

# GBB Glossary

**abstract event class** An event class that is intended to be used as a superclass for other event classes and is not intended to be instantiated (signalled directly).

**accessor methods** A pair of methods written or automatically generated for a generic function. One method reads the value of a particular slot, and the other method stores a value into the slot.

**Agenda control shell** A control shell based on a precondition/action model that gives you control over the execution order of KSAs. In the Agenda shell, the execution order is determined by a rating assigned to each KS.

**atomic value** Any Common Lisp value that isn't a list, except nil, which is both an atomic value and the empty list.

**blackboard** A tree or subtree of one or more space instances. That is, a blackboard comprises any space instance *and* its children *and* its children's children, and so on.

A root space instance that has no children is a blackboard; however, a root space instance that has children is not a blackboard when considered alone. Similarly, an interior space instance alone is not a blackboard. (Thus, a blackboard may or may not be a space-instance hierarchy.)

The following space-instance hierarchy contains five blackboards: one consisting of the entire hierarchy, one consisting of the hierarchy rooted at C, one consisting of only B, of only D, and of only E.



**blackboard database** In GBB, an object-oriented database that is a shared repository of problems, partial solutions, suggestions, and contributed information. The KSs interact anonymously using the blackboard database. The blackboard database is the container for all space instances and unit instances.

<b>ChalkBox</b>	In the GBB product family, the graphics toolkit that allows you to create object-oriented, graphical user interfaces. ChalkBox interfaces are portable on all hardware platforms and Common Lisp implementations on which GBB runs.
<b>CLOS</b>	The Common Lisp Object System.
<b>conses</b>	Linked cells used to represent elements of a list. Each cons consists of two components, a car (a cell that can hold any value) and a cdr (a cell that typically points to the remainder of the list).
<b>control shell</b>	A flexible control component used to direct the problem-solving activities of the KSs. The control shell is separate from the blackboard database and the KSs. GBB provides two general-purpose control shells, the FIFO shell and the Agenda shell.
<b>Control Shell Window</b>	A window displayed by the GBB Graphics System that allows you to interactively monitor control-shell activities and to stop or pause the control shell or invoke stepping of the control shell, as needed.
<b>demon (or daemon)</b>	A program, routine, or procedure used in other (nonGBB) programming methods that runs only upon being triggered in response to a specific situation within another program.
<b>dimension</b>	<p>An attribute of space instances and unit instances that provides a metric for positioning the unit instances on the space instances. A dimension is an extent that can be either continuous, consisting of a real-number interval, or collective, consisting of a set of labels (a list of symbols or Common Lisp objects).</p> <p>You can define as many dimensions as needed to best represent the kinds of unit instances used by your application. Dimensions play an important role in the retrieval of unit instances from the blackboard database and in graphical presentation of unit instances.</p>
<b>dimension value</b>	<p>The value of a unit instance relative to a particular dimension.</p> <p>A unit instance's dimension value determines where, relative to the same dimension of a space instance, the unit instance is to be positioned on the space instance. The dimension value can be the value of a nonlink or link slot of the unit instance, or a value computed from the value of a slot. As the slot value changes, the dimension value changes; and GBB repositions the unit instance on the space instance relative to the new dimension value.</p>
<b>dimension-value computation</b>	A definition that describes the location of one or more dimension values within a Common Lisp data structure. You define a dimension-value computation in order to specify one or more dimension values that are

(1) ranges, consisting of two numeric values representing minimum and maximum values; (2) computed from a complex data structure, which is a data structure made up of more than one value; or (3) composite.

<b>event</b>	<p>A signal indicating the occurrence of some activity within a GBB application (for example, the creation of a unit instance) or external to the application (for example, a mouse click or timed interruption).</p> <p>Events have two purposes: they trigger event functions, and they can cause the control shell to trigger a KS or perform other activities.</p>
<b>event buffer</b>	<p>A storage place for event instances that are created by the control shell to record events that are of interest to a KS.</p>
<b>event class</b>	<p>A class associated with an event. In GBB, event classes are also CLOS classes.</p> <p>GBB uses event classes for two purposes: to obtain information needed when signalling events and to add any event functions you want to associate with the events. Additionally, the control shell uses event classes to create event instances.</p>
<b>event instance</b>	<p>An instance of an event class. Event instances are used by the control shell to trigger a KS, or to obviate or retrigger a KSA.</p>
<b>event function</b>	<p>A user-defined function you associate with a particular event class, to be run when the event is signalled. You can specify any number of event functions for a given event class.</p>
<b>event printing</b>	<p>Printing, to a specified stream, information about an event that has been signalled.</p>
<b>event type</b>	<p>A value that indicates whether an event involves a unit instance and, if so, whether it also involves a specific slot in the unit instance or a specific space instance. Event types are <code>:non-unit</code>, <code>:unit</code>, <code>:unit-nonlink-slot</code>, <code>:unit-link-slot</code>, and <code>:unit-space</code>.</p>
<b>false</b>	<p>A value for Boolean tests that is represented by the symbol <code>nil</code>.</p>
<b>FIFO</b>	<p>A first-in, first-out ordering scheme used by the FIFO control shell.</p>
<b>FIFO control shell</b>	<p>GBB's most basic control shell, which is based on a FIFO ordering scheme (or, optionally, a LIFO ordering scheme). Generally, the FIFO shell causes all triggered KSs to be activated and executes the KSAs in the same order in which they are created.</p>



<b>KS</b>	A knowledge source, which is a program module that contains expertise needed to solve a particular problem. A KS contains two kinds of information: (1) the KS code that is executed by the control shell so the KS can contribute its expertise, and (2) specifications indicating when and with what information the KS code is to be executed.
<b>KS function</b>	A function that is called when a KSA is executed, in order to perform the problem-solving activity of the KS with which the KSA is associated. The KS function is one of the types of functions that can make up the code portion of a KS. (The code can include an activation predicate, precondition function, retrigger function, and so on.) A KS must contain a KS function; the other types of functions are optional.
<b>KS language</b>	Any language you can use to write the code for your application's KSs (that is, GBB/OPS, Common Lisp, or any language that can be called from Common Lisp).
<b>KS trigger</b>	The association of one or more events with a KS so that the occurrence of any one of the events will cause the control shell to act on the KS. Events associated with a KS as a triggering trigger cause the control shell to trigger the KS for activation, whereas events associated with a KS as an obviation trigger or a retriggering trigger cause the control shell to run the obviation predicate or the retriggering function for the KSAs of the KS.
<b>KSA</b>	A knowledge-source activation. Whereas the KS contains the code that implements the expertise of the KS, a KSA is the combination of the KS and a specific execution context. Numerous KSAs for a given KS can exist at the same time.
<b>LIFO</b>	A last-in, first-out ordering scheme that can be used by the control shell for scheduling KSAs for execution.
<b>link</b>	A bidirectional pointer between unit instances.
<b>link slot</b>	A slot declared to contain a link between unit instances.
<b>list</b>	A sequence consisting of zero or more elements, each of which can be an atom or another list. The elements of a list are represented by conses.
<b>metering</b>	The process of measuring all or particular phases of GBB or control-shell processing, in order to gather data you can use for analyzing and optimizing those processing activities.
<b>NetGBB</b>	A product that, as an extension of GBB, allows application developers to use networked GBB systems to construct enterprise-wide distributed

blackboard applications. NetGBB, coupled with GBB, provides developers with a blackboard framework and integrated network communication facilities designed to work together especially for distributed networks of blackboard systems.

<b>obviation</b>	In the control shell, removing from the queue of KSAs that are pending execution a KSA that is no longer necessary.
<b>path variable</b>	<p>A variable that specifies information GBB uses in determining the space instances on which to store instances of a unit class.</p> <p>More specifically, this kind of variable specifies a nonlink slot from which GBB is to obtain the value (or a portion of the value, if the slot contains a path-variable computation). You can include a path variable as a space name or space-instance index in the form you specify using the <code>:paths</code> option to the <b>define-unit-class</b> macro. (The <code>:paths</code> option specifies space instances and/or space-instance paths used to determine the space instance on which to store unit instances.) You can specify multiple path variables for a unit class.</p>
<b>path-variable computation</b>	A definition that describes the location of one or more path-variable values within a Common Lisp data structure. You define a path-variable computation in order to specify one or more path-variable values that are (1) ranges, consisting of two numeric values representing minimum and maximum values; (2) computed from a complex data structure, which is a data structure made up of more than one value; or (3) composite.
<b>place form</b>	A form that, when evaluated, accesses a value in some location; and, when used with the <b>setf</b> macro or related macros, causes the value to be updated.
<b>precondition function</b>	A function you can specify for a KS that will, upon triggering of the KS, analyze the circumstances under which the KS was triggered and determine whether to activate the KS. Precondition functions are used only by the Agenda control shell.
<b>predicate</b>	A function that tests for some condition involving its arguments and returns <code>nil</code> if the condition is false or some non- <code>nil</code> value if the condition is true.
<b>qualified space name</b>	<p>A specifier that disambiguates the path to a space, when the space has the same name as another space.</p> <p>The form of a qualified space name is as follows:</p> $([\text{:root}] \{space\text{-}name\}^+)$ <p>where <math>\{space\text{-}name\}^+</math> contains the name of the space preceded by the name of the space's parent, and so on, until the specifier is unambiguous.</p>

The `:root` keyword indicates that the space names in `{space-name}`<sup>+</sup> form a complete path from the root space.

<b>quiescence</b>	A condition that occurs in the control shell when there are no executable KSAs in the queue of pending KSAs.
<b>reader method</b>	A method used for obtaining the value of a slot. The <b>define-unit-class</b> macro automatically defines a reader method on the generic function you specify using the <code>:accessor</code> option for a given slot.
<b>retriggering</b>	Performing some control action on a KSA that is pending execution, based on the occurrence of some event (for example, changing the rating of the KSA based on new information).
<b>root space</b>	Typically, a space that has no parents (but may have children). However, a root space may also be a space within a hierarchy of spaces and at which a subhierarchy is rooted.
<b>scheduling loop</b>	The sequence of processing activities performed cyclically by the control shell, during which it processes its event buffer in order to determine which KSs to trigger, for which KSs to obviate or retrigger KSAs, and which KSA to execute.
<b>sequenced KS</b>	A KS that causes other KSs to be processed in the sequence in which you specify them within the same scheduling-loop cycle.
<b>signalling of an event</b>	Reporting the occurrence of an event. GBB signals predefined events; you must explicitly signal user-defined events.
<b>slot</b>	A place in a unit instance in which a value is stored and can be accessed and changed.
<b>slot accessor</b>	See <a href="#">accessor methods</a> .
<b>slot name</b>	The name of a unit-class slot.
<b>space definition</b>	The definition of a space. Space definitions are instantiated to create space instances, on which unit instances can be stored. GBB allows space definitions to be replicated as multiple space instances.
<b>space instance</b>	An atomic blackboard piece that serves as a container for storing unit instances. The attributes of each space instance are determined by its space definition.

<b>space-instance index</b>	<p>An integer that is associated with a space instance when it is created, to uniquely identify it. Unless you specify otherwise, the first instance of a space has the index 0, the second has the index 1, and so on.</p> <p>You need not include a 0 index in a space-instance path, because it is assumed. For example, the following space-instance path, which describes the path to the <code>vehicle-track</code> space instance that has an index of 1, contains three space names and the necessary indexes:</p> <pre>(space 1 hyp vehicle-track 1)</pre>
<b>space-instance path</b>	<p>A list describing a path from the root of a space-instance hierarchy to a specific space instance. The list contains space names, optionally interspersed with space-instance indexes and wildcards; and it must begin with either a wildcard or the name of a space that has no parents.</p> <p>Three examples follow:</p> <pre>(root A E) (space1 0 space2 0 hyp1 1) (* B)</pre>
<b>stepping</b>	Temporarily stopping processing at predefined points during GBB and control-shell operations; useful for debugging.
<b>stimulus unit instance</b>	The unit instance associated with a unit, unit-path, unit-nonlink-slot, or unit-link-slot type of event that caused the control shell to trigger a KS.
<b>trigger</b>	See <a href="#">KS trigger</a> .
<b>triggering</b>	The action of the control shell that makes a KS a candidate for activation.
<b>true</b>	A value for Boolean tests that is represented by any non-nil value (not necessarily <code>t</code> ).
<b>unit</b>	See <a href="#">unit instance</a> .
<b>unit class</b>	An object that determines the structure and behavior of a set of unit instances. A unit class defines characteristics such as slots, paths, dimensions, and permanent event functions for its instances. In GBB, unit classes are also CLOS classes.
<b>unit instance</b>	An instance of a unit class; a blackboard object. In GBB, unit instances are also CLOS objects.

<b>unit mapping</b>	A specification that indicates how instances of a unit class are to be stored on space instances. A unit mapping can partition a space instance into buckets, in which unit instances can then be stored.
<b>widget</b>	A geometric graphical object used as a user-interface component in ChalkBox. Typically, widgets are reusable and configurable.
<b>wildcards</b>	Reserved symbols you can use to refer to multiple space instances in a space-instance path.
<b>writer method</b>	A method used for modifying the value of the slot. The <b>define-unit-class</b> macro automatically defines a writer method on the generic function named ( <b>setf</b> <i>reader-function-name</i> ) for a given slot (where <i>reader-function-name</i> is the generic function on which the reader method is defined).

Copyright © 2000 by Knowledge Technologies International, Inc. All rights reserved.

GBB, GBB Runtime, ChalkBox, NetEval, NetGBB, and the KTI logo are trademarks of Knowledge Technologies International, Inc. Other brand or product names are trademarks or registered trademarks of their respective holders.

Knowledge Technologies International, Inc.  
401 Main Street  
Amherst, MA 01002  
Phone: (800) 577-8990, (413) 256-8990  
Fax: (413) 256-3179  
E-mail: [gbb-info@KTIworld.com](mailto:gbb-info@KTIworld.com)  
WWW: [www.KTIworld.com](http://www.KTIworld.com)

May 2000

