Contributors

Jakub Czyz

Nate Derbinsky

John Laird

Bob Marinier

Andy Nuxoll

Joseph Xu

Table of Contents

1. Document History 3

2. Soar-EpMem Motivation 4

3. Working Memory Structure 5

4. Storing Episodes 6

4.1. New Episode Creation 6

4.2. Episode Contents 6

4.3. Working Memory Activation 7

4.3.1. The wma Command 8

4.3.2. WMA Parameters 8

4.4. Soar-EpMem Storage 9

5. Retrieving Episodes 11

5.1. Soar-EpMem Commands 11

5.2. Non-Cue-Based Retrievals 11

5.2.1. Absolute NCB Retrievals 11

5.2.2. Relative NCB Retrievals 11

5.3. Cue-Based Retrievals 12

5.4. Retrieval Meta-Data 13

6. Soar-EpMem Parameters 15

6.1. Parameter Configuration 15

6.2. Parameter Descriptions 15

6.2.1. General 15

6.2.2. Storage 15

6.2.3. Representation 16

6.2.4. Space 16

6.3. Full Parameter Configuration 17

6.4. Parameter Behavior 17

7. Soar-EpMem Statistics 18

8. Trace Information 19

9. Soar-EpMem Performance 20

10. Soar-EpMem Programmer Reference 22

10.1. Soar-EpMem 22

10.1.1. Useful Commands 22

10.1.2. Parameters 22

10.1.3. Retrieval Agent Commands 23

10.1.4. Retrieval Agent Meta-Data 23

10.2. Working Memory Activation 24

10.2.1. Useful Commands 24

10.2.2. Parameters 24

# Document History

Version 0.1

Initial specification.

# Soar-EpMem Motivation

Soar-EpMem is a task-independent, architectural integration of an artificial episodic memory (EpMem) with Soar. The EpMem mechanism will automatically record episodes as a Soar agent executes. These episodes can later be queried and retrieved in order to improve performance on future tasks.

# Working Memory Structure

Upon creation of a new state within working memory, the architecture will automatically create a structure in working memory called **epmem**. Within this structure, agents issue requests to Soar-EpMem (see Section 5.1) by populating the **command** identifier with working memory elements (WMEs) and process Soar-EpMem generated WMEs in the **result** identifier (see Section 5.4).

# Storing Episodes

This section details Soar-EpMem storage of episodes, including new episode triggering, what is stored, interactions with Working Memory Activation (WMA), as well as where and in what format the episodes are stored.

## New Episode Creation

One functional requirement of an artificial episodic memory is that recording new episodes does not require deliberate action/consideration by the agent. Soar-EpMem provides automatic storage of new episodes during the Output phase of each decision cycle as dictated by the **trigger** parameter.

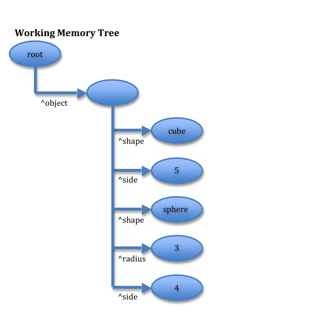
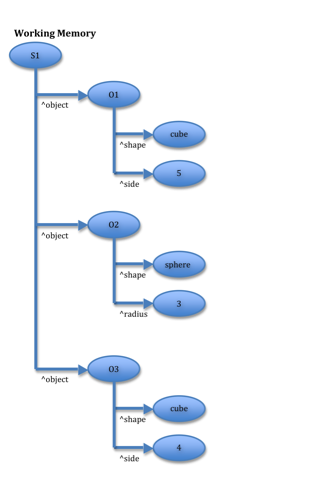
For debugging purposes, the **force** parameter allows the user to manually request that an episode be recorded during the Output phase of the current decision cycle. Behavior is as follows:

* The value of the **force** parameter is initialized to **off** every decision cycle.
* During the Output phase of any decision cycle where the **force** parameter has a value of **on**, a new episode is stored irrespective of the value of the **trigger** parameter.

## Episode Contents

For an artificial episodic memory to be task-independent it must record most, if not all, information available to the agent at the time of episode creation. Aside from exceptions discussed below, when Soar-EpMem records a new episode, it stores the entire contents of the top state of Working Memory. Consider the following exceptions:

* **Shared WMEs** – Soar agents can create graph structures in Working Memory using shared WMEs. To avoid cycles during storage/retrieval, Soar-EpMem does not store these shared structures. During episode creation, Soar-EpMem traverses Working Memory in a breadth-first manner and only records WMEs upon first encounter. In effect, this reduces Soar’s Working Memory graph to a Working Memory tree.
* **Multi-Valued Attributes** – Queries across multi-valued attributes can become computationally expensive. For performance reasons, storage of multi-valued attribute identifiers is controlled by the **provenance** parameter. When multi-valued attributes are not stored (**provenance** value **off**), children of multiple identifiers are collapsed into a single representative structure. As an example, consider the following illustrated conversion:



* **Manual Exclusions** – There are classes of WMEs that Soar agents may encounter that provide no benefit in context of EpMem. For instance, the “random” WME on the TankSoar input link structure provides a different random number on each update; this value is potentially useful to an agent designer, but most likely will not contribute to effective episodic learning. Moreover, excluding WMEs from storage can provide performance benefits (reduced memory consumption and storage/retrieval time). The **exclusions** parameter allows run-time management of a list of attribute names that will be ignored during Soar-EpMem storage.

## Working Memory Activation

During the episodic retrieval process (discussed in detail later), multiple episodes may match an agent’s query. Nuxoll has produced data that demonstrates improved retrieval quality when using Working Memory Activation (WMA) of WMEs as a form of selection bias. Thus, Soar-EpMem supports integration with WMA in Soar. For a theoretical discussion of the Soar implementation of WMA, consider reading *Comprehensive Working Memory Activation in Soar* (Nuxoll, A., Laird, J., James, M., ICCM 2004).

The following sections detail configuration of WMA, including the **wma** command and WMA parameters.

### The wma Command

Management of WMA within Soar makes use of the **wma** command. Executing the command with no options will print a table of current parameter information. Executing the command with the **--print** (or **-p**) option prints a table of currently activated WMEs for debugging purposes. Finally, the **wma** command has getter (**-g** or **--get**) and setter (**-s** or **--set**) options for retrieving/manipulating parameter values (discussed in the next section).

### WMA Parameters

The following table briefly describes the parameters available for manipulation using the **wma** command’s **get**/**set** options. Further text below provides more information regarding specific parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Acceptable Values | Default | Description |
| **activation** | on  off | on | Turns on/off WMA |
| **decay-rate** | [0,1] | 0.8 | Specifies the speed at which WMEs are decayed |
| **criteria** | o-agent  o-agent-arch  all | all | Specifies what WMEs will have decay values |
| **forgetting** | on  off | off | Turns on/off removal of WMEs with low activation values |
| **i-support** | none  no-create  uniform | uniform | Specifies the mode in which i-supported WMEs affect activation levels |
| **persistence** | on  off | off | Dictates whether an instantiation activates WMEs just once (default), or every cycle until it is retracted |
| **precision** | high  low | low | Level of precision (relates to performance) with which activation levels are calculated |

decay-rate

The **decay-rate** parameter controls the speed at which WMEs are decayed. A value of 0 will decay WMEs instantly, where 1 will not reduce initial activation level. Note that the value is internally multiplied by -1 (which is reflected upon retrieving the parameter value).

criteria

The criteria parameter specifies the classes of WMEs that will have decay values:

* **o-agent** – Only o-supported WMEs created by the agent (i.e. they have a supporting preference) are activated.
* **o-agent-arch** – All o-supported WMEs including architecturally created WMEs are activated.
* **all** – All WMEs are activated.

i-support

The i-support parameter specifies the mode in which i-supported WMEs affect activation levels:

* **none** – i-supported WMEs do not affect activation levels.
* **no-create** – i-supported WMEs boost the activation levels of all o-supported WMEs in the instantiations that test them. Each WME receives an equal boost, irrespective of "distance" (in the backtrace) from the tested WME.
* **uniform** – i-supported WMEs boost the activation levels of all o-supported WMEs in the instantiations that created or test them. Each WME receives an equal boost irrespective of "distance" (in the backtrace) from the tested WME.

## Soar-EpMem Storage

EpMem currently uses SQLite to facilitate efficient and standardized storage and querying of episodes. The episodic store can be maintained in memory or on disk (per the **database** and **path** parameters). If the store is located on disk, users can use any standard SQLite programs/components to access/query its contents.

Because the most efficient organization and representation of the episodic store is currently under investigation, Soar-EpMem exposes different storage modes via the **indexing** parameter. Our current data shows that the default method, **bigtree\_rit**, maintains high efficiency in most environments, and thus other settings are reserved for experimentation. To facilitate external access to the store, the text below represents the relational schema of the default organizational method:

ids (WME name/value registration)

* child\_id (integer, primary key) - element id
* parent\_id (integer) - element id of parent
* name (text) - WME name
* value - WME value, NULL for identifiers
* hash (integer) - hash of the name/value pair

times (Efficient lookup of valid temporal ids)

* id (integer, primary key) - valid temporal id

now (Registry of “now” element ranges)

* id (integer, primary key) - element id, relates to ids (child\_id)
* start (integer) - temporal id when element instance started

points (Registry of completed valid element ranges lasting one decision cycle)

* id (integer) - element id, relates to ids (child\_id)
* start (integer) - temporal id when element instance started/ended

episodes (Registry of completed valid element ranges)

* id (integer) - element id, relates to ids (child\_id)
* start (integer) - temporal id when element instance started
* end (integer) - temporal id when element instance ended
* node (integer) - RIT node value, expedites intersection searches

left\_nodes (RIT left table, temporary)

* min (integer)
* max (integer)

right\_nodes (RIT right table, temporary)

* node (integer)

weights (Temporary look-up table of element weights during query transactions)

* id (integer, primary key) - element id, relates to ids (child\_id)
* weight (real) - WMA value

It should be noted that currently only the **bigtree\_instance** indexing mode stores WMA values of episode WMEs. The remaining methods make use of the WMA values of cue WMEs at retrieval time.

# Retrieving Episodes

This section details the agent interface to Soar-EpMem retrievals, including command protocol, non-cue-based (NCB) retrievals, cue-based (CB) retrievals, and retrieval meta-data.

## Soar-EpMem Commands

An agent issues a command to the Soar-EpMem system by populating appropriate WMEs on the **command** identifier of a state’s **epmem** structure. At the end of each decision cycle, after creating a new episode is considered (and possibly implemented), Soar-EpMem processes each state’s EpMem **command** structure. Results, meta-data, and errors are placed on the **result** identifier of that state’s **epmem** structure (discussed in Section 5.4).

Only one type of command (which may consist of multiple WMEs) can be issued in a single decision cycle (though multiple states may issue commands). Malformed commands (including attempts at multiple commands) will result in an error.

After a command has been processed, Soar-EpMem will ignore it until some aspect of the **command** structure changes (via addition/removal of WMEs). When this occurs, the **result** structure is cleared and the new command (if one exists) is processed.

## Non-Cue-Based Retrievals

The following sections discuss issuing absolute and relative NCB retrieval commands.

### Absolute NCB Retrievals

At time of storage, each episode is attributed a unique temporal id. This id is the current value of **time** statistic (see Section 7) and is provided as the **memory-id** meta-data item of retrieved episodes (see Section 5.4). An absolute NCB retrieval is one that requests an episode by temporal id. An agent issues an absolute NCB retrieval by placing a WME on the **command** structure with name **retrieve** and value equal to the desired temporal id:

state.epmem.command.retrieve temporal-id

Supplying an invalid value for the **retrieve** command will result in an error.

The temporal id of the first episode in an episodic store will have value 1 and each subsequent episode’s temporal id will increase by 1. Thus the desired temporal id may be the mathematical result of operations performed on a known episode’s temporal id.

This implementation of Soar-EpMem does not implement any episode dynamics, including forgetting. Thus any integer temporal id greater than 0 and less than the current value of the **time** statistic will be valid. However, if forgetting is implemented in future versions, no such guarantee will be made.

### Relative NCB Retrievals

One interesting characteristic of episodic memory is the empirical ability to “play forward” episodes through time. Soar-EpMem implements this functionality through relative NCB retrievals.

The system stores the temporal id of the last successful retrieval (NCB or CB). Agents can indirectly make use of this information by issuing **next** or **previous** commands. Soar-EpMem executes these commands by attempt to retrieve the episode immediately proceeding/preceding the last successful retrieval (respectively). To issue one of these commands, the agent must create a new identifier with the appropriate command name on the **command** structure:

state.epmem.command.next <n>

state.epmem.command.previous <p>

If no such episode exists then an error is returned.

In this implementation of Soar-EpMem, if the temporal id of the last successfully retrieved episode is known to the agent (as could be the case by accessing result meta-data), these commands are identical to performing an absolute NCB after adding/subtracting 1 to the last temporal id (respectively). However, if an episode dynamic like forgetting is implemented, these relative commands are guaranteed to return the next/previous valid episode (assuming one exists).

## Cue-Based Retrievals

CB retrieval commands are used to search for an episode in the store that “best” matches an agent-supplied cue, while potentially adhering to optional modifiers. A cue is composed of WMEs that partially describe a top state of Working Memory in the retrieved episode. All CB retrieval requests must contain a single **query** cue, a single **neg-query** cue, or both. A **query** cue describes structures desired in the retrieved episode, whereas a **neg-query** cue describes non-desired structures. For example, the following Soar production creates a **query** cue consisting of a particular state name and a copy of a current value on the input link structure:

sp {sample\*query

(state <s> ^epmem.command <ec>

^io.input-link.foo <bar>)

-->

(<ec> ^query <q>)

(<q> ^name my-state-name

^io.input-link.foo <bar>)

}

In applying a cue to the episodic store, identifiers are not used to compare candidate episodes, but simply to establish the structure of non-identifiers. Thus, all valid cues must contain at least 1 non-identifier WME.

CB retrievals can be thought of as a nearest-neighbor search. First, all candidate episodes, (defined as episodes containing at least one non-identifier WME in at least one cue) are identified. Two quantities are calculated for each candidate episode, with respect to the supplied cue(s): the **cardinality** of the match (defined as the number of matching non-identifier WMEs) and the **activation** of the match (defined as the sum of the WMA decay values of each matching non-identifier WME). Note that each of these values is negated when applied to a **neg-query**. To form each candidate episode’s match score, these quantities are combined with respect to the **balance** parameter as follows:

Match Score = (balance)\*(cardinality) + (1 - balance)\*(activation)

The candidate episode with the greatest match score is then retrieved. In the case of identical winning match scores, the most recent episode is selected.

The CB retrieval process can be further tempered using optional modifiers:

* The **before** command requires that the retrieved episode come relatively before a supplied temporal id:

state.epmem.command.before temporal-id

* The **after** command requires that the retrieved episode come relatively after a supplied temporal id:

state.epmem.command.after temporal-id

* The **prohibit** command requires that the temporal id of the retrieved episode is not equal to a supplied temporal id:

state.epmem.command.prohibit temporal-id

Multiple **prohibit** command WMEs may be issued as modifiers to a single CB retrieval.

If no episode satisfies the cue(s) and optional modifiers an error is returned.

## Retrieval Meta-Data

The following list details the WMEs Soar-EpMem populates in the **result** identifier of the **epmem** structure wherein a command was issued:

* **retrieved <episode>**

If Soar-EpMem retrieves an episodic memory, that memory is placed here. This WME is an identifier that is treated as the root of the state that was used to create the episodic memory. If the **retrieve** command was issued with an invalid temporal id, the value of **retrieved** will be **no-memory**.

* **status**

This WME provides information about the result of a retrieval command:

* + **success** – the CB retrieval command resulted in a successful match
  + **failure** – the CB retrieval was legitimate but no matching episode was found
  + **bad-cmd** – the command was malformed or more than one command was issued
* **match-score**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is a decimal indicating the raw match score for that episode with respect to the cue(s).

* **cue-size**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is an integer indicating the number of non-identifier WMEs in the cue(s).

* **normalized-match-score**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is the decimal result of dividing the raw match score by the cue size. It can hypothetically be used as a measure of Soar-EpMem’s relative confidence in the retrieval.

* **match-cardinality**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is an integer indicating the number of non-identifier WMEs matched in the **query** cue minus those matched in the **neg-query** cue.

* **memory-id**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is an integer indicating the temporal id of the retrieved episode.

* **present-id**

This WME is created whenever an episode is successfully retrieved from a CB retrieval command. The WME value is an integer indicating the current temporal id, such as to provide a sense of “now” in EpMem terms. By comparing this value to the **memory-id** value, the agent can gain a sense of the relative time that has passed since the retrieved episode was recorded.

# Soar-EpMem Parameters

The following sections discuss how to configure the Soar-EpMem parameters discussed in previous sections.

## Parameter Configuration

Individual configuration parameters are retrieved and manipulated using the **get** and **set** switches of the **epmem** command:

epmem [-g|--get] <parameter>

epmem [-s|--set] <parameter> <value>

Agents can retrieve and change parameters in the actions of rules using the **cmd** function.

## Parameter Descriptions

All Soar-EpMem parameters are organized below. The *Protected* field is discussed in Section 6.4).

### General

|  |  |
| --- | --- |
| Purpose | Enable or disable Soar-EpMem |
| Parameter | **learning** |
| Values | |  |  | | --- | --- | | **off** | Disable Soar-EpMem | | **on** | Enable Soar-EpMem | |
| Default | **on** |
| Protected | no |

### Storage

|  |  |
| --- | --- |
| Purpose | Specifies whether the episodic store will be maintained in memory or on disk |
| Parameter | **database** |
| Values | |  |  | | --- | --- | | **file** | Episodic store is maintained on disk | | **memory** | Episodic store is maintained in memory | |
| Default | **file** |
| Protected | yes |

|  |  |
| --- | --- |
| Purpose | Specifies where on disk the episodic store will be saved |
| Parameter | **path** |
| Values | |  |  | | --- | --- | | **<empty>** | Soar-EpMem will create a temporary database file on disk during execution (and delete it after use) | | **<valid path>** | Soar-EpMem will use the specified path for its database file on disk - if the file doesn’t exist, it will be created | |
| Default | **<empty>** |
| Protected | yes |

### Representation

|  |  |
| --- | --- |
| Purpose | Specifies the store indexing mode |
| Parameter | **indexing** |
| Values | |  |  | | --- | --- | | **bigtree\_hybrid** | Supplements Nuxoll’s Interval method with a bit-string episode representation for retrievals | | **bigtree\_instance** | Implements Nuxoll’s Instance method | | **bigtree\_range** | Implements Nuxoll’s Interval method | | **bigtree\_rit** | Supplements Nuxoll’s Interval method with a Relational Interval Tree for retrievals | |
| Default | **bigtree\_rit** |
| Protected | yes |

|  |  |
| --- | --- |
| Purpose | Specifies whether multi-valued attribute structure is stored (*not implemented*) |
| Parameter | **provenance** |
| Values | |  |  | | --- | --- | | **off** | Multi-valued attribute structure is not stored | |
| Default | **off** |
| Protected | yes |

### Space

|  |  |
| --- | --- |
| Purpose | Specifies what triggers new episode creation |
| Parameter | **trigger** |
| Values | |  |  | | --- | --- | | **dc** | Episodes are recorded every decision cycle | | **none** | Episodes are not automatically recorded | | **output** | Episodes are recorded decision cycles when there is a change to the output link | |
| Default | **output** |
| Protected | no |

|  |  |
| --- | --- |
| Purpose | Forces creation of a new episode |
| Parameter | **force** |
| Values | |  |  | | --- | --- | | **off** | Episode recording is dependent upon the current trigger | | **on** | An episode will be recorded this decision cycle | |
| Default | **off** |
| Protected | no |

|  |  |
| --- | --- |
| Purpose | Specifies the degree to which cardinality and WMA are weighted in query processing |
| Parameter | **balance** |
| Values | Numeric, **[0,1]** |
| Default | **0.5** |
| Protected | no |

|  |  |
| --- | --- |
| Purpose | Specifies a list of WME attribute names that are ignored during episode creation |
| Parameter | **exclusions** |
| Values | |  |  | | --- | --- | | **<any string>** | If the supplied value does not currently exist within the exclusion list it is added, otherwise it is removed from the list. | |
| Default | **<empty>** |
| Protected | no |

## Full Parameter Configuration

Entering simply the **epmem** command (with no switches) will return full parameter configuration information. For example, assuming default configuration, the result of executing **epmem** is as follows:

>epmem

EpMem learning: on

Storage

-------

database: file

path:

Representation

--------------

indexing: bigtree\_rit

provenance: off

Space

-----

trigger: output

force: off

balance: 0.5

exclusions:

## Parameter Behavior

Upon attempting to set a Soar-EpMem parameter, the new value is validated. If the value is found to be invalid, the system will use the previous value.

The set of parameters listed above that have a “yes” in the *Protected* field cannot be changed once the Soar-EpMem system has been “initialized.” The Soar-EpMem system initializes during recording of the first episode since starting Soar or issuing the **close** switch of the **epmem** command (see Section 10.1.1).

# Soar-EpMem Statistics

Feedback from the Soar-EpMem system is retrieved using the **stats** switch of the **epmem** command:

epmem [-S|--stats] <statistic>

If a **statistic** argument is provided, the command returns the value of a specific statistic. The valid statistic arguments are listed below.

|  |  |
| --- | --- |
| Statistic | **time** |
| Description | Current episode id (starts at 1, increases) |
| Label | Time |

|  |  |
| --- | --- |
| Statistic | **mem\_usage** |
| Description | Current SQLite memory usage in bytes |
| Label | Memory Usage |

|  |  |
| --- | --- |
| Statistic | **mem\_high** |
| Description | Greatest SQLite memory usage in bytes since last database initialization |
| Label | Memory Highwater |

When using the **bigtree\_rit** indexing method, the following additional statistics may be requested for debugging performance issues in the Relational Interval Tree: **rit\_offset**, **rit\_left\_root**, **rit\_right\_root**, **rit\_min\_step**.

Agents can retrieve specific statistics in rule actions using the **cmd** function.

Entering the **epmem --stats** command with no statistic, or an invalid statistic, will return all statistics. A sample execution may look as follows:

>epmem --stats

Time: 0

Memory Usage: 0

Memory Highwater: 0

# Trace Information

To view Soar-EpMem debugging information, use the following watch switch:

watch [-e|--epmem]

This function is not enabled by default or through any watch level. At present, this watch level generates a message when an episode is recorded.

# Soar-EpMem Performance

In initial empirical usage, Soar-EpMem is very stable but its use incurs a significant cost in Soar performance. This section provides some Soar-EpMem usage data with a single TankSoar “simple-bot” agent (modified for EpMem) using all default WMA/EpMem parameter values, except **database** value of **memory** (Linux, 2.4GHz Xeon, 4GB RAM). The agent was required to store new episodes every decision cycle, and issued a CB retrieval request every other decision cycle.

Storage is an operation requiring time linear in the number of WMEs per episode. The NCB retrieval operation using the **bigtree\_rit** indexing method theoretically requires time logarithmic in the size of the episodic store.

CB retrievals currently require time linear in the size of the episodic store and typically dominate cycle time.

The **bigtree\_rit** indexing method requires memory linear in the changes in the Soar-EpMem agent’s Working Memory.

# Soar-EpMem Programmer Reference

The following tables list basic information about each of the Soar-EpMem related commands. It is not intended to substitute for this document, but a quick reference for commonly used commands and options.

## Soar-EpMem

### Useful Commands

|  |  |  |
| --- | --- | --- |
| Command |  | Description |
| epmem |  | Summary table of parameter settings |
| epmem [-g|--get] <parameter> |  | Retrieve a Soar-EpMem parameter value |
| epmem [-s|--set] <parameter> <value> |  | Set a Soar-EpMem parameter value |
| epmem [-S|--stats] <statistic> |  | Access Soar-EpMem statistics |
| epmem [-c|--close] |  | Close the current Soar-EpMem database |
|  |  |  |
| watch [-e|--epmem] |  | Soar-EpMem debugging trace |

### Parameters

Parameters noted with a \* are *protected*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **General** | | | | |
| Parameter Name |  | Acceptable Values |  | Default |
| learning |  | on  off |  | off |
|  |  |  |  |  |
| **Storage** | | | | |
| Parameter Name |  | Acceptable Values |  | Default |
| database\* |  | file  memory |  | file |
|  |  |  |  |  |
| path\* |  | <empty>  <system path> |  | <empty> |
|  |  |  |  |  |
| **Representation** | | | | |
| Parameter Name |  | Acceptable Values |  | Default |
| indexing\* |  | bigtree\_hybrid  bigtree\_instance  bigtree\_range  bigtree\_rit |  | bigtree\_rit |
|  |  |  |  |  |
| trigger |  | dc  none  output |  | output |
|  |  |  |  |  |
| **Space** | | | | |
| Parameter Name |  | Acceptable Values |  | Default |
| balance |  | [0,1] |  | 0.5 |
|  |  |  |  |  |
| exclusions |  | <any string> |  |  |

### Retrieval Agent Commands

*Absolute NCB Retrieval*

state.epmem.command.retrieve temporal-id

*Relative NCB Retrieval*

state.epmem.command.next

state.epmem.command.previous

*CB Retrieval*

state.epmem.command.query <cue>

and/or

state.epmem.command.neg-query <cue>

CB Retrieval Optional Modifiers

state.epmem.command.before temporal-id

state.epmem.command.after temporal-id

state.epmem.command.prohibit temporal-id

### Retrieval Agent Meta-Data

state.epmem.result

^retrieved <episode>

^status << success failure bad-cmd >>

^match-score double

^cue-size integer

^normalized-match-score double

^match-cardinality integer

^memory-id temporal-id

^present-id temporal-id

## Working Memory Activation

### Useful Commands

|  |  |  |
| --- | --- | --- |
| Command |  | Description |
| wma |  | Summary table of parameter settings |
| wma [-g|--get] <parameter> |  | Retrieve a WMA parameter value |
| wma [-s|--set] <parameter> <value> |  | Set a WMA parameter value |
| wma [-p|--print] |  | Prints a table of currently activated WMEs |

### Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter Name |  | Acceptable Values |  | Default |
| activation |  | on  off |  | on |
|  |  |  |  |  |
| decay-rate |  | [0,1] |  | 0.8 |
|  |  |  |  |  |
| criteria |  | o-agent  o-agent-arch  all |  | all |
|  |  |  |  |  |
| forgetting |  | off  on |  | off |
|  |  |  |  |  |
| i-support |  | none  no-create  uniform |  | uniform |
|  |  |  |  |  |
| persistence |  | off  on |  | off |
|  |  |  |  |  |
| precision |  | high  low |  | low |
|  |  |  |  |  |