OpenRules®

Open Source Business Decision Management System

Release 6.*

Installation Guide

OpenRules, Inc.

www.openrules.com

Dec-2016

Table of Contents

PRE-REQUISITES

OpenRules® requires:

- Java 2 Standard Edition: J2SE Development Kit release 1.6 or higher

You may execute the standard decision project provided in the workspace "openrules.dmn" or create similarly your own projects without writing a line of Java code. However, you may use Java to describe input/output objects and/or define some methods that you do not believe should be externalize. In this case to compile and execute your Java classes you also need to install

- Apache Ant 1.6 or higher*

OpenRules® works in any operating environment with standard Java. If you use Linux/Unix you are probably already well aware how to deal with Java and Ant. If you use Windows and are not very familiar with Java/Ant configuration issues, below are straight-forward recommendations.

Install JDK

1. Install Java Platform (JDK) 7u45 (or later) from http://java.sun.com/javase/downloads/index.jsp. Select JDK - Java SE Development Kit (not JRE). Click on the link for Windows x64 if your computer is 64 bit or Windows x85 otherwise.
2. Accept all defaults, and your JDK will be installed at c:\Program Files\Java\jdk1.8.0_05 or a similar directory using the actual release version you select.

Install ANT

1. Install Current Release of Ant (1.9.4 or later) from

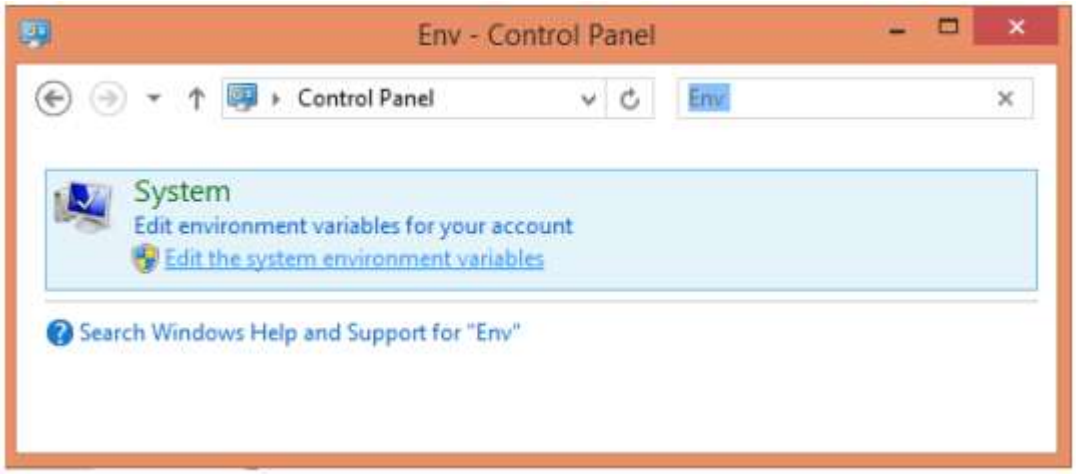
   http://ant.apache.org/bindownload.cgi.

2. Unzip the downloaded file to c:/apache-ant-1.9.4 and rename c:/apache-ant-1.9.4 to c:/ant.

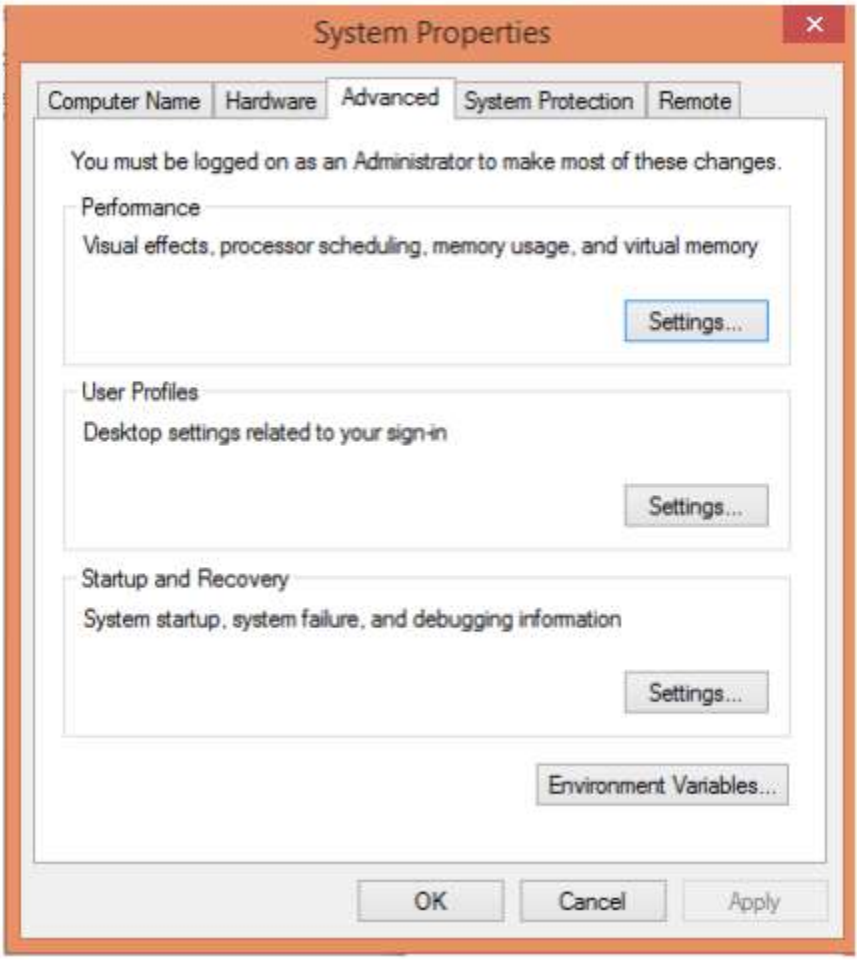
Set Environment Variables

Now you need to set up your environment variables. Here are detailed instructions for windows 8.

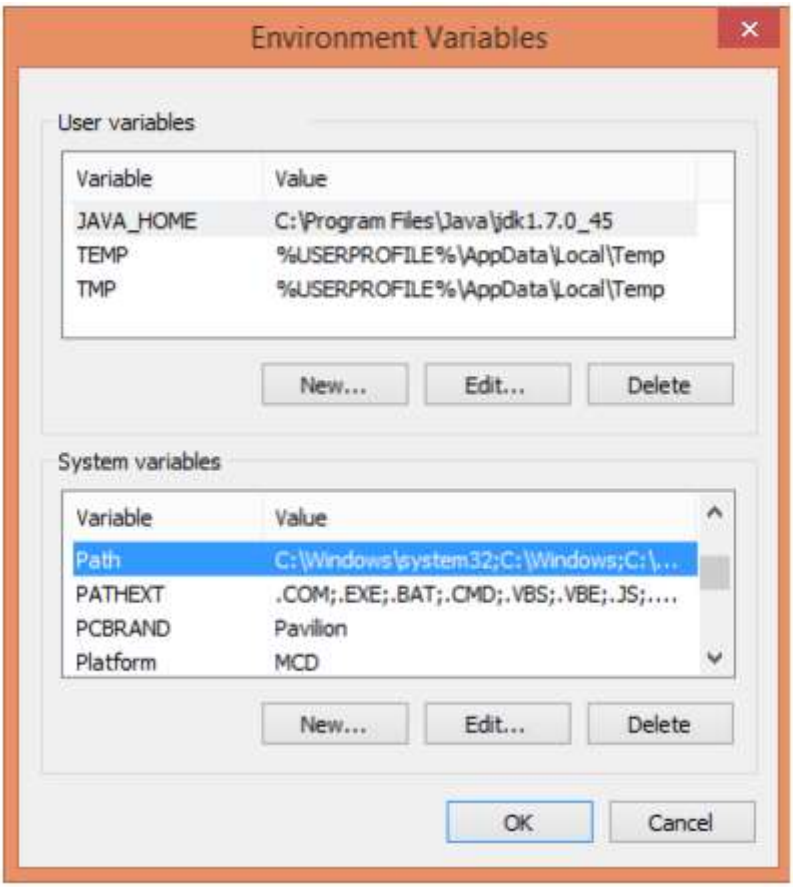
Right-click on the Windows Start button and select “Control Panel”. Type “Env” in the search box:



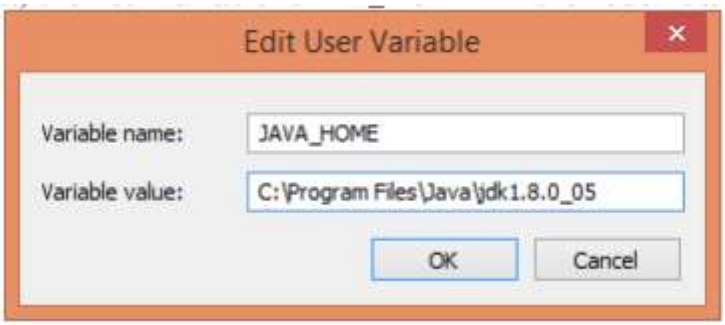
Select “Edit the system variables” and then select “Environment Variables…” from this dialog:



It will open the following dialog:

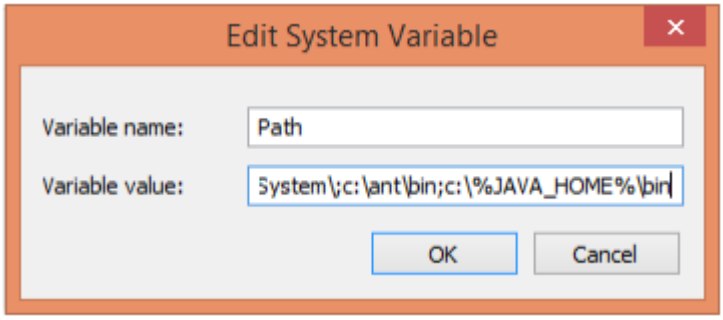


Add (or select) the user variable "JAVA_HOME" in the "User variables" list:



Define the Variable value using the actual directory, in which you downloaded JDK. Click "OK".

Now you need to select the system variable "Path" in the list "System variables". Click on the "Edit" and go to the very end of the "Variable Value" and add a semicolon following paths to the folders "bin" for the Ant and Java: …;`C:\ant\bin;C:\%JAVA_HOME%\bin`



Note. To access the Environment Variables for earlier versions of Windows do the following: left-click on the Windows Start-button, then right-click on My Computer, and choose Properties. Select the tab "Advanced" and then "Environment Variables".

Check the Installations

To make sure that you installed Java and Ant correctly, click on Start + Run and enter "cmd" to open a console window. Type

>java -version
>ant –version

It should show the correct versions of JDK and ANT, otherwise something was wrong in your variable settings.

DOWNLOAD

OpenRules® can be downloaded from http://openrules.com/download.htm. You will download one file "`openrules.decisions.zip`" that should be unzipped at your hard-drive (just avoid spaces in folder names). After unzip, the folder "`openrules.decisions`" will contain executable decision projects described in this document and several additional sample projects.

OpenRules® libraries and templates necessary for the execution of these projects are located in the main configuration folder "`openrules.config`". This folder is a simple placeholder for all OpenRules jar-files and related 3rd party jar-files. They are located in the subdirectory "lib". For simplicity, all sample projects depend on this project. It also contains build.xml file with a few targets used to compile, validate and run standard

projects. All standard sample projects depend on `openrules.config` project. However, you do not need this project if you copy all needed jars to you own directory and properly adjust your build-files.

The jar-file "`openrules.config/lib/openrules.all.java`" contains all necessary OpenRules® classes. All other jars are standard 3rd party jar-files that you may or may not need. Besides `openrules.all.java` a minimal OpenRules® configuration includes the following jars:

- `commons-logging-1.1.jar`
- `log4j-1.2.15.jar`
- `commons-lang-2.3.jar`
- `poi-3.10-FINAL-20140208.jar`
- `poi-ooxml-3.10-FINAL-20140208.jar`
- `poi-ooxml-schemas-3.10-FINAL-20140208.jar`
- `dom4j-1.6.1.jar`
- `xmlbeans-2.3.0.jar`

You may use different versions of these commonly used 3rd party tools.

Running Examples

You may execute any project by double-click on the file "`run.bat`" from a regular Window Explorer. If you use Linux, you should rename "`run.bat`" to "`run.sh`" (and make the proper changes).

If you make changes in the Java launcher (usually called Main.java in the folder "src"), you may compile your changes using "`compiled.bat`".

Working with Eclipse

To start using OpenRules® within Eclipse IDE, simply import all projects you are interested in from the downloaded folder "openrules.decisions" into your Eclipse workspace. The configuration project "openrules.config" always should be imported first. To take advantages of the OpenRules® Eclipse Plugin, install it directly from your Eclipse

using the following URL: http://openrules.com/downloads/protected/eclipse.

OPENRULES INSTALLATION STRUCTURE

You may download OpenRules® from http://www.openrules.com/download.htm. The standard OpenRules installation contains several projects that you may need (or not) based on your development needs. There are several OpenRules® configuration projects and several sample projects.

Sample Decision Models

OpenRules® standard installation comes in forms of different workspaces (folders) that consist of multiple projects. Each workspace should include the configuration project "openrules.config".

The evaluation version comes with the workspace "**openrules.dmn**" that includes the following sample projects with various DMN-based decision models:

Project Name	Description
DecisionHello	This is a basic decision model that decides how to greet a customer during different times of the day. For example, if a customer Robinson is a married woman and local time is 14:25, it produces a greeting like "Good Afternoon, Mrs. Robinson!". The model and test data are defined completely in Excel.
DecisionHelloWithErrors	This decision model is similar to "DecisionHello" but contains errors so you may see how they are diagnosed
DecisionHelloWithDates	This decision model is similar to "DecisionHello" but shows how to deal with Dates
DecisionHelloWithRequestResponse	This decision model is similar to "DecisionHello" but shows how along with the concept Customer to represent multiple business concepts such Request and Response
DecisionHelloJava	This decision model is similar to "DecisionHello" but test data comes from Java objects to demonstrate an integration with IT
DecisionModelTDM	This is an implementation of the Live Primer for "The Decision Model"

DecisionLoanOrigination	This is an implementation of the DMN example described in the Chapter 11
DecisionLoanPreQualification	This decision model specifies several decisions and related business rules for a loan pre-qualification process.
Decision1040EZ	This decision model demonstrate more complex rule decisions and rule families that calculate the US tax form 1040 described in this tutorial
DecisionPatientTherapy	This decision model defines medication and dosing rules as described in this tutorial
DecisionUpSell	This decision model demonstrates how deal with arrays of objects using complex up-sell rules organized in intuitive rule families. While presented rules use financial products, lists of products and services can be easily adjust to any industry.
DecisionVacationDays	This decision model specifies complex rules for calculation employee vacation days using simple formulas within a rule family.
openrules.config	The main configuration project that includes needed OpenRules® libraries and templates

You also may download the workspace "**openrules.decisions**" that includes sample projects that demonstrate additional decision modeling capabilities.

All sample projects are accompanied by readme-files, include test cases, and can be executed directly from a file manager or within an IDE such as Eclipse. Detailed descriptions of these projects can be found at the document "Getting Started".

Sample Rules Projects

OpenRules installation includes the following sample rule projects:

Project Name	Description
HelloRules	This is a basic rule project that decides how to greet a customer during different times of the day. For example, if a customer Robinson is a married woman and local time is 14:25, it produces a greeting like "*Good Afternoon, Mrs. Robinson!*". All rule tables and test data are defined completely in Excel.
HelloJava	This rule project is similar to "HelloRules" but test data comes from Java objects.
HelloGroovy	This Groovy project is similar to "HelloJava" but Excel rules are called from Groovy.
HelloJavaProtected	This rule project is similar to "HelloJava" but Excel files are protected
HelloJavaTemplates	This rule project shows how to use OpenRules templates by converting rules tables from the project "HelloRules" to summer and winter greeting rules that are based on the same template
HelloFromThreads	This project demonstrates how to organize a parallel execution of the same instance of the OpenRulesEngine in different threads and how to measure their performance
HelloJsr94	This rule project is similar to "HelloJava" but demonstrates how to launch OpenRules in accordance with JSR-94 standard Rule Engine API. This project depends on another project "*lib.jsr94*" that contains jar-files and sources for JSR-94 API.
HelloTwoEngineRuns	This project demonstrates how to run different Excel-based rules from the same OpenRulesEngine
HelloTwoEngines	This project demonstrates how to create and run different instances of the OpenRulesEngine
HelloXMLCustomer	This rule project shows how to use XML objects within a rule project
HelloXMLPeople	This rule project shows how to use array of XML objects within rule projects
AutoInsurance	This rules project shows how to organize complex rules-based calculations creating an auto insurance premium calculator. It demonstrates how to use OpenRules' data modeling mechanism for complex data structures with internal dependencies. It also

	demonstrates complex calculation methods.
HealthCare1	This project demonstrates how to implement the clinical guidelines. It defines business terms and facts, uses them to create and test the proper business rules. It does not use decision models. See the detailed description at http://openrules.com/docs/ClinicalGuidelines.Part1.htm.
Loan1	This rule project specifies several decisions and related rule families for a loan pre-qualification process (without use of decision models).
LoanExplain	This rule project demonstrates how to add an explanation mechanism to business rules by adding explanations of made decisions to the loan pre-qualification project "Loan1".
OverdraftProtection	This rules project shows how business rules are used to resolve such banking problem as overdraft. This rule service can be used to support a customer services rep while on the phone with a bank customer complaining about some problems with her account.
OneOrTwo	This project explains execution logic of decision tables showing that even trivial rules can be not so trivial.
Merge	This rules project shows how to merge and how not to merge cells within Excel-based rules tables.
DataArrays	This rules project shows how to access different data arrays defined in Excel from a Java program.
UpSellRules	This rule project demonstrates how deal with arrays of objects using complex up-sell rules organized in intuitive rule families. While presented rules use financial products, lists of products and services can be easily adjust to any industry.
Vacation	This project specifies complex rules for calculation employee vacation days using simple formulas within a rule family.
RuleRepository	This project demonstrates how to organize a rules repository using inter-related Excel rules books distributed between different directories and websites. It also shows how associate Java classes and methods with different rules that use them. Read more
RuleRepositoryDB	This project demonstrates how to organize a rules repository using a relational database. Read more
RuleRepositoryDBV	This project demonstrates how to organized a rules repository using a relational database with a built-in version control. Read more
SeasonRules	This rules project shows how to parameterize an OpenRules repository. It specifies different versions of similar rules inside

	inside the same repository. To define rules that offer different travel packages for different years and seasons, we use environment variables YEAR and SEASON inside include statements of the Environment tables that define which rules should be included into the project. Read more
ExternalRulesFromJava	This rules project shows how to use external rules defined in Java. Read more
ExternalRulesFromGUI	This rules project shows how to use external rules defined in run-time from a GUI. Read more
ExternalRulesFromDB	This rules project shows how to use external rules defined in database tables. Read more
ExternalRulesFromXML	This rules project shows how to use external rules defined in XML files. Read more
com.openrules.tools	An optional library with convenience Java classes that contains predefined methods shared by different OpenRules-based projects. It includes common printing methods, operators, a simple JDBC interface. It is a place where OpenRules and its customers contribute commonly used classes and methods. The library comes with all sources included so that it may be easily understood, modified or extended by customers.
openrules.config	The main configuration project that includes needed OpenRules libraries and templates. This project should be always present for other sample project to work.

All sample projects are accompanied by readme-files, include test cases, and can be executed directly from a file manager using "run.bat" or within an IDE such as Eclipse.

Sample Web Applications

OpenRules installation includes the following sample web applications and web services:

Project Name	Description
HelloForms	This is a simple rules-based Web Application. It demonstrates how to create a simple web application using Excel as a layout editor, deploy the application on Tomcat server, and run it from a Web browser. It is created based on a pure rules project "HelloRules" by adding a Web-based GUI.
DialogType0	This is a simple dialog template that includes all types of questions and two buttons “Next” and “Prev” that allow you to navigate throw pages.
DialogType1	Similar to DialogType0 plus a special Login page with a password and an ability to store/reload dialog sessions.

DialogType2	Similar to DialogType1 but adds a menu to navigate between pages.
DialogType3	A more complex questionnaire that may have many questions on each page. The next question to ask is always displayed on the top of the page and all answered questions below it.
DialogCreditCard	A simple web questionnaire for a credit card application
Dialog1040EZ	A real-world questionnaire that allows a user to interactively fill is notorious US tax form 1040EZ and automatically generate a resulting PDF document.
HelloJsp	This is a simple rules-based Web Application with business logic in OpenRules and presentation logic in JSP. It demonstrates how to create a simple JSP-based application with OpenRules, deploy the application on a Tomcat server, and run it from a Web browser.
HelloJsr94Jsp	This web application is similar to "HelloJcp" but uses JSR-94 Interface to create and run OpenRulesEngine for a web application. The main differences are in the file GreetingEngine.java.
HelloWebLogic	This is a simple rules-based Web Application demonstrates how to create a simple JSP-based application with OpenRules, deploy the application on the BEA WebLogic server, and run it from a Web browser.
HelloWebSphere	This is a simple rules-based Web Application demonstrates how to create a simple JSP-based application with OpenRules, deploy the application on the IBM WebSphere server, and run it from a Web browser.
HelloWS	This project demonstrates how to deploy basic rule project as a Web Service. It contains all components necessary to configure and deploy business rules defined in the main xls-file war/rules/HelloWS.xls and included xls-files from the sub directory war/rules/include. By default, this project deploys these rules as a Web Service on a running Tomcat server.
HelloWSJavaClient	This Java project demonstrates how to call a Web Service "HelloWS" from an Java application (client).
HelloWSCustomer	This Java project is similar to a Web Service "HelloWS" but also demonstrates how to use custom objects within generated WSDL
HelloWSCustomerClient	This Java project demonstrates how to call a Web Service "HelloWSCustomer" from an Java application (client).
Loan2	This web application provides a GUI for a basic rules project "Loan1". This is an example of a presentation-oriented Web Application that uses business rules for loan *pre*-qualification. It is based on the standard OpenRules library "*openrules.forms.lib*", and in particular uses "Dialog.xls" for interaction logic. More explanations for this

	project can be found at http://openrules.com/docs/man_forms.html.
HealthCare2	This web application provides a GUI for a basic rules project "HealthCare1". This is the second part of the clinical guidelines use case that models interaction sessions between a doctor and a rules-based system. See the detailed description at http://openrules.com/docs/ClinicalGuidelines.Part2.htm.
LoanDynamics	This is an example of a presentation-oriented Web Application that uses business rules for loan approval. This project supports a Loan Approval process with an ability to dynamically add/delete/modify different securities related to the initial loan application. Read more.
LoanStudent	This web application demonstrates how to use OpenRules Forms to define a complex student loan approval process.
NumberGuess	This is an example of a Web Application that implements a number guessing game that is frequently used as a sample application for different Web development techniques. See the detailed description at http://openrules.com/docs/man_game.html.
PartyManager	This web project demonstrates how to create dynamic web tables. Read more
ConfigForms	This web project demonstrates how to implement related combo-boxes. On the first form you enter information about a customer. Based on the entered attribute "Gender" the choices for the attribute "Interest" are dynamically changed. You also have two input fields with a choice of colors. When you select the first color it will be excluded from possible choices for the second color.
1040EZ	This is an example of an interactive web application with a complete implementation of the well-known US tax form 1040EZ. Supports complete business logic, data and a genuine form required by government regulations. This project also demonstrates PGF generation by automatically producing final tax documents in the PDF format using the standard PDF form. The project uses the standard OpenRules library openrules.forms.lib, and in particular uses Dialog.xls for interaction logic and PredefinedTypes.xls. It relies on the included demo version of the PDF generation library "pdfgen" from the Big Faceless Organization.
openrules.config	The main configuration project that includes needed OpenRules libraries and templates
openrules.forms.lib	This is a standard OpenRules library to support Web applications. It provides Excel files that supports typical interaction constructions, layouts and layout elements, and predefined data types.

To work with Web-based applications, you will need web containers or application servers

such as Apache Tomcat to be installed. You may download a free Tomcat container from http://tomcat.apache.org/download-60.cgi. For Windows download and run the file "apache-tomcat-6.0.29.exe" choosing 32-64 Windows Service Installer.

By default, all OpenRules web projects are tuned to work with Tomcat. The actual installation of your Tomcat is described in the file "build.properties" as

```
appserver.home=C:/apache-tomcat-6.0
```

Make sure that appserver.home points to your own Tomcat installation directory.

If you use IBM WebSphere or BEA WebLogic, there are two projects "HelloWebSphere" and "HelloWebLogic" that demonstrate how to work with these servers.

To deploy your rules projects as Web Services you will also need to install Apache Axis 1.4 – see the project "HelloWS". If you have problems to download Axis 1.4 you may get it from http://openrules.com/downloads/my/axis-1_4.zip

All sample projects are accompanied by readme-files, include test cases, and can be deployed and executed as web applications.

TECHNICAL SUPPORT

Direct all your technical questions to support@openrules.com or to this Discussion Group. Read more at http://openrules.com/services.htm.