AgentBuilder README Verson 1.4 (Solaris)

```
========================================================================
                             AgentBuilder®
               An Integrated Toolkit for Constructing
                     Intelligent Software Agents

                    © Copyright 2004 Acronymics, Inc.
                        http://www.acronymics.com
                       http://www.agentbuilder.com

========================================================================
========================================================================
                        AgentBuilder Pro 1.4     (7/30/04)
                        AgentBuilder Lite 1.4    (7/30/04)
========================================================================

IMPORTANT NOTE: If you are upgrading from a previous version of
AgentBuilder, your agencies and agents will need to be re-saved under
the new version of AgentBuilder. This can be done by bringing up the
properties dialog (for the agent or agency) and clicking on the
communications button. Click the Ok button for the communications
dialog. You will also need to remove all spaces embedded in the agent
name.

IMPORTANT NOTE:  Because of changes in Java 1.4, AgentBuilder agents
can no longer have embedded blank characters in the agents' names.
That is, "Example Agent 1" is no longer a valid agent name.  You
should use something like "ExampleAgent1" or "Example_Agent_1".
If you have existing agents built with AgentBuilder, you will need
to change the agent's names to remove the embedded blank characters
before running the agents in the new version of AgentBuilder.

NEW FEATURES
------------

Both AgentBuilder Lite and AgentBuilder Pro have been upgraded to
version 1.4. Both Lite and Pro include the following new features:

- Java 1.4 Support. The toolkit and agents now support both Java 1.4
  and Java 1.1.8. Developers can now use either version of Java as part
  of their development effort. AgentBuilder 1.4 is available for Windows.

- Special Note for AgentBuilder Pro users on Windows:
  If your system user name has an embedded blank character (e.g.,
  "John Doe") you may have trouble using the Agency Viewer tools.
  There is a simple work-around available that requires an addition to
  the AgentBuilder.bat file.  See the file in the Documentation directory
  titled "Modifying Windows AgentBuilder.bat File" for instructions.

- Bug Fixes. Version 1.4 includes numerous bug fixes and performance
  enhancements.  Download a free evaluation copy


========================================================================
        System Requirements For Java Runtime Environments ver 1.4
========================================================================

For information about the JRE for the Solaris environment  see

     http://java.sun.com/j2se/1.4.2/install.html

For important changes in Java since Java 1.3 see

    http://java.sun.com/j2se/1.4.2/compatibility.html
========================================================================
RUNNING AGENTBUILDER WITH A DIFFERENT JRE

AgentBuilder ships with the Java Version 1.4 JRE.  However, you can use
any JRE that you like with AgentBuilder.  Follow the instruction below
 to modify your AgentBuilder installation to run with a different JRE.

These instruction assume you want to use the latest JRE (1.6 at the time
this note was prepared).  These instructions are for the Windows
environment.  You can modify them for other environments.

     1. If you don't have the latest JRE, you can download if from here:
http://developers.sun.com/downloads/
     2. Install the JRE and note the install directory.
```

3. Locate the "AgentBuilder.bat" file in Windows Explorer. The file is located in the directory where AgentBuilder was installed. Open "AgentBuilder.bat" with an editor. To do this you can right-click on the file and select edit.

4. to change AgentBuilder to use the newly installed JRE (1.6.0 in this example).

```
     Replace: .\jre\bin\java
     With this: "C:\JDK1.6.0_14\bin\java"
```

Assuming you have installed the JRE in C:\JDK1.6.0_14. Save and close the "AgentBuilder.bat" file.

5. Repeat steps 3 and 4 for the "Engine.bat" file.

GETTING STARTED WITH AGENTBUILDER

==

BECAUSE OF CHANGES IN THE WAY THE SUN JRE IS PACKAGED AND DISTIBUTED, THE INSTALLATION STEPS FOR VERSION 1.4 ARE SLIGHTLY DIFFERENT THAN IN PREVIOUS VERSIONS. IN PARTICULAR, THE JRE IS NO LONGER AUTOMATICALLY INSTALLED IN THE AGENTBUILDER DIRECTORY STRUCTURE. YOU MUST INSTALL THE JRE AND THEN CREATE LINKS TO THE APPROPRIATE DIRECTORIES IN THE AGENTBUILDER DIRECTORY STRUCTURE.

PLEASE READ THE FOLLOWING MATERIAL CAREFULLY.

Solaris Installation NOTES

==========

1. Change to the directory where you will install AgentBuilder.

```
cd /usr/local/        # or wherever you install 3rd-party software
```

   The suggested installation location is in /usr/local/agentBuilder.

2. Unpack the file using one of the following methods. (You must be super user to modify system files.)

```
     gzip -dc agentBuilder-1.x-solaris-sparc.tar.gz | tar -pxf -
```

3. After unpacking, you will find one of the following files/directories if you have downloaded the version that includes the JRE. The CDROM versions of the software always include the JRE.

```
j2re-1_4_2_05-solaris-sparc.sh
j2re-1_4_2_05-solaris-sparcv9.sh
agentBuilder
```

   If you did not download the version with the JRE you will only see the agentBuilder directory listed

3. Now, cd to the agentBuilder directory where you will see the following directory structure:

```
./bin           - Contains script to run AgentBuilder
./html          - Contains html documentation
./lib           - Contains needed jar files
./lib/repository  - Contains system repository
```

   NOTE: there is no ./jre included.

   If you already have a Solaris JRE Version 1.4.2 on your machine then skip to step 5 of these instructions.

4. You now need to install the JRE on your machine. There are two executables provided for this purpose. First, cd to the location where you wish to install the jre; e.g., /usr/local.
   Now execute the j2re-1_4_2_05-solaris-sparc.sh program. You will be provided with an opportunity to accept the Sun license terms and then the JRE installed. If you have a 64 bit machine, you need to execute the j2re-1_4_2_05-solaris-sparcv9.sh as well.

   You can find complete instructions for installing the jre at
   http://java.sun.com/j2se/1.4.2/jre/install.html

5. You now need to provide a symbolic (soft) link to the JRE so AgentBuilder can find the JRE. The details of the link depend on where your JRE is located. If you installed the JRE in /usr/local then the link command will look something like

```
     ln -s /usr/local/j2re1.4.2_05 /usr/local/agentBuilder/jre
```

6. Add the AGENTBUILDER_HOME environment variable to your system (or local) .cshrc or whatever appropriate resource config file you use. It should point to the directory in which you have installed AgentBuilder. If you installed it in the default directory (/usr/local/agentBuilder/) then you DON'T have to do this.

5. Either add the agentBuilder/bin directory to your path, or create some links to it from a directory already included in your path. For example, we recommend creating two links from /usr/local/bin.

```
  (You will need to be su to do this)
   cd /usr/local/bin
   ln -s /usr/local/agentBuilder/bin/agentBuilder .
   ln -s /usr/local/agentBuilder/bin/engine .
```

Launching an agent engine from a Unix shell script

An agent engine can be started on its own Java virtual machine (separate from the AgentBuilder tools) via the engine shell script found in the the agentBuilder bin directory. This bin directory also contains the agentBuilder shell script, so it should already be in your path. Type:

```
    engine -i
```

on a command line to bring up the Agent Engine Options dialog, which allows you to specify the RADL file and run-time options to be used. See chapter 3 in the Reference Manual for an explanation of the options in the Agent Engine Options dialog. You can also start an agent engine directly by typing:

```
    engine radl-filename
```

Chapter 3 also describes the options that can be used with the engine script. Typing engine -h on a command line will print a listing of the available options.

KNOWN PROBLEMS with AgentBuilder 1.4

- If you are upgrading from a previous version of AgentBuilder, your agencies and agents will need to be re-saved under the Version 1.4 of AgentBuilder. This can be done by bringing up the properties dialog (for the agent or agency) and clicking on the communications button. Click the Ok button for the communications dialog and the properties dialog.
- You can no longer use embedded blanks in an agent's name. For example, you should name your agents SupplyAgent of Supply_Agent rather than simply Supply Agent. This is a limitation introduced with the 1.4 version of Java.

```
========================================================================
                        AgentBuilder Pro 1.3a     (6/15/01)
                        AgentBuilder Lite 1.3a    (6/15/01)
========================================================================
```

IMPORTANT NOTE: If you are upgrading from a previous version of AgentBuilder, your agencies and agents will need to be re-saved under the new version of AgentBuilder. This can be done by bringing up the properties dialog (for the agent or agency) and clicking on the communications button. Click the Ok button for the communications dialog and the properties dialog.

NEW FEATURES

Both AgentBuilder Lite and AgentBuilder Pro have been upgraded to version 1.3a. Both Lite and Pro include the following new features:

- Java 1.3 Support. The toolkit and agents now support both Java 1.3 and Java 1.1.8. Developers can now use either version of Java as part of their development effort. AgentBuilder 1.3 is available for Windows

98/NT/2000, Solaris, and Linux.

- CORBA Support. The toolkit and agents now support CORBA IIOP in
  addition to RMI for communications between agents.

- HP E-SPEAK Support. The toolkit and agents now support
  Hewlett-Packard's E-Speak protocol.

- TCP/IP Socket Support. The toolkit now allows construction of
  agents that use TCP/IP protocols. Now agents can now communicate
  using raw TCP/IP sockets.

- User-Friendly Agent Engine Options Dialog. The Agent Engine Options
  dialog has been updated to make it much easier to use.

- Bug Fixes. Version 1.3 includes numerous bug fixes and performance
  enhancements.

===

KNOWN PROBLEMS with AgentBuilder 1.3a

- It is possible to have agents that ran OK on previous versions but
  will not work properly on this new version. The problem is caused by
  changes in the Javavirtual machine.

- Under Windows 2000, on certain machines you may be unable to run any
  java application that contains a GUI. The workaround is to turn off
  hardware acceleration. You can access this setting by going to your
  display properties, selecting the settings tab, click on the advanced
  button, and then the Troubleshooting tab. You need to set it on the
  2nd from the left tick mark.

===

System Requirements For Java Runtime Environments ver 1.3

===

For installation instructions and system requirements for this
release, see

http://java.sun.com/j2se/1.3/install.html

WINDOWS 9x/NT/2000/XP NOTES

===================

1. Run the agentBuilder.exe program. It is a self extracting
installation program. The setup program will then guide you through
the installation of AgentBuilder onto your system.

2. There should be a new AgentBuilder Toolkit menu in your
Startup -> Programs folder.

WINDOWS PROBLEMS

Initial Run Problems

After installation the first time agentBuilder is run the java virtual
machine reports an error while trying to create the user's directory.
The error can be ignored.

Windows Networking

The "ExampleAgent4" and any agent which uses a PacCommSystem
or communicates with other agents requires the windows networking to
be setup properly. There are two ways to ensure this:

1) Add or modify your c:\windows\hosts file to contain your hostname.
(See the example hosts file in the distribution's html directory)
For NT the hosts file is in d:\winnt\system32\drivers\etc\hosts

2) Turn on DNS (domain name service). This requires going into the
control panel and editing your network setup. You MUST have a valid
DNS server in order to use this technique.

Shortcuts

If, for any reason, you need to re-create the shortcuts to AgentBuilder and/or AgentEngine, use the following properties.

```
AgentBuilder - Target = < install-dir >\agentBuilder.bat
               Working Directory = < install-dir >

AgentEngine - Target = < install-dir >\engine.bat
              Working Directory = < install-dir >
```

Launching an agent engine from the Windows Start menu

An agent engine can be started on its own Java virtual machine (separate from the AgentBuilder tools) by selecting AgentEngine in the AgentBuilder folder. Select

```
    Start -> Programs -> AgentBuilder Toolkit -> AgentEngine
```

This will bring up the Agent Engine Options dialog, which allows you to specify the RADL file and run-time options to be used. See chapter 12 in the user's guide for an explanation of the options in the Agent Engine Options dialog.

Example Agents

There are several example agents included with the toolkit.

Quick Tour Agency

A simple to more complex group of agents that are discussed in the User's Guide.

ExampleAgent1: The simplest possible agent. An agent with 1 rule that just prints "Hello World" to standard out.

ExampleAgent2: This agent is slightly more complex. It uses are rule to constantly print out "Hello World".

ExampleAgent3: This is a more complex agent than the other two. It uses a user defined class (PAC) and four rules to print, sleep the agent for a set amount of time and then exit the system.

ExampleAgent4: This is the most complex example agent. It utilizes a PAC to display an interface to the user, interact with the user and exit the system when given a command from the user.

HelloWorldAgency

HelloWorld: This is the same agent as Example Agent 4. It is here to demonstrate that an agent can be in its own agency.

SimpleBuyerSeller Agency

This is an example of a group of interactive agents. They depend upon each other for operation. In this example a PriceRequest PAC is used to record product name, quantity, store name and the quoted price. A PriceRequest PAC is sent from the Buyer to the store. The store fills in the PriceRequest object and returns the completed object to the buyer agent. The Buyer agent then prints this information out.

To run the buyer-seller agents you must run them all at once. Go to the AgentManager, load the SellerAgent and run it. Then load the Buyer Agent and run it. You should see the output in the agent engine console window.

Note: The communication information is critical for the buyer and store agents to run properly. The agents will work together with the default configuration ONLY if they are all run on the same machine. If they are to be run on different machines then the IP address (set in the agent properties dialog) must be set explicitly for each agent. You can not use the CURRENT_IP_ADDRESS, it must be a machine name like: washington.reticular.com

These agents are only intended to demonstrate inter-agent communications and rudimentary reasoning. The agents in the current version are very simple and are intended for illustration purposes only.

Simple Buyer-Seller with Protocol Agency

This is an example is a close replication of the Simple Buyer-Seller Agency, with the primary difference being that this agency is built using a protocol. It is useful as an example of getting familiar with the AgentBuilder Pro tools.

===

Built-In PACs

===

The following is a list of pre-defined pacs that exist for each new agent that is created. The JavaDoc pages are linked.

-
-
- Agent
- AgentInfo
- PacCommSystem
- RmiCommInfo
- Time
- KqmlMessage
-

===

HELP AND BUG REPORTING

===

There are several different ways of reporting problems and getting help. You can report problems by:

1. email us at support@agentbuilder.com

3. FAX us at (480) 615-1297

4. Voice Call at (480) 615-8543. Ask for AgentBuilder Support.

On-line Documentation can be found at:
  http://www.agentbuilder.com/

===

NOTES AND KNOWN PROBLEMS

===

Development Recomendations

- As with any software development, spend time designing your agent before you start, especially the rules. Writing the rules is the HARD part of building AgentBuilder agents, and designing them well before you start will speed up the whole processes.

- Choose your PAC names, method names and data names carefully. Changing them after you've built actions, PACs and rules breaks all of the rules and actions currently using them.

- Perform iterative development. This means to put one or two rules in your agents and then test them. There is an input window in the engine console to allow you to enter strings into the agent's mental state at runtime. AgentBuilder Pro has an Agent Debugger that makes this much easier.

- Debug your PACs as much as possible before developing your agent. Debugging your rules/actions/commitments at the same time will be very difficult.

- Make all of your PACs implement cloneable and serializable (unless they are interface PACs). You can run into runtime problems during testing if you don't.

- Use the import in the Object Modeler as much as possible. And remember you can delete methods/data that aren't going to be used.

- Changes to PACs will be required once you start using them in the rules is going to happen. If you change the API for your class you will need

      1) reimport it into the object model and save it.
      2) Update (or import the class) the object model from the
      PacEditor, then issue a save.
      3) Now the new API is usable in the rules.

- RunTime Exceptions

  - The TargetInvocationException is the grab bag of exceptions for the engine. It usually means that either you've accessed a classes that is null or a problem has ocurred in one of the PAC methods.

  - The ClassNotDefinedException means your CLASSPATH isn't correct or a naming problem has crept in.

  - The ClassCastException means either an invalid cast was specified in a rule or an argument to a method couldn't be typecast to the correct type.

  - The NULLPointerException can mean either a RADL file is incorrect and cannot be parsed OR an object at runtime was null.

- If your computer is low on RAM, buy more! Java programs require large amount of memory and run much faster if they are not swapping.

- Finally, don't be afraid to ask for help. By sending email to support@agentbuilder.com we can help you with some of your problems. Please send your RADL files and the error log in your requests for help.

AgentBuilder Repositories

If you look at the agentbuilder home directory (.AgentBuilder on unix or < install-dir > /users/current-user on windows) you will see a repository directory. This repository directory contains all of the information created and/or used by the developer for his/her agents. It is important to know that all dependencies for the agents must be imported into that repository. If you want to use information from another repository you must copy desired information from that repository into the current user's repository. Any direct manipulation of the files, such as renaming or deleting, will result in the loss of data.

Agents with Interfaces

The source code for the Example Agent 4 interface is included in the agentBuilder/src directory. We recommend that you look at how we built this agent and use it and its interface as a starting point for your interfaces.

Parsing Problems

One problem all developers run into is that the engine is unable to parse a RADL file generated by the tool. This can happen when you run the agent from the AgentManager or directly from the engine. What this means is that some part of your agent wasn't correctly constructed. The usual culprit is a rule that wasn't formed properly. It is fairly easy to built a pattern in the rule editor which fails. It may take some time to determine what is causing the problem at run time. The usual approach is to try to determine which rule, then which pattern is causing the problem. To avoid this problem it is highly recommended that you do iterative development, rather than building all of the rules and then trying to run the agent.

User PACs

There are restrictions on what types of methods and data can be visible in a PAC. Currently, an Object Model (built in the ontology tools) must either define all referenced classes, which are not supported Java types, in the same object model. This means that if you import a class which references another class in a method or as a data field, you must also import that class. Otherwise, when you import the classes into the PAC editor it will fail. Another restriction is that you cannot use arrays. This shortcoming will be addressed in a future release of AgentBuilder.

These types of problems will only manifest themselves when you import classes from an object model. The solution is to return to the object modeler and either delete the methods/data that references the class or import the class.

Agent Properties Communications Dialog

To enter or modify the port number, the user must first double-click on the port number cell. Once the user has entered a port number, the user must press the enter key in order for the port number to be modified. If the enter key is never pressed, the new value will never be set for port number.

There is a bug in the implementation of the table. If the user were to single click on the port number cell, the cell will receive key events. However, two things will happend, if the user attempts to modify the port number after single-clicking on the cell. One, the alphanumeric keys will be echoed, and all other keys will become control characters. And two, the port number will never actually be modified.

Window Menu behavior

There is currently a Windows NT limitation in the Windows menu in that it is not able to bring iconified windows to the foreground. This is an unsupported feature in the current JDK, and no workaround has yet been found.

OutOfMemoryError Exceptions

There is a known problem with OutOfMemoryError exceptions. If this should occur, try to save any unsaved data. AgentBuilder will have to be restarted.

```
=======================================================================
                        AgentBuilder Pro 1.2    (8/3/99)
                        AgentBuilder Lite 1.2    (8/3/99)
=======================================================================
```

IMPORTANT NOTE: If you are upgrading from a previous version of AgentBuilder earlier than 1.1 read the known problems for Lite 1.1 for help on the upgrade (at the end of this README).

- AgentBuilder 1.2 has both Java 1.1.8 and 1.2 support. Make sure you have the correct version for your application requirements. Both distributions include the production JRE for their respective platforms.

- Currently only Solaris Sparc and Windows 95/98/NT are supported, linux and SGI versions are due out soon.

NEW FEATURES

- Support for Java 1.2.

- The ability to run multiple agents in one virtual machine. Many changes went into implementing this new capability, including a new multi-agent console, the ability to run an entire agency and tools to allow the user to specify which agents run together in a single virtual machine.

- Printing capabilities have been added to the graphical tools (Object Modeler, Concept Mapper and Protocol Editor). This feature is limited to printing on a certain area size. This limitation will be improved in future releases.

- The ability to select and move multiple nodes at once in the graphical tools (Object Modeler, Concept Mapper and Protocol Editor).

- The ability to directly invoke methods on Return Variables (in the rule editor).

- Ability to name return variables as soon as they are created.

- The protocol update mechanism was improved to replace all "protocol related patterns" while leaving manually edited patterns in the rule. Previously, patterns were added, and never deleted, which resulted in more work for the user.

- Added the ability to detect when sending a message failed. The KqmlMessage object is flagged with an error and then asserted into the mental state. A "sendError" flag was added to the KqmlMessage PAC to support this new feature.

- A "user" field was added to the Agent PAC. This should be the name of the person (or organization) for whom the agent works.

```
KNOWN PROBLEMS with AgentBuilder 1.2
--------------------------------------
```

- The Help viewer doesn't work in Windows with JRE 1.2.1. You can still view the help pages by using your own browser(e.g. Netscape or Internet Explorer).

- It is possible to have agents that ran OK before but won't run with the new version. The problem is attributable to changes in the Java virtual machine.

- Occassionally a computeIntersection exception is thrown during tool usage. These can generally be attributed to problems in Sun's Swing release and can be ignored.

- Ocassionally, when the agent properties are updated (at least from the project manager) the system hangs. The save operation is completed, but a repaint isn't performed on the parent frame. Restarting the system is required.

- The file trace from the agent engine isn't getting the verbose output, only the System.out.println statements.

- Under Windows the file dialogs aren't working properly. If you choose the default directory it works, but if you try to specify the root directory it won't show files.

- Occassionally, when running the Agency Viewer on slower machines we've seen some failures. On the faster machines with lots of RAM it seems to handle over 10 agents at once (remember that each agent runs on its own virtual machine).

- Occassionally, when starting AgentBuilder the interface won't appear. When this happens, kill the process and restart it.

- Problems on SGI machines. There is a problem with some dialogs being displayed properly on SGIs. Usually a resize will cause the dialogs to be displayed properly. There is also a problem running AgentBuilder on another UNIX platform and displaying on a SGI. Removing the AgentBuilder.properties file before running the tools solves the problem.

- When changing the Agent Properties dialog, ocassionally the tool will hang. You will need to kill and restart AgentBuilder when this happens.

```
=========================================================================
                        AgentBuilder Pro 1.1    (3/23/99)
                        AgentBuilder Lite 1.1    (3/23/99)
=========================================================================
```

IMPORTANT NOTE: If you are upgrading from a previous version of AgentBuilder Lite (or the Pro beta) read the known problems for Lite 1.1 for help on the upgrade.

AgentBuilder Pro builds upon and uses the AgentBuilder Lite tools. A new version of Lite, 1.1, has been developed and is being released as

part of Pro.   The AgentBuilder Pro product features two new
subtoolkits for agency and protocol development.

- The Pro 1.1 release has some functional additions to the Protocol
editor, Object Modeler and Pac Editor. As well as several bug fixes
from the previous release.

- Both the Pro and Lite version are using the 1.1.7 of java. Both
distributions include the production JRE for their respective platforms.

- The agency tools are used for working with agencies (collections of
related agents). Using this manager, you can add/delete agents from an
agency, edit the properties of the agency itself, import communication
protocols and analyze/debug the agency at runtime. You will find the
following new tools:

  - Agency Manager
  - Agency Viewer (a runtime tool)
  - Role Editor

- The protocol tools are used for building communication protocols. A
communication protocol is a specification for agent interation in a
specific area. The protocols are decoupled from specific agencies and
only weakly tied with ontologies. For example, a protocol could be
used to specify communication in the Buyer Seller Agency (as described
in the Application Note). The protocols are specified graphically as
a state diagram. States represent a particular point in the
communication, whereas links represent communication between agents.
You will find the following new tools:

  - Protocol Manager
  - Protocol Editor

```
=======================================================================
                      AgentBuilder Lite 1.1
=======================================================================
```

IMPORTANT NOTE: If you are upgrading from a previous version of
AgentBuilder Lite read the known problems for Lite 1.1 for help on the
upgrade.

The update to AgentBuilder Lite 1.1 includes a significant number of
changes. The functionality provided by the development tools is
greatly enhanced, and we are providing a number of improvements to the
runtime system. You may notice that some of your existing agents no
longer run properly and/or in exactly the same manner. That is
because of some fundamental changes undertaken in the pattern matching
policy. Read through the policy changes below and you will be able to
determine the nature of the problems and how to fix them.

- The UNIX versions of 1.1 is running with the 1.1.7 JDK, the windows
version is running with 1.1.7.

- All of the managers have been combined into one window with tab
panes. This reduces the number of windows used by the tool by
5. To go to a manager, just click on the tab. If you click on the
agency or agent first, in the project window, and then their manager,
the manager loads the agency or agent automatically.

- Direct method invocation in the right-hand-side of the Rule Editor
has been added. This is extremely useful to you because you no
longer have to create Actions in order to invoke methods on their
PACs. Now the developer can see and click upon methods in the dialogs
in right-hand-side of Rule Editor. It also saves the extra work of
using the ConnectAction built-in action. Actions can still be useful,
but it is much easier to use direct method invocation.   Support for the
older agents is still included.

- When using direct method invocation the "run" method is always
executed on its own thread.

- Pattern matching policy

  - Mixing mental changes and actions is now permitted.
  The order the operations are performed in the order they are
  specified in the RHS (this differs from the old policy of ALL
  actions first and then all mental changes).

  - If you don't use a SET_VALUE_OF operation to modify an object , only
  a method with the "set" prefix will mark the object as modified for

the next pattern matching cycle.

  - Pattern matching now uses inheritance, matching against an object's parent class causes a match/

  - Copies (or clones) are made of all objects which are asserted. Copies are also made of all arguments to the NewObject operation, all other parameter passing is done by reference.

  - Retractions only mark objects for deletion at the end of the engine cycle. So it is possible for one rule to retract an object and another to modify it in the same cycle.

- Changes have been made in how ontologies are selected by an agent. Now the developer need not specify which ontologies to include in the agent. Instead this is automatically determined based on what the agent needs. As PACs are imported from ontologies, only the ontologies used are added to the list.

- Changes have been made to the Object Modeler, PAC Editor and the process of creating object models and turning them into PACs. The tool supports "open" object models (object models which include references to classes NOT contained within the object model). As long as the class is contained in ONE of the object models in the repository the class can be imported as a PAC.

  - The option of inheriting methods from parent classes is now allowed when importing a class into an object model.

  - All attributes, return types and parameters in a class (in an object model) need to be fully qualified.

  - Classes can be defined in only ONE object model.

  - The user's PACs can have Vectors, Hashtables, Object and Enumeration classes. (These have all been included in the Java types list.)

- The Pac Editor now supports short name mapping. See the reference guide for more details.

- Support for 1,2 and 3 dimensional arrays has been added. There is now an "Array" class in the Java lists that can be used to contain arrays of any object. The Array class is a "phantom" class which is used in making an array look more like an object. Support for more than 1 dimension will be added soon.

- Some type checking has been added to the Rule Editor to help the user develop only legal patterns.

- Some of the default PACs were incorrect and have been respecified. The default agent was rebuilt with the new set of PACs.

- The default ontology is always copied to the developer's repository if there isn't already one there.

- Window position and sizes are now stored and reused automatically.

- Various bugs from 1.0.1b were fixed.

```
=======================================================================
                              LAST WORDS
=======================================================================
```

AgentBuilder is a product of Acronymics,Inc.

2400 Brookhaven Drive
Alma, AR
email: informationAgent@AgentBuilder.com with Subject: Support

OR

support@agentbuilder.com
http://www.agentBuilder.com;
http://www.acronymics.com

support@agentbuilder.com
Last Modified November 14, 2011