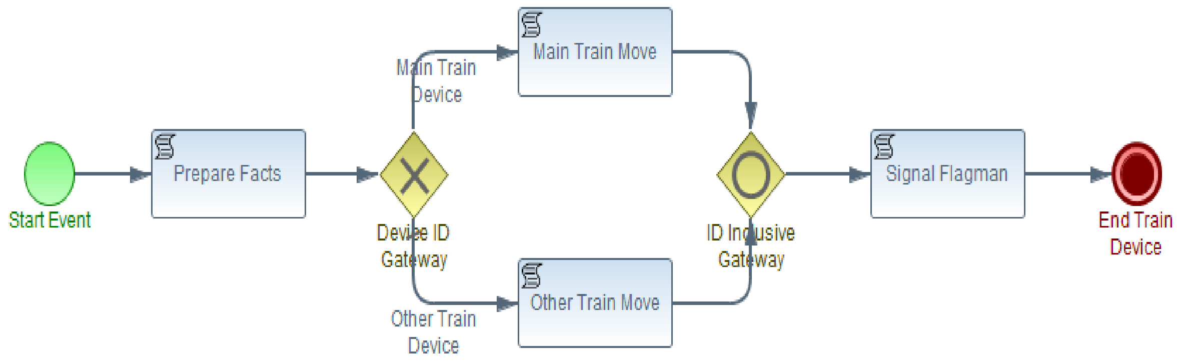
Business processes and rules are two core concepts which are defined as follows:

- **Business processes:** Represents what the business does.
- **Business rules:** Represents decisions that the business makes.

Although processes and rules are two different things there is a clear advantage if your end users are allowed to combine processes and rules. This means for example:

- Rules can define which processes to invoke
- Rules can specify decisions in that process
- Rules can augment (or even override) the behavior specified in the process (for example to handle exception cases)
- Assignment rules can be used to assign actors to (human) tasks
- Rules can be used to dynamically alter the behavior of your process

Example 5.4.2.1. EOSpy-AI Train Movement jBPM



In this example we will use EOSpy AI-IoT Drools-jBPM to solve a knowledge and notification problem. We have a train yard with IoT devices installed on the trains to notify us if they are in motion or not. Our rule is, if none or one train is moving, the train yard signal flagman can signal that the yard is safe. If more than one train is moving, then the signal flagman signals that the trains must move slowly.

With this use of IoT Drools-jBPM analysis and reasoning in IoT devices, we were able to orchestrate dissimilar devices (trains) that normally have no awareness of each other. This creates opportunities for direct integration of computer-based into the physical-world that has never been available before.

This IoT orchestration of devices gives us the ability for action after our AI decision. Also, we are able to use IoT devices to update mechanical trains, giving them new awareness of each other. Of course, we could improve on our IoT Drools-jBPM application and update each train of the others proximity. This would allow us to implement collision avoidance like what is available in automobiles.

Example 5.4.2.2. Signal Train Motion Rule

```
rule "Move-Stop Train Motion Rule"
     no-loop
   when
     // event keyPress_1 used to change state Move/Stop
      $event : ServerEvent( $eventId : id, (event == "keyPress_1" || event
== "keyPress_2") )
```

```
            $device : Devices( $deviceId : id,  $eventId == id )
        then
                    String result = ($device.status == "Move") ? "Stop" : "Move";
                    $device.status = result;
                    $device.action = "Ok";
                    com.arduinotron.ui.MainWindow.getInstance().log("Train Device "
    + $device.getName() + " - " + result);
                    modify( $device );
                    update( $device );
    end
```

This rule example demonstrates the use of a pushbutton signal as a switch. Each time you press the IoT button, the state is turned on (if it's off) or turned off (if on). This also demonstrates how we can repurpose a legacy signaling device to operate with new or different behavior.

Example 5.4.2.3. Train Motion Rules

```
    rule "More Than One Train is Moving - Slow Rule"
        when
                accumulate ( $device : Devices( $deviceId : id, $device.status
    == "Move" ); $cnt : count(1); $cnt > 1 )
    then
                com.arduinotron.ui.MainWindow.getInstance().log("<Slow> More
    Than One Train is Moving.");
    end
```

The train motion rules are all relatively similar, counting the number of trains reporting move status. By delegating important decisions to be taken into your rules system, the **business processes** become much more resilient to change. This IoTBPM example provides a "**smart**" environment for the train yard or AI-IoT Drools-jBPM artificial intelligent smart automation.

The signal flagman indicates a task that needs to be executed by human actors. The Drools-jBPM business processes management human interface and system interaction for the train yard signal flagman could be a light indicator or text message telling him what to signal for the yards condition.

Drools-jBPM supports a special human task node inside processes for modeling this interaction with human users. This human task node allows process designers to define the properties related to the task that the human actor needs to execute. In our example, we use a script task to execute our human task in our business process engine, for simplicity. When the task is ready to start, the engine will execute the script. When the script is completed, the task will also be completed.

With this use of IoT Drools-jBPM analysis and reasoning in IoT devices, we were able to orchestrate dissimilar devices (trains) that normally have no knowledge or awareness of each other. This demonstrates opportunities for direct integration of computer-based into the physical-world that has never been available before by repurposing legacy systems or mechanical systems instead of replacing the hardware or physical machine.

In the movement jBPM example, the IoTBPM device ID is used in the device ID gateway to differentiate between the main train and other train movement commands. This gives us the ability to have special commands in our business process for the movement of the main train in our train yard.

We currently support these additional data fields in the jBPM server event data model:

name, getName()	event, getEvent()
adderss, getAddress()	textMessage, getTextMessage()
temp, getTemp()	light, getLight()
keyPress, getKeypress()	alarm, getAlarm()

You can add additional fields to the data model and transmit via any device for EOSpy AI-IoT Drools-jBPM processing. All of these values can be evaluated in the Drools rules engine and the jBPM process.

5.7.2 Illuminance AI-IoTBPM Drools-jBPM Arduino Tron Agent Automation

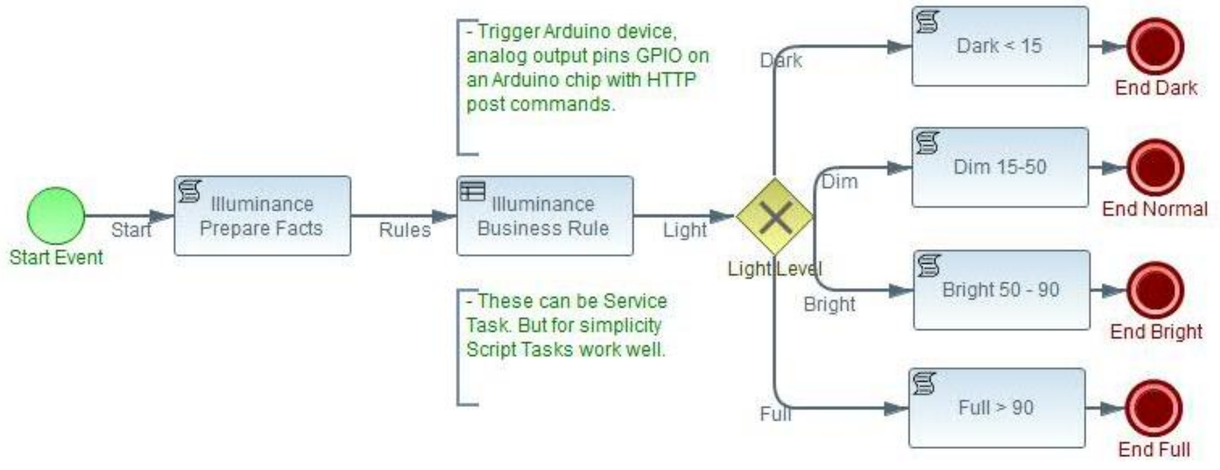
In the previous example, we looked at the use of EOSpy AI-IoT Drools-jBPM to solve a knowledge and notification problem. Now, let's look at an end-to-end AI-IoT Drools-jBPM example. In the illuminance example, we will demonstrate how to invoke Drools **business rules** from within our jBPM executing processes and how to handle the interactions between **process** and **rules**.

We will use the EOSpy IoT client to stream remote sensor information to the EOSpy AI-IoT Drools-jBPM application. Then we will use the AI-IoT Drools-jBPM on this IoT information stream to decide what behavior and business process functions to execute. Finally, we will execute the AI-IoTBPM Drools-jBPM decision using the EOSPY AI-IoT Arduino Tron Agent software interface, which allows us to send commands with the EOSPY AI-IoT software to control external Arduino connected IoT devices.

The AI-IoTBPM Arduino Tron Agent software uses a WiFi wireless transceiver interface to control and interact with module sensors and remote control devices. You can control any device from the EOSpy AI-IoT Arduino Tron Agent software or stream any interface over the WiFi internet. With the EOSpy AI-IoTBPM Arduino Tron Agent software, you can automatically turn on lights, appliances, cameras, and open doors from the Drools-jBPM Expert System processing model.

This gives us a complete IoT, to Drools-jBPM, to IoT, end-to-end process example. By streaming IoT information directly into our business process management application we can modify IoT behavior and functionality without having to physically change our IoT devices.

Example 5.5.2.1 Illuminance AI-IoTBPM Drools-jBPM


- Trigger Arduino device, analog output pins GPIO on an Arduino chip with HTTP post commands.
- These can be Service Task. But for simplicity Script Tasks work well.
Start Event
Start
Illuminance Prepare Facts
Rules
Illuminance Business Rule
Light
Light Level
Dark
Dim
Bright
Full
Dark < 15
Dim 15-50
Bright 50 - 90
Full > 90
End Dark
End Normal
End Bright
End Full

The EOSpy AI-IoTBPM Arduino Tron Agent NodeMCU can receive action commands from EOSpy via HTTP URL post for 7 Seg display, relay, buzzer, infrared emitter, LED, motor actuators, gate access, auditable speech, alarms, or any other device via an HTTP URL post command. You can use a command like the following to send actions to the EOSpy AI-IoTBPM Arduino Tron Agent:

```
com.arduinotron.server.AgentConnect.getInstance().sendPost("/LED3=ON");
```

In this illuminance example, we will use EOSpy AI-IoTBPM Drools-jBPM to manage the light level in an office. Our IoTBPM sensor will use a photoelectric sensor to set the light field to the illuminance level. If it's too dark, we want to turn on the light. If it's too bright, we want to draw the blinds. If the light level is less than 10, we will send a signal to the lamp switch to turn on. If the light level is greater than 100, we will send a signal to the curtain motor to close the blinds.

With this use of IoT Drools-jBPM analysis and reasoning in IoTBPM devices, we are able to orchestrate dissimilar devices that normally have no awareness of each other. This creates opportunities for direct integration of computer-based into the physical-world and allows us the ability to control IoTBPM behavior outside of the device. This allows devices to be deployed and modified or updated at any time.

After our IoTBPM illuminance light level is read, we use the illuminance business rule group to call and execute our rules for this process. A business rule task provides a mechanism for the process to provide input to a business rules engine and to get the output of calculations that the business rules engine might provide. This rule flow group identifies the IoTBPM device that is reporting the light level.

Example 5.4.2.2. Illuminance Rule Flow Group

```
//declare any rules to fire in the jBPM AI_IoT.Illuminance process
rule "Rule for device ID: 100111"
      ruleflow-group "Illuminance Rule Flow"
      when
            $event : ServerEvent( $eventId : id, $eventId == "100111" )
      then
            com.arduinotron.ui.MainWindow.getInstance().log("ruleflow-group
device ID: 100111 " + $event.getEvent() + " - " + $event.getName());
end
```

In the rules process class we execute a fileAllRules() prior to executing startProcess() for the jBPM.

```
// go! - fire all rules
long noOfRulesFired = this.kSession.fireAllRules();

// Start the process with knowledge session
instance = kSession.startProcess(processID, params);
```

This allows us to do some AI reasoning (discovery) prior to executing our business process. In this example, this allows us to evaluate the IoTBPM event in updated UI display with the information.

Example 5.4.2.3. IoTBPM Evaluate Event Rule

```
//declare rules to fire when fileAllRules() is executed
rule "Key Press 1 Rule"
    when
        $event : ServerEvent( $eventId : id, event == "keyPress_1" )
        $device : Devices( $deviceId : id,  $eventId == id )
    then
            $device.action = "Keypress 1";
            $device.status = "Light " + $event.getLight();
            com.arduinotron.ui.MainWindow.getInstance().log("Rule Key Press
1: " + $event.getEvent() + " - " + $event.getName() + " - " + " Light Value:
" + $event.getLight());
            modify( $device );
end
```

Set this URL address in the *eospy.properties* file *arduinoAgent=http://10.0.0.2* to configure the connection to the EOSpy AI-IoTBPM Arduino Tron Agent.

We then use a command like the following to send actions to the EOSpy AI-IoTBPM Arduino Tron Agent for the Dark Script Task, where we will send a signals to the lamp switch to turn on, and the Bright Script Task, where we will send a signal to the curtain motor to close the blinds.

```
com.arduinotron.server.AgentConnect.getInstance().sendPost("/LED3=ON");
```

The EOSpy AI-IoTBPM Arduino Tron Agent NodeMCU can receive action commands from EOSpy via HTTP URL post for 7 seg display, relay, buzzer, infrared emitter, LED, motor actuators, gate access, auditable speech, alarms, or any other device via an HTTP URL post command.

We can use a relay in this Arduino circuit to actually switch an electronic device on and off. A relay is an electromagnetic switch, which is controlled by a small current, and used to switch ON and OFF relatively a larger current. A relay is a good example of controlling the devices using a smaller Arduino device.

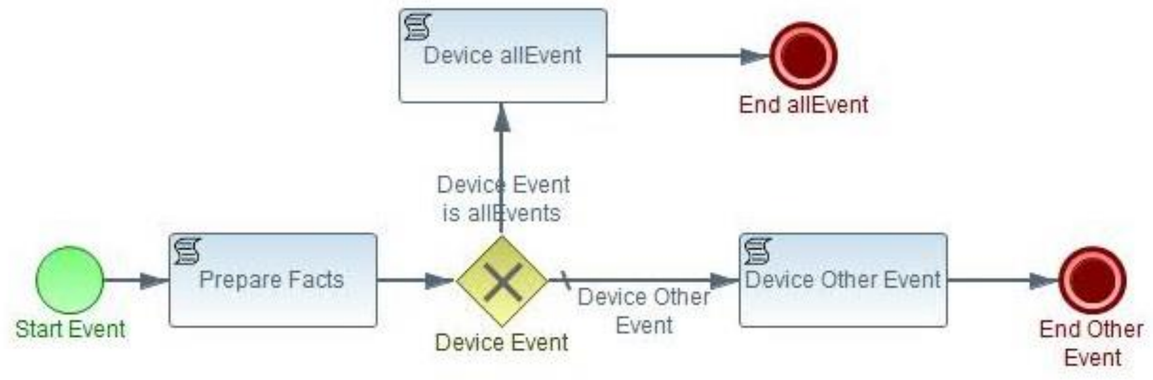
5.7.3 Gyroscope AI-IoTBPM Drools-jBPM Motion Sensors

Depending on the device, several sensors can provide information that lets you monitor the motion of the device. The sensors usually include an accelerometer (accel), gyroscope (gyro) and magnetometer (magnet) sensors. Motion sensors are useful for monitoring device movement such as tilt, shake, rotation, or swing. There are also additional sensors such as gravity, linear acceleration, rotation vector, significant motion, step counter, and step detector sensors.

By processing our IoT information/data through the jBPM business process management application we can modify IoT behavior and functionality, without having to physically change our IoT devices.

In the gyroscope example, we will examine the IoT device magnetometer (magnet) sensors to determine if the IoT device is upside-down. This would be important information for a vehicle, shipping container, medical supplies, or any other item that could be damaged or dangerous if turned over.

Example 5.5.3.1 Gyroscope Process AI-IoTBPM Drools-jBPM



In the jBPM process, in the following we look at the IoT device event that raised our notification. This could be a significant IoT reported event like airbags deployed, fire alarm, or hazardous materials.

The gyroscope rules will examine the IoT device magnetometer (magnet) sensors and determine if the device is 'face up' or 'face down'.

Example 5.4.2.3. Gyroscope IoTBPM Evaluate Event Rule

```
//declare rules to fire in for zBias
rule "Device Gyroscope Positive zBias"
      when
   $event : ServerEvent( $eventId : id, $magnet_z : magnet_z, $magnet_z > 0 )
      $device : Devices( $deviceId : id,  $eventId == id )
      then
            $device.action = $event.getEvent();
            $device.status = "Face Up";
            modify( $device );
end
```

You can try this example and, of course, any of the others by installing the EOSPY Client Android on your phone. To install the EOSPY Client application on your phone, download the EOSPY application from the Google App Store. To start the EOSPY Client, click on the Eagle icon on your phone. The EOSPY Client screen will appear. You can also download the EOSPY TI-SensorTag Client version.

Then you can turn your phone over and watch the state change in the gyroscope rules will examine.

Also, with the IoT client device, you can send custom textMessage, getTextMessage() and alarm, getAlarm() information to the EOSpy AI-IoT Drools-jBPM application for evaluation in the Drools-jBPM.

5.7.4 GPS Position AI-IoTBPM Drools-jBPM Artificial Intelligent GPS Automation

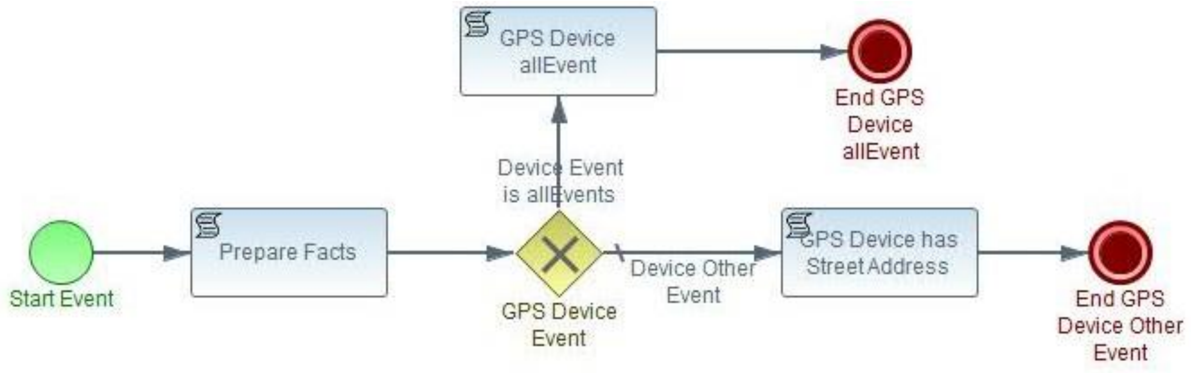
There are many IoT GPS options to choose from with EOSpy. First, you don't have to use a GPS module. If you have a fixed IoT device, consider hardcoding your GPS LAT/LON position. Then, if you have IoT device monitoring, for instance, a door open, you could pin that device location to a building map. This would give you a visual indicator of the location of which door was open on your building blueprint.

Look at the EOSPY AI-IoTBPM Arduino Tron Sensor sketch for an example of how you can find LAT/LON from an address and hardcoding a GPS LAT/LON position-location to the Arduino Tron application.

```
//Update these with LAT/LON GPS position values
//You can find LAT/LON from an address
https://www.latlong.net/convert-address-to-lat-long.html
String lat = "38.888160"; // position LAT
String lon = "-77.019868"; // position LON
```

The Arduino device has many options for GPS modules. The Arduino has methods to read information like date, time, location and satellites in view from the standard NMEA data streams. This NMEA data streams gives you the LAT/LON position to set in your LAT/LON data packet.

Example 5.5.4.1 GPS Position AI-IoTBPM Drools-jBPM



In the GPS position example, we can examine the IoT device event that transmitted the GPS position. This could be a significant IoT event reported from the device, such as motor started, airbags deployed, overheat, outside Geofence or over-speed.

Example 5.5.4.2.GPS Position Speed Over 60 Rule

```
//declare rule to fire when Device speed is over 60
rule "GPS Device Speed Alert"
      when
        $event : ServerEvent( $eventId : id, $lat : lat, $lon : lon,
        $altitude : altitude, $speed : speed, $speed > "60" )
      $device : Devices( $deviceId : id,  $eventId == id )
      then
            $device.action = "Alert";
            $device.status = $event.getEvent();
            com.arduinotron.ui.MainWindow.getInstance().log($event.getName()
+ " GPS Speed Alert lat:" + $lat + " lon:" + $lon + " speed:" + $speed);
            modify( $device )
end
```

This GPS position rule fires when the GPS device reported speed is greater than 60. The rule reports the position and speed that fired the rule. Depending on the GPS devices, the module can report additional information that is transmitted along with the LAT/LON data packet. This information may be different depending on the device type, environment, and conditions.

- **GPS Device Module Environment and Condition Rules**

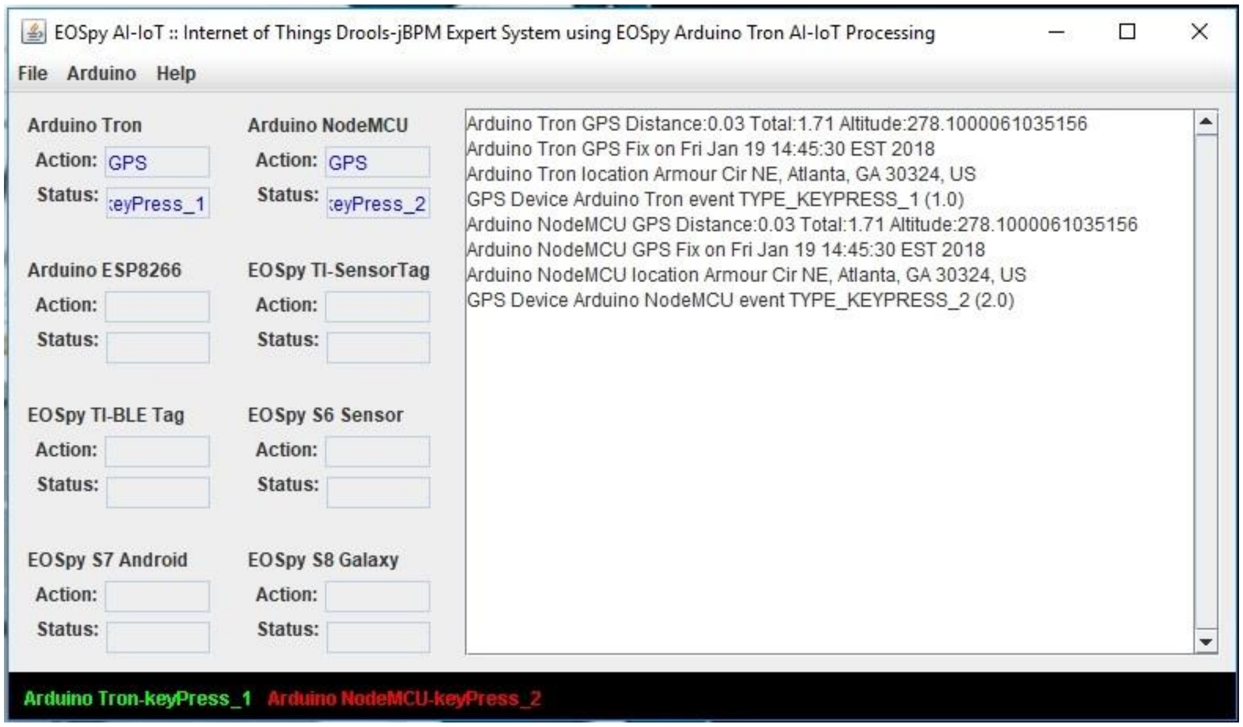
There are additional rules that respond to the GPS device, if the IoT device sends a text messages or if the device sent an alarm.

Example 5.5.4.2.GPS Device Send Alarm

```
//declare rule to fire when GPS Device sends Alarm message
rule "GPS Device Sent Alarm"
        when
          $event : ServerEvent( $eventId : id, $alarm : alarm, $alarm != null )
        $device : Devices( $deviceId : id,  $eventId == id )
        then
                com.arduinotron.ui.MainWindow.getInstance().log($event.getName()
+ " Sent Alarm: " + $alarm);
end
```

In addition, there are rules for processing a GPS device text message; the GPS device fixes information and GPS device address location. All of the information and messages are transmitted along in the LAT/LON data packet from the IoT device.

Example 5.5.4.3 GPS Position AI-IoTBPM Drools-jBPM



Arduino Tron Action: GPS Status: keyPress_1
Arduino NodeMCU Action: GPS Status: keyPress_2

```
Arduino Tron GPS Distance:0.03 Total:1.71 Altitude:278.1000061035156
Arduino Tron GPS Fix on Fri Jan 19 14:45:30 EST 2018
Arduino Tron location Armour Cir NE, Atlanta, GA 30324, US
GPS Device Arduino Tron event TYPE_KEYPRESS_1 (1.0)
Arduino NodeMCU GPS Distance:0.03 Total:1.71 Altitude:278.1000061035156
Arduino NodeMCU GPS Fix on Fri Jan 19 14:45:30 EST 2018
Arduino NodeMCU location Armour Cir NE, Atlanta, GA 30324, US
GPS Device Arduino NodeMCU event TYPE_KEYPRESS_2 (2.0)
```

Arduino Tron-keyPress_1 Arduino NodeMCU-keyPress_2

In the GPS position example, we can see the rules for GPS device distance, GPS device fix information, and GPS device address location being fired. If the GPS module transmitted additional information in the &alarm= or &textMessage= fields, then those rules would fire.

The jBPM Script Task node; GPS Device has street address runs a conditional Java script task.

```
log("GPS Device " +  kcontext.getVariable("name") + " event TYPE KEYPRESS 2 (2.0)");
```

A script task is executed by our business process engine. The modeler or implementer defines a script in a language that the engine can interpret. When the task is ready to start, the engine will execute the script. When the script is completed, the task will also be completed.

Script tasks can run action tasks and very complex evaluations and are convenient for combining data.

- **Android Things System on Module (SoM)**

Android Things let you build professional, mass-market products on a trusted platform, without previous knowledge of embedded system design. It reduces the large, upfront development costs and the risks inherent in getting your idea off the ground.

Android Things hardware provides a turnkey hardware platform to build your IoTBPM device. Android certified development boards based on System on Module (SoM) architecture give you the following benefits to get you started quickly:

- Integrated Parts - SoMs integrate the SoC (System-on-Chip), RAM, flash storage, WiFi, Bluetooth, and other components onto a single board and come with all of the necessary FCC certifications. When you want to mass produce your device, you can optimize your board design by flattening existing modules onto a PCB to save costs and space.
- A Google BSP - The Board Support Package (BSP) is managed by Google. Therefore, that means you don't have to do kernel or firmware development. This gives you a trusted platform to develop on with standard updates and fixes from Google.
- Differentiated Hardware - Our partners provide development boards with different SoMs and form factors to suit your needs, giving you choice and flexibility. When you're ready, take your prototypes to products by customizing them to fit a specific form-factor, all while running the same software.

Android Things SDK extends the core Android framework with additional APIs provided by the things support library, which lets you integrate with new types of hardware not found on mobile devices.

Developing apps for embedded devices is different from mobile in a few important ways such as:

- More flexible access to hardware peripherals and drivers than mobile devices
- System apps are not present to optimize startup and storage requirements
- Apps are launched automatically on startup to immerse your users in the app experience
- Devices expose only one app to users, instead of multiple, like with mobile devices

- **GPS Tracking Devices**

Many companies make various off-the-shelf GPS Tracking devices. Configuring these devices will vary a little from vendors. First, add the new device with a unique identifier into the EOSPY – Executive Order Sensor Processor System Server. Next, configure your device to use the appropriate EOSPY Server IP address and port number. If the device fails to report, check the IP Address and Device ID.

Device Unique Identifier - For most devices, you should use an IMEI (International Mobile Equipment Identity) number as a unique identifier. However, some devices have vendor-specific unique identifiers, for example, TK-103 devices use 12-digit identifier.

If you don’t know your device identifier, you can configure your device first and look at the server log file. When the server receives a message from an unknown device it writes a record containing a unique identifier of a new device. Look for records like “Unknown device – 123456789012345”; “Unknown device” 123456789012345 is your new device identifier.

Address and Port - To select the correct port, find your device in the list of supported devices. The port column of the corresponding row contains default port numbers for your device. If you want to use variations from the default ports, you can change them in the configuration file.

EOSPY supports more than 90 GPS communication protocols and more than 800 models of GPS tracking devices from popular GPS vendors. Review the list of supported devices for information about your GPS Tracking Device.

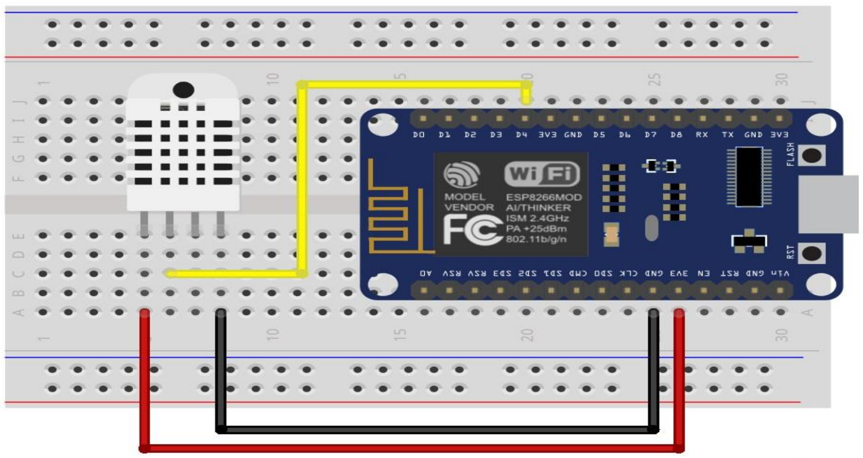
EOSpy live map server will display additional sensors data that you need to automate your processes and gain visibility into your business operations. This additional data could be ambient temperature, IR object temperature, humidity sensor, pressure sensor, ambient light, accelerometer, gyroscope, magnetometer, digital microphone, magnetic sensor, and magnetometer.

5.7.5 Environment AI-IoTBPM Drools-jBPM Artificial Intelligent Sensor Automation

In the previous examples, we looked at the use of EOSpy AI-IoT Drools-jBPM to solve a knowledge and notification problem and looked at an end-to-end AI-IoT Drools-jBPM example. We have demonstrated how to invoke Drools **business rules** from within our jBPM executing processes and how to handle the interactions between **process** and **rules**. These applications work with both EOSPY Android and the EOSPY Arduino Tron application installed on your Arduino IoT Device. Now let's look at a board build Arduino Tron IoTBPM reasoning device. This Arduino Tron device is an Industrial Internet of Things (IIoT) or Machine to Machine (M2M) device that uses sensors in the build with the Arduino device.

We will use the Arduino ESP8266 ESP-16S WiFi serial transceiver module with 4MB flash. Arduino provides very inexpensive **IoTBPM MQTT** telemetry transport platform. Download the EOSPY Arduino Tron sensor application from GIT. Update the with WiFi network values for network SSID (name) and network password. Update the EOSPY server IP address and unique unit ID values and add in EOSPY server. Also, we will use a DHT11 digital temperature and humidity sensor. See the Arduino Tron sensor sketch for more details and information.

Example 5.5.5.1 Environment Reasoning DHT11 Board Schematic


D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
Wi Fi
MODEL ESP8266MOD
VENDOR AI/THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin

This is the schematic you need to wire the DHT sensor to your ESP8266 board. We will use the EOSpy IoT client to stream remote Arduino ESP8266 DHT11/DHT22 temperature and humidity sensor information to the EOSpy AI-IoT Drools-jBPM application. Then we will use the AI-IoTBPM Drools-jBPM on this IoTBPM information stream to decide what behavior and business process functions to execute.

All the fields you have provided values for so far are for WiFi communications; the remaining fields are used in the Arduino Tron sensor application. To use a DHT11/DHT12 digital temperature and humidity sensor sees the Arduino Tron sensor sketch for more details and information.

```
const bool readPushButton = true;
                        // Values for the digitalRead value from gpiox button
const bool readDHT11Temp = true;
                // Values for the DHT11 digital temperature/humidity sensor
```

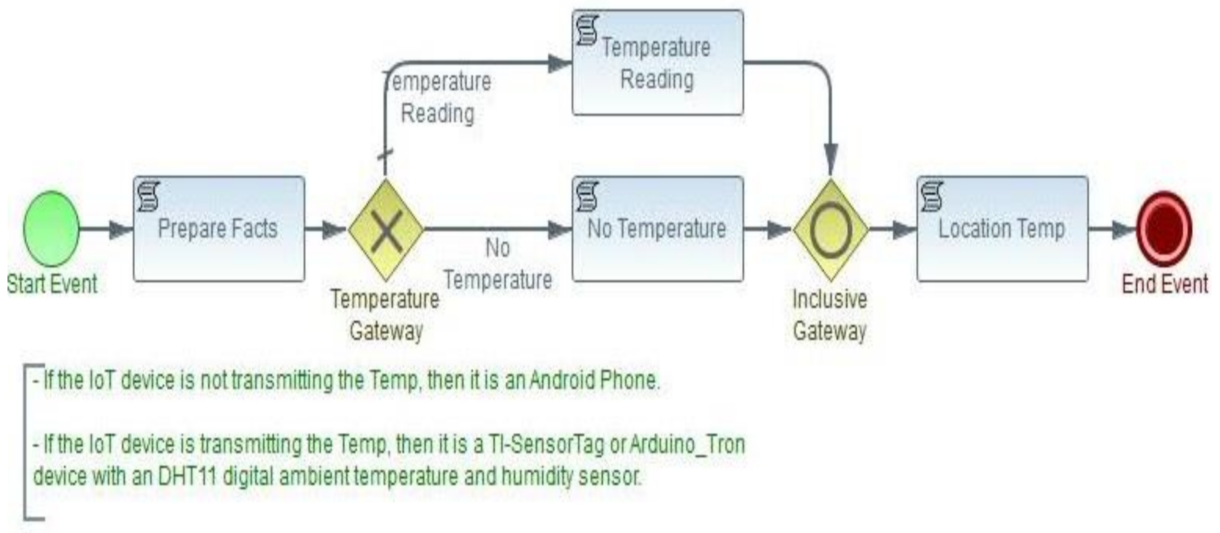
To use a DHT11 digital temperature and humidity sensor install the SimpleDHT library and uncommit the lines for dht11 in the Arduino Tron sensor sketch for more details and information. Change the readDHT11Temp value to true. Also, you can change the reporting time by modifying the timeCounter that calls the Arduino Tron sensor sketch function for reading the DHT11.

```
void readdht11(); // read DHT11 digital temperature and humidity sensor
```

Remember to uncommit all the lines for the dht11 in the Arduino Tron sensor sketch.

```
if ((err = dht11.read(pinDHT11, &temperature, & humidity, NULL)) !=
SimpleDHTErrSuccess) { <-- uncommit for dht11
```

Example 5.5.5.2 Environment Reasoning AI-IoTBPM Drools-jBPM


Start Event
Prepare Facts
Temperature Gateway
Temperature Reading
Temperature Reading
No Temperature
No Temperature
Inclusive Gateway
Location Temp
End Event
- If the IoT device is not transmitting the Temp, then it is an Android Phone.
- If the IoT device is transmitting the Temp, then it is a TI-SensorTag or Arduino_Tron device with an DHT11 digital ambient temperature and humidity sensor.

In the environment reasoning example, the jBPM process following we examine the IoTBPM device temperature event that is transmitted by the Arduino device. Also, we evaluate any significant IoTBPM reported temperature or humidity event such as too warm or too cold condition at our location.

Example 5.5.5.3. Environment Reasoning Raise alarm - Too Warm at Location

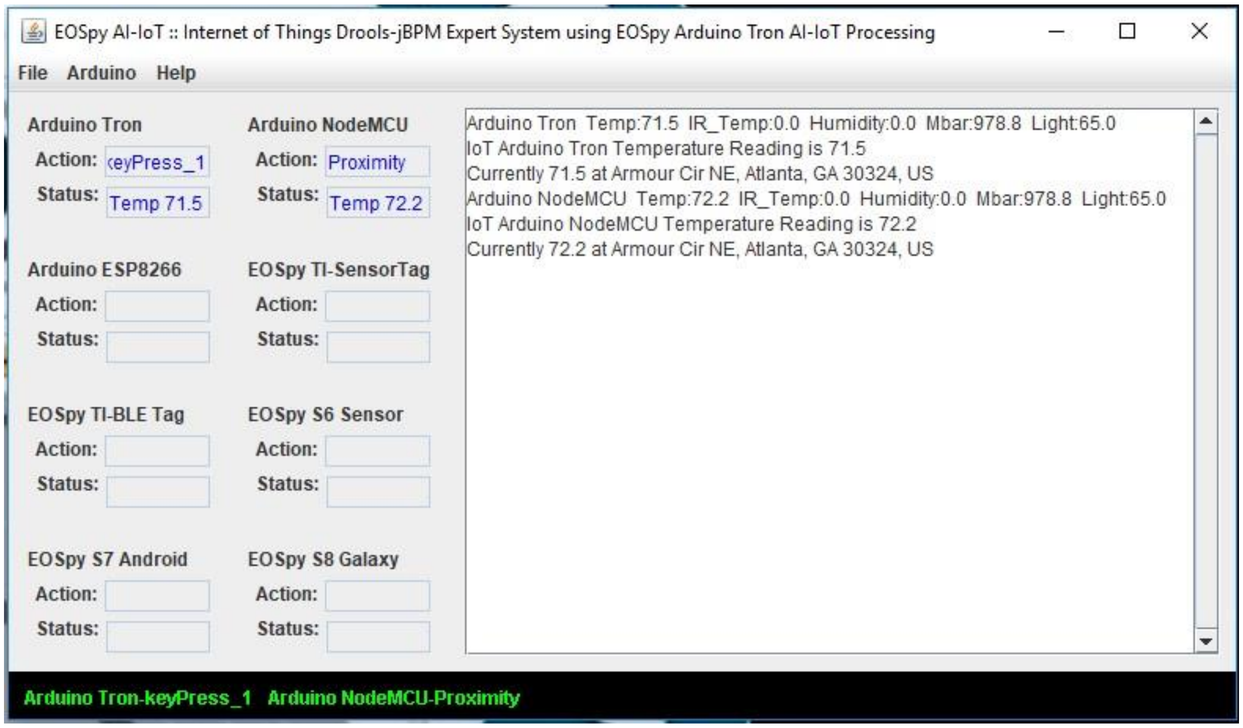
```
//declare rule to fire when Temp is over 75
rule "Raise alarm - Too Warm at Location"
        when
          $event : ServerEvent( $eventId : id, $temp : temp, $temp > "75" )
        $device : Devices( $deviceId : id,  $eventId == id )
        then
                $device.action = "Alert";
                $device.status = "Temp " + $temp;
                com.arduinotron.ui.MainWindow.getInstance().log(">>Raise Temp
Alarm " + $event.getName() + " Temperature:" + $temp);
                modify( $device );
end
```

This environment reasoning rule fires when the DHT11 device reports a temperature if greater than 75. A GPS device module can be added to our environment and condition rules, the rule then reports the position and temperature when the rule is fired. Depending on the GPS devices, the module can report additional information that is transmitted along with the LAT/LON data packet. There are additional rules that respond to the GPS device if the device sends a text message or if the device sent an alarm.

Example 5.5.5.4.Enviorment Reasoning Device Send Text Message

```
//declare rule to fire when GPS Device sends Alarm message
rule "IoTBPM Device Sent Text Message"
        when
          $event : ServerEvent( $eventId : id, $textMessage : textMessage,
$textMessage != null )
        $device : Devices( $deviceId : id,  $eventId == id )
        then
                com.arduinotron.ui.MainWindow.getInstance().log($event.getName()
+ " Text Message: " + $textMessage);
end
```

Example 5.5.5.4 Environment Processing AI-IoTBPM Drools-jBPM



Arduino Tron — Action: keyPress_1 — Status: Temp 71.5

Arduino NodeMCU — Action: Proximity — Status: Temp 72.2

```
Arduino Tron  Temp:71.5  IR_Temp:0.0  Humidity:0.0  Mbar:978.8  Light:65.0
IoT Arduino Tron Temperature Reading is 71.5
Currently 71.5 at Armour Cir NE, Atlanta, GA 30324, US
Arduino NodeMCU  Temp:72.2  IR_Temp:0.0  Humidity:0.0  Mbar:978.8  Light:65.0
IoT Arduino NodeMCU Temperature Reading is 72.2
Currently 72.2 at Armour Cir NE, Atlanta, GA 30324, US
```

Arduino Tron-keyPress_1   Arduino NodeMCU-Proximity

In addition, there are rules for processing an IoT device text message such as the IoT device fix information, and IoT device address location with current temperature at this address location.

IoT devices can be accessed anywhere in the world via the internet and exchange all types of sensor, conditional, enjoinment conditional data. In this example, we are exploiting only one sensor located somewhere else in the world. The NodeMCU ESP8266 microcontroller has hundreds of various sensors available for the Arduino boards. Most of these sensors are very inexpensive and work well together.

A partial list of the type of sensors available for the Arduino boards include: Soil module, Infrared sensor receiver module, Laser head sensor module, Temperature and humidity sensor module, Infrared emission sensor module, 5V relay module, Gyro Module, Finger detect heartbeat module, Microphone sensitivity sensor module, Metal touch sensor module, Flame sensor module, 3-color LED module, Hunt sensor module, Linear magnetic Hall sensors, Rotary encoder modules, Active buzzer module, Magic Light Cup modules, Small passive buzzer module, Digital temperature sensor module, Tilt switch module, Analogy Holzer magnetic sensor, Ultrasonic module, Mercury opening module, Hall magnetic sensor module, RGB LED SMD module, Mini Reed module, Bicolor LED common cathode module 3MM, Smart car avoid obstacle sensor infrared sensor photoelectric switch, Key switch module, Photo-resistor module, Breadboard power module, Tilt sensor module, Temperature sensor module, Vibration switch module, Microphone sound sensor module, Large reed module, Two-color LED module, Optical breaking module, Temperature sensor module, MP1584EN buck module, SD card reader module, PS2 Joystick game controller module, Automatically flashing LED module, DS1302 clock module, RFID Reader and Sensor, Proximity Sensor, Ultrasonic distance measurement module, Water level module and more.

The NodeMCU ESP8266 microcontroller is a very powerful and economic platform with many additional environmental and conditional sensors available. The NodeMCU ESP8266 can also interface with your existing device circuit boards. This gives you the ability to add WiFi to an existing circuit board.

The attractiveness lies within the fact that this inexpensive NodeMCU ESP8266 microcontroller board can be added to an existing device and programmed via the Arduino IDE to add new functionality like WiFi to an existing system: i.e., alarm system, fire monitoring, asset monitoring, and building access. Also, the data from this device can be stored in the Arduino memory for later import into our application for processing. This type of store and forward connectivity is useful when the sensors are out of WiFi range.

6 Arduino Tron AI-IoT Drools-jBPM Automation

6.1 Arduino Tron Automation Project

In in the previous sections we discussed the integration of Drools and BPM into our IoT projects. In the Arduino Tron AI-IoT Drools-jBPM application project examples, we demonstrated that actions and behavior of our IoT devices could be controlled from our BPM engine and analytical reasoning can be achieved through Drools Rules. Also, with the use of AI Drools-jBPM analysis and reasoning in IoT devices, we can orchestrate dissimilar devices that normally have no awareness of each other.

We are going to expand on what we have learned in the previous sections and add some new IoT components and devices that will be a part of our Arduino Tron AI-IoT Drools-jBPM Automation project. These new additions will give us additional situational awareness and allow us to interact more naturally with our IoT components and devices.

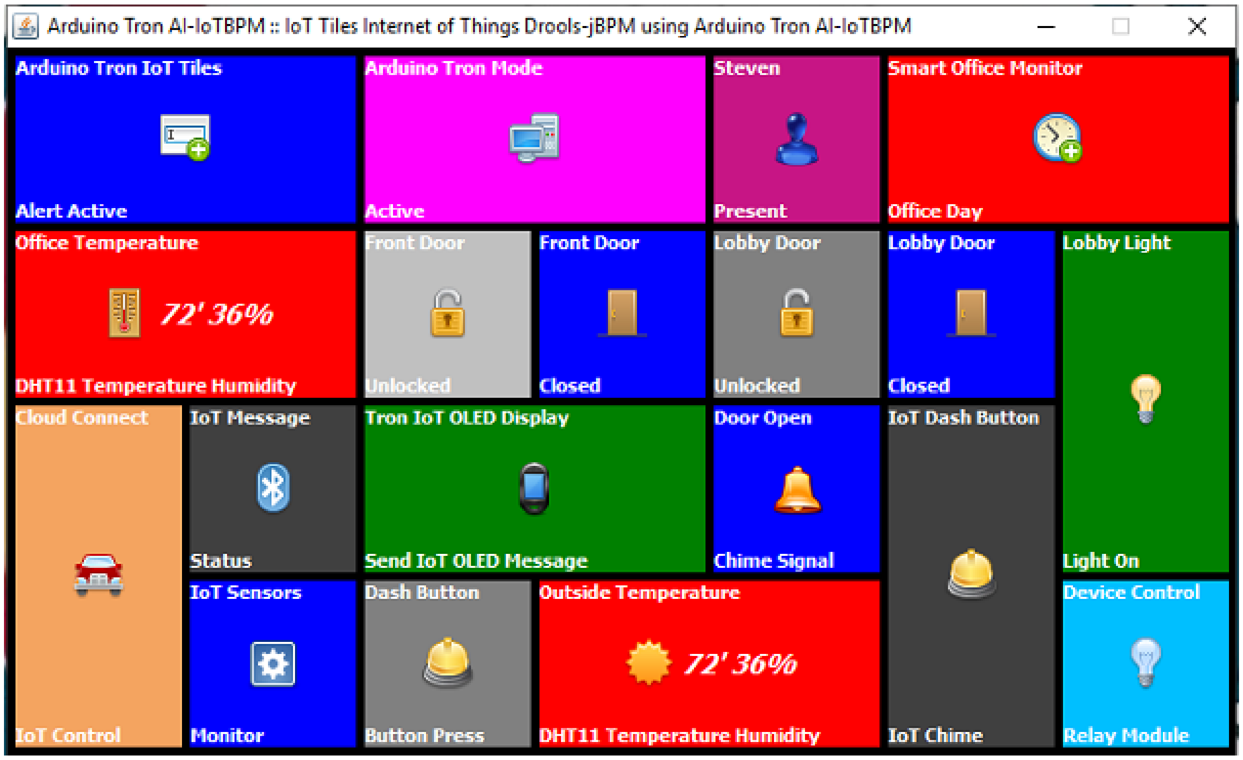
6.1.1 IoT Automation Situational Awareness (SA)

IoT Situational Awareness (SA) is the perception of environmental elements by our IoT devices and events with respect to time or space, the comprehension of their meaning, and the projection of their change of status to us (humans) in a perceivable way. This might be as simple as turning on a light or displaying a message or sounding an alarm. However, we want to know more, we want to know our SA.

Additionally, as distributed IoT sensors and applications become larger and more complex, the simple processing of raw sensor and actuation data streams becomes impractical. Instead, data streams must be fused into tangible facts, information that is combined with knowledge to perform meaningful notifications. What we have accomplished with our AI-IoT Drools-BPM implementation, is our Arduino Tron IoT devices are judging and making decisions after cognitive situational reasoning.

6.2 Arduino Tron IoT Tiles Control Panel

The Arduino Tron IoT Tiles is a control panel (dashboard) for Arduino Tron IoT Things, which controls IoT smart office automation and IoT monitoring from your desk. IoT Tiles allows you to send IoT commands directly to your IoT devices and monitor all your IoT device sensor and alert messages.


Arduino Tron AI-IoTBPM :: IoT Tiles Internet of Things Drools-jBPM using Arduino Tron AI-IoTBPM

Arduino Tron IoT Tiles — Alert Active
Arduino Tron Mode — Active
Steven — Present
Smart Office Monitor — Office Day
Office Temperature — 72' 36% — DHT11 Temperature Humidity
Front Door — Unlocked
Front Door — Closed
Lobby Door — Unlocked
Lobby Door — Closed
Lobby Light — Light On
Cloud Connect — IoT Control
IoT Message — Status
Tron IoT OLED Display — Send IoT OLED Message
Door Open — Chime Signal
IoT Dash Button — IoT Chime
IoT Sensors — Monitor
Dash Button — Button Press
Outside Temperature — 72' 36% — DHT11 Temperature Humidity
Device Control — Relay Module

Figure 6.1.2: Arduino Tron IoT Tiles Control Panel (Dashboard)

This single dashboard gives you two-way communications with all your IoT devices and Arduino Tron IoT Things. Also, it provides situational awareness, alerts and notification messages from each of your IoT devices. Arduino Tron IoT Things events are updated "instantly" on the IoT Tiles panel. You can view the state of your Arduino Tron IoT Things: Presence Sensors, Contact Sensors, Motion Sensors, Temperature, Water Sensors, Battery Status, Vibration Sensors, Luminosity Sensors, GPS (Location), Weather (Office and Outside), or any equipment from one panel from IoT Tiles panel.

From Arduino Tron IoT Tiles you can fully control all IoT devices from one panel, use it to control: Switches, Dimmable Lights, Control lights, Momentary Switches, Theme Lights, Control Thermostats, Heating/ Cooling Thermostats, Control Things, Door Locks, Entrance Door Access, Music Players, Cameras (Image Capture), and more. Control all manufacturing floor equipment from one panel.

- **Arduino Tron IoT Tiles Automation for Controlling these IoT Devices**

1. SECURITY CAMERAS

Office security and monitoring are paramount for your property or rental space. Integrate surveillance systems with office automation audio today to take advantage of cutting edge security features like sound-activated recording and motion detection. See who's approaching and leaving your office front door--or other sensitive spaces, like your server room.

2. LIGHTING AUTOMATION

Complete control of your office automation lighting from brightness to color from your IoT office control panel. Lighting automation lets you regulate lighting options in your office remotely and conveniently.

3. OUTDOOR LIGHTING

Dim and control building lighting from your IoT device, or program the lights to turn off automatically at a designated time. If you are using a smart door sensor, you can activate the walkway lights when the office door opens or when motion is detected.

4. DOORBELL

See and talk to delivery drivers or other visitors at your loading dock or main entrance - even if you're not at the office.

5. LOCKS & SECURITY

Let smart office locks control access to your business. Locks office automation enables you to control your locks via IoT devices. Never lose another physical key or worry about whether you locked your business doors or not.

6. CLIMATE CONTROL

Use your thermostat in your office automation control system and control temperature from a tablet or computer anywhere you are. Program your office's thermostats for the nights and weekends, so you're not air-conditioning or heating empty rooms.

7. MOTION SENSORS

Wireless motion detectors for notification. Program your lights to turn on and off when people enter and leave and have the system alert you when someone's entered your office.

8. WINDOW SHADES

Reduce energy costs by programming your shades to move with the sun. Great energy savings for heating and cooling also, ambient lighting.

9. WATER LEAK ALARM

Check for water in your office and get notified of leaks and floods when you are not in the office. Water leak alarms are designed to help mitigate damage and avoid leak conditions so that they can be fixed before they become much costlier problems. These flood alerts can be crucial, and they make this sensor one of the most popular for office automation alerts.

10. PRESSURE-SENSITIVE MATS

Pressure-sensitive mats alert you to comings and goings of visitors, employees, and intruders. These wireless monitors send alerts to you when someone steps on the mat. Most are suitable for indoor or outdoor use.

11. CONFERENCE ROOM RESERVATIONS

Smart-office automate a conference room reservation. As soon as you walk into a meeting room, RFID software recognizes you and books the room on the spot.

12. HUBS & CONTROLLERS

Office automation has enabled central control devices from which you control all your automation mechanisms. Arduino Tron IoT Tiles is affordable office automation options for you, customized this hub controller for your office needs.

The beauty of the Arduino Tron IoT Tiles is that all your IoT device technology becomes interactive with humans through one simple IoT Tiles panel dashboard on your computer. With the Arduino Tron IoT Tiles getting your IoT project working with a great human interface is a fast-easy solution.

6.3 Arduino Tron IoT Display

The Arduino Tron IoT Display is a small and inexpensive display that can sit on your desk or be mounted to a wall. It provides situational awareness, alerts and notification messages from your IoT devices. The Arduino Tron IoT Things events are displayed "instantly" on the IoT Display. You can think of the IoT Display as a single IoT Tiles panel window, focused on displaying priority alerts from you Arduino Tron IoT Things. Priority alerts like "Office Door" opened.

This very small and inexpensive Arduino Tron IoT Display uses an SSD1306 OLED backlight display module and ESP-01. The IoT Display ESP-01 module is a receiver which connects via the WiFi network to accept IoT data and display the alerts and messages to you instantly. The Arduino Tron IoT Display is a single point to display meaningful alerts for you from IoT devices. This helps you know what is going on with your IoT network and devices.

Figure 6.1.3: Arduino Tron IoT Display

It is much clearer for you to read the IoT Display alert message than for you to fumble for your cell phone to read an alert notification that may disappear. The Arduino Tron IoT Display is a notification module that can also provide information when installed into your equipment or a professional desk interface for your IoT devices.

The Arduino Tron IoT Display can be configured to continuously display the time, temperature and humidity, and then display flashing alert messages when they arrive. The temperature and humidity sensor values displayed on the Arduino Tron IoT Display module can be from your office, from outside or from any location in the world. You can read the temperature and humidity from any DHT11 transceiver sensor and create a fully functional weather station.

6.3.1 Arduino Tron Web Server Cloud Interface for IoT Device Control

IoT device management is the process of configuring, monitoring and maintaining the device software that provides its functional capabilities. Effective IoT device management is critical to establishing and maintaining the health, connectivity, and security of all of your IoT devices. The Arduino Tron Web Server provides device management to set IoT device parameters, activate and deactivate your devices, and grant access control of your IoT devices parameters from the Internet.

The beauty of the Arduino Tron Web Server is that the IoT device technology becomes interactive with humans through a simple Web Server browser interface. The integration between Arduino Tron Web Server and IoT devices presents a viable control and notification solution with a bright future – one that will connect people, things, and systems together as part of business-critical processes as never before.

The Arduino Tron lightweight Web Server provides an IoT dashboard, management, and control for remote management of your Internet-Enabled IoT devices. With the Arduino Tron Web Server getting your IoT project working in the cloud is a fast-easy solution. The Arduino Tron lightweight Web Server is a cloud-connected complete SoC (System on a Chip) architecture that integrates all components of a computer, WiFi and Web Server application software on an ESP-01 or ESP8266 WiFi chip for complete control of Internet of Things (IoT) devices from the cloud.

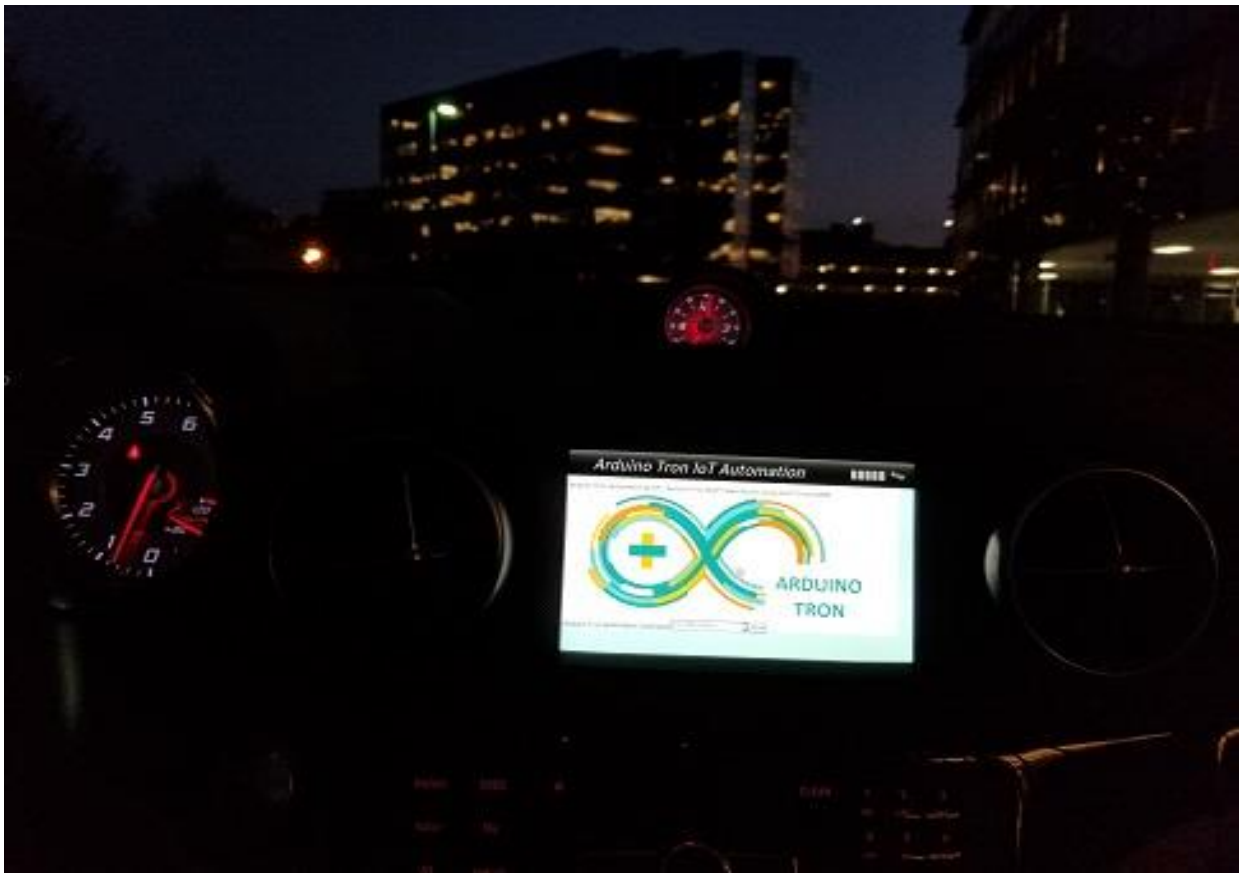


Figure 6.1.4: Arduino Tron Web Server Cloud Interface

The Arduino Tron Web Server Cloud Interface for IoT Device Control can be addressed from any web browser anywhere in the world. This connection is from a car in-dash display using the automobile's web browser to connect to the Arduino Tron Web Server Cloud Interface.

Some applications for the Arduino Tron Web Server Cloud Interface:

- Support for MQTT IoT device control and managing
- Bidirectional IoT device communication and data storage
- Send and receive IoT data between sensor readers or triggering devices
- Cross-device platform and server messaging
- Monitor device sensors and stored metadata
- Monitor and track device status or consumption levels in real-time
- Automated code executes to trigger alerts or IoT device actions

- **Arduino Tron Web Server Interface Sketch for Controlling IoT Devices**

The Arduino Tron Web Server software uses a web browser interface and a WiFi wireless transceiver interface to stream information to the BPM-Drools Server from any remote-control Arduino Web Server.

The Arduino Tron SoC ESP-01S or ESP8266 WiFi Transceiver Module with 4MB Flash memory both provide comprehensive IoT device control and management with a very small, inexpensive and lightweight IoT device cloud Internet-connected solution.

The configuration information needed for the Arduino Web Server is similar to the Arduino Tron sketch.

Update the with WiFi network values for network SSID (name) and network password.

```
// Update these with WiFi network values
const char* ssid     = "your-ssid"; // your network SSID (name)
const char* password = "your-password"; // your network password
```

Update the IoTBPM Server IP address and unique unit ID values and add in EOSPY Server.

```
// Update these with EOSpy service IP address and unique unit id values
byte server[] = { 10, 0, 0, 2 }; // Set EOSpy server IP address as bytes
String id = "334455"; // Arduino Tron Device unique unit id
```

You will need to set a different device unique identifier for each one of your Arduino Tron Sensor ESP8266 MQTT devices. Next match the Arduino Tron sensor device unique identifier to the device unique unit ID in the Arduino Tron server application. Above are all the fields you need to provide configure values for the remaining fields that are used in the Arduino Tron Web Server application.

6.4 Arduino Tron ESP-01 Wireless Alert Sensor

The Arduino Tron ESP-01 Wireless Alert Sensor is for automated remote monitoring and notification. This IoT device reduces the resources needed to manually monitor people, places and things that could be self-monitored. The Arduino Tron ESP-01 Wireless Alert Sensor offers an easy-to-use, wireless monitoring solution that uses existing WiFi networks and internet access to gather sensor data and alert notifications instantly.

Some uses for Arduino Tron ESP-01 Wireless Alert Sensor are refrigerator temperatures, plumbing leaks, door and window access, humidity or server room temperature, front door access and much more. The Arduino Tron ESP-01 Wireless Alert Sensor online cloud-connected IoT WiFi, complete SoC (System on a Chip) provides alerts when specific conditions are met. You can address security alerts and maintenance problems before they become expensive repairs.

Arduino Tron ESP-01 Wireless Alert Sensor can be used as a simple remote alert that will immediately send an alarm signal via the WiFi network back to the Arduino Tron AI-IoTBPM Drools-jBPM Server. The Arduino Tron Wireless Alert Sensor can be integrated with any wireless security system, mailbox, garage door, driveway alarms, motion sensor, delivery detect alerts, wireless door entry chime, doorbell or panic button alarms. The Arduino Tron Wireless Alert Sensor provides for audio (door chime) or device activation (such as lamp or appliance).

Arduino Tron ESP-01 Wireless Alert Sensor Applications:
- Warehouse door monitoring
- Freezer and cooler door monitoring
- Button or switch integration
- Production line tracking

- **Arduino Tron ESP-01 Wireless Alert Sensor Principles of Operation**

The Arduino Tron ESP-01 Wireless Alert Sensor can be used with any contact switch. The Arduino Tron ESP-01 Wireless Alert Sensor sends via the WiFi wireless transceiver to the BPM-Drools Server from any remote location in the world. The BPM-Drools Server can log the information and then activate any other automated process based on the event. This can be something as simple as an IoT Wireless Open-Door Chime or as complex as an employee access authorization and notification.

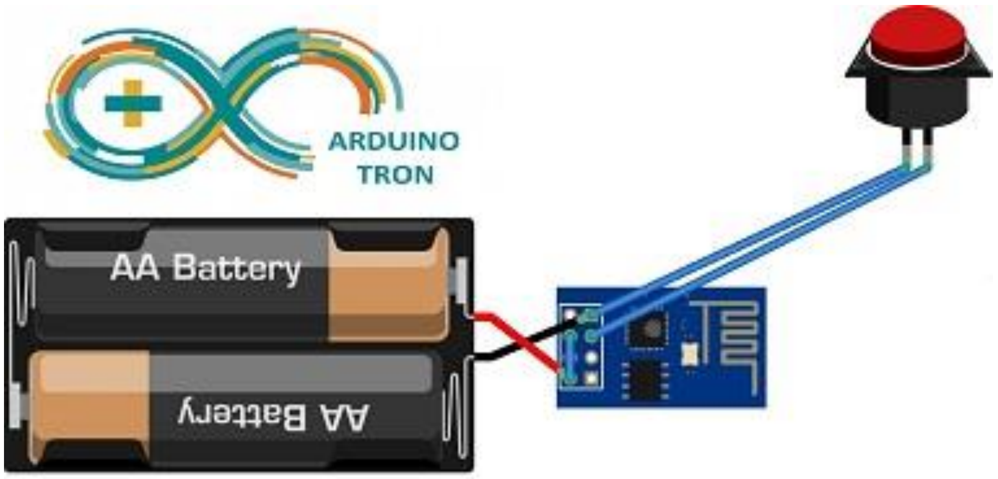
The configuration information needed for the Arduino Tron ESP-01 Wireless Alert Sensor is the same as the Arduino Tron sketch (for the ESP8266). The hardware build difference is that the ESP-01 is in deep sleep mode until RESET pin is connected to a LOW signal (pushbutton is pressed). *ESP.deepSleep(0);* The ESP-01 uses the CH_PD (enable / power down, must be pulled to GRD directly or via a switch). The CH_PD (chip select) is all that is needed for the operation of the module. This eliminates the use of the GPIO0 and GPIO2 that determines what mode the module starts up in.

6.5 Arduino Tron Wireless Dash Button

The Arduino Tron ESP-01 Wireless Dash Button Alert WiFi Transmitter is for automated remote monitoring and notification. The Arduino Tron IoT Dash Button is a programmable button based on the Arduino Tron WiFi Dash Button Alert sketch. This simple WiFi dash button device is easy to configure and designed to get you started quickly using the Arduino Tron AI-IoTBPM Drools-jBPM Server IoT and Arduino Tron Cloud Services.

You can configure the button's action in the cloud to count or track items, call or alert someone, start or stop something, order services, or even provide feedback. For example, you can click the button to unlock or open your office door, open your garage door, call an employee, call a customer service representative, track the use of common office items, or products, or remotely control your office appliances. The button can be used as a remote control for any shop floor equipment. You can integrate it with other IoT devices or even your own company's applications.

Arduino Tron ESP-01 Wireless Dash Button can be used as a simple remote alert that will immediately send an alarm signal via the WiFi network back to the Arduino Tron AI-IoTBPM Drools-jBPM Server. The Arduino Tron Wireless Dash Button can be integrated with any wireless security system, mailbox, garage door, driveway alarms, motion sensor, delivery detect alerts, wireless door entry chime, doorbell or panic button alarms. The Arduino Tron Wireless Alert Sensor provides for audio (door chime) or device activation (such as lamp or appliance).



The programmable Arduino Tron WiFi IoT Dash Button is a small device that can be mounted to a wall or button that sits on your desk or table.

It provides situational awareness, alerts and notification messages from your IoT devices.

The Arduino Tron Wireless Alert Sensor provides for audio (door chime) and can be integrated with any wireless security system, driveway alarms, motion sensor, delivery detect alerts, wireless door entry chime, doorbell or panic button alarms. This adds audio functionality to the Arduino Tron Wireless Alert Sensor, used for wireless door entry chime, doorbell, or panic button alarms.

6.6 Arduino Tron ESP-01 Agent Expansion Module

The Arduino Tron IoT ESP-01 Agent WiFi Relay Expansion Module board is a simple and easy-to-use expansion board that uses the ESP-01 to drive a relay and operate devices or machines wirelessly.

The Arduino Tron IoT ESP-01 Agent WiFi Relay Expansion Module board can be used to connect devices so that multiple devices or machines act in unison through the IoT BPM-Drools Server. The Arduino Tron IoT ESP-01 Agent WiFi Relay Expansion Module board can be used to connect to office door locks, activate security alarms, turn on office lights, control thermostats, answer the doorbell, open window shades, activate motion sensors, and to control any equipment on the shop floor machines.

The Arduino Tron Agent software interface allows you to send commands with the Arduino Tron AI-IoTBPM server software to control external Arduino connected devices. The Arduino Tron AI-IoTBPM server software uses a WiFi wireless transceiver interface to control and interact with module sensors and remote control devices. Control any device from the Arduino Tron Agent software or stream any interface over the WiFi internet. With the Arduino Tron Agent software you can automatically turn on lights, appliances, cameras, and lock/unlock doors from the Drools-jBPM expert system processing.

A long pin ESP-01 shield can add additional functionality to the WiFi Relay Expansion Module board, like a speaker or other push button function. The shield eliminates the problems of trying to modify the WiFi Relay Expansion Module board and lets you simply plug in the ESP-01. Then plug the shield into the board. With a speaker added to the WiFi Relay Expansion Module board, you can use the additional sketch commands: /CHIME and /TONE. This adds audio functionality to the Arduino Tron Wireless Alert Sensor, used for wireless door entry chime, doorbell, or panic button alarms.

6.7 Arduino Tron DHT11 Temperature and Humidity Sensor

The Arduino Tron DHT11 Temperature and Humidity Sensor ESP-01 WiFi Wireless Module allows you to send information from the DHT11 digital temperature and humidity sensor directly to the Arduino Tron AI-IoTBPM Drools-jBPM Server. This module uses the ESP-01 as the main controller and the temperature-humidity sensor is DHT11.

Also, you may use a DHT11 digital temperature and humidity sensor with the Arduino Tron Sensor sketch. The ESP8266 collects the environment temperature and humidity then uploads to the Arduino Tron AI-IoTBPM Drools-jBPM Server.

All the fields you have provided values for so far are for WiFi communications; the remaining fields are used in the Arduino Tron sensor application. To use a DHT11/DHT12 digital temperature and humidity sensor see the Arduino Tron sensor sketch for more details and information.

```
const bool readDHT11Temp = true;
                // Values for the DHT11 digital temperature/humidity sensor
```

To use a DHT11 digital temperature and humidity sensor, install the SimpleDHT library and uncommit the lines for dht11 in the Arduino Tron sensor (see the sketch for more details and information). Change the readDHT11Temp value to true. Also, you can change the reporting time by modifying the time-counter that calls the Arduino Tron sensor sketch function for reading the DHT11.

With the Arduino Tron DHT11 Temperature and Humidity Sensor ESP-01 WiFi Wireless Module configured, we can use DHT11 IoT information stream to decide what behavior and business process functions to execute based on temperature or humidity with the AI-IoTBPM Drools-jBPM Server.

6.8 Summary

Internet of Things (IoT) devices opens an opportunity to create a new generation of business processes that can benefit from IoT integration, taking advantage of networking, sensing capabilities, remote awareness and the ability for rational agents to take automated actions. A rational agent is an IoT agent that acts and that does 'the right thing'. The right thing obviously depends on the performance criterion defined for an agent, but also on an agent's prior knowledge of the environment, the sequence of observations the agent has made in the past and the choice of actions that an agent can perform.

While this data is useful, there is still "a-disconnect" in integrating these IoT devices with mission-critical business processes and corporate cloud data awareness.

The task of moving IoT devices beyond "connected" to "smart" is daunting. Moving beyond collecting IoT data and transitioning, to leveraging this new wealth of IoT data, to improving the smart decision-making process is the key to automation. Artificial Intelligence (**AI**) will help these IoT devices, environments and products to self-monitor, self-diagnose and eventually, self-direct. This is what we said in the opening of this book, "If a machine thinks, then a machine can do."

However, one of the key concepts in enabling this transition from connected to smart is the ability to perform **AI Analytics**. The traditional analytic models of pulling all data into a centralized source such as a data warehouse or analytic sandbox is going to be less useful. We are not trying to just analyze complex IoT data; we are trying to make "smart decisions" and take actions base on our AI Analytics of IoT devices.

***IoT Definition** – IoT is the integration of computer-based systems into our physical-world.*

With Arduino Tron (Artificial Intelligence – Internet of Things) AI-IoTBPM BPMN modular, this allows us to define both the business processes and IoT devices behavior at the same time using one diagram. The Arduino Tron AI-BRMS Drools itself is the heart of the agent that computes, and reasons based on the available data and its knowledge of the IoT sensors on the environment. With Arduino Tron adding Drools-jBPM to IoT is easy, we make the IoT devices "smart".

The power of the IoT device increases greatly when business process (jBPM) can use them to provide information about our real-world as well as execute IoT device actions as part of our business process. This increases the processing power of IoT devices enables them to take part in the execution of the business logic. The jBPM-BPMN modular allow us to define both the business processes and IoT devices behavior at the same time using one (BPM) diagram. In our examples, we demonstrated adding Drools-jBPM to IoT devices. Making **"Things Smart"** is the application of AI to IoT platform via DroolsRules Inference Reasoning, jBPM, and ES-Expert Systems Architecture.

With the use of AI Drools-jBPM analysis and reasoning in IoT devices, we can orchestrate dissimilar devices that normally have no awareness of each other. This creates opportunities for direct integration of computer-based systems into the physical-world that has never been available before. This results in improved efficiency, accuracy, and economic benefits by increased automation - reduced intervention. This IoT orchestration of IoT devices gives us the ability for action after our AI decision.

Artificial Intelligence (A.I.) is a very broad research that focuses on "making computers think like humans." Expert Systems (E.S.) use knowledge representation: a knowledge base for reasoning, i.e., we can process data with this knowledge base to infer conclusions.

Expert Systems and Knowledge-based Expert Systems are considered to be "applied artificial intelligence." We can reason to a conclusion (infer) beyond what we currently know.

Business Rules Management Systems builds additional value on top of a general-purpose Rules Engine by providing business user-focused systems for rule creation, management, deployment, collaboration, and analysis user tools. These further add to the value by making the Rules Engine easily implemented and ingrained into our existing Enterprise Systems.

The Arduino Tron AI-IoTBPM Drools-jBPM Expert System provides sophisticated jBPM and Drools processing. This allows us to define IoT behavior within business process and with the same level of abstraction of other atomic actions or events in our standard jBPMN.

7 EOSpy AI-IoTBPM Internet of Things Resource Constraint Solver

7.1 EOSpy AI-IoTBPM OptaPlanner Constraint Solver Reasoning

7.1.1 OptaPlanner AI-IoTBPM Smart Automation Constraint Solver

OptaPlanner is a constraint solver. It optimizes business resource planning use cases, such as vehicle routing, employee rostering, cloud optimization, task assignment, job scheduling, bin packing and many more. Every organization faces such scheduling obstacles: assign a limited set of constrained resources (employees, assets, time and money) to provide products or services. OptaPlanner delivers more efficient plans to improve service quality and reduce costs.

OptaPlanner is a lightweight, embeddable planning engine. It enables normal Java programmers to solve optimization problems efficiently. It is also compatible with other JVM languages (such as Kotlin and Scala). Constraints apply to the plain domain objects and can reuse existing code. There's no need to input them as mathematical equations. Under the hood, OptaPlanner combines sophisticated optimization heuristics and metaheuristics (such as Tabu search, simulated annealing, and late acceptance) with very efficient score calculation.

7.2 EOSpy-AI Resource Problem Description

Let's consider solving a technician resource assignment problem for our business requirement in our EOSpy-AI Internet of Things system. Our business model will have technicians out in the field, and we will receive trouble tickets. Our customers have different levels of service. We want to route our technicians to the customers with the highest revenue potential to maximize our business profits.

One of the characteristics is the calculation of the solution score using the rules engine. It means that we define our problem constraints implementing business rules and assigning a score to each one, making it easy to implement and scale our trouble ticket with customer service level requirements. You have technicians with different skill levels. Therefore, you don't want to assign a high skill level technician to a lower skill level ticket or to a customer that is receiving large discounts.

7.3 EOSpy-AI Solving Resource Assignment Problem

EOSpy-AI will solve the resource assignment problem using the Tabu search acceptor. In our Drools EOSpy-AI example, you have a company that provides technicians with different skills to customers based on the skills, location, and availability of the technicians, and you want to automatically choose the best technician for every trouble ticket request. Additionally, you have customers with varying service charge rates. Also, we want to route your technicians to the customers with the highest revenue potential to maximize your business profits. This may mean passing a closer lower level customer to travel to a higher charge customer because we can bill the tech at a higher rate.

7.4 EOSpy OptaPlanner AI-IoTBPM Planning Engine

The OptiPlanner is a **lightweight, embeddable planning engine that optimizes planning problems** and is well suited for both EOSpy-AI CLOUD-based and STREAM-based AI-IoTBPM problems. It solves use cases such as:

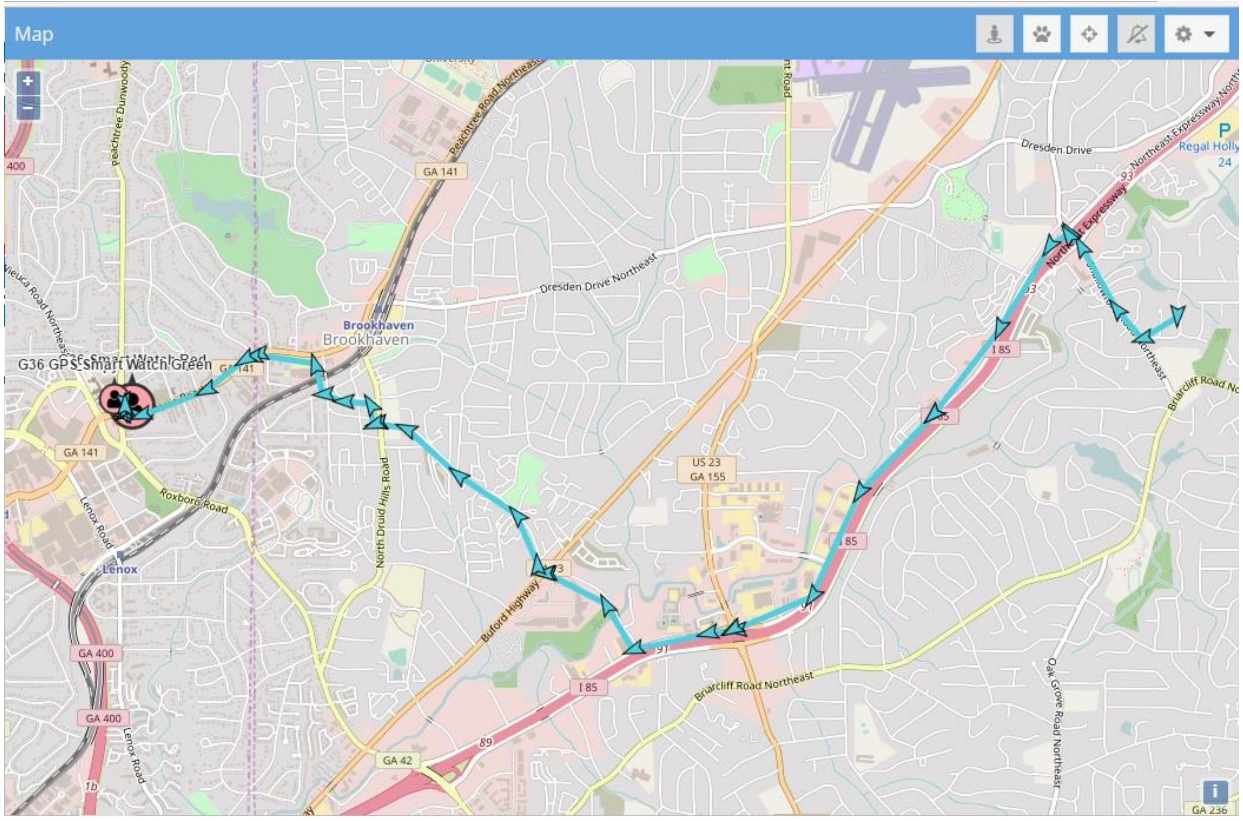
- **Employee shift rostering**: timetabling nurses, repairmen, technician (GPS)
- **Agenda scheduling**: scheduling meetings, appointments, maintenance jobs, advertisements
- **Employee timetabling**: scheduling lessons, courses, exams, conference presentations
- **Vehicle routing**: planning vehicles (trucks, trains, boats, airplanes (with freight and/or people)
- **Bin packing**: filling containers, trucks, ships/storage warehouses, and cloud computer nodes
- **Job scheduling**: planning car assembly lines, machine queue planning, workforce planning
- **Tracking stock**: minimizing waste, asset tracking, rental/lease tracking (item GPS tracking)

Every organization faces planning problems - Provide products and services with a limited set of constrained resources (employees, assets, time and money).

The OptiPlanner module is used to optimize automated planning problems combining search algorithms with the core of the rules engine. It can be used to solve lots of different use cases such as scheduling, routing, timetables and receive the IoTBPM GPS information from our EOSpy IoTBPM clients from our service technicians in the field.

The OptiPlanner provides a benchmarker that is useful to find the solver type that best fits into your problem domain. It will help you in the process of comparing the performance of solvers creating reports with statistics and information that we can use to tweak the configurations.

OptiPlanner supports several optimization algorithms to efficiently wade through that incredibly large number of possible solutions. Depending on the use case, some optimization algorithms perform better than others. You can switch the optimization algorithm by changing the solver configuration.



Example 6.4.1. EOSpy-AI best solution found planner route

IoTBPM Streaming Event Correlation Map Analysis

- The goal for streaming event correlation layer is to leverage topology/service model insights and provide a correlated status mapping of the IoTBPM smart things sensors and data.
- IoTBPM sensors and monitors will listen to queues and fetch common event objects produced by the data ingestion layer and insert them into JBoss BRMS Drools as “facts” and the @role metadata tag to the fact type: “events” and map conditions accordingly on the EOSpy map.
- JBoss BRMS (based on RETE algorithm and forward chaining) shall, in-real time, evaluate the appropriate business rules that need to get fired based on the appropriate IoTBPM conditions.

7.4.1 EOSpy-AI class explanation

In EOSpy-AI, we will need resource assignment of technicians with different *skills* and *training levels* to service a trouble ticket request. Define the *Skill*, *Location,* and *Training Level* enum set class definitions.

7.4.2 Technician Java class explanation

In the EOSpy-AI Drools expert system, the Technician Java class is defined as followings:

Example 6.4.2.1. Java Technician class definition

```
package com.arduinotron.ai_iot;

public class Technician {
	private Location location;
	private TrainingLevel trainingLevel;
	private boolean busy;
	private Set<Skill> skills;

	public Technician(Location location, TrainingLevel training, boolean busy,
Set<Skill> skills) {
		this.location = location;
		this.trainingLevel = training;
		this.busy = busy;
		this.skills = skills;
	}

	public Technician(Technician technician) {
		this(technician.location,
		technician.trainingLevel,
		technician.busy,
		(technician.skills.isEmpty()) ? Collections.<Skill> emptySet() :
technician.skills);
}
```

7.4.3 Service Request Java class explanation

In the EOSpy-AI system, the last class is the service request or trouble ticket. The service request will specify the required skills and desired location of the requested technician. This class must be annotated with a *@PlanningEntity* annotation, which is discussed in detail next.

Example 6.4.3.1. Service Request class definition

```
package com.arduinotron.ai_iot;

@PlanningEntity
public class ServiceRequest {
	private Location location;
	private Set<Skill> neededSkills;
	private Technician technician;

	public ServiceRequest(Location location, Set<Skill> neededSkills) {
		this.location = location;
		this.neededSkills = neededSkills;
	}

	public ServiceRequest(ServiceRequest serviceRequest) {
		this(serviceRequest.location, EnumSet.
copyOf(serviceRequest.neededSkills));
		if (serviceRequest.technician != null) {
				setTechnician(new Technician(serviceRequest. technician));
		}
	}

@PlanningVariable
@ValueRangeFromSolutionProperty(propertyName = "serviceRequests")
```

With our EOSpy-AI domain model defined, we can start creating the classes needed by OptiPlanner. The TechnicianMove class, which must implement the drools.planner move interface, and generate the changes between different solutions.

The *ServiceRequest* entity class was annotated with the *@PlanningEntity* annotation. The planning entity is a POJO that changes during the solving phase. In our EOSpy-AI example, only the *ServiceRequest* property values will change during the execution of the planning steps.

The properties of the planning entities that change during planning must be annotated with the *@PlanningVariable* annotation and also must specify that the values range using one of the following available annotations, which are pretty descriptive:

```
@ValueRangeFromSolutionProperty(propertyName = "propertyNameInSolution")
@ValueRangeFromPlanningEntityProperty(propertyName = "propertyNameInEntity")
@ValueRangeUndefinied
```

The class that implements the Drools Planner move interface is where the EOSpy-AI OptiPlanner concepts come into light. It represents a change from a solution A to a solution B, but to understand a little more you have to know that local search solves a problem by making a move on the current solution that changes it into a better solution, acting very much like a human:

Example 6.4.4.1. Java Technician Move class definition

```
package com.arduinotron.ai_iot;

public class TechnicianMove implements Move {
	private ServiceRequest serviceRequest;
	private Technician technician;

	public TechnicianMove(ServiceRequest serviceRequest, Technician technician) {
		this.serviceRequest = serviceRequest;
		this.technician = technician;
	}

	@Override
	public boolean isMoveDoable(WorkingMemory wm) {
		return !serviceRequest.getTechnician().equals(technician);
	}

	@Override
	public Move createUndoMove(WorkingMemory wm) {
		return new TechnicianMove(serviceRequest,
serviceRequest.getTechnician());
	}

	@Override
	public void doMove(WorkingMemory wm) {
			FactHandle serviceRequestHandle =
wm.getFactHandle(serviceRequest);
			serviceRequest.setTechnician(technician);
			wm.update(serviceRequestHandle, serviceRequest);
	}
}
```

The *isMoveDoable(WorkingMemory)* method is automatically used by OptiPlanner to filter the non-doable moves. A non-doable method is:

- A move that changes nothing on the current solution

- A move that is impossible to do on the current solution

In a non-doable move is when the move wants to assign the same technician to a service request who is already assigned in the current solution. Next, you will see an implementation of the *isMoveDoable(WorkingMemory)* method:

```
@Override
public boolean isMoveDoable(WorkingMemory workingMemory) {
	return !serviceRequest.getTechnician().equals(technician);
}
```

OptiPlanner also needs to know how to undo the last move, by simply implementing the *createUndoMove(WorkingMemory)* method, which is shown as:

```
@Override
public Move createUndoMove(WorkingMemory workingMemory) {
	return new TechnicianMove(serviceRequest, serviceRequest.getTechnician());
}
```

EOSpy-AI should know how to create a new move by calling the *doMove(WorkingMemory)* method. Inside this method, it assigns the current technician to the service request and updates working memory:

```
@Override
public void doMove(WorkingMemory workingMemory) {
	FactHandle serviceRequestHandle =
workingMemory.getFactHandle(serviceRequest);
	serviceRequest.setTechnician(technician);
	workingMemory.update(serviceRequestHandle, serviceRequest);
}
```

Next is the generation of EOSpy-AI moves. At this moment we can only make a single move, but we need to generate a move set. It's up to you and your business case, but you will have to create a move set that can be sequentially combined to reach all the possible solutions.

The move set is generated with a *MoveFactory* implementation. We used a *CachedMovefactory* and implemented the *createCachedMoveList(Solution)* method to generate the move set. Inside this method, we generated a move for each of the requested services with all the available technicians, generating a full move set with all the possible combinations:

```
package com.arduinotron.ai_iot;

public class SelectNextTechnicians extends CachedMoveFactory {
	@Override
	public List<Move> createCachedMoveList(Solution solution) {
		TechniciansSolution techSolution = (TechniciansSolution) solution;
		List<Move> moves = new ArrayList<Move>();
		for (ServiceRequest sr : techSolution.getServiceRequests()) {
			for (Technician technician : techSolution.getTechnicians()) {
				moves.add(new TechnicianMove(sr, technician));
			}
		}
		return moves;
	}
}
```

Now that EOSpy-AI has generated the move set, we have to implement the *Solution* interface using the *SimpleScore* object. The *SimpleScore* object will contain the score of the current solution, which is calculated/recalculated after each step execution, and the best solution found after the execution will be the one with the highest score.

The solution was implemented for EOSpy-AI example using a *SimpleScore* score implementation:

```
package com.arduinotron.ai_iot;

public class TechniciansSolution implements Solution<SimpleScore> {
	private List<Technician> technicians;
	private List<ServiceRequest> serviceRequests;
	private SimpleScore score;

	public TechniciansSolution(List<Technician> technicians, List<ServiceRequest>
serviceRequests) {
		this.technicians = technicians;
		this.serviceRequests = serviceRequests;
	}

@Override
	public Solution<SimpleScore> cloneSolution() {
		TechniciansSolution solution = new TechniciansSolution();
			solution.score = score;
			solution.technicians = technicians;
			List<ServiceRequest> clonedServices = new
ArrayList<ServiceRequest>(serviceRequests.size());
			for (ServiceRequest sr : serviceRequests) {
				clonedServices.add(new ServiceRequest(sr));
			}
			solution.serviceRequests = clonedServices;
			return solution;
	}
}
```

7.4.4 EOSpy-AI Technician Rules explanation

In EOSpy-AI the score calculation is done using score rules, which makes the process of adding more constraints relatively easy and scalable. The *scoreCalculator* global will be automatically inserted into the working memory and has to be used to calculate the score of the current step. Ideally, this score has to be updated in a single rule using the weights generated by the score rules.

In EOSpy-AI, we can find several constraints, and one such constraint is that ideally, the technician must be in the same location as the service request. In the rule consequence, we need to logically insert a new *IntConstraintOcurrence* fact with a value. In this case, the value should be 1, to specify the weight of the broken constraint:

Example 6.4.4.1. Technician Rules definition

```
	rule "sameCity"
	    when
	        $sd : ServiceRequest(technician.location != location)
	    then
	        insertLogical(new IntConstraintOccurrence("sameCity", 1, $sd));
	end
```

Another constraint is the technician availability, which must be available to potentially be selected as the best solution. In this rule consequence, we assigned a higher weight than the previous rule constraint because it has more importance:

```
rule "isBusy"
    when
        $sd : ServiceRequest(technician.busy == true)
    then
        insertLogical(new IntConstraintOccurrence("isBusy", 6, $sd));
end
```

Finally, the last constraint is related to the technician skills required by the service requested. You can see, it has more importance than the location but less than the technician availability:

```
rule "skillMatch"
    when
        $sd : ServiceRequest($neededSkills : neededSkills, $tec : technician)
    then
        Set<Skill> tempSkills = EnumSet.copyOf($neededSkills);
        tempSkills.removeAll($tec.getSkills());
        insertLogical(new IntConstraintOccurrence("skillMatch",
             tempSkills.size() * 3, $sd));
end
```

The last rule is used to calculate the score of the current step, aggregating the weights of the *IntConstraintOccurrence* objects inserted by other rules. This rule was also assigned with a lower salience value to be only evaluated after the score rules.

Even though the use of *salience* is not the best practice, we need it to control the rule evaluation order to gain performance in the score calculation:

```
rule "hardConstraintsBroken"
salience -1 // Do the other rules first (optional, for performance)
    when
        $hardTotal : Number() from accumulate(
              IntConstraintOccurrence($weight : weight), sum($weight))
    then
        scoreCalculator.setScore(- $hardTotal.intValue());
end
```

Once the score rules are defined, we have to create the solver configuration file. It's a simple XML file that will configure the solving algorithm behavior. The currently available solver in OptiPlanner is local search, which can be configured to use Tabu search and simulated annealing acceptors.

The acceptor is used to activate a Tabu search or simulated annealing and will assign an accept-change value for each generated move. In this recipe, we are using Tabu search, and we are going to configure one of the several Tabu types. The solution Tabu is one of the recommended ones because it tends to give the best results and requires little or no tweaking.

Finally, we arrive at the *forager* section. A forager has the responsibility to gather all the accepted moves and pick the one that will be the next step and normally will be the accepted move with the highest score.

Here, we can configure whether and how the forager will pick the next move early using one of the following values. Using the *NEVER* value, a move will never be picked early, but there are two more possible values:

- *NEVER*: A move will never be picked early. This is the default value.

- *FIRST_BEST_SCORE_IMPROVING*: Picks the first accepted move that improves the best score. If there is none, then it behaves exactly the same as *NEVER*.
- *FIRST_LAST_STEP_SCORE_IMPROVING*: Picks the first accepted move that improves the last step score. If there is none, then it behaves exactly the same as *NEVER*.

It is possible that the selected technician isn't the absolute best option, but it should be an effective solution to the problem. Finally, you may want to know that there are many different solutions:

- A **possible solution** is a solution that does or does not break any number of constraints. Planning problems tend to have a large number of possible solutions; however, most of them are worthless.
- A **feasible solution** does not break any hard constraints. Sometimes, there are no feasible solutions, and every feasible solution is a possible solution.
- An **optimal solution** is one with the highest score. There is always at least one optimal solution to a planning problem.

7.5 Summary

In this example, we have seen Drools stream mode for complex event processing and OptiPlanner for solving complex IoTBPM event reasoning problems. Events in Drools are immutable objects with strong time-related relationships. CEP has a great value, especially if we need to make complex decisions over a high number of events. We've seen the use of time/length sliding windows and temporal operators.

This example also discussed the Drools type declarations which can define metadata on top of the existing types or define new types. As was demonstrated, new types are useful for rule decomposition.

Follow EOSpy website www.eospy.com for additional information on EOSpy-AI