# Apex 3.0 Beta Release Notes

## Introduction

Apex 3.0 Beta is a major revision of Apex that provides enhanced AI reasoning and autonomy services, a revamped graphical user interface (Sherpa), an improved interface for defining applications, and numerous other enhancements.

## Enhancements

Sherpa

- A new web browser-like user-interaction model including: backward/forward history control, an explicit focal object represented as a URL, correspondence between data objects of interest (structures manipulated by AI reasoning services) and items that can selected as the focal object, and the ability to create multiple viewer instances (*floating* windows).

- Redesigned so that view types are now aligned with the types of AI services provided by Apex:

    - Task Agenda. A new view illustrating the state of task execution for an agent at any given instant. Every task on the agenda is displayed in a hierarchical table that can be expanded or collapsed to show or hide subtasks, respectively. Detailed information about each task, including state, resources, and times are displayed in columns.

    - Monitor View. A new view that depicts a monitor graphically. The monitor and its component conditions are displayed in a color-coded graph that clearly reveals their satisfaction status.

    - PDL View. A new view that, in one column, lists all procedures for a given agent, sorted alphabetically, by bundle, or by calling hierarchy. A second column displays the code for a selected procedure.

    - State Variable View. A new view that presents, as a table, the values of all state variables for a selected agent over time.

    - Object Diagram. A revamped version of the previous, limited diagram facility. Diagrams now can be attached to *any* object. Two kinds of diagrams are supported: photo (for example, JPG image) and wireframe (two-dimensional line drawing). Diagrams can have mouse-sensitive regions that correspond and link to the actual Apex objects they represent. By default, all objects have a graph diagram, which is a box and arrow representation of the object's subpart structure.

    - Trace View. A revamped version of the previous Trace View. Events are now decomposed into their logical parts and displayed in a table with numerous additional (and customizable) columns. Hyperlinked event components are clickable for quick navigation to their related views.

- Object Tree. An enhanced version of the previous Object Tree. All objects in an Apex application are now accessible in an expandable/collapsible hierarchy rooted at the application itself. Any object can be selected to make it the new focal object and to request desired views.

- Inspection View. An enhanced version of the previous Inspection View, made clearer by removing uninteresting class slots.

- Enhanced step/pause interface. A new pull-down menu allows the setting of specific times, time intervals, or step intervals at which to automatically pause an application.

AI Reasoning Services

- New framework and language extensions for advanced monitoring and querying, including support for measurements, estimations, trends, as well as complex and time-based conditions.

- Ability to specify logging policies for measurements (the new `log` step level clause).

- Support for *indexical* variables (variables surrounded by "+" characters that denote functions called with the current task to obtain a value).

- A new syntax (surrounding angle brackets) for expressing when ordinary PDL variables are expected to be bound, thus preventing unintended bindings.

- A new procedure type called Primitives to represent the low level skills of an agent. Primitives replace Activities (and obviate the need for the requisite CLOS programming).

- Sequential and ranked procedures now support procedure and step level clauses (they also now require explicit `step` clauses).

- New syntax shortcuts for procedures:

  - The step name (tag) can be omitted. When omitted, a name for the step is automatically generated from the first symbol in the task description, for example:

```
(step (reach ?object with ?hand))
…
(step (grasp ?object with ?hand) (waitfor  ?reach))
```

  - The index of a procedure can be given as the first or second argument to the procedure, instead of using an index clause, for example:

```
(procedure :sequential (pick up ?object with ?hand) …)
```

- New step-level clauses `repeating` and `responding`, for repetition and response policies (replacing the `period` clause), `log` for logging policies of measurements, and `on-start` and `on-end` for specifying Lisp code to evaluate around task executions.

- New procedure-level clauses `terminate` (alternative to step-level clause of the same name) and `expected-duration`.

- Better handling of cogevents received by an agent: order is preserved, asynchronous events are not lost, and the `*transient-cogevents*` flag is obsoleted.

- New `schedule` form for scheduling actions with PDL or from Lisp.

Application support

- Simplified interface for creating agents and resources.
- Publish-subscribe mechanism for communication between agents.
- Support for ISO-standard duration expressions.

Miscellaneous enhancements

- Improved performance in speed and memory usage.
- Numerous bug fixes and miscellaneous enhancements in both Apex and Sherpa.
- Many new example applications.
- A new tutorial on creating Apex applications.

## Apex license information

Apex is built upon Allegro Common Lisp (R) produced by Franz, Inc. Franz requires users of Apex to accept its license agreement, which pertains primarily to the use of the Franz components of the Apex software. The agreement is available in a file named `agreement.txt` located in the Apex installation directory.

## Apex documentation

The following documents contain information relevant to Apex:

- The *Apex Reference Manual* contains detailed instructions for using Apex and creating Apex applications.
- The tutorial *Paper Covers Rock: An Apex Tutorial* teaches you how to write basic Apex applications, by writing more and more complication versions of the game Rock, Scissors, Paper.

Apex documentation may be revised after the release of Apex 3.0. To check if you have the latest version of a document, note the document revision date and check for the Apex website for updates.

## Known issues

On the Macintosh, Sherpa works best when the "Look & Feel" setting (in the View menu) is set to Metal. There may be minor display or interaction problems if other settings are used.

On the Macintosh, object diagram display is slow. You need to wait a bit for the diagram to finish rendering. You may also need to zoom-in and zoom-out to see the diagram properly sized.

A network timeout error sometimes leaves Sherpa unresponsive. If this happens you must quit Sherpa manually (that is, from the Force Quit menu in the Finder) and restart it.

Some features will not work when running Sherpa and Apex on different machines. Most importantly, you can only load applications in Sherpa from the Recent Applications menu, because Load opens a file browser on Sherpa's host, while Apex looks on its host for files. The workaround for loading a new application is to load it using the Apex Listener (see Appendix A); subsequently it will appear in the Recent Applications menu.

## Upgrading existing applications

You must make the following changes to your existing applications to enable them to work in Apex 3.0.

The `:init-sim` argument of `defapplication` is no longer supported. A new argument `:type` is used to specify the application's type. See the `defapplication` section in Chapter 5 of the Apex Reference Manual.

The PDL built-in form `start-activity` (but not the Lisp function) has been obsoleted and must be replaced with definition and calling of primitive procedure. For example, given the following code:

```
(procedure
  (index (pick up ?object with ?hand))
  …
  (step s3 (start-activity ?hand reaching :duration '(3 secs)
             :object ?object))
    …
  (step s6 (terminate) (waitfor (completed ?s3))))

(defclass reaching (resource-activity) …)
(defmethod initialize-activity ((act reaching)) …)
(defmethod complete-activity ((act reaching)) …)
```

Replace this code with:

```
(procedure
  (index (pick up ?object with ?hand))
   …
  (step s3 (reach ?object with ?hand taking '(3 secs))
  …
  (step s6 (terminate) (waitfor ?s3)))

(primitive
  (index (reach ?object with ?hand taking ?duration))
  (duration ?duration)
  (on-start <behavior of initialize-activity above>)
    (on-completion <behavior of complete-activity above>))
```

In general, the rules for transforming start-activity based code to the new format is as follows (not all are necessarily applicable).

- Replace the `activity` class named in the call to `start-activity` with a new `primitive` procedure.

- Replace the call to `start-activity` with a call to this new primitive.
- The first argument to `start-activity` is a resource. Either pass this resource to the primitive (i.e. make it a variable in the primitive's index) or reference it directly inside the primitive. In either case, add a `profile` clause in the primitive for this resource.
- If the `start-activity` call has a `:duration` argument, add a `duration` clause to the primitive. The duration value can either be passed into the primitive as a variable in its index, or coded explicitly in the primitive.
- Other arguments passed to `start-activity` (e.g. `:object`) can be passed to the primitive as variables in its index.
- Replace the `update-activity` method of the `activity` class with an `update` clause in the primitive.
- Replace the `complete-activity` method of the `activity` class with an `on-completion` clause in the primitive.
- Replace the `initialize-activity` method of the `activity` class with an `on-start` clause in the primitive.

Remove `apex-info` forms from files (they will now only generate a warning if left in).

Replace `period` clauses with either `repeating` or `responding` (`period` will still work, but now prints a warning that it is deprecated).