



# Gedae 5.1 Release Notes

December 2006

Address: Gedae, Inc.  
1247 N Church St, STE 5  
Moorestown, NJ 08057

Telephone: (856) 231-4458  
FAX: (856) 231-1403  
Internet: [www.gedae.com](http://www.gedae.com)

# Note to Champ AVII and Champ AVIV Users:

Please read SCR2423 at the end of these release notes for important installation information.

## 1 New Features

### SCR2373: Add Ability to Set Break Points

A user can now load a breakpoint description file into gedae to break on primitive and queue events. If the graph is located in:

FGlibraries/boxes/<pathname>

(for example FGlibraries/boxes/demo/comm/e\_comm)

Then the breakpoint files must be found in the FGlibraries directory (\$FPATH on Windows):

FGlibraries/breakpoints/<pathname>

(for example FGlibraies/breakpoints/demo/comm./e\_comm/bkpts)

Each line of the breakpoint file describes a breakpoint. Primitive breakpoints are described as:

```
box=<boxname> [type=<event type>] [fired=<range specifier>]  
[exec=<range_specfier>] [granularity=<range_specifier>]
```

Where

- <boxname> is the hierarchical name of the box element in the graph (not including the toplevel graph name).
- type is the type of a box event
- <event\_type> is one of
  - begin – event occurring immediately before a primitive executes
  - apply – event occurring after a primitive executes its Apply method
  - reset – event occurring after a primitive executes its Reset method
  - eos – event occurring after a primitive executes its EndOfSegment method
- fired is the total number of times the box has fired (accumulation of granularity for all executions of the box)
- exec is the total number of times the box's Apply method has been called
- granularity is the granularity at which the box is run

We define the <range\_specifier> by example

<b>fired=&lt;range_specifier&gt;</b>	<b>interpretation</b>
fired=10	fired=10
fired=(100,200)	100<fired<200
fired=[100,200)	100<=fired<200
fired=(100,200]	100<fired<=200
fired=[100,200]	100<=fired<=200
fired=(100,...)	100<fired
fired=(...,100)	fired<100
fired=[100,...)	100<=fired
fired=(...,100]	fired<=100

Breakpoints can be set on user events as:

box=<boxname> type=user [number=<range\_specifier>] [value=<range\_specifier>]

Where

- number is the number passed as the first parameter to embUserEvent, embEndUserEvent, embUserIntEvent or embUserFloatEvent
- value is the value passed as the second parameter to embUserIntEvent or embUserFloatEvent

The second major breakpoint type is the queue breakpoint. A queue breakpoint has the form:

queue=<queuename> [type=<event\_type>] [tokens=<range\_specifier>]  
[total\_produced=<range\_specifier>] [total\_consumed=<range\_specifier>]

Where

- tokens are the number of valid tokens in the queue
- total\_produced are the total number of tokens that have been produced into the queue since the application started
- total\_consumed are the total number of tokens that have been consumed from the queue since the application started

An example breakpoint description file for the graph demo/comm./e\_comm is shown below:

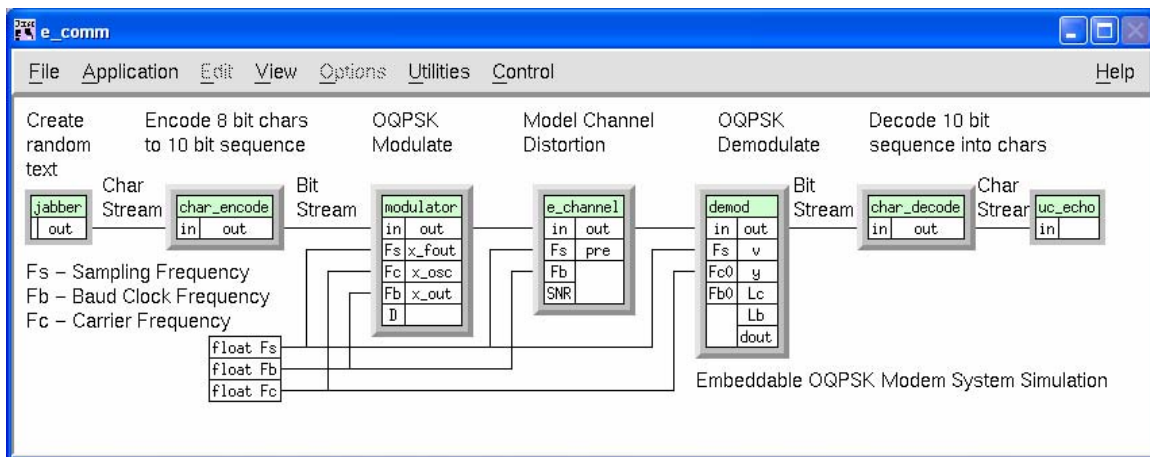
```
box=jabber type=begin fired=0
box=demod.demod2 type=reset
box=e_channel.norm_noise type=begin fired =0
box=modulator.fft_filter type=begin exec=[10,15]
box=e_channel.norm_noise type=apply fired=[40000,50000]
box=demod.decide type=apply fired=[10000,10020]
```

```

queue=modulator.fft_filter<in type=produce total_produced=[100000,150000]
queue=modulator.fft_filter<in type=consume total_consumed=[150000,200000]
queue=demod.decide<in type=consume total_consumed=[13000,13020]
box=uc_echo type=begin fired=[10000,10500] granularity=[1,50]
box=uc_echo type=begin fired=[10000,10500] granularity=[51,100]
box=uc_echo type=begin fired=[10000,10500] granularity=[101,150]
box=uc_echo type=begin fired=[10000,10500] granularity=[151,200]
box=uc_echo type=begin fired=[10000,10500] granularity=[201,256]

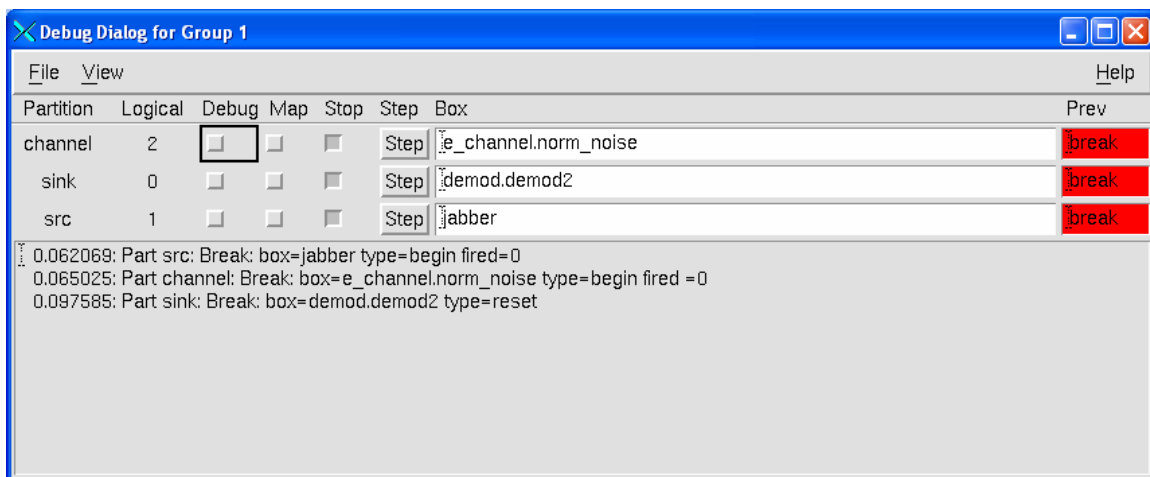
```

The e\_comm graph is illustrated below



**Note:** Breakpoints only take effect when event tracing is enabled.

Running the e\_comm demo with the above breakpoint file, with Trace enabled, and using a partitioning and mapping that maps the graph to 3 processors causes the breakpoint dialog to pop up as below:



At this point all three partitions have stopped at different breakpoints and the breakpoints where they stopped are listed in the text area at the bottom of the dialog. A user can at

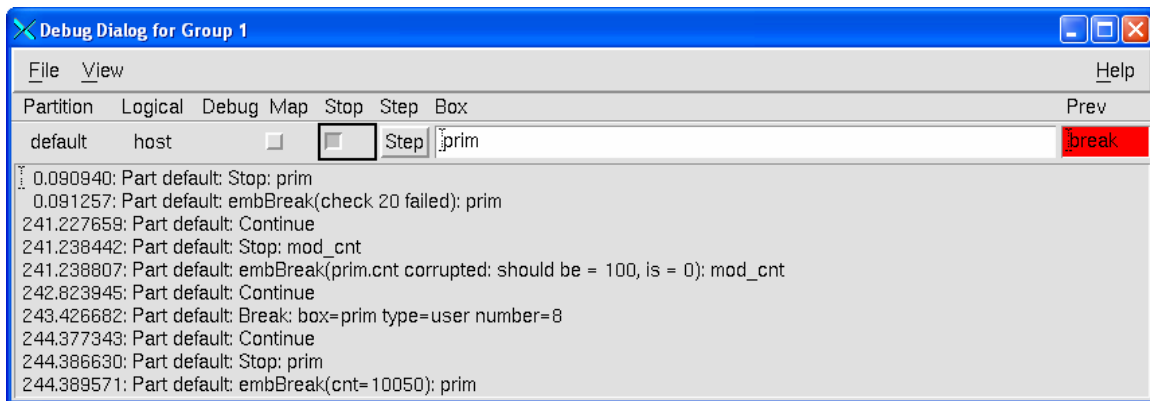
this point start any partition by unsetting the Stop toggle for that partition. A user can single step through the primitives in the partition. And a user can bring up the Structure display for the primitives in the graph and examine input and output data values.

## Function embBreak

A user can force Gedae to invoke a breakpoint from a primitive by calling embBreak. For example an Apply method can monitor a variable cnt and force a break as

```
Apply: {  
  ...  
  if (cnt == 10050) {  
    embBreak("cnt=10050");  
  }  
  cnt++;  
  ...  
}
```

Such programmed breakpoints can allow the user to force a breakpoint using more complicated conditional expressions than are available in the Gedae breakpoint file description language. It is anticipated that the embBreak function will be used for debug purposes only. When an embBreak point is hit the debug dialog reports it as:



The above dialog shows several breakpoint occurrences. The bottom two lines shows that primitive "prim" has called the function embBreak with the text string "cnt=10050".

## SCR2397: Support Hexadecimal and Octal numbers in Canvas Data

The Gedae Flow Graph Editor was extended to allow specification of hex and octal data on the canvas using the 0x prefix for hex and the 0 prefix for octal. E.G., "int A = 037" evaluates to 31 decimal and "int B = 0xfe" evaluates to 254 decimal and "int C = 0xfe-1" evaluates to 253 decimal

## SCR2403: User registered monitoring functions

A new capability to allow a user to register a function from a primitive Start method that will be called after every primitive firing has been created. The new capability was designed to allow users to monitor Local primitive variables to ensure that they are not being modified outside of the primitive. Other uses of the capability however would be possible. The user registers the function by calling `embRegisterDbgMonitor(void (*dbg_monitor)(void *data,int isme),void *data)`; Where `dbg_monitor` is the function that gets called after every primitive firing. The function takes two parameters. The first is a pointer to the data that is registered as the second parameter. The `isme` parameter is a Boolean variable that indicates if the `dbg_monitor` is being called after the registering primitive or after another primitive. The `isme` parameter allows the monitor function to note the last value of a state variable when the registering primitive was called and report a state change as an error when a nonregistering primitive was called.

One way to use the `dbg_monitor` function is to call `embBreak` if the value being monitored was changed outside of the primitive. For example if a primitive has a local variable `cnt` declared as:

```
Local: {
    int cnt;
}
```

Then the variable can be monitored by the `checkCnt` function as:

```
Include: {
typedef struct {
    int *state_cnt;
    int last_cnt;
} CheckRec;

void checkCnt(void *data, int isme) {
    CheckRec *cr = data;
    /* check to see if cnt is corrupted */
    if (!isme && *cr->state_cnt != cr->last_cnt) {
        char message[1000];
        sprintf(message,"prim.cnt corrupted: should be = %d, is = %d",
                cr->last_cnt,*cr->state_cnt);
        embBreak(message);
    }
    cr->last_cnt = *cr->state_cnt;
}
}
```

Where the `CheckRec` structure is allocated by the Start method and the pointer to count is registered as:

```
Start: {
    CheckRec *cr = embCalloc("default",1,sizeof(CheckRec));
    first = 1;
    cr->state_cnt = _state->cnt;
    cr->last_cnt = cnt;
    embRegisterDbgMonitor(checkCnt,cr);
}
```

```
}
```

Note in the above example the pointer to the Local variable cnt is obtained by using `_state->cnt`.

### SCR2407: Add exit code to exitFGE

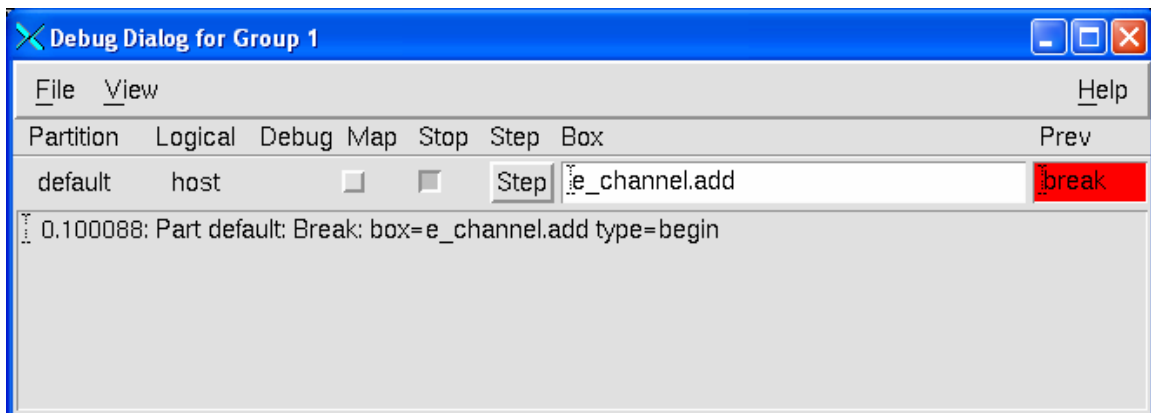
The function `exitFGE` now has the prototype `void exitFGE(int error_code)`; The `error_code` parameter is passed through to the `exit()` function and returned as an error code from the executable.

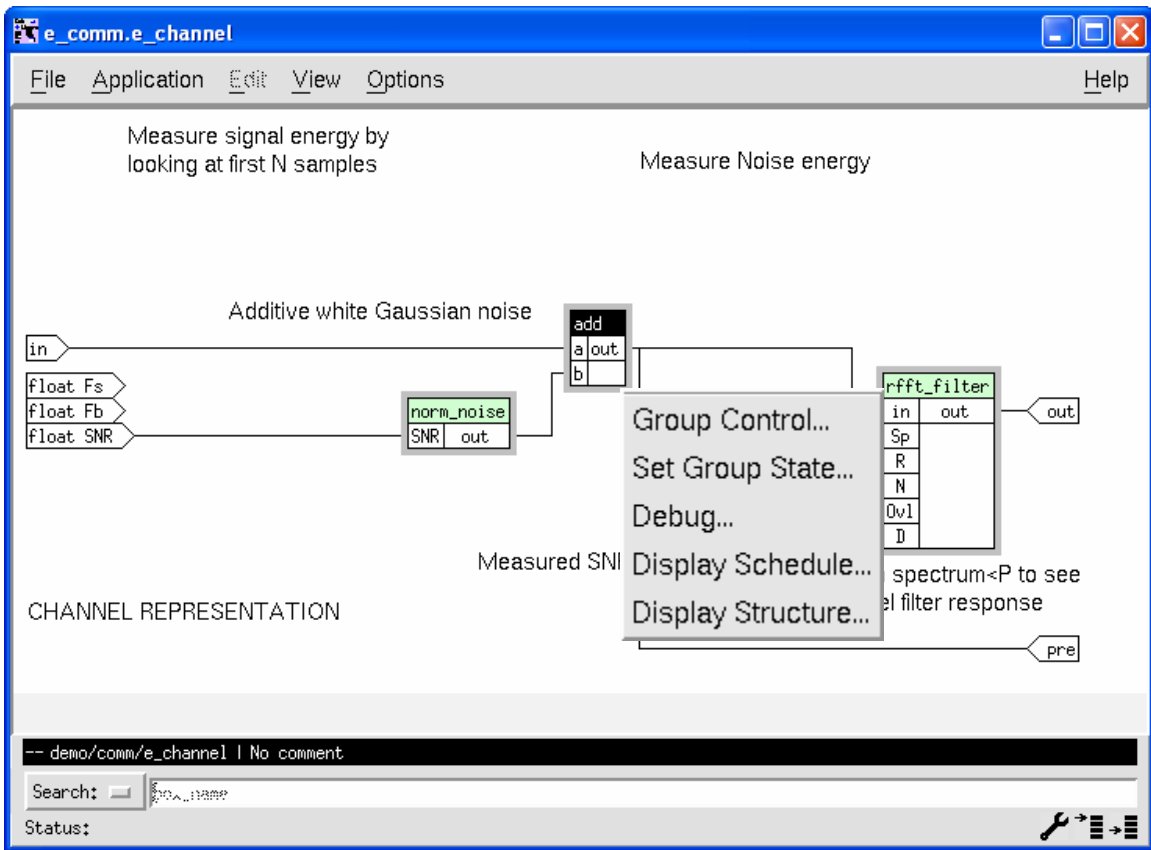
### SCR2412: Add ability to dump memory from memory display structure.

Primitive stream input and output values can now be dumped to a file from the Gedae structure display for that primitive. For example if we run the `demo/comm./e_comm` graph using the following breakpoints file:

```
box=e_channel.add type=begin  
box=e_channel.add type=apply
```

Then when the graph is started (with trace enabled) the debug dialog pops up as:





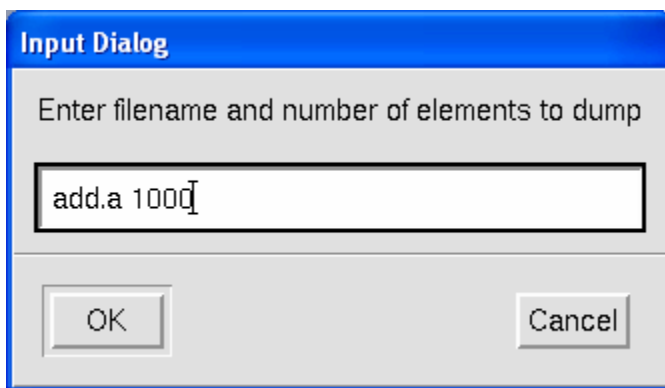
Right clicking on the add box allows the user to select “Display Structure” bringing up the display of the add boxes state structure:

Name	Value
embeddable/stream/add	...
int granularity	5376
float *a	0x01809C18; -0,0243587
int N_a	5376
float *b	0x01804818; 0,0141357

This structure shows the value of input streams a and b before the box fires. We can right click on the a input and select the Dump To File item from the popup menu:

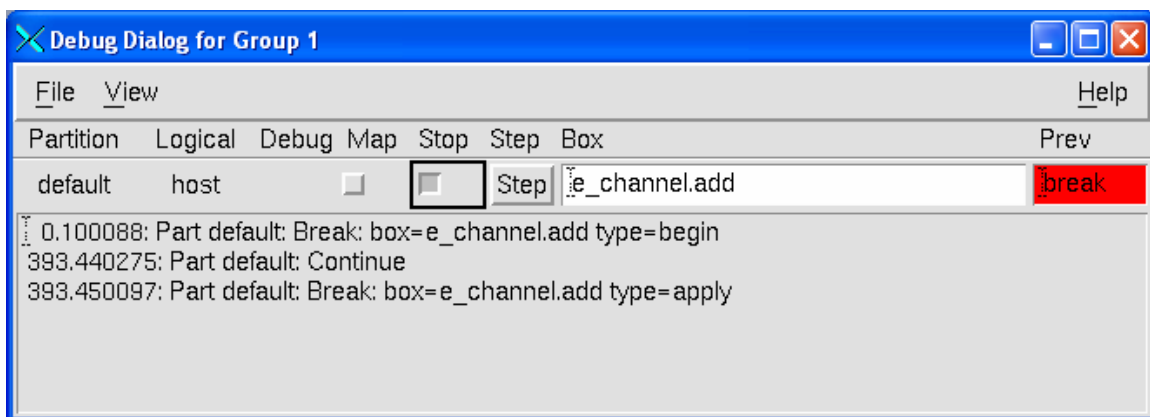
- Expand
- Collapse
- Hide
- Show All Boxes
- Expand Struct
- Expand Immediate
- Dump To File

We can then enter the name of the file to dump the data to and the maximum number of values to dump:

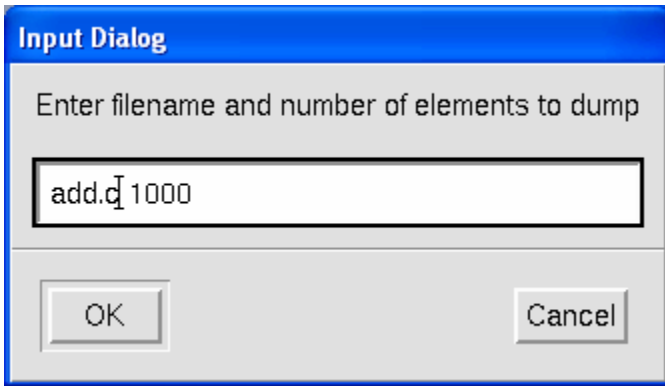


Similarly we can dump 1000 values from the b input to file add.b

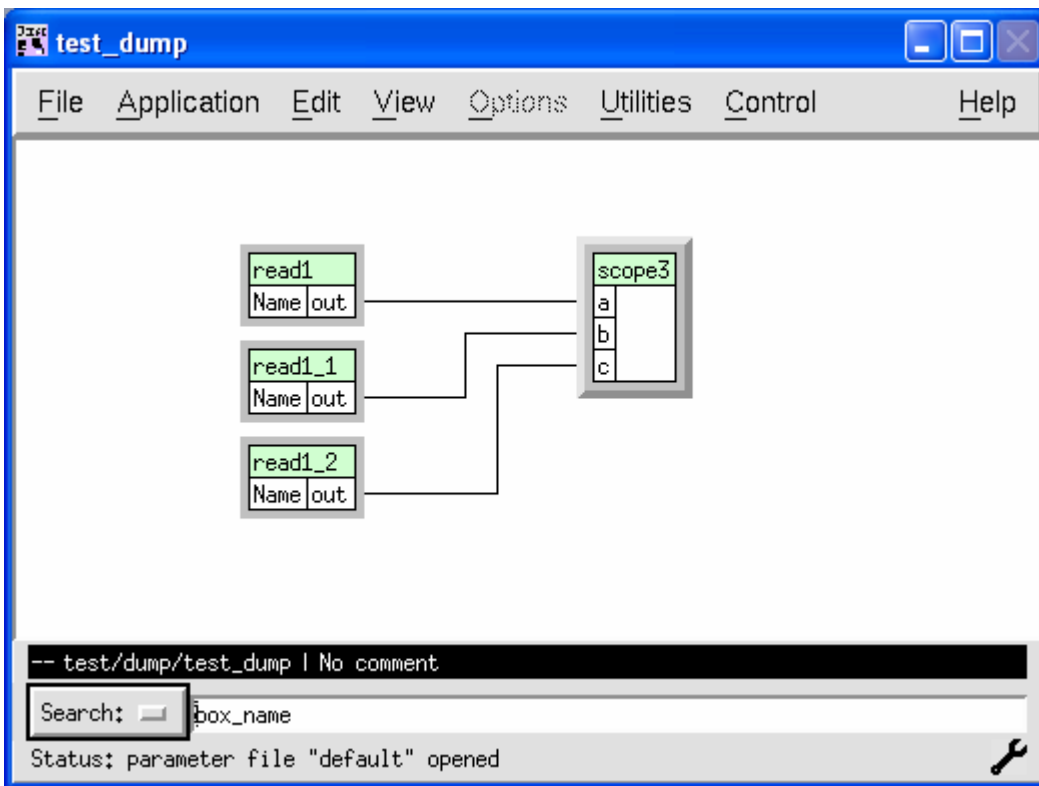
At this point we hit the Stop toggle on the debug dialog and the graph proceeds to the next breakpoint



This is the breakpoint that occurs after the box has fired. Since the a input is in place with the output the structure display now shows the output value of the add box in the a input. We select the a input and dump it to file add.c as:



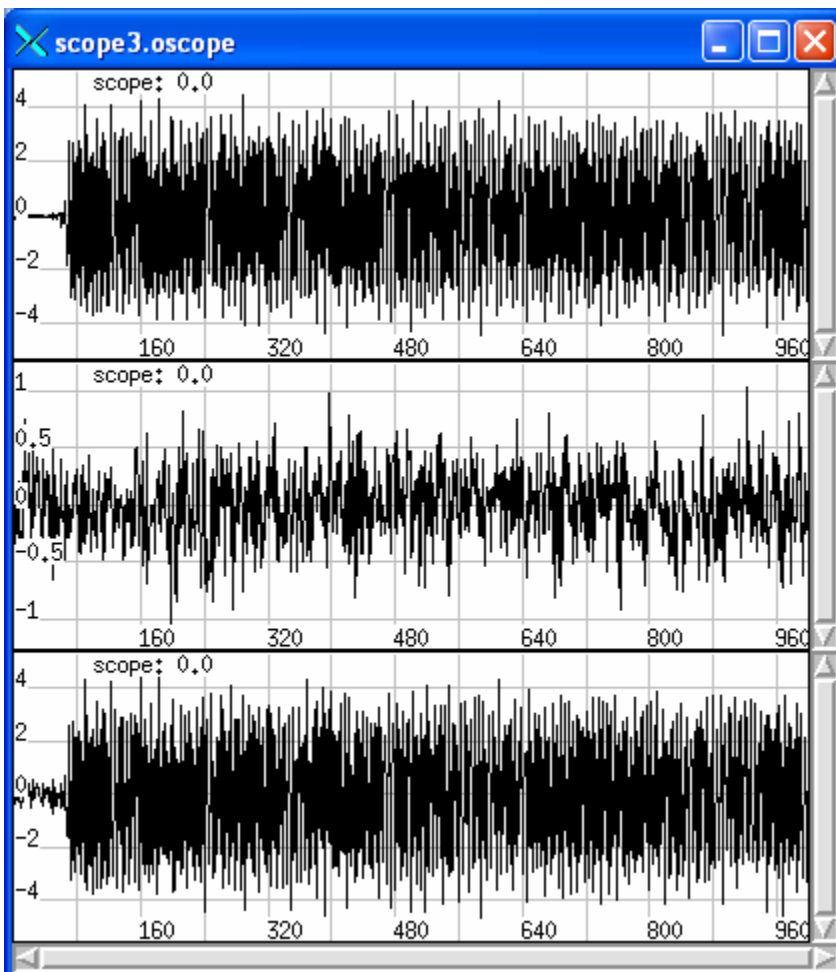
At this point the input values to the add box should be stored in files add.a, add.b and the output value should be stored in add.c. We can verify that the add box has executed correctly using the following graph:



With the parameters to the file read boxes set to read the files add.a, add.b and add.c.

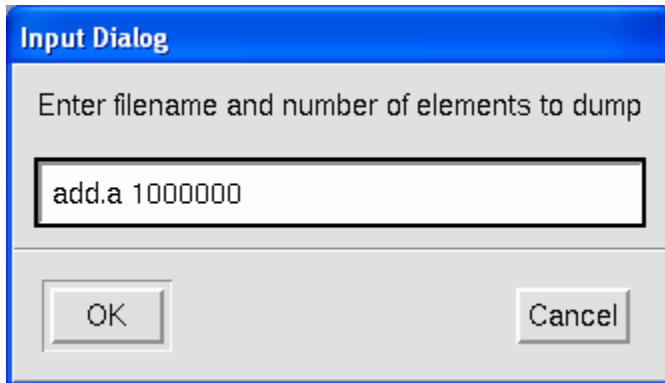
Name	Value
char Name[6]	add.a
read1	
char Name[6]	add.b
read1_1	
char Name[6]	add.c
read1_2	
scope3	

Running this graph displays the results of adding input a to be to produce c:

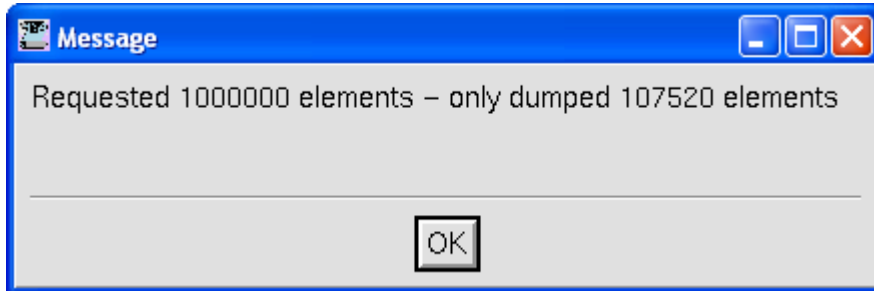


Typically the user can use the granularity of the primitive to determine how many values to dump. If more values than can be safely read from the memory address are requested

the user is informed of how many values were actually dumped. For example requesting the values from the a input as:



The user is informed:



This is a preliminary capability. We plan to use this capability as a starting point for being able to display or save any data values in the graph while the graph is running.

### **SCR2331: Get configuration control status of all boxes in graph**

The menu item File->Status All has been added to get the status of all boxes in the graph from the configuration management tool.

## 2 Bugs Fixed

### **Case 1148: VxWorks `embWallclock` Function Misses Wrap**

When collecting trace information, the `embWallclock` function timer can wrap without being detected. This failure to detect the timer wrap causes VxWorks processor timelines to appear compressed. This problem has been fixed.

### **Case 1239: Static Schedule in Segmented Subgraph Not in Scope of Segment Controller Asserts**

An assert error is caused by a static schedule in a segmented subgraph that has no input in the scope of the segment controller. For example, a constant box driving a merge input in the subgraph can cause the error. The error manifests itself as an assert as shown below:

```
Error: assert(dq->eos[inptr].begin_level > 0) failed.  
File: ../../source/segment.c, Line: 935
```

The fix for this problem is that the user is notified of this illegal graph during graph development.

### **SCR2314 Launch Info Dialog Make button does not work correctly**

If a group is set to run with a target command program the launch package Make button does not produce a launch package that has the target command program specified. The problem was fixed so that when the group is set to make a target command program the Make button now creates the launch package the same way the run button does. When the group is set not create a target command program the Make causes a command program to be built as was done in previous versions of Gedae.

### **SCR2383: Allocation Error during application load phase causes protocol error**

If during the application load phase (downloading of the application description to the target processor) the target processor cannot allocate the requested amount of memory instead of reporting the error to the host a protocol error occurs.

### **SCR2392: Group Control Dialog Tables should be deleted when entering dirty state.**

If a group control dialog table such as the Partition Table, Mapping Table or Queue Table is popped up and the graph is edited (for example a box is deleted) then the tables should

all be destroyed. Because they stay up if the user tries to interact with them Gedae segfaults as the Group that the tables referenced is no longer valid.

### **SCR2388: Prevent recursive calling of X event handler**

Recursive calls to the X Event Handler function need to be prevented. Such recursive calls have been observed when setting the priority of a primitive from the Firing Granularity Table after the Gedae graph has been run. Preventing the recursive call has avoided this problem.

### **SCR2390: Problem with DSA on Champ AVIV**

Data that is written by the sender is getting overwritten by the receiver with the old values that are still in cache. This occurs because on the AVIV (as opposed to the AVII) the cacheInvalidate flushes the cache in addition to doing the invalidation. The problem was fixed by invalidating the cache before the receiver gives the sender permission to send a new message (during the recvDone primitive's execution).

### **SCR2385: e\_dvasm.c ramps scalar value when stride is 1**

In the stride 1 branch of e\_dvasm.c, c is incremented accidentally (as if it were a pointer).

### **SCR2372: Setting partition of graph containing a typedef segfaults**

Setting the partition of a graph that contains a typedef box causes Gedae to segfault.

### **SCR2402: Stack overflow in gedae when creating compile script with long commands**

If Gedae tries to generate compile command lines over 1000 characters in length then Gedae will crash with a stack overflow. This problem was fixed by increasing the limit to 10000 characters. Also a warning is printed if the compile line exceeds 5000 characters.

### **SCR2404: Undefined data types streams between host and target with different endians**

If the host and target processors have a different endian then the transfer of data types that are not one of char, short, int or float do not get converted correctly. To provide a partial solution to this problem Gedae has been modified to use the integer byte swapping routines for transferring unknown data types. This method will work as long as all 32 bit and 64 bit words 32 bit byte reversed and the data structures being transferred contain no 8 or 16 bit types (chars and shorts).

### **SCR2384: Redo exclusive lock counter**

The exclusive lock counter that prevents one exclusive branch from firing until the preceding exclusive branch has finished was not working correctly. Subgraphs driven by multiple exclusive outputs and having more than one static schedule at the input interface could get into a state where the lock counter erroneously went negative. A simplified lock counter has been developed that keeps one lock count for the entire exclusive set member

rather than a different lock count for each exclusive queue. This new method avoids the problem.

### **SCR2389: Multiply defined symbols during launch package creation**

When Gedae builds the script for incrementally linking the library used to create the host process for a command program in adds some of the object files to the library twice. For some BSPs this double addition of the objects to the library causes the final link to fail with a multiply defined symbol error.

### **SCR2405: Exclusive queue with different number of destinations on some family members**

Graphs having exclusive queues that fan-out to multiple destinations on some family indices and to fewer paths on others can cause a segfault.

### **SCR2374: Remove perl distribution for non-Windows versions**

We include perl in all our releases. Sometimes our perl conflicts with the Perl installed on the system, preventing successful installation. Remove the perl from the Unix versions of Gedae and edit the build scripts to use perl from the path instead of absolute path to the system directories.

### **SCR2417: Target Command Programs no longer work**

Command programs created by marking the CP column in the mapping table no longer work.

### **SCR2396: Change Editor to Kill toplevel graph during File->Open or File->New**

The editor was modified to Kill the toplevel graph (erase all memory of it including changes) when doing a File->Open, a File->New or opening one of the previously opened graphs. If the existing toplevel graph has been modified the user is notified about the modification and asked if the operation should continue. Prior to this change any edits to subgraphs common to the original graph and the recently opened graph caused Gedae to segfault.

### **SCR2382: Arithmetic overflow when calculating firing granularity of dynamically dependent box**

Dynamically dependent primitives running at large natural granularities can generate an arithmetic overflow when calculating the dependent primitives firing granularity.

### **SCR2408: gedae\_library\_serial.dat file out of sync**

The code generation of boxes and of the executables that use those boxes both depend on the gedae\_library\_serial.data file which assigns numbers to the directory paths in which the boxes are stored. The intention is that only the development environment should write to the file while gen reads it. However, it appeared that the file was not being properly

flushed by the development environment meaning that the gen function was using different serial numbers than the development environment. The effect of this was that there were undefined symbols in the generated executables after a fresh makeGEDAE CLEAN.

### **SCR2411: Graphs with Exclusion cannot be run twice**

Graphs having exclusive queues cannot be run twice. That is if the user executes the following sequence: Control->Run followed by Control->Terminate followed by Control->Run, then the graph stalls on the second invocation of Control->Run.

### **SCR2413: Complex data values not transferred to host correctly from different endian target**

Values of type complex are not transferred correctly from a target processor to a host processor if the endians of the two processor types differ. This error can be observed in the Gedae Structure Display for elements in the display that are complex pointer values. The error was observed on a Windows platform controlling a ChampAVIV but would be observed on any platforms that required floating point conversions between the target and host.

### **SCR2416: "Create Subgraph" when output connected to route box**

When the user creates a subgraph and an output of the subgraph is connected to a route box, Gedae can segfault.

### **SCR2415: The Gedae -launch command line parameter ignores launchInfo dialog group settings.**

Generating an application as: `gedae -file demo/comm/e_comm -pa default -gr embedded_cp -conf test_ent -launch launch/e_comm_cp` Ignores any of the group settings (like directory name) set from the launch package group control dialog.

### **SCR2418: Segfault when saving group settings with empty string on Linux**

If you attempt to save group settings with an empty string, Gedae will segfault on Linux while doing a strcmp with the string.

## **3 Known Bugs**

### **Case 1024: Accommodate Different Exceed Versions in initGEDAE**

In order for Gedae to work with Exceed/XDK 9.0, makeGEDAE has to be modified.

Currently users must view the FAQ for 7.1 changes  
[http://www.gedae.com/SUPPORT/FAQ/exceed\\_7\\_1.html](http://www.gedae.com/SUPPORT/FAQ/exceed_7_1.html)  
and then remove xlibcon from makeGEDAE and ent files mentioned in the FAQ.

## **Case 1026: Broadcast Transfer Mechanisms**

Currently all Gedae data communication is point to point; however, many target processors support efficient broadcast mechanisms. The goal of this task is to extend the Gedae BSP API to allow broadcast data transfers to be part of the BSP. A second goal is to fix the DSA mechanisms that cannot be implemented when there is fan-out (see Case 1149). Rather than fixing that problem directly, a broadcast DSA capability will be implemented.

## **Case 1056: FGU of Hierarchical Typedef Boxes**

FGU does not transfer hierarchical typedef boxes correctly. The `typedef` used to define the input of the box is set to the old directory rather than the new.

## **Case 1057: Graph Stalls**

A rare condition can cause a graph to stall (or segfault) when the controlled static schedule is partitioned to two processors in the following form:

A->B->A

The problem scenario is that the schedule is partitioned into three parts, with the first and last parts mapped to the same processor. Usually Gedae puts the parts mapped to processor A in the same static schedule; however, to allow efficient pipelining, Gedae splits the two parts mapped to processor A into two different static schedules. They are numbered n.1 and n.2 (for example 2.1 and 2.2). To see if any schedules have been broken into two parts, the user can pop up the Schedule Info Dialog and see if any of the schedule names contain a decimal point.

The decimal point in the schedule name does not necessarily indicate a problem. The problem only occurs when the data source driving the processing is faster than the graph, causing the control message queue to back up and overflow. The condition is rare because the problem only happens when the graph is not keeping up with the input data rates.

## **Case 1078: Gedae Overwrites Protected Files**

Gedae overwrites files that are protected as a result of being checked out of a CMS. This overwriting causes a problem because it defeats the CMS system.

### **Case 1108: Setting Default Subschedule Queue Capacities Correctly**

The queue policy and capacity of a queue feeding a subschedule may not be set correctly.

### **Case 1122: Embedded Build Can Require a `makeGEDAE CLEAN`**

If an application is repartitioned, then the target executables do not get relinked. The problem is that all the .o files are older than the targets, and the fact that there is a new link line does not force the target library and target executables to rebuild.

### **Case 1139: Unterminated Comments**

Unterminated comments cause the Gedae parser to segfault.

### **Case 1140: Parser Problem**

The Gedae parser does not handle an odd number of quotes (") well.

### **Case 1142: Arrays of Strings Not Allowed**

Gedae currently allows string array graph parameters to be declared as:

```
const string X[] = {"hello", "world"}
```

or

```
string X[i] = [i]Y
```

where Y is a family of strings.

In either case, the values so declared are not correctly set, and therefore, should be considered illegal.

## **Case 1143: Inconsistent Data Type Declarations**

Gedae aborts when the same stream type is multiply defined. For example, suppose two primitives define data types with the same name but different definitions. If the primitives use these types in their `Input`, `Local` or `Output` sections, then Gedae aborts.

## **Case 1144: Function `appFree` Memory Leak**

A command program running on VxWorks does not free all the resources allocated (memory, sockets, etc). The `appFree` function must release everything allocated. Gedae should automatically generate a call to `appFree` for the standard `exec-host` command program.

## **Case 1145: External Code Does Not Recompile**

`Make` is not called after a successful run, so changes to code listed in the `Personal_Emb_Obj_List` do not get recompiled. To force the recompile, it is currently necessary to change something from the Gedae GUI – such as, saving a primitive or toggling the Group "Run on Embedded" toggle off and on.

## **Case 1146: Large Graphs Fail to Display on Flattened Graph**

If a graph is too large, then it cannot be displayed on the flattened graph. That is, if the flattened width or height exceeds the allowable pixmap width or height.

## **Case 1147: Primitive Cannot Recompile**

If a primitive `Input`, `Output` or `Local` section is modified at runtime, then Gedae segfaults when the primitive is recompiled, and the graph is rerun. Currently, the user must exit Gedae after a primitive `Input`, `Output` or `Local` section has been modified.

## **Case 1149: DSA with Fan-out Does Not Work for Some BSPs**

If a box output fans-out to several boxes mapped to several different processors, then the DSA communication mechanism does not work correctly for Mercury and Sky BSPs.

## **Case 1150: Trace Table Saved on NT Not Readable on Solaris**

Files saved from NT are byte reversed from what is expected on Solaris. Files saved on a big-endian platform cannot be read by Gedae from a little-endian platform.

## **Case 1151: FFT Primitives Only Work with Power of 2 Sized Vectors**

The FFT boxes do not support non-power-of-2 lengths; however, the comments make no mention of this fact. If these boxes only support a power-of-2, then it would be useful to have a separate set of boxes that support a non-power-of-2.

## **Case 1155: Constants Propagated Through typedef Boxes**

Constants propagated through typedef boxes cannot be used for instantiation.

## **Case 1159: Stream Box with `push` in Hostless Launch Package**

If a stream box contains a call to `push` and it is made part of a hostless launch package, then the launch package will fail to compile, as the code for the `push` is not included in the standalone library.

## **Case 1249: Transferring Doubles between Host and Target with Different Endian (Dy4av2)**

Currently, the Gedae BSP does not support providing byte swapping of doubles when transferred between the host and target processors.

## **SCR2012: Running Two VxWorks Processes on the Same Processor**

This problem occurs when trying to run two separate Gedae generated VxWorks executables on the same processor; however, the entry point for each executable has the same name, `VxWorks_main`, making this impossible.

## **SCR2019: Graphs with Host to Target Control Ports Fail on Linux and Solaris**

The problem is that the host is not performing mail box services while it is waiting to establish a control port to the target processors. Unix processors require the host to be performing these services in order to make connections.

### **SCR2022: Inplace Box Scheduling Problems**

Copy boxes occasionally need to be added to a graph by the user to avoid scheduling problems associated with primitive outputs marked as being inplace with an input.

### **SCR2024: Connecting Graph Parameter to User Define Type Segfaults**

Gedae erroneously allows standard C parameter types to be connected to user defined parameter types. This type of connection causes Gedae to segfault.

### **SCR2025: Undefined Initialization Function Symbols**

Occasionally when linking a target executable, the linker complains about undefined function names such as `I0002_add`. A workaround for this problem is to do a `makGEDAE CLEAN` followed by a `makeGEDAE`.

### **SCR2047: Allow Changing Length of Parameter String at Runtime**

Changing the value of a string parameter to a stream may cause Gedae to stop executing. Gedae stops executing if the length of the string is changed.

### **SCR2054: Modifying a Running Graph Segfaults**

Modifying a Gedae graph that is running can cause a segfault. This problem has been reported several times but has not been duplicated by the Gedae support group. Most edits are disabled during graph execution.

### **SCR2062: Outlaw Segmented Static Schedules Controlled by Nondet Inputs with Multiple Boxes**

Segmented static schedules controlled exclusively by nondet inputs and that contain more than one box should be outlawed. These graphs are currently considered problematic and can produce unexpected results.

### **SCR2064: Outlaw Pointer Streams Followed by Delays or Overlap**

Pointer streams followed by delay boxes or boxes with input overlap parameters do not work and should be outlawed.

### **SCR2065: Pointer Stream Followed by an Inplace Box**

A pointer stream cannot be followed by an inplace box that has an Apply method because the execution of the box will modify the pointer's value. A workaround is to add a copy box between the pointer and the inplace box.

### **SCR2078: Gedae Can Go to Sleep if Processes Are Polling**

A Gedae process can go to sleep if a process is polling. The sleeping is only seen on the NT BSP, which is currently the only BSP that implements the sleep capability. Gedae should only sleep when all schedules are in the paused state.

### **SCR2104 Handling of Null Segments in Distributed Graphs**

If a segmenter controls a segmented subgraph that is distributed and if the segmenter produces null segments, then parts of the distributed graph that are not directly controlled by queues will not see the end-of-segment.

### **SCR2113 Failed Launch Package Creation**

The "failure making launch package" error is received when an absolute path is entered in the directory field of the Launch Info Dialog.

### **SCR2116 Segfault when "Map Parts" Fields Are Navigated While the Graph Is Running**

Gedae segfaults if the Map Parts Dialog fields are navigated while the graph is running.

### **SCR2131 Improve Flattened Graph Placement**

After expanding the flattened graph, reduce the gaps between the boxes. There is currently a great deal of extra white space in the expanded version of the flattened graph.

### **SCR2157: Need to Turn on Launch Package Tracing**

Launch packages by default start with tracing turned off. Attaching the development environment to the running application may turn the tracing on but any events before attaching are missed.

## **SCR2158: Trace Table send/recv Webs Do Not Work from an Attached Launch Package**

When the Gedae Development Environment attaches to a launch package, the Trace Table send/recv webs do not work. No send/recv webs are displayed.

## **SCR2221: Search on type doesn't follow route boxes**

If you search on the type of the input to a scope1, it doesn't recognize the type. This is because there is a route box between the input and the next primitive.

## **SCR2244: Using f6 to disconnect a constant can cause a segfault.**

Disconnecting a constant source from a constant destination using the f6 button causes a segfault. This because a stub is now connected to the constant destination and the evaluation of the constant fails.

## **SCR2245: Gedae does not prompt user to save graph**

If a graph has been modified using the f6 cursor and that was the only modification then Gedae does not prompt the user to save the graph when exiting Gedae.

## **SCR2261: embTerminateError called from Reset doesn't stop primitive**

If embTerminateNormal or embTerminateError are called from a Reset method of a graph that has more than one static schedule, then the static schedule containing the primitive is not terminated as it should be. The schedule Apply methods will be called even though one of the schedules Reset's failed.

## **SCR2267: Non Family output Connected to Family Input causes Graph to Crash**

If a non-family stream source is connected to a family stream input gedae segfaults when the user runs the graph. This type of graph should be detected as an error and Gedae should not be allowed to run.

## **SCR2282: Limitations on the Use of Typedef Primitives**

Typedef Primitives will not work if connected to variable vectors or matrices. Typedef primitives will not work if connected to route boxes.

## **SCR2302: MCP3 Parameter Support**

MCP3 BSP does not support parameters yet.

### **SCR2303: MCP3 Granularity Support**

MCP3 BSP does not support granularity for FPGA->PPC comm yet.

### **SCR2312: The Gedae dy4av2 BSP function e\_zvrcip can kill process**

Calling the Gedae Dy4av2 BSP function e\_zvrcip can cause the process to exit with an arithmetic exception. The problem is that even if the number whose reciprocal is being taken has real and imaginary parts not both zero a divide by zero error still occurs.

### **SCR2349: FGU does not work with long pathnames**

When moving a FGU tar from Solaris to Windows, the Windows system cannot reproduce long filenames. This seems to be a problem with the Windows tar executable.

### **SCR2360: Trigger boxes with Reset method but no input parameters don't get fired**

A trigger box that has a Reset method but no other inputs does not get fired when the user selects Control->Run (but does on a Control->Reset).

### **SCR2361: Eval boxes with no inputs are not included in launch package**

This feature can be confusing as when the Eval box calls getcwd. In this case the launch package will use the wd from which it was created - not the one it is run from...

### **SCR2371: Series of Create Subgraph, Save As, Undo causes segfault**

You can open MotionSimulation, make osc\_1, addK subK and sqr a subgraph named xxx, save the graph xxx as yyy, go back to MotionSimulation and undo the change.

### **SCR2378: Create Subgraph does not support multidimensional families**

Create Subgraph cannot handle boxes and data that use multidimensional families.

### **SCR2391: Primitives using allocArray should free with freeArray**

All primitives that use allocArray should be checked to see that in their Destroy or Terminate methods they free the arrays using freeArray. Doing nothing about freeing can cause memory leaks during development at runtime on VxWorks type systems. Freeing with free instead of freeArray can cause segfaults.

### **SCR2399: Gedae built in function names should be ignored if used in non-function context**

Gedae built in functions like "time" should be ignored if not used as a function. That is time(x) should be converted by the gedae parser but x.time (or even x.time(x)) should not.

### **SCR2400: Family of array parameters to a trigger box does not codegen right**

An input parameter like float [N]in[Max] does not codegen correctly for trigger boxes. The data values are not correct. If only one family member is set, then the box still executes however the array dimensions are invalid.

### **SCR2406: Primitive with EndOfSegment but no Apply does not work.**

If a primitive has an EndOfSegment method but no Apply method the primitive is not included in the running application so the EndOfSegment method is never called.

### **SCR2419: FGTable edit box does not recognize termination**

Gedae does not accept some FGTable entry while the graph is running. If you are trying to edit a FGTable entry, then terminate the graph so you can change the value, the FGTables do not automatically recognize the graph has terminated. The user must unselect the entry area then reselect it to enter the data.

### **SCR2420: Microphones sporadically stop collecting data on AFG for Linux**

The Audio For Gedae (AFG) input primitives sometimes stop getting data from the microphones.

### **SCR2422: Edit primitive in \$EDITOR when syntax error on load**

If there is a syntax error when loading a primitive, bring it up in the \$EDITOR or primitive editor so it can be fixed.

### **SCR2423: Dy4Av2 Installation Problem**

We needed to remove the files rsh.exe and cygwin1.dll from %gedae%\nt\bin because delivering these files and placing %gedae%\nt\bin in the users PATH variable caused many user to have conflicting versions of cygwin1.dll in their path. It is now necessary to copy cygwin1.dll and rsh.exe from the cygwin bin directory into %gedae%\nt\bin. For example these files can be copied as:

```
copy c:\cygwin\bin\cygwin1.dll %gedae%\nt\bin
copy c:\cygwin\bin\rsh.exe %gedae%\nt\bin
```