I-X Syntax

Jeff Dalton, Austin Tate and Stephen Potter
Artificial Intelligence Applications Institute
Centre for Intelligent Systems and their Applications
School of Informatics, The University of Edinburgh
Appleton Tower, Crichton Street, Edinburgh EH8 9LE, UK

Web: http://i-x.info
E-mail: query@i-x.info

Version 4.0 – 31 August 2004

See the separate documentation in the ix-syntax folder for the latest version of the I-X and <I-N-C-A> messages and XML formats used within the I-X system.

The following information gives an outline of these messages but should not be considered definitive.

An I-X Process Panel can send or be sent a number of XML format messages when communicating with other agents or systems to give it issues to address, activities to perform and reports to note. A Test agent (called I-Test) is provided to give a simple way to try this out. The format of these messages is as follows:

```
issue ::=
   <issue
         status="status"
         priority="priority"
         sender-id="name"
         ref="name"
         report-back="yes-no">
      <pattern><list>pattern-element...</list></pattern>
   </issue>

activity ::=
   <activity
         status="status"
         priority="priority"
         sender-id="name"
         ref="name"
         report-back="yes-no">
      <pattern>pattern-element...</pattern>
   </activity>

constraint ::=
   <constraint
         type="name"
         relation="name">
      <parameters>
         <list>
           <pattern-assignment>
             <pattern>pattern element...</pattern>
             <value>pattern element...</value>
           </pattern-assignment>
         </list>
      </parameters>
   </constraint>

constraint types allowed at present are “world-state” and the relations
for this are “condition” and “effect”.  The value must be given, but a
default “value” can be set to “true”.

report ::=
   <report
         report-type="report-type"
         priority="priority"
         sender-id="name"
         ref="name">
```

```
      <text>string</text>
   </report>

chat-message ::=
   <chat-message
         sender-id="name">
      <text>string</text>
   </chat-message>

pattern-element ::=
      <symbol>name</symbol> |
      <string>text</string> |
      <item-var>?name</item-var> |
      <integer>digits</integer> |
      <long>digits</double> |
      <double>...</integer> |
      <list>pattern-element...</list> |
      other pattern-elements are possible

report-type ::= success | failure | progress | information | event

priority ::= lowest | low | normal | high | highest

yes-no ::= yes | no

status ::= blank | complete | executing | possible | impossible | n/a
```

Notes

Strings and symbols that contain some special symbols need to have these encoded. Use "&" for ampersands, "<" for less-than, and "&gt" for greater-than.

In attribute values, double quote should be encoded as """.

It is possible to have a variable match all of the remaining elements in a list by using the special symbol "&rest" followed by an ordinary variable. I.e., `<symbol>&amp;rest</symbol><item-var>?name</item-var>`

```
domain ::=
   <domain>
      <name>string</name>
      <refinements><list>refinement...</list></refinements>
   </domain>

refinement ::=
   <refinement>
      <name>string</name>
      <variables><list>variable...</list></variables>
      <pattern><list>...</list></pattern>
      <issues><list>issue...</list></issues>
      <nodes><list>node-spec...</list></nodes>
      <constraints><list>constraint...</list></constraints>
      <orderings><list>ordering...</list></orderings>
      <comments>string</comments>
   </refinement>

variable ::=
   <item-var>?name</item-var>
```

```
issue ::=
   <issue
         status="status"
         priority="priority"
         sender-id="name"
         ref="name"
         report-back="yes-no">
      <pattern><list>pattern-element...</list></pattern>
   </issue>

node-spec ::=
   <node-spec id="name">
      <pattern><list>pattern-element...</list></pattern>
   </node-spec>

constraint ::=
   <constraint
         type="name">
      <parameters><list>pattern-element...</list></parameters>
   </constraint>

ordering ::=
   <ordering>
      <from>node-end-ref</from>
      <to>node-end-ref</to>
   </ordering>

node-end-ref ::=
   <node-end-ref
         end="end"
         node="name">
   </node-end-ref>

end ::= begin | end
```

Variables begin with “?” and can be used anywhere. Unbound variables appear in the process panel and can be bound by the user of the panel (and in later versions by external query capabilities).