I-X Process Panels – User Guide

Austin Tate, Jeff Dalton, Jussi Stader, Stephen Potter and Jessica Chen-Burger
Artificial Intelligence Applications Institute
Centre for Intelligent Systems and their Applications
School of Informatics
The University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, UK

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

Version 2.3 – 3rd October 2002

1 Introduction to I-X and I-X Process Panels (I-P^2)

1.1 I-X Research Programme

I-X is a research programme with a number of different aspects intended to create a well-founded approach to allow humans and computer systems to cooperate in the creation or modification of some product such as a plan, design or physical entity – i.e. it supports **synthesis tasks**. I-X may also be used to support more general collaborative activity.

The I-X research draws on earlier work on O-Plan (Tate et.al. 1998; 2000; 2002), <I-N-OVA> (Tate, 1996), the Enterprise Project (Fraser and Tate, 1995; Stader, 1996; Uschold, et.al., 1998) and the TBPM project (Stader, 2000) but seeks to make the framework generic and to clarify terminology, simplify the approach taken, and increase re-usability and applicability of the core ideas.

The I-X research programme includes the following threads or work areas:

1. **I-Core**, which is the core architecture, and an underlying ontology for activity and processes termed <I-N-C-A> (Issues, Nodes, Constraints and Annotations), and the terminology used to describe systems or applications built in the I-X framework.

2. **I-P^2**, which are I-X Process Panels used to support user tasks and cooperation.

3. **I-DE**, which is the I-X Domain Editor, which is itself an I-X application but is also used to create and maintain the domain description, process models and activity specifications used elsewhere.

4. **I-Plan**, which is the I-X Planning System. This is also used within I-P^2 and other applications as it provides generic facilities for supporting planning, process refinement, dynamic response to changing needs, etc.

5. **I-Views**, which are viewers for processes and products, and which are employed in other applications of I-X. I-Views can be for a wide range of modalities and types of user.

6. **I-Faces**, which are underlying support utilities to allow for the creation of user interfaces (User I-Faces), inter-agent communications (Communications I-Faces) and repository access (Repository I-Faces).

7. **I-X Applications** of the above work areas in a variety of areas. These currently include:
    a. Coalition Operations (CoAX)
    b. Emergency and Unusual Procedure Assistance (I-Aid, I-Help, I-Rescue)
    c. Support Desks (I-Support)
    d. Multi-Perspective Knowledge Modelling and Management (I-AKT)
    e. Medical Best Practice Procedures or Protocols (I-Medic)
    f. Natural Language Presentations of Procedures and Plans (I-Tell)
    g. Collaborative meeting and task support (I-Space: I-Room and I-World).
    h. Intelligent Messaging (I-Me).

8. **I-X Student Projects**, which are deepening and refining a number of aspects of the I-X research programme.

9. **I-X Technology Transfer**, including work on standards committees, especially for process, plan, activity and capability models.

1.2 I-X Process Panels (I-P^2)

The aim of an I-X Process Panel (I-P^2) is to act as a workflow, reporting and messaging "catch all" for its user. It can act in conjunction with other panels for other users if desired.

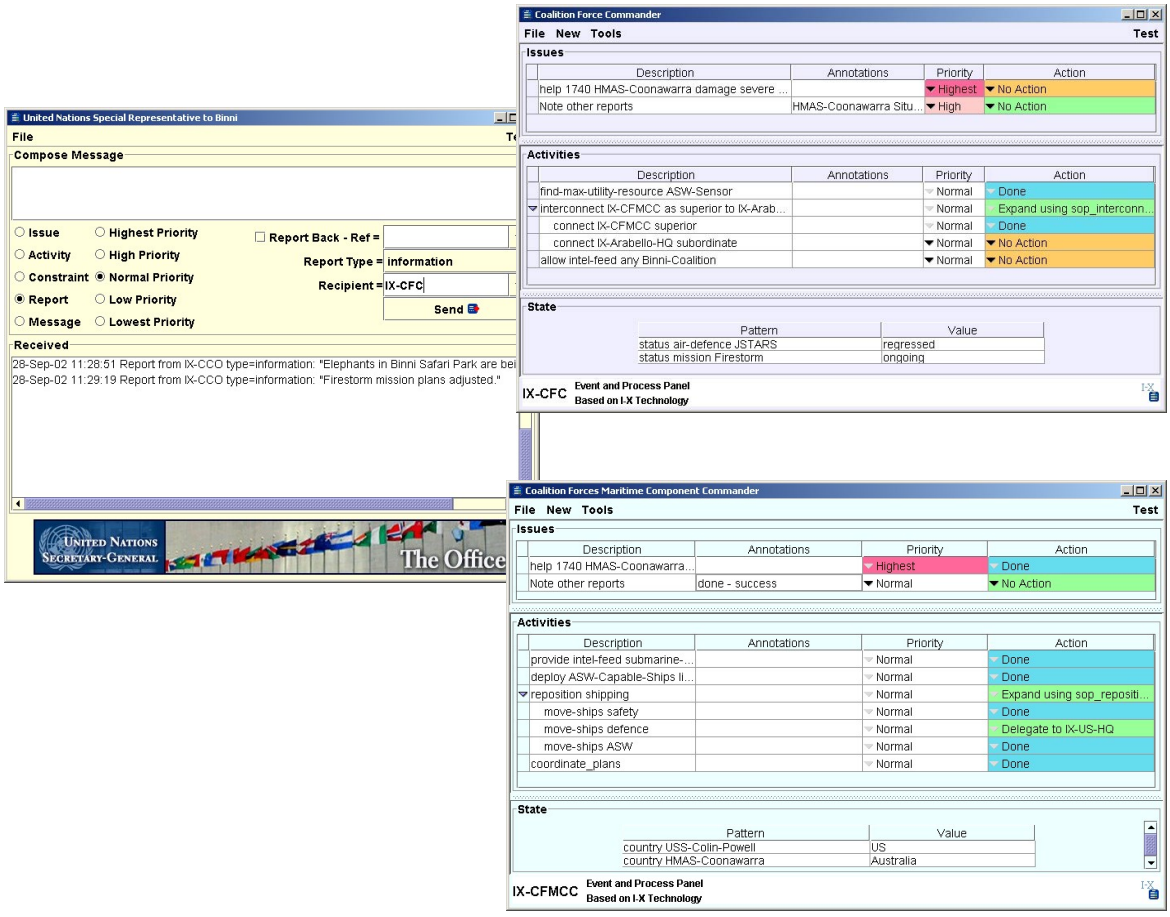
- Can take ANY requirement to:
  - Handle an issue
  - Perform an activity
  - [later: Maintain a constraint]
  - [later: Note an annotation]

- Deals with these via:
  - Manual (user) activity
  - Internal capabilities
  - External capabilities (invoke or query)
  - Reroute or delegate to other panels or agents (escalate, pass or delegate)
  - Plan and execute a composite of these capabilities (expand)

- Receives reports and messages and, where possible, interprets them to:
  - Understand current status of issues, activities, constraints and annotations
  - Understand current world state, especially status of process products
  - Help control the situation

- Copes with partial knowledge

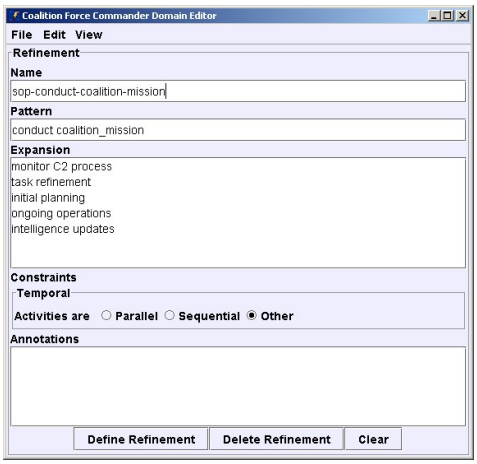
Three example process panels are shown in the figure below. These panels are from a demonstration of agent systems within a military Coalition context – part of the Coalition Agents eXperiment – CoAX (Allsopp et.al. 2001, 2002).



An I-X Process Panel supports a user or collaborative users in selecting and carrying out "processes" and creating or modifying "process products". Both processes and process products are abstractly considered to be made up on **"Nodes"** (activities in a process, or parts of a process product) which may have parts called sub-nodes making up a hierarchical description of the process or product. The nodes are related by a set of detailed **"Constraints"** of various kinds. A set of **"Issues"** is associated with the processes or process products to represent unsatisfied requirements, problems raised as a result of analysis or critiquing, etc.

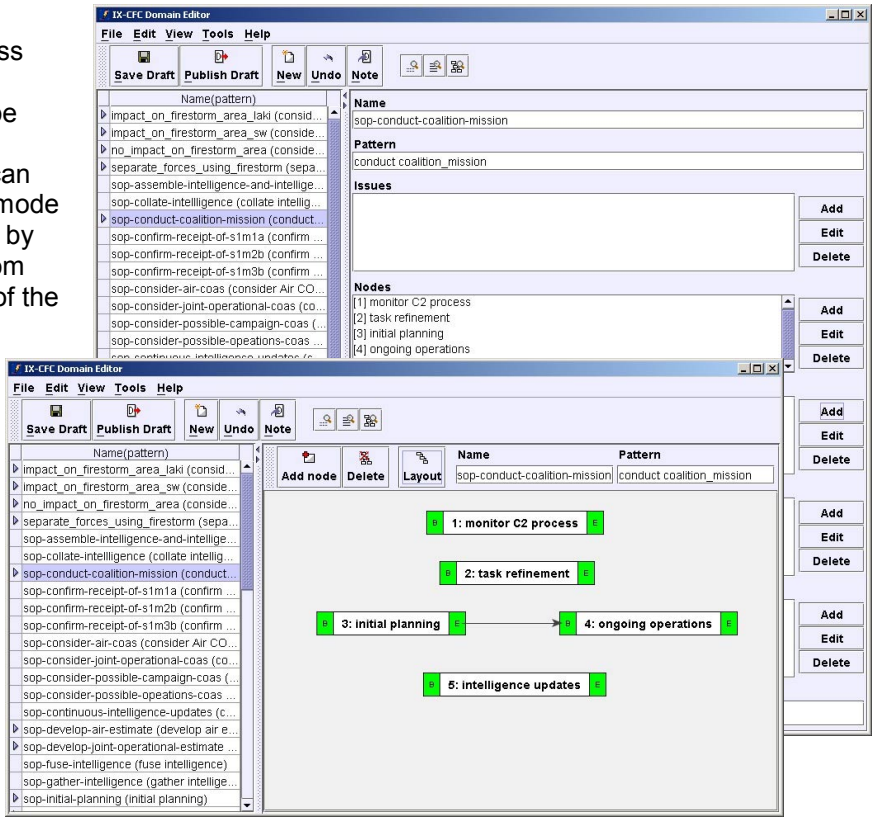
Processes and process products in I-X are represented in the <I-N-C-A> (Issues – Nodes – Constraints – Annotations) model of synthesised artifacts (Tate, 2000).

1.3 I-X Domain Editor (I-DE)



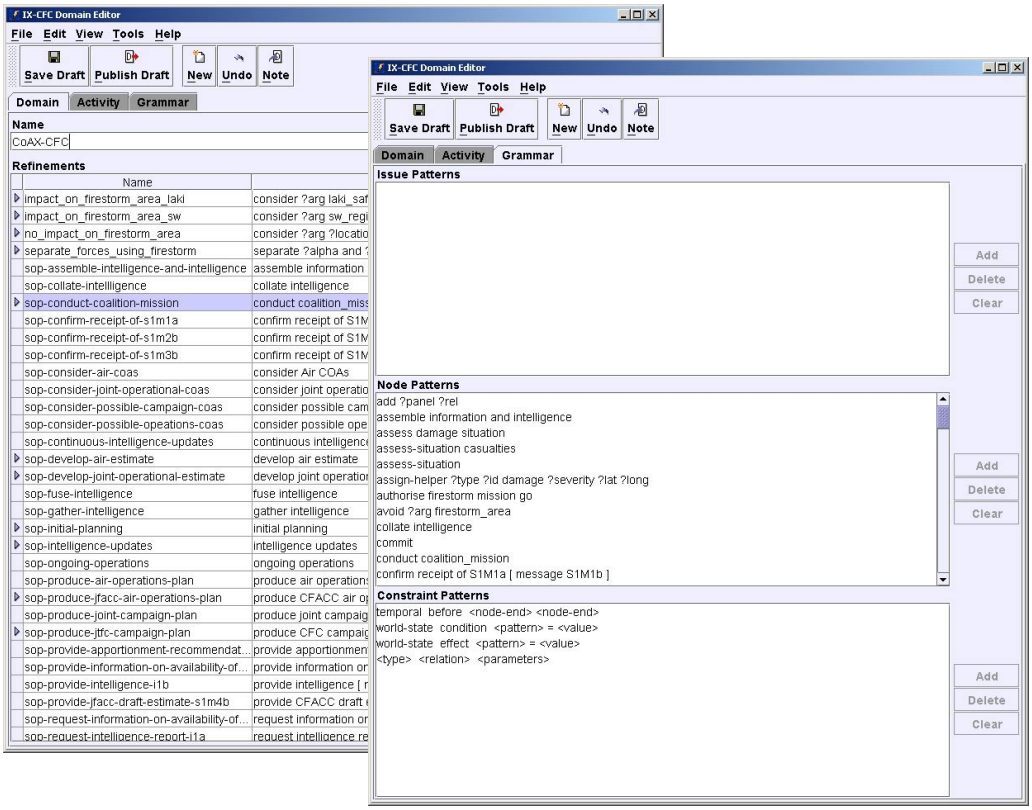
The process descriptions used by I-X Process Panels are kept in a domain library. This can be loaded when a panel is started, and can be added to dynamically by a user of a panel.

Simple Mode - the process panels contain a simple, form-based domain and process editor (right). This allows simple task breakdown structures to be specified along with a temporal constraint that the sub-steps should all be sequentially ordered or all kept in parallel.

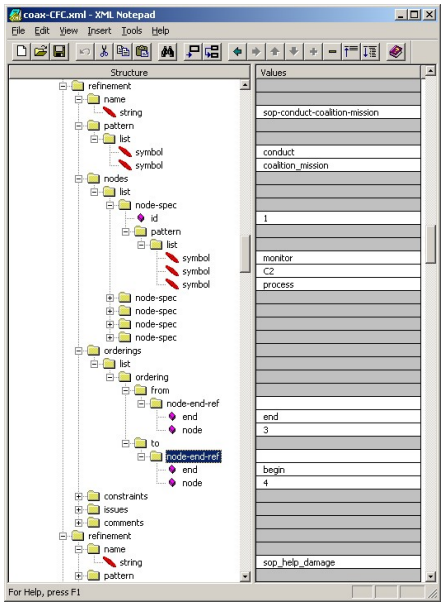


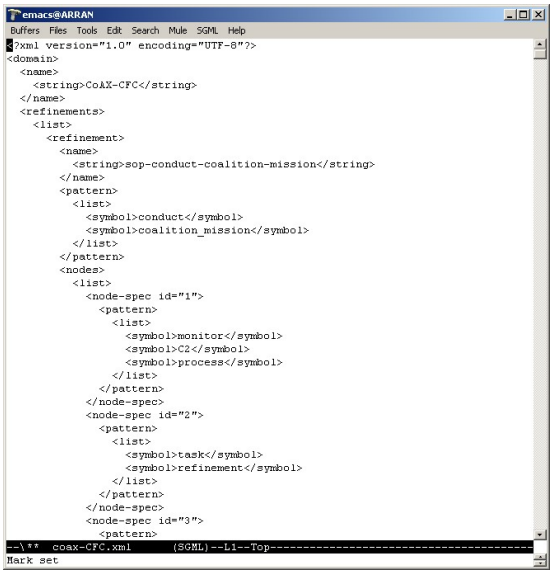
Advanced Mode - a more powerful domain and process editor allows for multiple perspectives and views to be used to create rich process models beyond those that can be created with the simple mode editor. This can be reached by selecting advanced view from the Views pull down menu of the domain editor. I-DE is also available as a stand-alone application to maintain a set of domain and process libraries. The advanced editor provides a “minimal” view which is deliberately similar to the Simple Mode Editor (and could one day replace it). It also provides a “comprehensive” view that allows the user to specify more complex temporal and world-state (condition/effect) constraints. Other constraints, like spatial ones or constraints on resources, can also be specified using the advanced view. A graphical view provides an alternative view to the form-based views. The graphical view illustrates precedence relationships between the sub-steps

of a process. This view can also be used to specify task breakdown structures via the expansion of nodes in the graph. In the advanced view, a tabbed option is available to allow access to related information about a domain model. The minimal, comprehensive and graphical views are all available via the Activities tab. Also available are “Domain” and “Grammar” tabs to view further details.



Use of XML and Text Editors - the process and domain models are maintained in XML. You can also modify them using an XML Editing Tool - such as the freely available Microsoft XML Notepad (see http://msdn.microsoft.com/xml/notepad/intro.asp) or a text editor.





2 Quick Start Guide

To quickly get started using the I-X Process Panels and run the demonstrations follow the following procedure. This is specifically for Microsoft Windows platforms – but a similar procedure can be used on Unix/Linux also.

1. Obtain and uncompress the I-X system distribution. This will create a single directory with all necessary files. It can be placed anywhere.

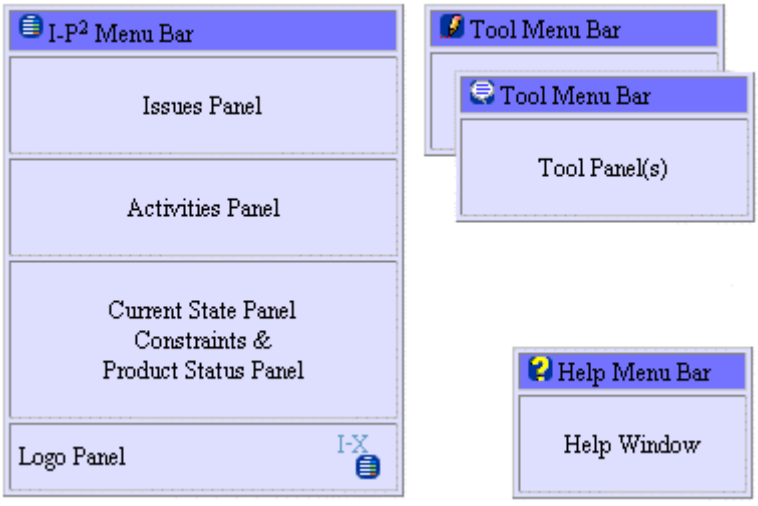
2. Ensure that you have a working Java Development Kit environment with the necessary Java programs on your current path, i.e. you should be able to run a command "java" from any location. You will need to alter the script provided in scripts\win\java-command.bat to set the path explicitly if this is not the case.

3. An example for using the I-X Process Panels is available in apps\isample. You can start up a demonstration by executing (e.g. by double clicking on) 3 of the batch script files in this sub-directory called scripts\win. Run xml-itest-nameserver.bat (done first to provide a simple name/address lookup service to the other process panels), and then xml-supervisor.bat and xml-operator.bat. A process panel customised to have a name based on your system user name and machine domain name is available by running xml-ime.bat. Similar scripts are available for Unix systems in scripts\unix.

4. You can then test the panels by sending sample issues between panels – one way to do that is to use the Test menu on I-Test to send sample issues, activities, constraints, reports or chat-style messages to other panels.

5. To use communications strategies other than simple inbuilt ones (simple and xml - which are available immediately), it is necessary to edit some scripts to set the location of the relevant code on your computer. Edit the script files in comms*\scripts*.

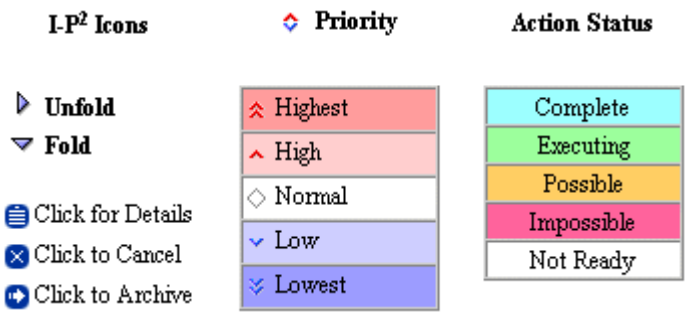
3 Using an I-X Process Panel



An I-X Process Panel (I-P^2) contains a number of sub-panels that describe:

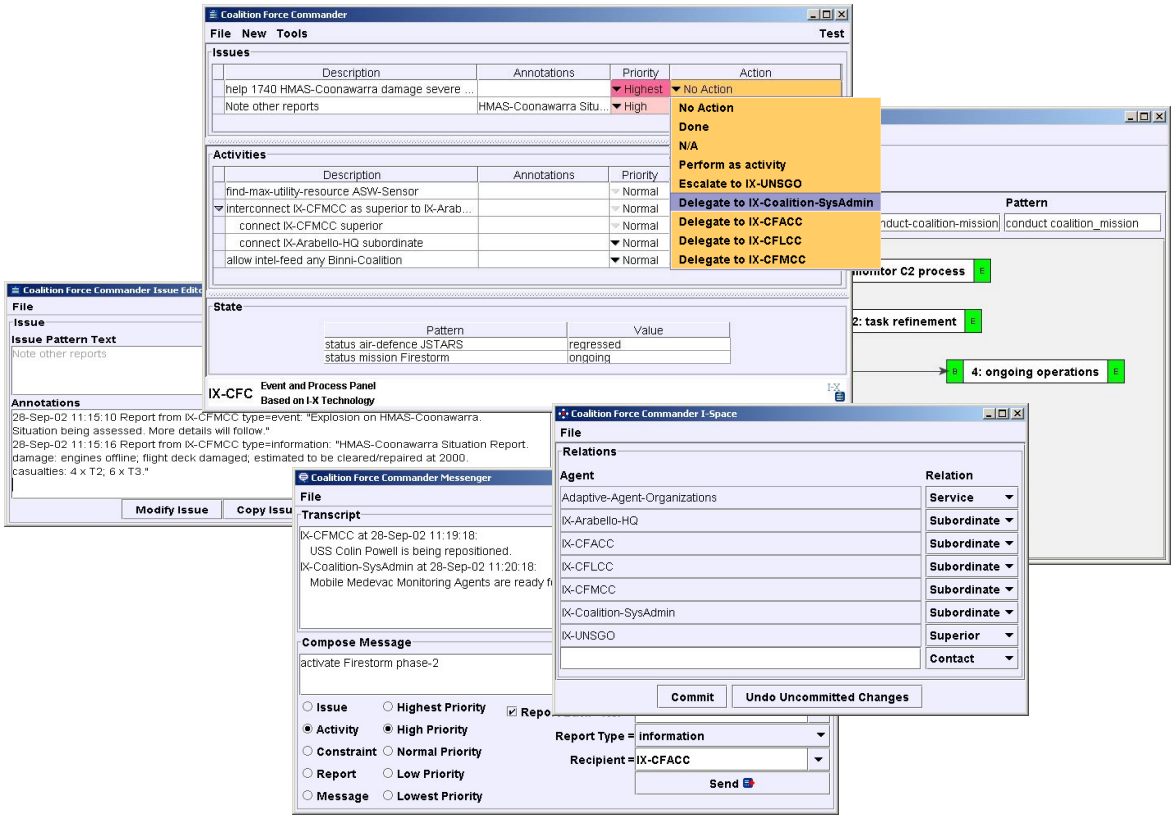
- A set of "issues" to be "handled".
- A set of "activities" to be "performed".
- Current state information reflecting the current set of "constraints" to be "respected". This includes the status of a range of "process products" being created or manipulated by the processes

The panel supports it's user in handling issues, deciding on a course of action and performing activities, and maintaining awareness of the current state, constraints, process products, etc.



Entries on panels can be expanded using information provided in the process library used by a panel, or the entries can be passed between panels.

Right click on a line to get a context sensitive menu that describes operations you can perform on the entry. This includes where relevant the ability to pop-up a window with more details of the entry (say an activity or an issue), or to expand or contract the display of some levels of hierarchically specified activities, to send information about the entry to the Messenger tool for sending on to others (perhaps in a modified form), etc.



A "tools" menu is available to make accessible the following:

- A domain or process library editor to view, edit or add to the list of process descriptions which may be used to "expand" entries on the process panel.
- A tool to view and change the relationships of the current panel to others ("I-Space").
- An instant messaging or "chat" tool to communicate in free format or the encouraged <I-N-C-A> structured forms with other I-X Process Panels and other systems ("intelligent messaging" or "semantically augmented messaging").

4 Using the I-DE Domain Editor Tool

The main window of the Domain Editor (the frame) contains several editor panels for editing different aspects (or constructs) of the domain. Currently the editors available are

- the Global Domain Editor, which edits information about the domain itself (e.g. the domain name)
- the Activity Editor, which edits information about activities and how they break down into sub-activities (refinement)
- the Grammar Editor, which currently only shows the patterns that are in use in the domain

An editor panel may itself have different "views" that are used to display and edit the panel's constructs. The Activity Editor has three such views:

1. Minimal View: a simplified version of the activity and its refinement. The main simplification is that no constraints are shown
2. Comprehensive View: a view that can display and edit all of an activity's specification
3. Graphical View: a graphical view that uses nodes and arcs to show an activity's sub-activities and the temporal relationships between them.

4.1 Domain Editor Window

This window provides access to the most functions of the overall domain editor via its menu bar and access to the most commonly used functions via its tool bar. The window can display in one of three styles: simple, tabbed, and card style. The style can be changed via the Options in the File menu.

The Menu Bar

The menu bar has 5 standard menus:

1. File for closing the Domain Editor and for file access (open/save). All functions here manipulate the domain as a whole, not individual constructs;
2. Edit for manipulating the current construct, i.e. the construct that is currently shown in the Domain Editor's panel;
3. View for changing which panel is shown in the Domain Editor and - if applicable - for changing which view is shown in that panel;
4. Tools for additional support like consistency checks etc.;
5. Help for access to this manual, other help, and information about the application.

The Tool Bar

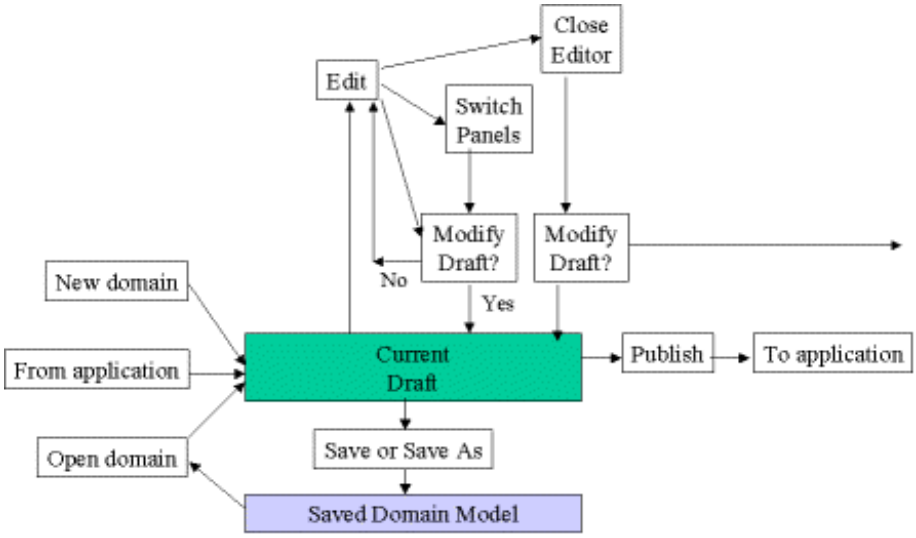
The tool bar provides access to the most commonly used functions via buttons. All these functions are also available via the menu bar (in most cases, the image on the toolbar button is shown in the menu next to the corresponding menu item. The toolbar can be switched on and off via Options in the File menu. Moving the mouse over a toolbar button will, after a

while, display a "tool tip text" that gives a brief explanation of the button's function.

4.2 Working with the Domain Editor

The Domain Editor maintains different levels of updates. The original domain model that the editor is started with is considered a public domain model, which other applications may be using for their own purposes (e.g. within a process panel). This public domain model is kept as it is unless it is explicitly “published” by the Domain Editor's user. (Note that this is true whether the Domain Editor is used in stand-alone mode or as part of another application). There is also a "draft domain model" which is the one that is being edited. The Domain Editor keeps track of any changes that are made to the draft domain model so that updates to the original domain model can be made explicitly.

Saving and Reverting



There are 3 levels of saving:

1. When a construct has been edited in the Domain Editor Panel, initially these changes may be made only in the panel itself, not in the domain construct that is being edited. In order to transfer changes from the panel into the construct in the draft domain, the user has to modify the draft, i.e. note the changes into the draft domain via the toolbar button or the Edit menu. When the user has edited a construct and not modified the draft, the system will prompt the user to note or discard changes if the user decides to switch constructs, views, or panels, or if the user decides to save or publish the draft domain.

2. Modifying the draft (noting changes) does not save to file, so the next level of saving is to save the draft domain to file. As with all editing applications, it is recommended to do this frequently to ensure that work is not lost. Saving the draft domain to file will write the whole domain with all its constructs into a file in XML format. This can later be loaded into the Domain Editor for further editing, or it can be accessed by other applications.

3. The underlying public domain is not changed by any of the above (simple editing, modifying draft, or saving the draft domain to file). The only way to update the public domain is to publish the draft domain via the toolbar button or the File menu. When this happens, the changes are made to the original domain and these chages will be seen by any applications that have registered as listeners to this domain. Note that publishing is always done for a whole domain, not for individual constructs. Note also that publishing a domain will not save it to file, but the same effect can be achieved by saving the draft domain to file just before or straight after publishing. At that point the draft domain and

the public domain can be represented by the same XML structures. It is a good idea to publish from time to time even if the Domain Editor is running stand-alone because it will make the editor more efficient.

It is worth noting that the Global Domain Editor Panel, i.e. the panel that is responsible for editing details about the domain like its name, only considers domain details as part of its editing remit, not the constructs within the domain.

While there is no "undo" function that undoes individual editing steps or editing of individual fields, the following functions are available, corresponding to the 3 levels of saving:

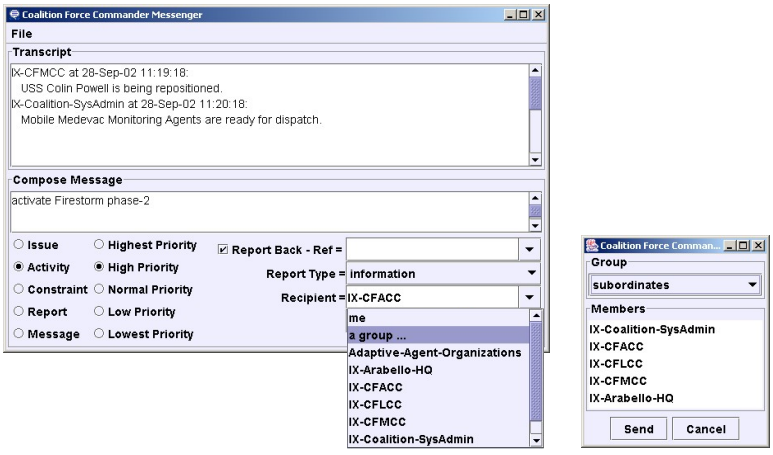
1. Undo: revert a construct to the last time it was saved to the draft domain (via Edit menu), i.e. undo all changes that have only been made in the editing panel;

2. Revert to published: revert a construct to the public version (via Edit menu), i.e. undo all changes to this construct since the domain was last published;

3. Re-load: revert the whole domain to the last time it was saved to file by opening that file via the File menu.

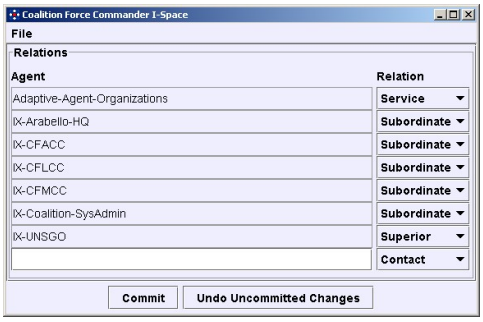
There is a fourth "revert" function for convenience: "discard changes to draft" which reverts the whole domain to the public version, i.e. undo all changes to all constructs since the domain was last published.

5 Using the I-X Messenger Tool



The I-X Messenger tool is used to compose and send messages to other panels and agents. It also shows any “chat” messages received form other agents (in the Transcript window). You can send messages to your own panel (“me”) and there is a simple group sending facility (which will be expanded in future releases).

6 Using the I-Space Tool



The I-Space tool allows for the management of the organisational relationships of the current panel (referred to as “me”) to other panels, agents and external services. New agent names can be added ion the type in box at the bottom. Existing agents or panels can have their relationship altered. The Commit button is used to inform the process panel of any addition or changes to relationships of existing entries. You can undo any changes made to the I-Space table that have not been committed already.

The relationships allow for the setting up appropriate “Action” menu entries for items on the panel. The relationships provided are:

Relationship	Action Menu Item
Superior	Escalate to (with report back)
Peer	Pass to (with report back)
Subordinate	Delegate to (with report back)
Service	Invoke (with report back)
Contact	None
None	None

Providing an external-capabilities description of the verbs associated with any agent can be used to selectively show the agent in the Action Item menu only for the given verbs. If no verb association is provided, it is assumed that all Superiors, peers and Subordinates can take any item (with any verb). It is expected that an external-capabilities description is given for a service or it will not appear on the menu at all.

7 Creating your own I-X Process Panel

A single process panel or a small cluster of panels in superior, peer or subordinate relationships to one another can be quickly adapted to a new application. We will later support more dynamic and adaptable combinations of multiple panels in more complex organisational structures (which we called I-Spaces), but much of the necessary support for this is not yet sufficiently generic to provide in an easily altered form.

An example I-P^2 application is provided in the apps\isample directory, which can be copied and adapted as follows:

- Copy the whole Isample directory to become a new directory with a name of your choice (e.g. apps\appname).
- In the directory config alter the property file names and contents of those files as you wish to tailor display names and labels used on the process panels.
- You can modify the ways in which the panel “actions” are set up using “I-Space” relationships such as superior, peer and subordinate.
- In the directory images add in any logo or logos for panels as you wish. Replacing the logo images\isample-logo.gif will mean the default logo is amended without further changes.

- Tailor a panel to a new application usually involves providing a suitable “domain model” that describes ways in which activities can be refined into more detailed sub-activities. Domain models in I-P^2 are stored in XML format (although a Lisp-oriented format is also available) and for I-Sample Process Panels are in the domain-library directory. A domain editor (see below) is provided to create or amend domain models, and domain models can be augmented while a Process Panel is running.
- You can add appropriate “Test Menu” entries (see below) .

A wide range of parameters can be specified to simply customise a range of things about each panel. A property file for a panel can specify most of these and can be set using, e.g.,

ix.ip2.Ip2 -load config/isample-supervisor.props

For example, the file config/isample-supervisor.props contains such things as:

```
symbol-name=Supervisor
display-name=Supervisor I-X Process Panel
logo-line-1=Supervisor I-X Process Panel
logo-line-2=Based on I-X Technology
logo-image=images/isample-logo.gif
domain=isample-supervisor.xml
subordinates=Operator
```

You can also set one property individually when the process panel program is started using a command-line argument, such as

```
"-display-name=App-Name Whatever"
```

The domain model including process descriptions available to the panel can be preloaded from a domain library file (e.g. as in the case above which loads the isample-supervisor.xml file describing sample processes that the panel is made aware of and can use to “expand” entries put onto the panel.

It can be convenient to provide some example issues, activities or other entries that can be added to a panel, which could have come from other systems or panels. It can also be convenient to provide messages that could be sent to other panels and agents. This allows simple demonstrations and testing to occur. The contents of the "Test" menu can be set using an XML file describing the entries, and informing the panel about this file using the
-test-menu=<pathname>
parameter – see the appendix for details of all parameters. An example test menu file follows. “me” for the “to-name” means the message is sent to the current panel rather than externally. The menu text is what actually appears as an entry in the Test menu. The $to item in the menu-text string (if present) is substituted by the “to-name” of the panel or agent to which then message is sent.

```
<?xml version="1.0" encoding="UTF-8"?>
<list>

   <test-item
           menu-text="Send $to a request for transport"
         to-name="Supervisor">
      <contents>
         <activity priority="high"
                   report-back="yes">
            <pattern>
               <list>
                  <symbol>transport_by_helicopter</symbol>
                  <item-var>?wounded</item-var>
                  <symbol>field_hospital_a</symbol>
               </list>
```

```
            </pattern>
          </activity>
        </contents>
      </test-item>

      ...

</list>
```

More details of setting up a test menu are provided in the appendix.

You can further tailor an I-X Process Panel to a specific application by renaming and amending the I-Sample Process Panel code as follows:

- You can rename the java\isample directory to be java\appname and, in that directory, rename Isample.java to be AppName.java or whatever you wish. Delete the compiled class files included there.
- Edit this renamed file to change the package name from isample to appname.
- Change the class name Isample to AppName whereever it occurs.
- Change any strings that refer to I-Sample to App-Name as you wish.
- You can provide a customised “state” viewer for panels. A description of how to do this is in the I-X Developer Guide.
- Recompile the AppName.java code with the compile script provided.
- In the directory scripts\win (and unix) alter the script names and script contents as necessary to refer to the new name rather than the basic ix.ip2.Ip2 class or the custom isample.Isample class.

The I-X Process Panels can have a number of “issue handlers” which can handle issues in specific ways:

- **Escalate, Pass and Delegate:** One type of handling is to reroute the issue to other users or panels. At present the issue handlers are defined using information supplied in the “I-Space” description for a process panel which is currently done by specifying the superiors, peers, subordinates and contacts parameters (whether via the command line or via the property file provided on panel start up).

- **Invoke:** External agents defined as being a “Service” in the I-Space tool appear on the action menu as capabilities to address items that they are registered as being able to handle (though defining their “external-capabilities” in panel properties.

- **Connect:** Actions of forma (connect panel-a relationship) have a connect handler that sets the appropriate entries in I-Space automatically.

8 Creating your own I-X Domain/Process Library

Each I-X Process Panel can make use of a domain model or process library which describes ways in which issues can be handled or high level activities can be broken down into more detailed activities which may be performed. A panel can operate without such process descriptions, but becomes more useful and helpful if it has such knowledge. A process library can be loaded when a panel is started up, and additional process descriptions can be provided while it is running, and indeed they can be saved at any stage to amend the stored version for later preloading.

You can create the process descriptions with the I-X Domain and Process Editor provided. It can be run on its own or can be called from within a Process Panel from the Tools menu. Since the process descriptions are actually stored in a simple XML format, it is also possible to use any XML editor to change the descriptions if you wish. The format of the XML is as follows:

```
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

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

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

```
yes-no ::= yes | no

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

Example domain models and refinements can be found in the apps\isample\domain-library directory. 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). The representation used is based on the <I-N-OVA> constraint representation of activity (Tate, 1996).

9 Further Tailoring

An I-X Developer Guide is available with details on more ways to tailor I-X Process Panels and systems. This usually involves Java programming add-ons.

9.1 Communications Strategy

I-X Process Panels can also be used with any of a number of "Communications Strategies". Example strategies are provided for the DARPA CoABS Grid ("grid"), the University of West Florida Institute of Human and Machine Cognition (UWF/IHMC) KAoS ("kaos"), the Jabber (www.jabber.org) XML framework ("jabber"), and the UK EPSRC-sponsored Advanced Knowledge Technologies AKT Bus ("akt"). Also provided is an adaptor for a simple direct link between panels possibly supported by a simple name server (referred to as the "simple" or "xml" communications strategy). Writing a suitable Communications Strategy can provide other message transport routes. More details are available in the I-X Developer Guide.

9.2 Custom World State Viewer

It is possible to replace the simple table view used for the current world state. Viewers that show the state information clustered into the various objects or process products being handled, and giving their attributes and values in a convenient form can be provided. Graphical images of the process products can be added where required. This could include map-based data to show the position of the objects.

10 References

Allsopp, D., Beautement, P., Bradshaw, J.M., Carson, J., Kirton, M., Suri, N. and Tate, A. (2001) "Software Agents as Facilitators of Coherent Coalition Operations", 6th International Command and Control Research and Technology Symposium, US Naval Academy, Annapolis, Maryland, USA, 19-21 June 2001.

Allsopp, D., Beautement, P., Bradshaw, J.M., Durfee, E.H., Kirton, M., Knoblock, C.A., Suri, N., Tate, A. and Thompson, C.W. (2002) "Coalition Agents Experiment: Multi-Agent Co-operation in an International Coalition Setting", Special Issue on Knowledge Systems for Coalition Operations (KSCO), IEEE Intelligent Systems, June 2002.

Fraser, J. and Tate, A. (1995) "The Enterprise Tool Set -- An Open Enterprise Architecture", Proceedings of the Workshop on Intelligent Manufacturing Systems, International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, August 1995.

Stader J., Moore J., Chung P., McBriar I., Ravinranathan M., Macintosh A.. (2000) "Applying Intelligent Workflow Management in the Chemicals Industries"; in “The Workflow Handbook 2001”, L. Fisher (ed), Published in association with the Workflow Management Coalition (WfMC), pp 161-181, Oct 2000.

Stader J. (1996) “Results of the Enterprise Project”, in Proceedings of Expert Systems '96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, UK, December 1996.

Tate, A. (1996) "The <I-N-OVA> Constraint Model of Plans", Proceedings of the Third International Conference on Artificial Intelligence Planning Systems, (ed. Drabble, B.), pp. 221-228, Edinburgh, UK, May 1996, AAAI Press.

Tate, A. (1998) “Roots of SPAR”, in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13 (1), March 1998, Cambridge University Press.

Tate, A. (2000) “<I-N-OVA> and <I-N-CA> - Representing Plans and other Synthesized Artifacts as a Set of Constraints”, AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems, at the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A., Dalton, J. and Levine, J. (1998) "Generation of Multiple Qualitatively Different Plan Options", Fourth International Conference on AI Planning Systems (AIPS-98), Pittsburgh, PA, USA, June 1998.

Tate, A., Dalton, J. and Levine, J. (2000) “O-Plan: a Web-based AI Planning Agent”, AAAI-2000 Intelligent Systems Demonstrator, in Proceedings of the National Conference of the American Association of Artificial Intelligence (AAAI-2000), Austin, Texas, USA, August 2000.

Tate, A., Levine, J., Dalton, J. and Nixon, A. (2002) “Task Achieving Agents on the World Wide Web”, in “Creating the Semantic Web”, Fensel, D., Hendler, J., Liebermann, H. and Wahlster, W. (eds.), MIT Press, 2001.

Uschold, M., King, M., Moralee, S. and Zorgios, Y. (1998) "The Enterprise Ontology", in "Special Issue on Ontologies", Knowledge Engineering Review, Vol.13(1), March, 1998, Cambridge University Press.

Appendix A: I-P^2 Parameters

IPC

```
    -symbol-name=symbol

  Note that symbol-name, by default, also becomes the ipc-name. If
  not provided, the default symbol-name is set to
  "IX-<user-name>@<machine-name>".


    -ipc-name=name

  Note that ipc-name is only available for special uses.  It is
  recommended that symbol-name is used generally to name agents, and
  this will by default be used for the ipc-name.  Wherever agent
  names are required, the ipc-name is used.

    -ipc=strategyName
    -ipc=class
```

strategyName can be simple (default) or xml using built in support. With suitable communications strategy add-ons other strategies can be specified such as: grid, kaos, jabber or akt.

See the javadoc for `IPC.getCommunicationStrategy(String strategyName)`

Default/Simple and XML Communication Strategies

```
    -port=number
```

Tells the agent to use a specific port number rather than to ask the underlying operating system to allocate a free one. This is especially useful in environments with a firewall.

```
    -host=hostname
```

Used to tell the agent what to call the machine it is running on when the default name will be incorrect. The default is the name returned by
InetAddress.getLocalHost().getHostName()

```
    -run-name-server
```

Tells the agent to run a name-server.

```
    -name-server
    -name-server=servername:port
    -no name-server
```

Tells the agent whether to use a name-server to look up the addresses of other agents, and if so what host and port to connect to. The name-server servername:port defaults to localhost:5555

Jabber Communication Strategy

```
    -jabber-server=hostname (e.g. jabber.org or akt.aiai.ed.ac.uk)
    -jabber-username=username
    -jabber-password=password
    -jabber-resource=resource (I-X by default)
```

Usually leave the jabber-resource at its default value. The assumption is made that all other panels with which this panel will communicate will share the same resource name. Hence, the resource name is used for two purposes:

1. to distinguish I-X panel jabber clients from non-I-X panel jabber clients, and;
2. to distinguish I-X panel clients belonging to a particular I-X 'cluster' from other I-X panels.

Caution should be applied when setting this parameter to anything other than its default value.

```
-jabber-presence=keyword (e.g. Online)
-jabber-allow-queuing=boolean (default false)
```

Can set true to allow asynchronous communications between panels that are not on-line at the same time (queuing is provided by Jabber servers). The default mode (false) will indicate a communications failure if the target recipient resource is not on-line or, if no resource is specified, the target user has no on-line resources at all. (In the latter case, if no resource is specified and the target user has an I-X Process Panel on-line, then this Panel will be the preferred destination for the message.)

IXAgent

```
-debug=boolean
```

Set to true for more detailed diagnostics in the Java console window.

```
-classic=boolean
```

Use alternative (simpler) interface for table views and other user interface elements.

```
-domain-editor-class=classname
        Possible values are currently:
              ix.iview.DomainEditor
              ix.iview.SimpleDomainEditor
```

Select domain editor to use.

IP2 Domain Library

```
-domain-library=pathname (URL syntax)

-domain=filename
```

Initial Panel Contents (Initial Plan)

```
-plan=pathname (URL syntax)
```

```
An XML file can be provided to specify the initial plan or initial
panel contents on start up.
```

IP2 Visual Appearance

```
-display-name=text
```

```
Note that display-name is set to "<symbol-name> Process Panel" if
not explicitly set.
```

```
-logo-line-1=text
-logo-line-2=text
-logo-image=pathname (URL syntax)
-metal-theme-secondary-3=colour
```

```
    -frame-size=WIDTHxHEIGHT

  The initial width and height of the process panel in the form
  WIDTHxHEIGHT (e.g. 800x400)

    -font-increment=integer

The relative font size for text, buttons and labels in I-X process
Panels.  E,g, 2, 4 or –2.  Odd numbers (e.g. 1 or 3) may not have
bold fonts installed on all systems).
```

IXAgent I-Space

```
    -superiors=namelist
    -subordinates=namelist
    -peers=namelist
    -contacts=namelist

    -external-capabilities=name:verb,...

  Note that namelist is a list of names of other process panels or
  external agent resources. A namelist is comma separated, and the
  list cannot contain spaces.

  external-capabilities specifies the verbs associated with any
  panel name or external agent name.  If the panel or agent is not
  in a defined relationship, then it is considered to be a
  "resource" and will only be available for the specific verbs
  specified.  If it is already in a defined relationship (usually
  for other I-X Process Panels) then it is treated as a restriction
  specification, so the relevant action menu entries only come up
  for those specific verbs – rather than for any pattern verb.
```

Test Menu

```
    -test-menu=pathname (URL syntax)

  An XML file can be provided to set the Test menu entries that
  appear in the top right corner of a process panel, and which can
  be convenient for testing and demonstrations.
```

General Notes

1. Filenames and pathnames are relative to the current directory when an application is run. This is usually the root directory for an I-X application using default start up scripts (i.e., <I-X-base-directory>\apps\<app-name>\).

2. When providing command line arguments, the "-" is not part of the parameter name. It is just command line syntax.

3. -load is not a parameter. It is syntax that says to load name=value lines from a file. More than one -load may be specified.

4. -no and -not can negate the following parameter (which is written without the initial "-"). It is equivalent to giving the parameter the value "false" but can be used in cases where it would seem odd to explicitly say "=false".

Appendix B: <I-N-C-A> XML Message Formats

An I-X Process Panel can be sent a number of XML format messages from 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>`

Appendix C: Test Menu Setup

Test-menu Files

To use a test-menu file, have a command-line arg or .props file entry test-menu=filename, e.g., ip2 -test-menu=somedir/test-sequences.xml

The file contains a list; each element describes a single entry on the test menu. In outline, the file therefore looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  ...
</list>
```

Three types of entry are allowed:

```
TEST-ITEM ::=
  <test-item
       delay-before="INT">
    <menu-text>STRING</menu-text>
    <to-name>STRING</to-name>
    <contents>SENDABLE</contents>
  </test-item>

TEST-SEQUENCE ::=
  <test-sequence>
    <menu-text>STRING</menu-text>
    <test-items><list>TEST-ITEM...</list></test-items>
  </test-sequence>

TEST-SEQUENCE-GENERATOR ::=
  <test-sequence-generator
       initial-delay="INT"
       delay-between="INT">
    <menu-text>STRING</menu-text>
    <template>TEST-ITEM</template>
    <to-names><list>...</list></to-names>
    <content-strings><list>...</list></content-strings>
  </test-sequence-generator>
```

A SENDABLE is an issue, activity, constraint, report or chat-message.

Note that string-valued fields (such as menu-text) may be written as attributes instead of as elements if the string is sufficiently simple.

Test items, sequences, and sequence-generators do not have to come from files. They are ordinary Java objects that can also be constructed in Java. However, it is often more convenient to specify them in XML.

In each of the above syntaxes, the menu-text is a string that is displayed in the "Test" menu in the top right corner of an I-X Process panel. For a test-item only, any occurrence of "$to" in the menu-text will be replaced by the value of the same test-item's to-name. (That is not done for a sequence, because the messages in a sequence might be to different destinations.)

A to-name is the symbol-name of the agent the test-item's contents should be sent to.

The delay-before in a test-item is the number of milliseconds to wait before sending the contents. Delay-before defaults to 0.

Note that within the SENDABLE test item contents you can specify a sender-id if you wish it to look like an item came from that agent or panel. The syntax for SENDABLE items is included in an earlier section.

Here is an example.

```
<test-item
    menu-text="Give $to a report-back example issue"
    to-name="me">
  <contents>
    <issue priority="high"
          report-back="yes">
      <pattern>
        <list>
          <symbol>note</symbol>
          <string>sample note text</string>
        </list>
      </pattern>
    </issue>
  </contents>
</test-item>
```

That test-item would appear in the "Test" menu as "Give me a report-back example issue" and when selected would send the panel an issue with priority=high, report-back=yes, etc.

Here is a test-item that sends a report after a delay of 4 seconds:

```
<test-item menu-text="Send $to a single report with a delay"
        to-name="me"
        delay-before="4000">
  <contents>
    <report report-type="information"
          text="Here's some information" />
  </contents>
</test-item>
```

A test-sequence contains a list of test-items. The menu-text of those items is ignored (and needn't be specified). The test-sequences' own menu-text appears in the "Test" menu.

When the test-sequence is selected from the "Test" menu, it processes the list of test-items in order. For each item, it waits for the item's delay-before milliseconds and then sends the item's contents to the agent named by the item's to-name. This allows a sequence to send messages to a variety of different destinations. By using delay-before values of zero, it is possible to get several messages to be sent (almost) at once.

Each item in a test-sequence may have a different type of contents. That makes it possible to send an issue to one agent, a report to another, and so forth.

However, in some cases, all of the items in a sequence will have certain things in common; and then it may be possible to use a test-sequence-generator.

A test-sequence-generator contains a single test-item that is used as a template. The menu-text of that item is ignored (and needn't be specified), but all other fields may be significant.

A test-sequence may contain either a list of to-names or a list of content-strings, but not both.

If it contains a list of to-names, a sequence is constructed by making a copy of the template for each of the to-names, replacing the copy's to-name each time. The resulting sequence will send essentially the same message to a series of agents

If there is a list of content-strings instead, the sequence contains one copy for each of the content-strings, with the "main contents" of the copy replaced by the corresponding content-string, suitably interpreted. This sequence will send a series of similar messages of the same type (issue, report, or whatever) to a single agent: the agent specified by the to-name of the template.

The interpretation of a content string depends on the class of the template's contents. If the template contains an issue or activity, the content string is treated as a pattern and parsed in the usual way (with ?names being variables etc); if the template contains a report or chat-message, the content-string is placed in the object's text field. Constraints are not yet supported (although they may appear in ordinary test-sequences.)

The test-items in the generated sequence have delay-before values determined as follows: If the generator specifies an initial-delay, it becomes the delay-before of the first item in the generated sequence. If the generator specifies a delay-between, it becomes the delay-before of all subsequent items in the sequence. Otherwise, the template' delay-before is preserved.

Here is an example that when selected will send a series of chat-messages:

```
  <test-sequence-generator initial-delay="0" delay-between="2000"
                  menu-text=”Send me some example chat messages”>
    <content-strings>
      <list>
        <string>Sample chat text 1</string>
        <string>Sample chat text two</string>
        <string>More chat</string>
        <string>This time there will be
several lines of text
  and maybe some indentation
  just for variety
and a last line</string>
      </list>
    </content-strings>
    <template>
      <test-item>
        <to-name>
          <string>me</string>
        </to-name>
        <contents>
          <chat-message />
        </contents>
      </test-item>
    </template>
  </test-sequence-generator>
```

Menu separators can be specified in the test-menu file. The XML syntax is

```
   <test-separator />
```

Just include it in the list between the test items you want to separate.

Using the "Test" Menu

Entries in the "Test" menu do not have to send messages. They don't even need to involve any of the objects described here. However, this section will describe only the cases that can be specified by a test-menu file.

If a single message is to be sent, without a delay, it is sent as soon as its "Test"-menu entry is selected. That is so regardless of whether it came from a single test-item or from a 1-item sequence.

Otherwise, a new thread is created to supervise the message sending. While that thread is running, the corresponding menu entry is prefixed by "Stop: " and, if selected, stops the thread. When the thread terminates, the entry reverts to its original form.

However, all messages are actually sent by the GUI event thread just as in the normal operation of a panel). The other thread exists only to control the timing.

Messages sent to "me" are given directly to the panel rather than going via the communication strategy.