



Voyager *Wizard*[™] Rules Developer Guide

Version 1.0 for Voyager 8.0-RC3

Table of Contents

Introduction	4
Overview	4
Preface	4
Rule Agent Requirements	4
JBoss Rules information	5
VOYAGER Wizard Ruleset Builder	5
Contacting Technical Support	5
Running Rules in a Voyager Agent	6
Loading a VOYAGER Wizard Ruleset	6
Loading Initial Working Memory	6
Loading Globals	6
The InferencingAgentAction	6

<This page intentionally left blank>

Introduction

Overview

[Voyager](#)[™] provides the ability to run inference engine rules in a mobile agent and report the results back to the caller. Rules can be specified either by constructing a [Voyager](#) Ruleset object (either programmatically or via the [Voyager](#) Ruleset Builder) or by writing a JBoss Rules file and executing it directly. Both techniques are covered in this document.

Preface

The purpose of this manual is to provide information necessary to execute inference engine rules within a [Voyager](#) agent. This is not meant to be a treatise on writing inference rules. This guide assumes a basic knowledge of Java and distributed computing concepts. [Voyager](#) uses the JBoss Rules engine to execute inference rules.

This preface covers the following topics:

- Rule Agent Requirements
- JBoss rules information
- VOYAGER *Wizard* Ruleset Builder
- Contacting technical support

Rule Agent Requirements

Before attempting to use the Ruleset Builder features of [Voyager](#).

- A Java Development Kit (JDK) installed on your computer. Voyager requires JDK 1.4.2 or later. You can download the latest release of JDK from www.javasoft.com at no charge.
- A working version of [Voyager](#) installed.
- A searchable version of the Voyager documentation is available at the Recursion Software web site at www.recursionsw.com.

Copyright © 2006 – 2010 Recursion Software, Inc.
All Rights Reserved

JBoss Rules information

Information about JBoss Rules syntax can be found in the following places:

- Main site: <http://labs.jboss.com/portal/jbossrules/index.html>
- Documentation: <http://labs.jboss.com/portal/jbossrules/docs/index.html>

See the documentation for information on the syntax of rule conditions (left hand side of a rule) and consequence. (right hand side of a rule) The Ruleset Builder provides a framework for writing inference rules, but the syntax of the rule parts must still be compiled and executed by the JBoss Rules engine.

VOYAGER *Wizard* Ruleset Builder

The VOYAGER *Wizard* Ruleset Builder can be used to author rules for execution in a [Voyager](#) agent. The VOYAGER *Wizard* Ruleset Builder helps the developer structure rules into a Goal oriented fashion.

Contacting Technical Support

Recursion Software welcomes your problem reports, and appreciates all comments and suggestions for improving Voyager. Please send all feedback to the Recursion Software Technical Support department.

Technical support for Voyager is available via the web, email, and phone. The Recursion Software Support website, at <http://support.recursionsw.com>, provides access to searchable FAQs (Frequently Asked Questions), technical articles, and other information. It also provides access to the Recursion Software Technical Support incident tracking system, where you can submit support issues and track them. You can also contact Technical Support by sending email to psupport@recursionsw.com or by calling (972) 731-8800.

Running Rules in a Voyager Agent

Loading a VOYAGER *Wizard* Ruleset

The VOYAGER *Wizard* uses Java Beans XML serialization to passivate Rulesets to persistent storage. Use the `java.beans.XMLDecoder` to retrieve them from persistent storage.

```
java.beans.XMLDecoder d =
    new java.beans.XMLDecoder(
        new java.io.BufferedInputStream(
            new java.io.FileInputStream("MyRulesetFile.vrs")
        )
    );
Ruleset ruleset = (Ruleset)d.readObject();
d.close();
```

Loading Initial Working Memory

Your Ruleset may require that some objects be present in Working Memory when rule execution begins. You can accomplish this by populating a `java.util.Collection` with the objects you want in Working Memory before execution. These objects must implement `java.io.Serializable` because they will be transported along with the Agent to the `AgentPeer` before execution.

Loading Globals

Your Ruleset may make use of Globals during the execution of the Consequence portion of its rules. You have several options when instantiating Globals for an agent. The first is to instantiate a Global before launching the agent. This option is necessary if the Global contains some state that must be specified at launch time. The second option is to not instantiate the Global and allow it to be instantiated on the `AgentPeer`. This is desirable if you do not want the overhead of sending the Global object to the `AgentPeer` along with the Agent. To do this the Global object must have an accessible no-arg constructor.

As with Working Memory objects, Global objects must implement `java.io.Serializable`. This is because the contents of each Global object are reported back after Agent execution is finished unless otherwise specified.

The InferencingAgentAction

All three of the above items, a Ruleset, initial Working Memory contents, and Globals, combine to create an `InferencingAgentAction` object:

```
XMLDecoder d =
    new XMLDecoder(
        new BufferedInputStream(
            new FileInputStream("MyRulesetFile.vrs")
        )
    );

Ruleset ruleset = (Ruleset)d.readObject();

d.close();

Collection<Serializable> wmContents = new LinkedList<Serializable>();

...populate wmContents...

Map<Global,Serializable> globals = new TreeMap<Global,Serializable>();

...populate globals...

IAgentAction agent = new
InferencingAgentAction(ruleset,wmContents,globals);

...launch agent...
```

The InferencingAgentAction can then be deployed to the AgentPeer for execution.