Eggplant for Mac
Release Notes

The notes below provide descriptions of the new features and changes introduced with each release of Eggplant. You are strongly encouraged to read about the relevant changes whenever you upgrade from an earlier version.

Potential compatibility issues are highlighted in **RED**.

Release v11.01 (17-August-2011)

Bug Fixes / Tweaks:

- Added support for UltraVNC 1.09 connections.
- Fixed a potential crash when setting the searchRectangle.

Release v11.0 (15-August-2011)

Highlights:

- Added a multi-purpose **ReadText()** function to read text from any screen using OCR.
- Added the ability to search all scripts within a suite.
- Added the ability to easily **repeat** a script block for a given length of time, or until a specified time.
- There is now a generic Text Search engine for finding text strings on the screen.
  If you have NOT modified your text preferences this is now the **default** Text Finding Engine.
- Added support for Mac OS X 10.7 (LION)

User Interface:

- Added a Search Field for Scripts, allowing you to search script names or the full content of every script in the suite.

- Added a "Last Run" column on the Scripts pane of the Suite window, that shows the time when each script was last run. The time for a successful run will be shown in green; for a failed run in red. Clicking the last run time will show the results for that script run in the Results pane.

- Added a connection type pop-up to the VNC Connection panel that allows specifying explicitly which protocol to use for a VNC Connection.

- Enabled editing of script names directly in the Scripts tab. Simply select a script then click its name a second time to begin editing. Double-clicking quickly will still open the script.

- Eggplant will no longer allow you to create two images of the same name in the same folder, regardless of what file type extensions they use.

Scripting:

- Changed the initial setting of **the defaultLineDelimiter** to be the set of all common line endings (CRLF, LF,

CR, LineSeparator, ParagraphSeparator) rather than just the Return (LF) character.

- Added **ReadText()** function which takes one or two parameters. The first (required) parameter is either a rectangle specifying the area of the remote screen from which text is to be read, or a point or image indicating the location of some text to read. The function returns whatever text was recognized from that part of the screen. In the case of a rectangle, it will read all text contained within that rectangle. In the case of a point, it will attempt to locate text near that point, and will read and return as much text as it can find adjacent to that text on the same line.

  The second (optional) parameter is a property list of TextReading options.

  Example:

```
        put readText((40,0,400,30))
--> Finder   File    Edit   View   Go

        put readText((200,10))
--> Edit
```

- Added a **HighlightRectangle** command which draws a rectangle highlight in the Remote Screen window. It requires a first parameter which is the rectangle to highlight. An optional second parameter is a property list of options, which may include color and duration properties. The **color** option specifies the color to use for drawing the rectangle, which defaults to red. The **duration** property specifies the length of time the highlight rectangle will be shown. If not specified it will be shown for the default ImageHighlightDuration time. Specifying zero for the duration will cause the highlight to remain until the next time a highlight rectangle is shown, either by another call to this command, or by any screen search command that draws highlights.

- A new Text Finding Engine (**OCR Search**) now exists in the Eggplant Preferences -> Text section.
  If you have NOT modified your text preferences this is NOW the default Text Finding Engine.
  If you have modified your text preferences you can create a new platform that uses OCR. You can always change your Default Text Platform using the Text Preferences. When using the **OCR Search** text engine you can specify text in any image finding command. You don't need to specify font characteristics but you can select language options for various styles.

  Example:

```
        click(Text: "File")  // If OCR is your default platform
        moveTo(Text: "1:34 AM", TextPlatform:"OCR") // If OCR is not the default platform
```

- The **RunWithNewResults** command will no longer change the status of any connections. Connections that are open when the command is issued will stay open when the called script begins, and connections that are open when the called script finishes will stay open in the calling script.

- Added an **every** option to the **offset** and **range** functions, to obtain the offset location of every occurrence of one text string within another or the full range (starting and ending positions) of every instance of some text within a larger text string. In addition to the natural language syntax, these can also be called using new **everyOffset()** and **everyRange()** functions.

```
put every range of "at" in "concatenation" into atRanges
put atRanges -- (5 to 6,9 to 10)
```

- Added initial support for **units** associated with numeric values. Any variable holding a numeric value can have a unit type associated with it. Two types of units are officially supported at present: a time interval (or duration) type with a unit of **seconds** and a byte size (or data size) type with a unit of **bytes**. Values created with the special syntaxes for these types will automatically be assigned with those unit types:

```
put 2 hours into waitTime -- a duration with units of 'seconds'

put 16 megabytes into capacity -- a byteSize with units of 'bytes'
```

- Added **the unitsEnabled** global property to specify whether units should be shown when displaying a value (the initial setting is false):

```
put 1 minute 30 seconds into delay

put delay -- 90

set the unitsEnabled to true
```

```
put delay -- 90 seconds
```

- Added the ability to **repeat for** *duration* to create a repeat loop that terminates after a specified length of time has passed. This type of repeat loop will execute the body of the repeat block, then check whether the specified length of time has elapsed since the repeat began. If so, the loop is terminated, otherwise the body of the repeat loop is executed repeatedly with the time being checked at the top of the loop each time until the specified time duration has passed.

```
repeat for 3 seconds
          add 1 to counter
end repeat
put "Counted to " & counter & " in 3 seconds"
```

- Extended the **is a** operator to allow testing whether a value is a particular type of unit. For example you can test whether a value is a duration using the "is a duration" (or "is a timeInterval") operator, or whether it is a byte size using the "is a byteSize" (or "is a dataSize") operator:

```
put 3 minutes into x
put x is a duration -- true
put 5 megabytes is a dataSize -- true
```

- Added the **units** property of a variable to access or change the units associated with that variable. For timeInterval values, the units will be "seconds"; for a byte size value it will be "bytes".

```
put 1 minute 30 seconds into delay
put delay's units -- seconds
set delay's units to "bytes"
```

  Other units may be used but are not specifically supported, so should be used experimentally with the understanding that the behavior may change as new unit types are added in the future:

```
set the unitsEnabled to true
put 12 into snack -- (must be a number before assigning units)
set snack's units to "peanuts"
put snack -- 12 peanuts
subtract 5 from snack -- (eat a few)
put snack -- 7 peanuts
```

- Added ability to **repeat until** *time* to allow a script to repeat a series of steps until a particular date/time has arrived.

```
repeat until "5:00 PM"
          doSomeWork
          wait a minute
end repeat
```

- Added ability to **wait until** *time* to allow a script to pause execution until a particular date/time arrives. If the time specified is already in the past, execution proceeds immediately.

```
wait until "5:00 PM"
wait until now + 1 second -- evaluated repeatedly; will wait forever
wait 1 second -- do this instead
```

- Added a **sorted** operator that provides all the functionality of the **sort** command. As an operator, it produces a sorted value without changing the source value. The sorted operator takes the form "*value* **sorted** *options*" where options include the same options available for the **sort** command, plus "**as** *chunkTypes*" to sort the value as something other than a list of items.

```
put (99,3,2,5,32,36,45737,26) sorted -- (2,3,5,26,32,36,99,45737)
put randomList sorted in ascending order into sortedList
put "Eggplant is great" sorted as words -- "Eggplant great is"
put "Eggplant is great" sorted as words by the length of each
          -- "is great Eggplant"
```

- Added **the asTextLimit** global property to set a limit on the size of internally-generated text representations of values. If a text representation is requested that exceeds this limit an exception is now thrown. This protects the system from possible crashes caused by extremely large values. The default limit is ten million characters.

- Added a **globallyUniqueString()** function to return a unique string each time it is called. The value is unique across systems, application instances, and individual calls to the function.

- Added some additional pieces of information returned by the **SystemInfo()** function, and by the **OSInfo()** function on Linux and Windows systems.

- Added **the predefinedVariables** global property. This is a property list containing the definitions of all of the predefined variables. Any changes made to this property will affect any subsequent use of an unassigned variable whose name is one of its keys. By modifying this property, predefined variables can be created or removed, or their value can be changed.

```
put st -- "st"
set the predefinedVariables.st to "SenseTalk"
put st -- "SenseTalk"
```

- Added some additional predefined variables, in these categories:

  Common symbols:
  period (**.**), **fullStop** (**.**), **semicolon** (**;**), **questionMark** (**?**), **exclamationMark** (**!**), **numberSign** (**#**), **circumflexAccent** (**^**), **ampersand** (**&**), **asterisk** (*****), **leftParenthesis** (**(**), **rightParenthesis** (**)**), **leftSquareBracket** (**[**), **rightSquareBracket** (**]**), **leftCurlyBracket** (**{**), **rightCurlyBracket** (**}**), **underscore** (_), **apostrophe** (**'**), **singleQuote** (**'**), **tilde** (**~**), **graveAccent** (**`**), **caretSign** (ʌ), **referenceMark** (※), **doubleExclamationMark** (**!!**), **doubleQuestionMark** (**??**)

  Numeric symbols:
  lessThanSign (**<**), **greaterThanSign** (**>**), **equalsSign** (**=**)

  Miscellaneous symbols:
  whiteDiamond (◇), **fisheye** (◉), **blackCircle** (●), **whiteCircle** (○), **largeCircle** (◯), **dottedCircle** (◌), **bullseye** (◎), **whiteBullet** (◦), **blackSun** (☀), **whiteSun** (☼), **cloud** (☁), **umbrella** (☂), **umbrellaWithRainDrops** (☔), **snowman** (☃), **cometSymbol** (☄), **blackStar** (★), **whiteStar** (☆), **lightningSymbol** (☇), **thunderstormSymbol** (☈), **sunSymbol** (☉), **blackTelephone** (☎), **whiteTelephone** (☏), **hotBeverage** (☕), **shamrock** (☘), **skullAndCrossbones** (☠), **cautionSign** (☡), **radioactiveSign** (☢), **biohazardSign** (☣), **caduceus** (☤), **ankh** (☥), **peaceSymbol** (☮), **yinYang** (☯), **firstQuarterMoon** (☽), **lastQuarterMoon** (☾), **femaleSign** (♀), **maleSign** (♂), **snowflake** (❄), **hotSprings** (♨), **whiteFlag** (⚐), **blackFlag** (⚑), **heavyBlackHeart** (❤), **blackSpadeSuit** (♠), **blackHeartSuit** (♥), **blackDiamondSuit** (♦), **blackClubSuit** (♣), **whiteSpadeSuit** (♤), **whiteHeartSuit** (♡), **whiteDiamondSuit** (♢), **whiteClubSuit** (♧), **recyclingSymbol** (♲), **blackRecyclingSymbol** (♻), **whiteSmilingFace** (☺), **blackSmilingFace** (☻), **whiteFrowningFace** (☹).

- Added **increasing** and **decreasing** as synonyms for **ascending** and **descending** for specifying the order of a sort operation.

Bug Fixes / Tweaks:

- The **imageFound()** function will now throw an exception if called with only one parameter that is a number. If the first parameter is a number it is treated as the time to search, so at least one image must also be supplied. To search for an image whose name is a number without also giving a time parameter, include the image's file type extension
- Fixed an occasional problem cutting text from a script.
- Fixed a problem where changes made to the Text properties were not made available to the AHDB windows. Now changes made in preferences will cause the AHDB context to reload the default values.
- Fixed a problem that could occasionally lead the **remoteClipboard()** function called with a time parameter to throw an exception incorrectly indicating that no new value was available.
- Fixed a crashing/memory problem when setting **the SearchRectangle** repeatedly in a script.
- Fixed a bug in the cursor location toolbar which would report an incorrect value when the screen was scrolled.
- Fixed a possible hang when running a script from the command line.
- Fixed an issue setting the fonts used for scripts and logs.
- Fixed a problem with the "**New Script…**" menu item being disabled when a window other than a script or suite window was selected.
- Improved the behavior when typing the name of an image to save that is the same as (or begins with) the name of an image folder or collection.
- Fixed a memory leak of a socket, which could lead to a crash when checking connection availability.
- Fixed a problem with display of values in the Duration column on the Licenses panel.
- Fixed a bug with the Remove button on the Licenses panel deleting the last license rather than the selected license.
- Fixed a problem using the Insert.. pop-up button to insert a RightClick command.
- Fixed a problem creating TextPlatforms or TextStyles that only differed by case.
- Fixed an occasional crash when cutting text from a script.
- Fixed a possible crash after canceling the close of an unsaved script.
- Fixed a bug that would throw an exception if Add Image was used to add an image to a commented line in a script.
- Fixed time values specified without a particular date to be treated as a time on the current day (previously a bug was treating these as being on the first day of the current year).
- Fixed a bug with deleting the first item of a list while iterating over that list (could cause the second item of the list to be skipped).
- Fixed a bug (introduced in 10.22) with displaying lists or property lists containing trees (including results of "all nodes of" expressions). The trees would appear as an empty property list.
- Fixed some problems with the **standardFormat()** function, which in some cases (such as values containing ">" followed by a return) would produce an incorrect representation that could not be turned back into the original value using the value() function.
- Fixed a problem with checking whether a range **is within** another range.
- Fixed a bug that would display ranges within lists or property lists as lists, rather than as ranges.
- Fixed a bug that would throw an exception if a **next repeat** statement was executed more than 10 times within a try block inside a repeat loop, with the message "SenseTalk Runtime Exception - TRY statement exceeds maximum nesting limit per handler (10)".
- Fixed a bug (introduced in 10.22) with displaying lists or property lists containing references to objects.
- Fixed a script formatting bug (introduced in 10.2) that would cause problems when pasting any odd-length text string at a location exactly 99 characters before the end of a script that is at least 200 characters long.
- Fixed a bug with standardFormat() that would throw an exception in certain cases (such as when a property list contained a standard Cocoa object).
- Improved the error message when the call depth limit is exceeded to provide a helpful hint. This error was also renamed from "SRUN_RecursionLimitExceeded" to "STMaxCallDepthExceeded".
- Fixed a number of script memory leaks, including some when accessing chunks of a value.
- Fixed a crash when accessing lines of a file in certain situations.
- Fixed a crash when setting a variable to one of its items that is an integer.

- Fixed a bug that could lead to a crash if an exception occurred within a sort condition inside a try block.
- Fixed a problem with formatting of lists and property lists that would fail to change the quote format back after rendering a list within a property list or vice versa.
- Added additional details to warning when unable to locate suite for a messenger.

Release v10.23 (22-October-2010)

Highlights:

- Includes new version **Vine Server 3.12**.
- Includes Latest **Eggplant Floating License Server**.
- Fixed a number of bugs and compatibility issues – (See Below).

Scripting:

- Added a **sorted** operator for beta testing. This takes the form "*value* **sorted** *options*" where options include the same options available for the **sort** command, plus "**as** *chunkTypes*" if you want to sort the value as something other than a list of items. This feature is considered beta, as it needs more testing and may undergo some finishing tweaks before the next release.

```
put (99,3,2,5,32,36,45737,26) sorted -- (2,3,5,26,32,36,99,45737)
put randomList sorted in ascending order into sortedList
put "Eggplant is great" sorted as words -- "Eggplant great is"
```

- Added **millionth** as a valid ordinal number.

- Added **millisec**, **millisecs**, **microsec**, and **microsecs** as abbreviations for millisecond, milliseconds, microsecond, and microseconds.

Bug Fixes / Tweaks:

- Fixed a problem updating the display of the Remote Screen on 10.4 and 10.5.
- Fixed some issues with storing data into byte chunks that would previously convert values to and from text internally. Not only was that inefficient, but it could throw exceptions depending on the particular data values and the defaultStringEncoding.
- Fixed the standardFormat function to format lists with a space after the opening brace so that nested lists can be evaluated correctly by the value function without being interpreted as block quotes.
- Fixed the "number of lines of" function and "each line of" expressions to allow a list for "delimited by" and use the lineDelimiter if delimiter is omitted or empty.
- Fixed a problem with error reporting when an error occurred in evaluating the setup conditions of a repeat loop, that would report as being an error in the preceding line of the script.
- Fixed a bug with accessing the properties of an object that was passed into a handler as a single-item list containing the object.
- Fixed a bug with accessing the currentIndex property or iterating using nextValue or nextAssignedValue of a list when that list contains a single value which is an object.
- Fixed a bug with assigning a list of chunks to a value that would treat them as references ("put items (3,4) of x into y").
- Fixed a bug when performing vector arithmetic using a list containing references to other values, that would alter the referenced values.

Release v10.22 (24-September-2010)

Highlights:

- Added Windows playback and compression of Movies after the recording finishes.
- Improved VNC performance and compatibility including KVM enhancements.
- Fixed a number of bugs and added many other features – (See Below).

VNC:

- Added control to optionally exclude VNC screen synchronization using the
  ```
  defaults write Eggplant RequireSync NO
  ```
- Fixed a crashing problem when using the Zlib based encodings. ZlibHextile, ZlibRaw, and ZRLE should all be more stable.
- Added support for maintaining a connection when changing SUT screen resolutions.

Scripting:

- Added a step to perform compression of movies after the recording finishes. This also makes them compatible with QuickTime on Windows. To disable this option use:
  ```
  defaults write Eggplant MovieCompressionEnabled NO
  ```

- Added optional "indent" property to the listFormat and the propertyListFormat. Setting this property to a suitable string such as " " will cause individual list items or property list entries to be displayed on separate lines with values indented by multiples of the indent value according to the nesting of the structure.
  ```
  set list to (apple, banana, cherry)
  put list -- (apple,banana,cherry)
  set the listFormat's indent to "  " -- indent by 2 spaces
  put list --
  (
    apple,
    banana,
    cherry
  )
  ```

Bug Fixes / Tweaks:

- Improved how Eggplant handles locked suites.
- Fixed a problem where eggPlant doesn't remember the location of the current script when it reloads from an external change.
- Fixed a problem where Remote Windows sometimes tracked the mouse when not the active window.
- Fixed a problem reading Images recorded on Windows and Linux versions of Eggplant.
- Fixed a problem with GetOptions reporting about FullScreenFailsafe.
- Fixed a problem when trying to record a movie with a long file path, we now record to /tmp and then move the file.
- Fixed a bug that prevented iterating over the same range or list multiple times in a nested manner.
- Fixed a bug when saving a script to a different Suite.
- Fixed a bug where non-active Remote Windows would report that they were running the script.
- Fixed a bug that would prevent similar objects (that were equal in value, but distinct objects) from both being included in the helpers or early helpers of another object.
- Fixed a crash when the SUT screen resolution changes.
- Fixed a problem where enabling/disabling some encodings wouldn't take effect.

- Fixed a bug using the TypeText pull-down menu.
- Fixed a crashing bug when minimized remote windows had their connections lost or screen sizes change.
- For performance reasons, minimized windows no longer update their tiles by default. To enable this use

```
defaults write Eggplant LiveMiniWindows YES
```

Release v10.2 (20-July-2010)

Highlights:

- New Text Recording feature for Remote Window allows recording most key commands.
- Dramatically improved formatting performance when working with very large scripts.
- Improved VNC performance and compatibility including KVM enhancements.
- Fixed a number of bugs and added many other features – (Please read the full release notes for details).

User Interface:

- You can set a Default search type for new images in the Viewer Preferences.
- Bug Submission Panel now provides feedback on if the submission was accepted.
- Improved Control over helper suites, allows setting Home and Default Suite directory relative paths by using tilde(~/) and dot(./) respectively. Select a helper and then click the path to edit.
- The New Text Recording feature allows you to just start typing when in Capture Mode in the Remote Screen Window. A panel showing the TypeText command to be inserted will allow you to preview your command before sending it to the SUT. This feature replaces the insert TypeText action.
- Results for scripts called with RunWithNewResults can now be easily jumped to by using the Linker Arrow in the Result Viewer.
- The Format menu has been removed, as creating scripts in RTF format is no longer supported. Similarly, the Default Script File Format has been removed from the Script preferences pane.
- The Restore Defaults action on the Text preferences panel now displays a dialog warning that it will remove any custom text platform information.
- Improved search results in Eggplant's HelpViewer to not search beyond newline ranges.
- The Eggplant Tutorials are no longer included in the HelpViewer as they have been moved online. A new command on the Help menu provides a link to the online tutorials.

VNC:

- Dramatically reduced the number of screen updates Eggplant requests during typical exchange with the VNC server and improves compatibility with x11vnc.

- Added the ability to forgo full-screen updates before image searches fail. These updates are difficult for slower VNC servers and certain KVM devices to handle. To disable full-screen updates issue this command in a terminal:
  `default write Eggplant FullScreenFailsafe NO`

- Added **the mouseRepositionSpeed** global property to allow setting the reposition to happen slowly, rather than jumping. Generally this is not recommended as sliding the mouse can cause side effects on the screen, but for KVM and other devices it is sometimes necessary.

Scripting:

- The current setting of **the ScriptLogging** property is no longer reset by the **RunWithNewResults** command. The ScriptLogging property is now shared in common between the master script and any scripts called by the RunWithNewResults command, so any change to the property in one script will carry over to the others.

- Quoted lowercase strings used in the context of the auto-released **ShiftKey** will now be interpreted as upper-case letters by all VNC servers. Example: TypeText shiftKey, "n" will print a capital "N" previously it was dependent on the target VNC Server.

- Added a **range** function to find the range of characters where one string is found within another. The range function has the same syntax as the **offset** function, providing many options for locating an occurrence of the target text before or after a given location within the source text:

```
put the range of "cat" in "concatenation" into catRange
put catRange -- 4 to 6
put chars catRange of "concatenation" -- "cat"
get the range of "Error" in myText before the end considering case
```

- Fixed the **offset** function to work properly with lists, returning the list item offset of a value or range of values within the source list, rather than the text offset. The new **range** function also works this way, returning the range of list items within the source list where the target list is found.

- Added **lineSeparator** and **paragraphSeparator** as built-in predefined variables whose values are the Unicode line separator (0x2028) and paragraph separator (0x2029) characters, respectively. Added these characters to the built-in list of line delimiters used when parsing scripts, and resulting from setting the defaultLineDelimiter to empty, so this list is now: (CRLF, LF, CR, LineSeparator, ParagraphSeparator).

- Added some additional predefined variables, extending these categories as indicated:

  Common symbols:
  checkMark (✓), **blackDiamond** (◆), **lowerLeftPencil** (✎), **helmSymbol** (⎈)

  Keyboard key symbols:
  appleLogo (), **alternativeKeySymbol** (⌥), **blankKeySymbol** (␣), **capsLockKeySymbol** (⇪), **clearKeySymbol** (⌧), **pageUpKeySymbol** (⇞), **pageDownKeySymbol** (⇟), **tabKeySymbol** (⇥), **tabLeftKeySymbol** (⇤), **returnLeftKeySymbol** (↩), **returnRightKeySymbol** (↪), **leftArrowSymbol** (←), **rightArrowSymbol** (→), **upArrowSymbol** (↑), **downArrowSymbol** (↓), **contextualMenuKeySymbol** (✳)

- Step values in a range are now honored when the range is used as the index in a chunk expression, and the original value type is retained (rather than always producing a list). For example, these all work properly now:

```
put chars 1..5 by 2 of "abcdefgh" -- "ace"
put items 2..6 by 2 of "a,b,c,d,e,f,g,h" -- "b,d,f"
put items 2..6 by 2 of (a,b,c,d,e,f,g,h) -- (b,d,f)
put chars 8..1 of "abcdefgh" -- "hgfedcba"
put chars 8..1 as list of "abcdefgh" -- (h,g,f,e,d,c,b,a)
```

- Changed the value of the special **end** constant (**the endValue** global property) to something less likely to ever cause trouble.

- Changed the **folder()** function (the folder, my folder, the folder of aFileOrPath, the directory, etc.) to return a fileDescription object rather than a simple string. The asText property of the object is set to the full path of the folder rather than to the short name as in most other fileDescription objects.

- Added **date** and **time** as special dynamic predefined variables (similar to **today** and **now**) that evaluate as the value of the date() and time() functions if they haven't been assigned another value.

- A file path can now be specified in "Windows" format, including drive letters (like "C:") and using backslashes instead of slashes. THIS WILL PREVENT THE USE OF "\" AS A CHARACTER IN A FILE

NAME.

- A file path can now be given as a list. Each item in the list is treated as one component of the path. A list beginning with "/" (or "\") as the first item, or with a Windows drive letter (such as "C:") as the first item and "/" (or "\") as the second item, will be treated as an absolute path.

- Added a **pathList()** function that takes any file path string or list and returns that file path as a list of individual path components in a standard format.

- Added a **filePath()** function that takes any file path string or list and returns that file path as a string in a standard format, with slashes as the separator between path components.

- Added a **windowsFilePath()** function that takes any file path string or list and returns that file path as a string in the standard format for Windows file systems, with backslashes as the separator between path components.

- Added a **resolvedFilePath()** function that takes any file path string or list and returns that file path as a string in a "resolved" standard format, which is the full absolute path to the file, taking into account the current folder if necessary, and resolving components such as ".." and "~".

- Updated the **shell()** function (and command) to work on Windows. On the Windows platform, **the shellCommand** global property is now set to empty by default (and shouldn't be changed).

  When **the shellCommand** is empty, the **shell** function now treats its first parameter as the command to execute and any additional parameters as parameters to be passed to that command. If only a single parameter is passed, it is split by any spaces that are present in the parameter to derive the command and its parameters.

```
set the shellCommand to empty -- (the default on Windows)
put shell("ls -l") -- runs the 'ls' command with '-l' parameter
put shell("ls", "-l") -- does the same thing
```

  When **the shellCommand** is not empty (the usual case on Mac and Linux, where it is set to "/bin/sh" by default), if multiple parameters are passed to the **shell** function they are treated as separate commands to all be executed in sequence within a single shell context.

- Enhanced performance of the **format()** function, and fixed it to be usable on Windows. This function also now supports additional format codes %a, %A, and %@ (%a and %A are not available on Windows).

Bug Fixes / Tweaks:

- Fixed a bug with the Hot Spot and other image attributes not being preserved when moving a Suite from Linux or Windows to the Mac.
- Fixed a bug with the Hot Spot field in the image drawer not updating after an image's hot spot is moved by command-dragging it.
- Fixed a problem with the Make Collection button not always properly updating the display.
- Fixed a bug with a scripted TIG not working correctly when the TIG script is not located in the current suite.
- Updated the autocomplete feature to include some missing property names.
- Adjusted the EggplantLicenseServer to run from inside the application by default making it much easier to grant a permanent firewall exclusion.
- Fixed a problem calling ConnectionInfo() on a server that didn't exist - this now returns an empty property list.
- Fixed a problem using the Connect() commands which could result in an exception.
- Fixed a problem where the Connect() command was case-sensitive on properties that were passed in.
- Removed the ICON^m file from inside the suite. It created a conflict for some SCM tools and is no longer needed on 10.5 and 10.6.
- Fixed a problem on Snow Leopard dropping Images from the same suite into the ScriptEditor.

- Fixed a problem passing file names into the PLAY command.
- Fixed a problem reading a file as data. This would improperly throw an exception if the file couldn't be interpreted as a string using the defaultStringEncoding. An expression of the form "file x as data" will always return the full contents of the file as binary data now.
- Fixed a problem with iterating over a list in one handler while in the process of iterating over that same list in another handler (the outer iteration would be terminated early, get stuck on a single value forever, or affected in other similar ways).
- Fixed a problem in which changing the properties of a value passed to another handler could affect the original value even though it was not passed by reference.
- Fixed a crash that could occur when making changes to a tree that was passed in as a parameter from another handler.
- Fixed a bug that would cause incorrect results (such as operations performed more than once) when a value within a range was changed while iterating over the values in that range.
- Fixed a bug that could produce unpredictable results with control structures (such as repeat loops, if/then blocks, or try/catch blocks) that were extremely long (thousands of lines).
- Fixed a bug testing the existence of an object property when the property name is in a variable.
- Fixed a problem with some error messages (such as when attempting to access a URL) not reporting NSInvalidArgumentException *** -[NSCFType stValue]: unrecognized selector sent to instance.
- Improved some error messages.
- Fixed a bug which would cause a crash when performing various operations on a list or property list that contained a circular reference to itself.
- Fixed a bug with deleting an item from a range.
- Fixed a bug with copying list items between lists or to a different place within a list, that would incorrectly treat the copy like a reference to the original item.
- Fixed a bug with passing a list item as a parameter to another handler, that would in effect pass it by reference.
- Fixed a bug with sorting items in a list nested within another list.
- Fixed a bug that would throw an exception ("STInvalidOperation -currentIterationIndex called on non-object STValue") when trying to iterate "by reference" over the items of a list (using "repeat with each item of theList by reference") when the list was received as a parameter or is itself a reference.
- Fixed a bug with **the exception** not being set to empty on a single-line **try** statement that doesn't throw an exception.
- Fixed a problem introduced in the compiler in release 1.36 (Eggplant 10.03) that would treat the plain words 'date' and 'time' as function calls rather than variables.
- Fixed a problem with forcing dates to compare as text in some situations ("the time as text is someTime" would compare as times rather than text).
- Always use normal SenseTalk formatting when storing structured values into a text chunk or a file.
- Fixed a problem with storing into files in some situations. This now correctly stores the respective values into the two files (previously it did nothing):

```
put ("abc","def") into (file "/tmp/a", file "/tmp/b")
```

- Fixed a bug that would return empty when using a range to access items in a list, such as:

```
put items 5..10 of 20..1
```

- Enhanced script formatting to allow formatting a partial script; made formatting more robust.

Release v10.11 (08-October-2009)

Highlights:

Bug Fixes / Tweaks:

- Fixed a problem when trimming Text Images coming from External Text Image Generators (e.g. Windows).
- Text Preferences will now warn if the new TextPlatform.plist can not be saved properly.

- Eggplant will now attempt to create the ~/Library/Eggplant directory if it does not exist.
- Ability to control the TextImageTolerance at a global level.

Release v10.1 (30-September-2009)

Highlights:

- Support for Mac OS X 10.6 Snow Leopard.
- Eggplant preferences are now stored under the domain **Eggplant** for both Linux and Mac platforms. See below for more detail.
- Text Platforms are now stored in the file **~/Library/Eggplant/TextPlatforms.plist**. See below for more detail.

User Interface:

- Enhanced the license panel to allow control of floating license attributes.
- Added the ability to enable/disable caching at the TextPlatform level.
- Added the ability to reset TextPlatforms to their default values.
- Added the ability to enable/disable caching at the TextPlatform level.
- Clearer enabling/disabling of controls when doing Image Doctor searches.
- Note: Apple's Preview application on Mac OS X 10.6 is not able to display Images and Screenshots captured in earlier versions of Eggplant; those images are fine when viewed in Eggplant, Safari, Photoshop and most other places. Images saved in this version will be viewable by Preview.

Scripting:

- When running in FullScreen mode Ask or Answer panels will now be displayed on top of the remote screen.
- **Line** chunks are now defined as chunks of text separated by any of a list of delimiter strings (as specified by the lineDelimiter property), rather than always being separated by return characters.
- Added **the lineDelimiter** local property and **the defaultLineDelimiter** global property to permit changing the set of strings used as delimiters between "line" chunks. Either property can be set to a list of delimiter strings. The initial setting of the defaultLineDelimiter is simply the return constant (also known as linefeed or LF) for compatibility with earlier versions. Setting the defaultLineDelimiter to empty is a shortcut to set it to the list (CRLF, LF, CR) to match any of the common line endings. The lineDelimiter property is local to each handler execution context (or frame) and is set to the value of the defaultLineDelimiter at the start of each handler.

  The order of the delimiters is significant, with preference being given to delimiters earlier in the list. So setting the lineDelimiter to (CRLF, LF, CR) will match CRLF (a carriage return character followed by a linefeed) as a single delimiter rather than as two separate line delimiters.
- Added **the characterFiller**, **the wordFiller**, and **the lineFiller** global properties to give control over how strings are extended when putting values into non-existent (out of range) character, word, or line chunks.

  The characterFiller is initially set to "." (a period), so putting something into a character position beyond the current value in a container will fill in the intervening positions with dots (NOTE: this is a change from earlier behavior):

```
put "A" into notes
put "G" into character 7 of notes
put notes -- "A.....G"
```

  The characterFiller isn't restricted to a single character. A longer string will be repeated as needed to fill in the space:

```
set the characterFiller to "Ha"
```

```
set laugh to empty
put "!" into character 7 of laugh
put laugh -- "HaHaHa!"
```

The wordFiller is initially set to "?" to represent any unspecified words (NOTE: this is a change from earlier behavior):

```
set phrase to "start here"
put "the end" into word 7 of phrase
put phrase -- "start here ? ? ? ? the end"
```

Set the wordFiller to empty for the same behavior as earlier releases (simply fill in with a single space), or to a different value:

```
set the wordFiller to "umm..."
put "Hello and" into greeting
set word 5 of greeting to "welcome!"
put greeting -- "Hello and umm... umm... welcome!"
```

The lineFiller is initially set to **return** for the same behavior as earlier versions, but may be set to a different line ending (such as CR or CRLF) or some other text if desired.

- Added **top(), bottom(), left(),** and **right()** functions to get the location of the indicated edge of a rectangle (a single value).
- Added **center(), topCenter(), bottomCenter(), leftCenter() and rightCenter()** functions to get the point at the center of a rectangle, or at the center of the top, bottom, left or right edge, respectively, of a rectangle.
- Added the ability to test whether any value is an iterator using the "is an iterator" operator (lists, ranges, and objects with an objectType of "iterator" are all iterators).

```
put (1,3,5,9) is an iterator -- true
```

- Fixed a problem with multiple each expressions within an expression that could cause a hang. It is now possible to use more complex expressions that combine each expressions, like these:

```
put each item of "A".."C" & each item of 1..4
          -- ((A1,A2,A3,A4),(B1,B2,B3,B4),(C1,C2,C3,C4))
put each item of 1..10 times each item of 1..10 into timesTable
```

- Parentheses can now be used around "for each" expressions and "where" clauses to enhance readability and limit the scope of the expression:

```
put each.first && each.last (for each item in employeeList)
put each line of sales (where item 3 of each > 5000)
```

- Added **is at least** and **is at most** as synonyms for the **greater than or equal to** and **less than or equal to** operators. Also added **is no less than** and **is no more than** variants for these operators:

```
set x to 5
put x is at least 5 -- true
put x is no more than 7 -- true
put x is at most 5 -- true
put x is no less than 5 -- true
```

- Assigning values to chunks with out-of-range negative indexes now properly inserts chunks at the beginning of the container as necessary:

```
set list to (1,2,3)
put "X" into item -6 of list
put list -- (X,,,1,2,3)

put "A" into goingDown
put "G" into character -7 of goingDown
put goingDown -- "G.....A"
```

- Changed name of the initialHandler script in a folder object from "<initialHandler>" to "_initialHandler_" (for greater compatibility across different file systems).

Bug Fixes / Tweaks:

- Eggplant preferences (aka defaults) are now read primarily under the Eggplant domain. If it does not find a value there it will look to the old com.redstonesoftware.Eggplant domain to get it. When preferences are saved in Eggplant that will always happen to the new Eggplant domain.

- Text Platforms are no longer part of the preferences (aka defaults) system and are now stored in the file ~/Library/Eggplant/TextPlatforms.plist. When 10.1 first launches it will relocate your Text platform preferences to there.
- Fixed a bug where the first call of an image in a scripted TIG as a Text Image Generator was ignored.
- Fixed a bug where image search type was reset when doing Image Doctor: Replace Image.
- Fixed a bug where Eggplant could hang from Command Line when Auto Doctor was enabled.
- Fixed a bug when making a collection of an image with a DOT in it's name.
- Fixed a problem with converted (from text) date values not being treated as noon on the given date.
- Fixed a bug with performing arithmetic operations with ranges:

```
put 1..9 times 3 -- (3,6,9,12,15,18,21,24,27)
```

- Fixed a bug when accessing **target** that would throw an NSInvalidArgumentException.
- Fixed a bug with sending a function message directly to SenseTalk using the syntax "SenseTalk.function()" or "SenseTalk's function()".
- Fixed a bug with changing the **step** property of a range (would set the step to the end value).

Release v10.03 (17-July-2008)

Highlights:

- Fixed a number of bugs and other minor issues.
- Includes Vine Server 3.11 which fixes problems with ZLibHextile encoding.

User Interface:

- Added the ability to reset script colorization to the default values.

Bug Fixes / Tweaks:

- Corrected numerous problems that resulted from an inaccessible results folder, including confusing exceptions and occasional crashes.
- Fixed a bug when editing script preferences before opening any scripts that would remove script colors and indents.
- Fixed a bug that caused Eggplant output to become garbled when setting **the defaultStringEncoding.**
- Fixed a class of exceptions that could be raised from the Connection List when no connections were present or when some were removed while Eggplant was running.

Release v10.02 (29-June-2009)

Highlights:

- Enhanced support for working with binary data.
- Added range expressions to easily create a sequence of values.
- Added support for iterators in repeat loops and each expressions.
- Added a command line option to report script failure status in a more standard way.
- Fixed a number of bugs and other minor issues.

- New improved version of Vine Server included for Mac SUTs (version 3.1).

User Interface:

- A padlock icon is now displayed in a Suite Editor window when Eggplant is unable to save updates in the suite, to indicate that changes may not be saved (typically this is due to lack of write permission). The lock icon will go away if Eggplant is later able to write the suite information.
- Improved the behavior of dynamic breakpoints. Breakpoints can now only be set on executable lines of a script, not on blank lines or comments. This should result in more predictable behavior.
- Saving a new image representation will no longer change the working image directory.
- Added support for .tff as an image file extension.

Command Line Interface:

- Added a **ReportFailures** flag to request that the exit code report the number of script failures (rather than successes) for the run. Either specify "-ReportFailures YES" on the command line, or use:
  `defaults write Eggplant ReportFailures -bool TRUE`

Scripting:

- Added improved support for working with binary data:

  • Added an **asData()** function, most commonly called using the new **as data** operator.
  For example, the raw binary contents of a file or URL can be accessed using **as data**:

  ```
  set myData to file "/tmp/data" as data
  ```

  It is also possible to write binary data to a file:

  ```
  put myData as data into file "/tmp/data"
  ```

  • When two known binary data values are compared for equality, they are compared byte for byte to see that they have exactly the same binary contents.

  • When values are converted between text and binary data, the current setting of **the defaultStringEncoding** global property is used to determine the encoding to be used.

  • A new **byte** chunk expression type gives access to any byte or range of bytes within a data value:

  ```
  put bytes 10 to 13 of myData into subData
  ```

  • Values read from a file by the **read** command are treated as text unless the quantity to read is explicitly specified as bytes.

- Added **ranges**. A range is a convenient way to describe a range or sequence of values by giving start and end values, and optionally a step value. A range expression can describe a range of numbers, dates or times, or characters by simply using the double-dot operator "**..**" or the word **to** between the start and end value:

  ```
  put 13..19 into teenRange
  put "April 1" to "June 30" by weeks
  ```

A range can be treated like a list, and will automatically generate its values as needed.

- A list can now be treated as an iterator. This includes being able to ask a list for its nextValue and currentIndex, and to change its currentIndex during iteration. The currentIndex is adjusted automatically any time items are inserted or deleted in the list. A **repeat with each** loop that iterates over the values in a list now uses the list's currentIndex property, so it is now safe to make changes to the list's contents while iterating.

- Added a **nextAssignedValue** function to get the next value that has been assigned in a list beyond its currentIndex.

- Added **is a multiple of** and **is divisible by** operators to test whether one value is an exact (evenly divisible) multiple of another.

```
put 21 is evenly divisible by 7 -- true
put 33 isn't a multiple of 2 -- true
```

- Made a small change to the **asDate()** and **asTime()** functions (often called using the **as date** and **as time** operators). These no longer act as simple synonyms for the **date()** and **time()** functions -- instead of using the standard formats, they will pick up the format to use from the text itself:

```
put date("mar 16, 1927") -- "03/16/27" (the same as before)
put asDate("mar 16, 1927") -- "Mar 16, 1927" (new behavior)
```

- Added a **format** property of date/time values, allowing direct access to the format, independently of the value. The format can be set to change the way the value displays.

- Added **numberWords()**, **ordinalWords()**, **timeInterval()**, **timeIntervalWords()**, **byteSize()**, and **byteSizeWords()** functions to convert a number, an interval of time (in seconds), or a file size (in bytes) to a friendlier format. The text that results from any of these functions (except ordinalWords) can be converted back to a number using the **value()** function.

```
put numberWords(90) -- "ninety"
put numberWords(427) -- "four hundred twenty-seven"
put ordinalWords(90) -- "ninetieth"
put timeInterval(90) -- "1 minute 30 seconds"
put timeIntervalWords(90) -- "one minute thirty seconds"
put timeIntervalWords(11520) -- "three hours twelve minutes"
put byteSize(5242880) -- "5 megabytes"
put byteSizeWords(90) -- "ninety bytes"
```

- Added new geometric functions to get the corners of a rectangle: **topLeft()**, **topRight()**, **bottomLeft()**, and **bottomRight()**.

- Added a new **processInfo()** function to return a property list containing information about the current process, including its name, parameters, and process id.

- Added a new **run()** function, analogous to the **run** command, to allow calling the initial handler of a script or other object explicitly as a function. This also allows calling a script whose name is generated dynamically at runtime. The run function needs to be called in a way that sends the run message to the object, such as using dot notation:

```
get ("test" & testNumber).run(value1, value2)
```

- Added a **C time** (or **CTime**) format, used by some C-based systems (including Python), plus **long**, **short**, and **abbreviated** variations:

```
put the C time -- Fri Jan 4 15:20:26 2008
put the long CTime -- Friday January 4 15:20:26 2008
put the short c time -- Jan 4 15:20 2008
put the abbrev C time -- Jan 4 15:20:26 2008
```

- A new **for each** expression extends the functionality of **each** expressions. More complex expressions can now be evaluated that use "each" to refer to each value provided by an each expression, by following that expression with **for each** as in this example:

```
put each & "^2 = " & each^2 for each item in (2,3,4,5)
          --> ("2^2 = 4","3^2 = 9","4^2 = 16","5^2 = 25")
```

- It is also possible to chain **each** expressions when working with nested lists:

```
set myList to ((1,2,3),(4,5))
put each item of each item of myList & "x" -- ((1x,2x,3x),(4x,5x))
```

- Added **iterator** support. An **iterator** is a SenseTalk object that provides a series of values, by implementing a **nextValue** handler. Each time the nextValue handler is called it should return the next value in the series. When no more values are available it should return the new special constant **end** (also available as **the endValue** global property). The only requirement for an object to be used as an iterator in addition to handling nextValue messages, is that it must have an objectType of "iterator". For example, here's a simple script that can serve as an iterator for the values 1 to 10:

```
properties
          objectType: "iterator",
end properties


to handle nextValue
          add 1 to my counter
          if my counter > 10 then return end -- no more values!
          return my counter
end nextValue
```

  An iterator can be used in a repeat loop like this (assuming the script above is called Counter):

```
set myIterator to a new Counter
repeat with each num in myIterator
          put num
end repeat
```

  or in an **each** expression:

```
put each item in new Counter
```

  A **startIteration** message is sent to the iterator object before iteration begins, to allow it to reset to a starting condition. So, to make the Counter object reusable, add this handler to its script:

```
to startIteration
          set my counter to 0
end startIteration
```

- Added **the centuryCutoff** local property (and its global counterpart **the defaultCenturyCutoff**) to control adjusting the year, when converting strings that use 2-digit years to a date. When a two-digit year is encountered, it is assumed to be in the present year or a future year up through the year indicated by the centuryCutoff. Otherwise, a two-digit year that is greater than the centuryCutoff is taken to be in the past (either earlier in the present century or in the previous century). By default the centuryCutoff is set to be 10 years in the future (so in 2008, a two-digit year will be assumed to represent a year between 1919 and 2018). Set the centuryCutoff to -1 to turn this functionality off (to allow two-digit years to represent years in the first century CE).

- Added **the wordQuotes** and **the defaultWordQuotes** local and global properties. These control the meaning of quoted "word" chunks in a script. Normally, any word that begins with a double quotation mark is treated as a quoted word that extends to the next quotation mark. The wordQuotes property can be used to define a different character or string (or opening/closing pair of strings) to be used as the quotation marks. Or, set the wordQuotes to "None" to disable quoting of words altogether.

- The use of "as" operators on values in property list expressions will now flag the value within the property list as the indicated type. This can be useful, for example, when constructing trees from property lists to force a value to be treated as literal text:

```
set myTree to tree(string:"<abc>stuff</abc>" as text)
```

- The literal strings "end" and "eof" may now be used as terminating values for the **read ... until** command, to read until the next occurrence of that text. To read until the end of the file or available data, use **read ... until end** or **read ... until eof** (without quotes around "end" or "eof").

- The **else** portion of a selector expression (except for containers) is now optional. If an else expression is not given, empty is assumed, so it is now possible to do things like this:

```
put "Total size is " & count & "byte" & (if count > 1 then "s")
```

- Made a small change to the syntax of the **run** command. A comma after the script name is now ignored (previously it would be interpreted as implying an empty first parameter). This should help avoid some confusion, allowing it to be called like a generic command, but may potentially be a compatibility issue (highly unlikely).

- Added **carriageReturn** as a predefined variable, equivalent to **creturn** or **cr**.

- Added support for 'ldt ' date types, 'shor' short integers, 'magn' unsigned integers, and 'alis' aliases in return values from AppleScript.

- Allow real numbers (with decimal fraction) to be used in combination with number words (e.g. "1.5 billion").

- Changed the output of the **standardFormat()** function slightly for lists, to use { } rather than ( ), so archived single-item lists can be restored properly.

- Changed the output of the **standardFormat()** function for trees, so they can be archived and restored properly by the value() function.

- Small change to make the syntax for calling the formatted date/time functions more flexible:

```
put international date of "3/16/26"
```

- Added **case sensitive** as a synonym for **with case** or **considering case**, and **case insensitive** as a synonym for **without case** or **ignoring case**.

- Date/time values are now quoted when shown as text literals in lists and property lists.

- Added **throw exception** as a synonym for the **throw** command.

- Added the ability to call a function dynamically, using an expression in parentheses in place of a function name:

```
set myFunc to "length"
put (myFunc)("abcd") -- calls length("abcd") and displays 4
```

- Fixed the "number of occurrences" function to use more intelligent default chunk types based on the source value when no chunk type is given. Characters are now assumed unless the source value is a list or an object, in which case list items or property values are assumed, respectively.

Updates to the **Tree XModule** (XML) functionality:

- Added a **treeFromHTML()** function to convert HTML text to a tree.

- Added **documentTreeFromXML()** and **documentTreeFromHTML()** functions to convert XML or HTML text to a "Document" type tree that will generate a full XML document when converted to text.

- Added **every node** as a synonym for **all nodes** when accessing a list of nodes matching an XPath expression.

- Added a new function **STTreeVersion()** to identify the version of the Tree XModule that is loaded. This function currently returns a value of 2 to reflect the changes and additions made in the STTreeNode XModule.

- Changed the names reported by the **loadedModules()** function for the Color and Tree XModules to "ST_Color" and "ST_Tree". Any code that checks for these modules by their old names ("STBackEndColor" and "STBackEndNodes") should be modified accordingly.

- Only trees whose nodeType is Document, Element, or DTD are now treated as lists and can therefore have subtrees (previously, all trees would return true for the 'is a list' operator, but other types would throw an exception on any attempt to insert subtrees).

- A list can now be converted (using "as tree" etc.) to an unnamed tree (with an empty tag). This also applies to lists within property lists that are being converted.

- Minor change to comparison of trees to allow document trees to compare as equal if their only difference is due to the presence or absence of version and characterEncoding properties with default values.

- Changed the **asTree** function to parse XML as a document type rather than an element type if the version property is given, even if it is the default 1.0 version.

- Fixed a bug with **node** expressions. They now conform correctly to XPath expressions, starting at the root of the tree rather than one level down.
- Fixed a memory leak when creating some types of trees.
- Fixed a bug with comparison of trees that would incorrectly treat trees as equal when they differed only in text node content.
- Fixed a bug to allow constructing a tree from a property list containing some values that are already trees.
- Fixed a bug with treating a tree text node as a container for purposes of mathematical operations (e.g. "add 1 to item 2 of myTree").
- Fixed a problem with inserting some XML elements into a parent tree.
- Fixed a memory leak when setting the _xmlinfo property of a tree.

Bug Fixes / Tweaks:

- Fixed a bug opening suite and running scripts when the results directory was not writable.
- Fixed a bug with entering commands in the Ad Hoc Do Box while a script is paused by the Image Doctor.
- Fixed an occasional bug when generating text images with the native Mac TIG on Mac OS X 10.4.
- Fixed an intermittent problem accessing the current clipboard contents with the remoteClipboard() function when connected to Vine Server on a Mac SUT (fixed in Vine Server 3.1).
- Fixed a bug where images set as Tolerant would ignore pre-stored alpha channels. New images will now retain their alpha values when imported. For compatibility reasons old images do not do this, just copy an image to allow it's stored alpha channel to be acknowledged.
- Fixed a bug where images originally stored as Pulsing or Text had the alpha values saved into the base image.
- Fixed a bug where images with any pre-existing alpha were assumed to be pulsing images when first read.
- Fixed a bug which prevented storing the original capture location when capturing images from a script.

- Fixed a bug which prevented saving images with an ImageInfo property into the results folder.
- Fixed a bug in which breakpoints in a commented-out section would pause execution.
- Fixed a bug in which a breakpoint immediately following selected code would pause execution at the end of a Run Selection.
- Improved jumpy display of breakpoints while typing curly braces during script editing.
- When the first item in the Control menu is inactive, it no longer says "Enter Observe Mode".
- Disable Thumbnails by default, switched to use 10.4 NSWorkspace API and removed IconFamily.
- Improved Image Doctor's "Add Representation" action to not turn an image into a collection when it was already being accessed by the script as part of a collection.
- Fixed a problem redrawing the previous capture area when moving the capture rect after it was previously moved by the Add Representation action.
- Improved performance and reduced memory requirements by not allocating helpers list for most objects, resulting in more efficient messaging as well. This change reduced the time required to run our SenseTalk test scripts by nearly 9% !
- Adding empty to a value that is not a number or a date will now throw an exception.
- Fixed a problem with assigning values to multiple nested properties of a previously undefined variable.
- Fixed a problem with some operators changing values by reference in some situations.
- Fixed a problem interpreting some numbers spelled out as words (e.g. "two hundred fifty-six point one").
- Extended recognition of ordinal number words up to one less than the millionth chunk ("the nine hundred ninety-nine thousand nine hundred ninety-ninth item" works).
- Fixed calling a list of functions on a value using the syntax for accessing a list of properties. Previously if the value was not an object only the first function would be called and the rest ignored. Example:

```
put (length, uppercase, lowercase) of "Test" -- (4,"TEST","test")
```

- Attempting to access a non-existent property of a file will now throw an exception (rather than simply returning empty).
- Fixed a problem with recognizing time values of the form "01:00 PM" (which would ignore the "PM") by changing the order of formats in the timeInputFormat global property.
- Fixed the results of some edge cases of the **modulo** operator when using non-integer operands.
- Fixed a bug that could treat a long string (~1000 chars) as though it were an object in some cases, leading to occasional bizarre results (such as saying that the string's length was 3).
- Fixed the lastPathComponent() and fileExtension() functions to properly treat fileDescription objects as referring to the file using their long name, not their "asText" name.
- Fixed a problem with fileDescription() and other file-path-related functions when called on script objects using a generic property syntax (".", "'s" or "of"). These functions now properly work with the file path of the object rather than its string value.
- Fixed a problem that could block objects in the backScripts or frontScripts from receiving messages if a similar-appearing object appeared earlier in the message path.
- Fixed a bug that could cause some handlers to be ignored when sending messages in some situations.
- Fixed a bug with inserting something nested into a non-list item within a list.
- Fixed a bug when assigning a value to an item within a list that is a reference to another container, that would replace the item in the list with a new value rather than assigning the value to the referenced container.
- Fixed a bug with some unsupported uses of **each** expressions (particularly multiple chained each expressions) that could cause a variety of compiler or strange runtime errors.
- Fixed a problem with **empty** not being testable as a boolean ('empty is a boolean' returned false).
- Fixed a bug with checking the existence of or otherwise working with a file with path "" or "/".
- Fixed a bug with evaluating expressions that refer to a property, which could produce strange results under certain circumstances.
- Fixed a bug with lazy evaluation of object properties sometimes occurring too late, in the wrong context.
- Fixed a bug with modifying properties of 'target'.
- Fixed the handlerNames() function to always try to treat its parameter as an object.
- Fixed a crash that could occur when putting a string into a chunk of itself.
- Fixed a memory leak when working with chunk expressions (could be quite significant in scripts that use chunks a lot).
- Fixed the **pass original message to *object*** command to exit the local handler if the message is successfully delivered. By adding the words **and continue** at the end of this command the local handler will continue whether or not the object handled the message (this is a new option, with the old -- formerly

incorrect -- behavior).

- Fixed a bug with over-zealous object lookup treating the strings "me" and "target" as the corresponding objects in some contexts, rather than as simple strings.
- Fixed a problem with return value sometimes carrying over to another context (seen as a double output in some situations).
- Fixed a problem with synchronizing the result value (seen as delayed result output while paused in the debugger).
- Improved caching of URL data for better performance in some situations.
- Changed internal string handling to improve performance (for possibly a 4% or better overall decrease in script execution times).

Release v4.12 (07-July-2008)

Highlights:

- Added preference to save breakpoints across Eggplant runs and a menu item to clear all breakpoints.
- Fixed a number of bugs and other minor issues.

User Interface:

- Breakpoints can optionally be saved between runs, the preference is found in Eggplant->Preferences->Script Editing.
- Added a menu item to Remove All Breakpoints in all open scripts.

Scripting:

- Added **the result** value on EveryImageLocation(). This returns a list of Image Info property lists that match the returned coordinates from the function.

Bug Fixes / Tweaks:

- Fixed a problem where Viewer Windows would stop updating visually after the Run Option or Mail preference panes were opened.
- Fixed a visual problem where moving cursors could leave a trail on the Viewer Window display.
- Improvement to clear image doctor rectangles when starting up a script.
- Fixed a problem with KeyDown and KeyUp not colorizing correctly.
- Fixed a rare lockup when running a command in the Ad-Hoc Do Box.
- Improved logging of results from calls in the Ad-Hoc Do Box
- Fixed a bug with subsequent runs involving ask or answer panels showing errors from previous run.

Release v4.11 (20-May-2008)

Highlights:

- Added support for emulating a middle mouse button when in Live Mode.

- Fixed a number of bugs and other minor issues.

User Interface:

- Added the ability to assign a modifier key to emulate a middle mouse button. On the Viewer Window pane of the Preferences panel, select a modifier from the popup menu. Then hold the selected modifier key down when clicking in Live Mode and Eggplant will send a middle mouse click event to the remote system.

Scripting:

- Added **the wordDelimiter** and **the defaultWordDelimiter** local and global properties. These control the meaning of "word" chunks in a script. The wordDelimiter defines a set of characters. Words are separated from each other by any number or combination of characters from that set.

```
set phrase to "Alas, poor Yorick! I knew him, Horatio"
put word 3 of phrase -- Yorick!
set the wordDelimiter to " ,!"
put word 3 of phrase -- Yorick
```

Bug Fixes / Tweaks:

- In an **each** expression, "item" is now assumed if no chunk type is specified. So, for example, the following are equivalent:

```
put each item of the keys of plist
put each of the keys of plist
```

- The Ask and Answer panels will now grow as needed to accommodate large amounts of text.
- Fixed a problem with the answer command improperly showing a Cancel button when not requested.
- Fixed a problem with auto-indentation of comments on the first line within a properties declaration.
- Fixed a problem with colorization of line continuation characters and following comments.
- Fixed a problem with user spacing not being respected following a block quote.
- Updated the SenseTalk Reference manual to include all recently added functionality.
- Fixed a problem with hyperlinks in the SenseTalk Reference table of contents.
- Fixed a problem with setting the HotSpot on an image that also specifies a ClipRectangle.
- Fixed a bug that would cause a lockup if the New Folder button on the Capture Image Sheet was used in Full Screen mode.
- Fixed a bug that could crash Eggplant when reconnecting to a VNC server with a new size.
- Added "SearchRect" as a valid image property (abbreviation for SearchRectangle).
- Fixed problems using Bug Reporting Panel
- Fixed a problem where Eggplant would crash when saving a script as it was closed.
- Fixed a problem with Eggplant segmentation faulting when passed empty parameters via Command Line on Mac OS X 10.4
- Fixed a problem with the "Select A Handler" popup in the Script Editor toolbar.
- Fixed a bug with repeat with each...by reference -- when using the same variable repeatedly it would sometimes skip items. Cleaned up some other potential problems that might have resulted in items being improperly added to the end of a list.
- Improved error messages, particularly those reported by the value() function for an invalid expression.
- Allow "a new property list" as a synonym for "a new object".
- Fixed a rare crash with loading a script file containing properties declarations while a script is running in another thread.
- Fixed a crash when opening a socket to an unknown host. The script will now throw an appropriate

exception in this case.

- Fixed a bug with copied objects in a list reporting an incorrect long id.
- Fixed a bug in callStack() that would erroneously list the line number as 0 in frames following a backEnd (non-script) frame.

Release v4.1 (25-February-2008)

Highlights:

- Markup support for all **Text Image Generators** and a new (TIG) to generate text images for use with Linux and other systems
- Enhanced **Keyboard Control**, with new KeyDown and KeyUp commands and simplified use of modifier keys.
- Full **XML** parsing and generation support.
- **Full Screen** mode for working with the SUT, including a new floating **Control Panel**.
- Integrated **Help Viewer**, with full searching across all documentation.
- Support for inserting **Text Image** property lists using the Script Editor.
- Eggplant 4.1 is compatible with Mac OS X **10.5 (Leopard)** and **10.4 (Tiger)** but is **not compatible with version 10.3 (Panther)**.

Text Image Generators:

- This release adds a new **Pango TIG** to generate text images for use with Linux and other systems. This TIG uses the open source Pango library to lay out and render the text. This is the same engine used by GNOME/GTK+, by FireFox and ThunderBird, and many other projects. PLEASE NOTE, however, that there is no single standard for rendering text on Linux systems -- each desktop environment, each development tool, and even each individual program may render text differently. With that said, the Pango TIG significantly extends Eggplant's text generation capabilities, and adds new options that weren't previously available.

  Installation. The Pango TIG is distributed as a separate bundle which must be installed on the Eggplant machine where it will be used. Download the appropriate Pango.bundle from http://www.testplant.com/downloads and copy it into the /Library/Frameworks/ directory on the machine.

  Setup. After installing the Pango bundle, bring up the Eggplant Preferences panel and select the Text pane. Click the Add... button to add a new Platform, and enter a name for the new text platform, such as "Linux" (or some other name of your choice). Then, in the Text Image Generator section of the panel, select "Pango (Linux)" from the popup list.

  Use. To use the Pango TIG, there are three options available to you: click the "Set As Default" button to make Linux your default text platform (not recommended unless that is the main platform you'll be working with); issue the command "set the currentTextPlatform to Linux" in any script prior to specifying text images; or include the property "textPlatform:Linux" in any text image property list.

  Text property lists may optionally include one additional property that is only recognized by the Pango TIG: the **antialias** property should be set to a boolean value (Yes or No, True or False, etc.) and controls whether the generated text will use antialiasing (by default, antialiasing is ON).

- **All three of the Text Image Generators** provided by TestPlant now provide the ability to include some text style changes within the generated text. To enable this feature, set the **useMarkup** property to true in any text property list, or set **the defaultUseMarkup** global property to true in your script. *[Note to beta testers:*

useMarkup replaces the rawPango property, which should no longer be used.]

When not using markup, the text value will be used literally on **all** TIGs, including the Windows TIG **(this was not true for Windows in the past, for strings containing '&' characters)**.

When using markup, the various TIGs provide support for markup as follows:
- All TIGs support using `&` for **&**, `<` for **<**, and `>` for **>** (the Windows and Native Mac TIGs allow '&' by itself, but this is discouraged in order to keep compatibility across platforms)

In addition:
- The Native Mac TIG supports standard HTML markup (and is fairly forgiving about missing or mismatched end tags, although it is recommended that you avoid relying on this behavior)
- The Pango TIG supports a more limited set of tags (and is strict about tag matching; plus, all tags must be in lowercase). The full description of the Pango markup codes can be found on this web page: http://library.gnome.org/devel/pango/unstable/PangoMarkupFormat.html
- The Windows TIG supports the tags <u> and </u> for underlining.

This markup capability gives you considerable control and flexibility, letting you include markup tags to set *individual parts* of your text in different fonts, colors, sizes, or styles. For example "This is cool !!" would represent the text "**This** is *cool* !!" for both the Mac and Pango TIGs.

- **Caching** was previously enabled by default. **Text Image Caching is Now OFF by default.** Because all three of the TestPlant provided TIGs operate quickly enough to be used on the fly scripts will generate TIG images as needed when run and as needed when reviewed in the logs.

User Interface:

- A new **Help Viewer** window (available from the Help menu) provides access to all of the Eggplant documentation. A search feature performs a search for text across the complete set of documents, greatly simplifying access to information. Help buttons (a question mark icon) have been added to some input panels and other places throughout the Eggplant interface, which provide direct access to relevant portions of the documentation.

- A **Full Screen** mode (available from the Control menu) zooms the active Remote Screen connection to occupy the entire screen, providing an "immersive" experience of working with the SUT directly.

- A new **Control Panel** (available from the Control menu) works with the active Remote Screen connection, providing an alternate means of accessing much of the functionality available from the Remote Screen toolbar. This panel floats above all other windows for easy access, and is especially useful when working in Full Screen mode. Control-click (or Right-click) the panel to customize the items which appear on the panel.

- A new **Text Image** item on the Insert... menu in the Script Editor (available on the toolbar and now also from the context menu in a script) supports inserting text image property lists in addition to using existing captured images. You may hold down the Shift key while accessing the Insert... menu to pre-select a particular command or function.

- The Script Editor sheet for inserting a command or function using an existing image (as well as the new sheet for inserting a text image) now include a pop-up list for selecting the command or function to be inserted.

- The **TypeText** command sheet (for generating a TypeText command from a Remote Screen window) offers a new "Insert Unquoted" option. Clicking this button will insert the command with the exact text entered in the panel without quoting it. This allows constants and predefined literals such as return, tab, controlKey, and so forth to be entered. So, for example, to generate a command to type Control-D you could enter `controlKey, "D"` in the panel and press the "Insert Unquoted" button.

- The pulldown list of font names displayed in the Font combo box (on both the Text Preferences pane and the Text Image Sheet when generating a text image command) now shows font family names rather than specific font names. These names are more readable (e.g. "Fang Song" instead of "SIL-FangSong-Reg-Jian") and also correspond more closely with the font names used on Windows and Linux systems (and the

TIGs for those systems). For Mac OS X generated text, font names are acceptable in either form.

Scripting:

- **XML Support.** This release includes a new **STTreeNode** XModule (bundle) providing full support for XML parsing and generation. This module adds to SenseTalk a new **tree** container type – a hierarchical data structure. A tree behaves as both a list and a property list (with some restrictions). As a list, a tree contains items – its children, sometimes called "nodes" – which are also trees. As a property list, a tree has properties which correspond to "attributes" of a node in XML terminology.

  In addition to the tree container type, the STTreeNode XModule adds the following features:

NOTE: all of the XML tree-related features described below are only available when the STTreeNode xmodule is loaded, which normally happens automatically when Eggplant is launched - the loadedModules() function may be used to verify its presence.

  - Added a **treeFromXML()** function, which can be used to parse XML data and return a tree.

  - Added a **tree()** or **asTree()** function, which can also be called using the special syntax **as {a} tree**. This can be used to convert a value into a tree data structure. XML data can be loaded as a tree from a file or URL with a single statement:

```
put url "http://some.address/data.xml" as a tree into myTree
```

  When **tree()** or **as a tree** is called with a parameter that is a property list that has an **asTree** property, the value of that property is used. If the property list / object has an **asTreeExpression** property, the value of that property is evaluated as an expression (equivalent to calling the treeFromXML() function) to obtain the tree value. If the object has neither of these properties, an **asTree** function message is sent exclusively to the object and its helpers to obtain the tree value.

  If the object doesn't supply a tree representation of itself in any of the above ways, it is taken to be a direct property list representation of a tree structure or a node. The property list may include these properties (and values): **_tag** or **_element** (tag name of an element); **_attributes** (attributes of an element node); **_children** (list of child nodes); **_text** (contents of a text node); **_comment** or "**--**" (contents of a comment node); **_processingInstruction** or **_pi** (contents of a processing instruction node); "**?**" followed by processing instruction name (body of a processing instruction node); **_XMLinfo** (property list of special XML document attributes).

```
put ( _tag:book, _children:"The Rose" ) as tree
                                    -- <book>The Rose</book>
```

  A simplified format can also be used:

```
put (book:"The Rose") as tree -- <book>The Rose</book>
```

  A simple element node with attributes but no content::

```
put (_tag:"pg", _attributes:(id:43)) as tree -- <pg id="43"></pg>
```

  A simplified format can also be used in this case:

```
put (_tag:"pg", id:43) as tree -- <pg id="43"></pg>
```

  If the target is not an object (and is not already a tree), the target's string value is evaluated as XML to obtain the tree value.

  - Added a **node** expression which will return a single node of a tree that matches an XPath expression:

```
put node "sales/month[2]" of orders into FebruarySales
```

  Use **all nodes** to return a list of every node of a tree that matches an XPath expression:

```
put all nodes "sales/month" of orders into monthlySales
```

A **node** expression (but NOT **all nodes**) can be stored into:

```
put 4 into node "product/quantity" of invoice
```

- Added a **nodePath()** function, which takes a tree node (or reference to a node in a tree) as a parameter and returns the path to that node.

- Added **the treeFormat** global property: a property list with property **prettyPrint** which defines whether trees are displayed nicely formatted (the default). Set this value to false to turn off pretty formatting of trees when they are displayed as XML text. The treeFormat also includes a **useStandardFormat** property. If set to true, trees will always be converted to property lists using a standard format, with _tag, _attributes, and _children properties. When set to false (the default) converting a tree into a property list (using "as property list" or "as object") will result in a simplified property list format being used for nodes that have a tag that isn't a reserved property name and/or don't have any attributes that are reserved property names.

```
set the treeFormat's prettyPrint to false
```

- Added **the treeInputFormat** global property: a property list with property **alwaysJoinText** which defines whether text nodes are combined automatically whenever a change is made to a tree that results in two text nodes adjacent to each other. Setting it to **false** allows sequential text nodes to be present in trees which may be useful for some applications, but may prevent Xpath node access from working properly. The default setting is **true** which is needed to ensure that node access (using a **node** or **all nodes** expression works reliably.

```
set the treeInputFormat's alwaysJoinText to false
```

- The **is a** operator can be used to test whether something **is a tree**, and may also be used to test for specific XML node types (e.g. document, element, comment, text, or processingInstruction):

```
if item 3 of myTree is a comment then delete item 3 of myTree
```

- **Enhanced Keyboard Control**. This release introduces additional keyboard control in the form of two new low-level commands (KeyDown and KeyUp) plus a simplified way to use modifier keys in the TypeText command.

- The **TypeText** command has been simplified with the addition of new codes for the temporary use of modifier keys. Previously, separate codes were always used for pressing and releasing the modifier keys. Using the new generic codes, TypeText will release the modifiers for you. So, for example, to send Ctrl-Alt-Del to a Windows machine, you can now do this:

```
TypeText controlKey, windowsAltKey, deleteKey
```

  This is equivalent to the earlier (and still supported) statement:

```
TypeText controlDown, windowsAltDown, deleteKey, controlUp, windowsAltUp
```

  More specifically, when one of the new generic modifier key codes (such as controlKey) is used with the TypeText command, the modifier key will be pressed down at the point where that code is encountered, and will continue to be held down for the remainder of key presses within that TypeText command. Any modifier keys pressed in this way are then automatically released at the end of the command. The generic key codes for the modifier keys are: shiftKey, controlKey, optionKey (or altKey), windowsKey and windowsAltKey (for Windows), and commandKey (for the Mac).

- With the new auto-release modifier keys we are now deprecating **TypeCommand.** The command will continue to work as it did in Eggplant 4.0 but all sequences can now be more consistently generated using the **TypeText commandKey** syntax:

```
TypeText commandKey, "Q"
TypeText commandKey, DeleteKey
```

- The internal values assigned to the special key names used with the TypeText command (and with the new KeyDown and KeyUp commands) have been changed. Previously, these names (like deleteKey) all represented sequences of characters beginning with a backslash (\) as shown in Appendix C of the Eggplant Reference manual. Now they have been assigned unique one-character codes, using unassigned Unicode values. When displayed by Eggplant or recorded in the log file for the run, they appear in a user-friendly format, like this: 「deleteKey」. The backslash codes may still be used, but won't be displayed in the new format.

- The new **KeyDown** and **KeyUp** commands work almost identically to the TypeText command, except that where TypeText depresses and then releases each key (with the exception of a few special "down" and "up" codes used for modifier keys), the KeyDown command will depress each indicated key and leave it down, and the KeyUp command will release each indicated key. This makes it possible to perform certain operations in which a non-modifier key must be held down. For instance, to test the cool dictionary feature in Mac OS X Tiger (which requires holding down the Command, Control, and "D" keys while pointing at a word), you could do something like this:

```
KeyDown controlKey, commandKey, "d" -- press the magic keys
MoveTo "someWord" -- point at a word
WaitFor 8, "theDefinition" -- wait for the definition to be displayed
KeyUp controlKey, commandKey, "d" -- release the magic keys
```

  To make it easy to ensure that no keys are still being held down at some point in your script, there are three special codes that can be used *only* with the **KeyUp** command: allKeys to release all keys that may be down; allModifierKeys to release only the modifier keys (shift, control, option, command) that are down; and allNonModifierKeys to release any keys other than the modifier keys:

```
KeyUp allKeys -- make sure nothing is being held down
```

- The **getOptions()** function has been enhanced to allow specifying the name of a group of options in order to retrieve the values of all of the options in that group at once. Three group names are currently recognized: "runOptions" returns the options that can be changed from the Run Options pane within Eggplant's preferences panel (this is also the default set of options that are returned when no parameter is given); "textOptions" returns options associated with Text Image generation; and "otherOptions" returns several other options. You can also pass more than one group name as parameters, for a combined set of options. So, for example, you could save most commonly-changed options for restoring later using something like this:

```
put getOptions("runOptions", "otherOptions") into savedOptions

-- do stuff here involving changing some options

setOptions savedOptions -- restore previous settings
```

- Added an **asList()** function, which can also be called using the special syntax **as {a} list**. When the target is an object which has an **asList** property, the value of that property is used. If the object has an **asListExpression** property, the value of that property is evaluated as an expression (equivalent to calling the value() function) to obtain the list value. If the object has neither of these properties, an asList function message is sent exclusively to the object and its helpers. If the target is not an object (and is not already a list), the target's string value is evaluated as an expression to obtain the list value.

- Added an **asObject()** function, which can also be called using the special syntax **as {an} object** or **as {a} property list**. When the target is not already an object / property list (or a tree), its string value is evaluated as an object expression (equivalent to calling the value() function) to obtain the object value. If the target is a tree, it is converted to a property list representation of that tree, as governed by **the treeFormat**'s **useStandardFormat** property.

- Enhanced the **ask** and **answer** commands to accept a timeout parameter. If the user hasn't responded before the timeout expires, the panel will automatically be dismissed. In this case, `it` will be set to empty, and the result will be "timeout". Both commands also include an "icon" option to specify the name or path of an icon to display on the panel in place of the regular application icon.

```
ask "Enter your name:" icon "/tmp/face.tiff" timeout 2 minutes
```

  The **ask** command may also include a "hint" and a "message". The hint or placeholder is text that appears in light gray in the input field any time it is empty and not selected. The message is additional text displayed below the title on the panel, similar to the answer command.

  The **answer** command may now include names for up to 4 buttons rather than the 3 that were allowed previously.

- Added **hostName()**, **hostNames()**, **hostAddresses()**, **machine()**, **OSInfo()**, **platform()**, **processor()**, **systemInfo()**, **systemVersion()**, and **userInfo()** functions to provide information about the machine on which SenseTalk is running and the current user.

- Added a **specialFolderPath()** function to return the path to a number of special folders. Call this function

with the name of a special folder to get the path to that folder. Folders supported include: home, system, library, applications, demo applications, developer applications, admin applications, developer, users, documentation, documents, core services, desktop, caches, application support, fonts, preferences, temporary (or temp), root, or the name of a user preceded by a tilde (~) to get the path to that user's home folder. An optional second parameter may be given to specify the domain for the folder. The domain may be one of: user, local, network, system, or all (note: when a domain of "all" is given, more than one path may be returned, as a list). If no domain is specified, a domain appropriate for the folder being requested will be assumed.

```
put specialFolderPath("applications") -- "/Applications"
put specialFolderPath("~brenda") -- "/Users/brenda"
put specialFolderPath("library", "local") -- "/Library"
```

- Added an **availableStringEncodings()** function to return a list of the names of all the available string encoding formats. In some cases there may be more than one name for the same encoding.

- Added a **capitalized()** function to return a string with the first letter of each word capitalized.

- Added **lowercase()** and **uppercase()** as preferred synonyms for **toLower()** and **toUpper()**.

- Added **UTCOffset()** as a preferred synonym for **secondsFromGMT()**. UTC (Coordinated Universal Time) is the preferred modern designation for standard time, rather than the historical term GMT (Greenwich Mean Time). [see http://en.wikipedia.org/wiki/Coordinated_Universal_Time]

- Added a **simple date** format (and **long**, **short**, and **abbreviated** variations), which omit the year:
```
put the simple date -- 01/04
put the long simple date -- January 4
put the short simple date -- 1/4
put the abbrev simple date -- Jan 4
```

- Added a **basic date** format (and **long**, **short**, and **abbreviated** variations), which include the year:
```
put the basic date -- Jan 4, 2008
put the long basic date -- January 4, 2008
put the short basic date -- Jan 4 08
put the abbrev basic date -- January 4 08
```

- Added a **basic time** format (and **long**, **short**, and **abbreviated** variations), which include the time:
```
put the basic time -- Jan 4, 2008 03:20 PM
put the long basic time -- January 4, 2008 03:20:26 PM
put the short basic time -- Jan 4 08 3:20 PM
put the abbrev basic time -- January 4 08 03:20:26 PM
```

- Added a **common date** format (and **long**, **short**, and **abbreviated** variations):
```
put the common date -- 1 Jan 2008
put the long common date -- 01 January 2008
put the short common date -- 1 Jan 08
put the abbrev common date -- 01 Jan 2008
```

- Added a **common time** format (and **long**, **short**, and **abbreviated** variations):
```
put the common time -- 1 Jan 2008 03:20 PM
put the long common time -- 01 January 2008 03:20:26 PM
put the short common time -- 1 Jan 08 3:20 PM
put the abbrev common time -- 01 Jan 2008 03:20:26 PM
```

- Added a mechanism for providing custom (user-defined) predefined variables. At startup SenseTalk looks for application or bundle resource files with the extension ".predef" and loads them. Files with this extension should be plain text files containing an archived property list. The SenseTalk value() function will be used to read this property list and register its keys and values with the SenseTalk engine as predefined variables.

  If two resources provide values for the same predefined variable name, the last value loaded will be the one that is used for that variable. Since the value being loaded, however, is a SenseTalk expression, it is possible to check for an already-defined value and use it instead, or incorporate it into the new value.

- Added a number of new predefined variables, including some symbols in the following categories (a predefined variable called **predefsLoaded** contains a list of the categories that were loaded):

  Common symbols:
  ellipsis (…), **hyphen** (-), **nonBreakingHyphen** (-), **figureDash** (-), **enDash** (–), **emDash** (—), **dagger** (†), **doubleDagger** (‡), **bullet** (•), **triangularBullet** (‣), **nonBreakingSpace** ( ), **atSign** (@), **careOfSign** (℅), **serviceMarkSign** (℠), **telephoneSign** (℡), **tradeMarkSign** (™), **facsimileSign** (℻), **numeroSign** (№), **invertedExclamationMark** (¡), **invertedQuestionMark** (¿), **verticalBar** (|), **brokenBar** (¦), **sectionSign** (§), **copyrightSign** (©), **registeredSign** (®), **pilcrowSign** or **paragraphSign** (¶), **middleDot** (·), **cedilla** (¸), **leftDoubleAngleQuotationMark** («), **rightDoubleAngleQuotationMark** (»)

  Currency symbols:
  centSign (¢), **poundSign** (£), **currencySign** (¤), **yenSign** (¥), **euroSign** (€), **dollarSign** ($)

  Numeric and mathematical symbols:
  percentSign (%), **perMilleSign** (‰), **perTenThousandSign** (‱), **degreeSign** (°), **superscriptOne** (¹), **superscriptTwo** (²), **superscriptThree** (³), **microSign** (µ), **plusSign** (+), **minusSign** (–), **multiplicationSign** (×), **divisionSign** (÷), **plusOrMinusSign** (±), **minusOrPlusSign** (∓), **squareRootSign** (√), **cubeRootSign** (∛), **infinitySign** (∞), **notSign** (¬), **equalSign** (=), **almostEqualSign** (≈), **approximatelyEqualSign** (≅),

  notEqualSign (≠), **lessThanOrEqualSign** (≤), **greaterThanOrEqualSign** (≥), **fractionOneQuarter** (¼), **fractionOneHalf** (½), **fractionThreeQuarters** (¾)

  Keyboard key symbols:
  commandKeySymbol (⌘), **optionKeySymbol** (⌥), **controlKeySymbol** (^), **shiftKeySymbol** (⇧), **eraseRightKeySymbol** (⌦), **eraseLeftKeySymbol** (⌫), **escapeKeySymbol** (⎋), **returnKeySymbol** (⏎), **ejectKeySymbol** (⏏)

  ```
  put copyrightSign && "2008" -- © 2008
  ```

- Added **the resultHistory** global property to return a list of information about the most recent non-empty settings of the result. This allows access to non-empty results after **the result** for an operation is no longer available. The resultHistory is kept trimmed to no more than **the resultHistoryLimit** most recent items.

- Enhanced the **repeat with each** syntax to automatically assume items and use the variable **it** when no chunk type or variable name is specified in the abbreviated form of repeat. So for example the following are now equivalent:

  ```
  repeat with each item of the keys of myPlist -- the old way (still works fine)
  repeat with each of the keys of myPlist -- new simpler syntax
  ```

- Fixed a problem with information being lost when converting some time values from text. When set from a property list (including automatically from the timeFormat global property) **the timeInputFormat** global property now automatically keeps shorter forms of the same format ordered after longer (more detailed) forms, in order to parse each input as fully as possible.

- Enhanced the **seconds()** function to accept a date as parameter, and return the seconds since the start of the millennium for that date.

- **When referring to a property list, the term "item" is now assumed to refer to *list* items rather than *text* items**. This simplifies working with a list of property lists, and should avoid some confusion when working with a property list. In the unlikely event that the old behavior is needed, explicitly specify "text item" rather than "item".

- Added a **handlerNames()** function, and removed "handlerNames" as a special property of an object.

- **Changed the script special property of an object to no longer be a "hidden" property**. This allows

"script" to be treated more like a normal property of a property list. The script property is still "restricted" in that its value is always a string. And of course if you send messages to the object, the script defines its behavior (so, for example, you're perfectly welcome to set an object's script to a number, but if you then try to "run" that object, you'll get a syntax error).

- The **replace** and **delete** commands now set a result (as returned by the result() function) to provide information about the number of occurrences that were replaced or deleted.

Bug Fixes / Tweaks:

- Eggplant has Rich clipboard turned off by default, it can be enabled in Preferences->VNC but we recommend that it remain off when using the Clipboard for testing.
- Passwords in the Connection List are now stored in an encrypted format.
- Drag and drop of text is now supported in the Script Editor.
- Fixed a problem where we continued to load more and more watchers on a result directory, causing massive slowdowns in extreme cases.
- Fixed a 10.5 problem where we would stop receiving notice of updates to scripts and suites.
- Fixed a problem where we would register for updates on scripts that weren't visible in the GUI.
- Fixed a memory leak with the **CaptureScreen** command.
- Fixed a problem with the ConnectionInfo() function returning inappropriate information when there was no connection.
- Fixed a problem with the ConnectionInfo() function not accepting a parameter.
- Fixed a bug in which the SearchRectangle would sometimes get set to an invalid rectangle upon connecting to a SUT, which in certain cases could result in Eggplant being unable to find any images on the SUT screen.
- Improved error reporting when an image is not found: if the SearchRectangle is set to any area less than the full screen, its setting is included in the report.
- Improved error reporting when loading data from a **URL**. In particular, if the data can't be converted to a string using **the defaultStringEncoding** an appropriate error will be reported rather than simply returning empty.
- Fixed a problem with **each** expressions not producing a nested list of lists when each value was a list, but instead (incorrectly) resulting in a single concatenated list.
- Fixed a problem with putting a property list into a file, url, or text chunk. This will now store the property list as text, rather than throwing an STIncompatibleValueException exception.
- Fixed a problem with quoting of text containing quotes and ending in ">".
- Added STXT as a recognized text type in values returned by a call to do AppleScript.
- Fixed a crashing bug when sorting a single (non-list) value as list items.
- Fixed a crashing bug (with "validateToolbarItem") involving certain toolbar customizations.
- Fixed a bug that would result in certain scripts failing to parse.
- Fixed a crash with sometimes completing the Generate Script Command Panel
- Fixed crashing bugs with ImageDoctor and changing attributes
- Fixed exceptoins with Make Collection button when capturing image
- Added sorting of Scripts by Name, Mod Date or Size in Suite Panel
- Removed sub-suites from being searched when using the OpenSuite command.
- Fixed the imageColorAtLocation() function, which would throw an exception
- Fixed a problem with the everyImageLocation() function reporting the wrong image name when no occurrences were found.
- Fixed a problem with the message displayed in Quit dialog when a script abort has been requested.
- Changed the LogWarning command to log its name in lowercase like other commands.
- When a new object is created, the built-in makeNewObject() function now sends the "initialize" message exclusively to the new object (and its helpers), not through the full message path.
- Fixed a problem with directly accessing the properties of an object's helpers.
- Fixed a hang when formatting some incomplete scripts (e.g. ending with "to" on a line by itself).
- Fixed a crashing bug with "delete any x in y" or "replace any x in y" when x does not appear in y.
- Fixed a crashing bug that could occur when converting a recursively nested list or property list to text.

- Fixed a crashing bug with "repeat with each item delimited by x of it" (in cases where a delimiter was explicitly supplied and the repeat variable is the same as the variable being iterated over).

Release v4.01 (03-July-2007)

Highlights:

- This maintenance release corrects a number of bugs and minor issues.

User Interface:

- The detailed log title on the Results tab now includes the date and time of the run, and also indicates whether log filtering (search term and/or log entry type) is in effect.

Bug Fixes / Tweaks:

- Improved the log search/filter behavior on the Results tab when switching logs or filter types.
- Choose a suitable script/suite context when running commands from the Ad Hoc Do Box before any script has been run.
- Disabled Image Doctor behavior when running commands from the Ad Hoc Do Box or generating a script command.
- Disabled Image Doctor behavior when working with generated text images, which can't be properly doctored.
- Fixed a problem with making image corrections and then resuming a script when using the ImageDoctor set to "manual" (script was not always using the corrected image).
- Added the ability to cancel loading externally edited scripts and added the ability to set it to always reload.
- Added the ability to ignore the RTF to plain text conversion warning.
- Fixed a problem where CLI execution would use the GUI license first instead of execution licenses.
- Fixed a problem with the Image Doctor not displaying a Helper suite containing the image to be doctored.
- Fixed a problem with the Image Doctor panel being left up if it is not dismissed by one of its action buttons.
- Fixed a display problem with image Discrepancy values that were entered as a number of pixels rather than a percent.
- Fixed a display problem when the Mouse Right Click Key preference was set to "None".
- Fixed a problem with the "Automatically upgrade suites..." preference checkbox not saving properly.
- Fixed the saving behavior on upgrades to only save scripts that were changed by the upgrade.
- Fixed a bug where Schedules would not be run if the first item was marked as disabled.
- Fixed a bug where New Script... and Delete Script would stop working on a suite upgraded that session.
- Fixed a bug when Image Doctor's Search button was clicked with no active connection.
- Fixed a bug where the Text column of the Log view could go away.

Release v4.0 (19-June-2007)

Highlights:

- **Image Doctor** functionality aids in diagnosing image-finding problems.

- **Auto Doctor** recovers from image-finding problems during script execution.
- **Image Collections** allow multiple image representations to be stored and accessed as a group (by folder).
- **Image Import** enables bringing images into a suite from external sources. Multiple image formats are now supported.
- **Log Search** enables searching and filtering of log entries on the Results tab.
- **Image Search** field allows you to locate images on the Images tab by name and/or description.
- **Image Highlighting** in the Remote Screen window shows you where Eggplant actions are occurring.
- **Open Process** command enables bidirectional communication with another process for better interaction and control.
- Better control over dynamic use of suites with two new properties: **the initialSuites** and **the finalSuites**.
- New **Drag** and **MoveTo** behaviors and a new **Drop** command simplify generating drag operations in a script. Automatic upgrading of older scripts enables all scripts to make use of the new commands.
- Additional image search capabilities: individual color tolerances, and pixel discrepancy.
- Enhanced list operations in scripts with new **each** expressions and **where** clauses.
- **Vine Server 2.2** is included with this release. It is recommended for use on all Mac SUTs.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.

Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- New **Drag** and **MoveTo** behavior. The functionality of the **drag** and **moveTo** commands have been updated and improved in connection with the introduction of image collections. To facilitate this change with minimal impact on old scripts, two new commands – **dragAndDrop** and **moveToEach** – have been introduced that replicate the old behavior. An automatic upgrade mechanism (after user confirmation) has been created to update suites to use the new behavior. Suites that haven't been updated will continue to work using the old behavior. The upgrade will fix all common cases, but could be fooled by use of drag or moveTo commands inside a **do** or **send** command (all *extremely* unlikely, but if you use **do** a lot, please be aware of this possibility).

- Interpretation of dates is more restrictive. The natural language interpretation of dates in earlier versions of Eggplant/SenseTalk was overly aggressive about recognizing things as dates, which caused problems. By default, the natural language processing is now turned off. See the description of **the timeInputFormat** below for details on how to restore the old behavior if necessary for your scripts.

- Calling **new <objectName>** will now copy properties of the prototype object. The behavior of SenseTalk's **new** expressions has been changed to automatically copy the properties of the prototype object into the newly created object. Our customers have told us that this is really the expected behavior, and we agree. However, any scripts which rely on new objects being empty (the old behavior) will need to be modified to use **new empty <objectName>** instead of simply **new <objectName>**.

- Calling functions vs. accessing properties using the dot (.) or apostrophe-s ('s) syntax and parentheses. The mechanism for passing optional parameters when using dot or apostrophe-s syntax has been clarified and made more consistent. Using parentheses after an identifier now *always* signifies a function call (previously parentheses could also be used in accessing a property). **It is unlikely that any scripts will be affected by this change**. However, if a script uses an expression like `the connectionInfo's status()` to access the status property of the connectionInfo property list, for example, the parentheses will have to be removed for it to continue to function properly. See the scripting section below for more information.

User Interface:

- **Script Auto-Upgrade.** When the new version of Eggplant is first launched, if you have used earlier versions you will be presented with a panel requesting to update the scripts older suites to Eggplant 4.0 standards. If you choose to have suites upgraded automatically, each time you open an older suite the auto-upgrade process will examine each script in the suite, changing any **drag** and **moveTo** commands to the new **dragAndDrop** and **moveToEach** commands. The suite will then be marked as upgraded, so the process won't be repeated.

  If you choose to upgrade suites manually, the scripts in older suites will be left unchanged and will be run in a compatibility mode. Scripts in these suites will not be able to use the new **drag** or **moveTo** functionality. The Settings tab in the Suite window shows the version of the suite, and provides an Upgrade button to allow you to upgrade an individual suite. There is also a checkbox located under "Other Options" on the General Preferences panel titled "Automatically upgrade suites to the current version" that can be used to reinitiate auto-upgrading of older suites the next time they are accessed.

- **Image Doctor.** When a script fails to find an image while the Image Doctor functionality is enabled, instead of immediately throwing an exception, a panel will be presented asking what you would like to do. At this point, you can do any of the following:
  - manually correct any problem that may have occurred on the SUT that is preventing Eggplant from finding the image, or,
  - if the image appears to be present on the SUT screen, you may invoke the Image Doctor to diagnose the situation and recommend corrective actions, or,
  - let the process proceed, most likely resulting in an "Image Not Found" exception.

  The Image Doctor's diagnostic tools can also be invoked manually for any image, from the new Image Doctor section on the Images pane of the suite window. Just select an image and click the Search button at the bottom of the pane (the Search button changes to "Cancel" while a search is in progress, allowing you to cancel the search). Whether it is invoked manually, or as the result of an image search failure during script execution, the Image Doctor will use a number of different approaches to search for the image on the Remote screen.

  A **Standard Search** (using the image's current search settings) will be performed.
  A **Dynamic Tolerance** search will look for the image with a range of color tolerance values.
  An **Alternate Types** search will look for the image using different search types (Text, etc.)
  A **Discrepancy Search** will attempt to find the image by allowing for an increasing number of incorrect pixels.
  In addition, the **Original Location** where the image was captured will be shown.

  Each diagnostic will use a different color to highlight any locations on the Remote screen where it detected a possible match for the image. By clicking an individual diagnostic, the match locations detected by that approach will be highlighted in that diagnostic's color and will appear in front of the highlights for other approaches.

  Actions. With a particular diagnostic selected, click the Action pull-down menu at the bottom of the Images pane (or right-click or control-click on a diagnostic). The *Show Original Image* action will briefly display the image at the "found" locations on the Remote screen. This can be a useful aid in identifying what may be different between the captured image and what is currently visible on the screen. Depending on the diagnostic tool that is selected, and whether it was able to find any possible matches, the *Fix Image* action will make corrections to the image's tolerance or discrepancy settings, or will change its search type for you. You can also choose to capture a new image, either replacing the current image (*Recapture Image*), or adding a new representation (*Add Representation*) thereby turning the image into an image collection. The Edit Script action will open a script editor window to the current line in the executing script to facilitate making changes to the script, such as timing adjustments.

  If the Image Doctor was invoked during a script run, the script will be paused, waiting on your actions. You can make corrections to the image and retry the search operation in the script. To do this, click the Continue button in the Run window. The operation will be retried (using any new settings and additional representations for the image). If the image is found, the script will continue execution from that point. If the image is not found, the Image Doctor's exception panel will be displayed again, allowing you to decide how to proceed.

  *Availability Notice for **Eggplant Green**: The Image Doctor's diagnostics can be run manually in Eggplant Green but will not be invoked for you during script execution. The full capabilities are available only in*

Eggplant Purple.

- **Auto Doctor.** When a script fails to find an image while the Auto Doctor functionality is enabled, instead of immediately throwing an exception, the Image Doctor's diagnostic capabilities will be invoked to attempt to find the image using more extensive means. If a suitable match can be found, Eggplant will log a warning indicating the adjustments that were required in order to find the image, and the script will proceed. The image itself is left unchanged.

  *Availability Notice for **Eggplant Green**: The Auto Doctor capabilities are available only in Eggplant Purple.*

- **Image Highlighting.** While a script is running, each successful image search will briefly highlight the located image in orange on the Remote screen.

- **Image Discrepancy.** The Info drawer for an image now includes an additional field: Discrepancy. This indicates either the percentage of pixels, or the actual number of pixels in the image, that are allowed not to match and still treat the image as found. Setting this to anything other than zero (the default) can significantly slow down the search for the image. However, in certain rare situations this may prove to be useful. For example, consider the case of searching for a word in a text field. If there may be a blinking insertion point somewhere within the word being searched for, Eggplant might fail to find the image if it happens to look only when the insertion point is present. Allowing Eggplant to ignore up to 20 incorrect pixels might enable it to find the word every time.

- **Image Tolerance.** Eggplant image searches nearly always allow a certain amount of tolerance for color variations within an image. The Tolerance field shown in the Info drawer for an image displays the tolerance level for that image. Instead of entering a single value here, you may now enter three values separated by commas, to specify different tolerances for each of the red, green, and blue components of the color value.

- **Image Import.** A new Add button on the Images tab displays an Open dialogue sheet that allows you to select one or more images and/or folders containing images to import into the suite.

- **Image Type Support.** Images in any of several different formats can now be used by Eggplant. To use a pre-existing image, simply drag it into the Images tab of a suite. Image formats supported include: TIFF, ICNS, ICO, PICT, GIF, BMP, PNG, PDF, and JPG / JPEG. Not all variations of any given format can be used on all systems. In particular, on Mac OS X 10.3.9 (Panther), only images that are in a full RGB or RGBA format will work. On Mac OS X 10.4 (Tiger) most color format variations of the supported file types should be accepted.

- **Image Search.** The Images tab now includes a search field for selecting images. Simply type in the field to select images whose names (or optionally, descriptions) include the search text. The pulldown menu (the magnifying glass) can be used to change the type of search, in order to search only image names (the initial setting), search image descriptions, or search in both names and descriptions. Press return in the search field after entering a search term to record that search term in the pulldown menu for instant recall.

- **Log Search.** The Results tab now includes a search field for selecting lines in the detailed log. Simply type in the field to select rows for commands that begin with those letters, or containing those letters in the image or text fields. The pulldown menu (the magnifying glass) can be used to filter for Interesting entries (logs, errors, exceptions, etc.) or errors and warnings only. Press return in the search field after entering a search term to record that search term in the pulldown menu for instant recall.

- **Log Actions.** An Action pulldown menu (also available as a context menu) has been added for the detailed log on the Results tab. The commands on this menu perform actions based on the selected line in the detailed log. The "View Script Line" action will open the Script Editor and select the command that generated the selected entry in the log. The "Show Image" action will switch to the Images tab and select the image associated with the selected entry in the log. The "Apply Fix" action applies only to log entries that were generated by the Auto Doctor; choosing this action will permanently apply the correction made by the Auto Doctor to the image indicated on the selected log entry, altering one or more properties of that image.

- **Handler Popup.** The handler name popup in the Script Editor toolbar now displays handler names in mixed case as shown in the script.

- **Mouse Coordinates.** A new Cursor Location toolbar item is available for the Remote screen toolbar that will display the coordinates of the mouse on the Remote screen when in Live mode.

- **Create Image Collection.** There is a **Create Image Collection** button on the Save Image sheet in the Remote window to facilitate creating collections for existing images. If you highlight an image you can "upgrade" it to a collection (folder) with the existing image inside it.

- **Drop Item on Toolbar.** There is a new **Drop** toolbar item available for the Remote window. This toolbar item can be used along with the Drag and MoveTo toolbar items to generate the commands to carry out a complete drag operation.

Scripting:

- **Image Collections.** Wherever an image name is expected in a script, you can now pass an "image descriptor", which may take any of a number of forms: the name of an image, the name of an image collection, an image property list, a text image property list, or a list of several image descriptors. Of these, image collections and lists of images are new in this release.

  An Image Collection is actually a folder containing one or more images. Rather than specifying the name of an actual image, you can specify the name of the folder and Eggplant will look for any of the images in that folder and its subfolders. This capability may be used in a number of different ways. Typically, each collection will represent a single interface element that has a number of different appearances: selected or unselected, for Mac or Windows, in French or English.

- **Collection Filters.** By specifying a collection, Eggplant will search for any of the images in the collection, and can therefore find the element in any of its possible states. While this can be very convenient, it also comes at a price in performance, since Eggplant may have to do a lot of looking for things that aren't there before it gets around to finding the one that is there. To help with this, and to provide additional control, you can specify a collection filter that will select which image(s) from a collection should be used. You can set the global filter like this:

  ```
  set the collectionFilter to (name:"_german")
  ```

  This specifies that only images whose name contains the string "_german" should be used. For example, if you have an image collection called "OkayButton", it might contain images called "Okay_english", "Okay_english_selected", "Okay_german", "Okay_german_selected" and so forth. By setting the filter above, only the image containing "_german" would be used and the others would be skipped.

  A collectionFilter can also be specified as a property in an image property list:

  ```
  click (imageName:"MyImage", collectionFilter:(name:"german"))
  ```

  In addition to or instead of a "name" filter, you can also specify a "description" filter, which will check the description of each image to see if it contains a matching string. If the requested description value is present in the description, that image will be used, otherwise it will be rejected.

  NOTE: Because every Eggplant command and function that deals with images can now accept image collections and image lists, they now all have the functionality that was previously associated with the "Any" commands and functions (**ClickAny**, **AnyImageFound()**, etc.). For this reason, the "Any" form of those commands and functions are now considered to be obsolete: instead of using ClickAny, for instance, you can simply use Click and provide as many image descriptor parameters as needed.

- Added **the imageDoctor** global property to control how the Image Doctor behaves. When set to "Manual" the Image Doctor panel will be shown when an image match fails. When set to "Auto" the **Auto Doctor** will be invoked when an image match fails. The default is "Manual".

- Added **the initialSuites** and **the finalSuites** global properties for users to provide better control over dynamic suite usage. These two new properties are both user modifiable lists of suites that Eggplant will use as resources to locate images and other scripts. The suites listed in **the initialSuites** are checked first (in the order given) before the current suite, and those listed in **the finalSuites** are checked after the current suite. The **OpenSuite** and **CloseSuite** commands, and **Helper Suite** behavior are unchanged and will continue to work as they always have after **the initialSuites** and before **the finalSuites**.

- Changed the **drag** command to **dragAndDrop** and introduced a new **drag** command and a **drop** command. The new drag command only takes a starting location for the drag (from a single image, fixed location, or image collection, list, or multiple parameters). When it is executed, it moves the mouse to the specified location and puts the mouse button down. A subsequent **drop** command will drag the mouse to the drop location and release the button. Intervening **moveTo** commands can be used between the **drag** and **drop** commands to drag the mouse to intermediate points. The **dragAndDrop** command can be used to perform a complete drag and drop operation with a single command using the old **drag** behavior.

- Changed the **moveTo** command to **moveToEach** and introduced a new **moveTo** command. The new moveTo command works with image collections including image lists, but only moves to a single location. Additional moveTo commands can be used to move the mouse to a sequence of locations. The **moveToEach** command can be used to move the mouse to a series of locations with a single command using the old **moveTo** behavior.

- Added the ability to specify different amounts of **color tolerance** for the red, green, and blue components of each pixel color in an image search. To do this, in an image property list, specify a list of three values for the tolerance property rather than a single value:

```
click (image:"MyImage", tolerance:(10,30,30))
```

  You can omit a value to use the default tolerance for that component:

```
click (image:"MyImage", tolerance:(10,,30))
```

- Added the ability to allow a search to find an image on the SUT screen even when a certain number of pixels fail to match. To do this, use the new **discrepancy** property in an image property list, giving either the percentage of pixels within the image or the actual number of pixels that may be ignored. Using this option will usually result in a search taking more time to find (or fail to find) an image:

```
click (image:"MyImage", discrepancy:3%) -- allow up to 3% unmatched pixels
click (image:"MyImage", discrepancy:40) -- allow up to 40 pixel errors
```

- Added **each** expressions for working with lists. Any expression of the form "each *chunkType* of *sourceValue*" will yield a list containing all of the chunks of that type:

```
put each character of "Sweet!" -- ("S","w","e","e","t","!")
put each word of "Wisdom begins in wonder"
          -- ("Wisdom","begins","in","wonder")
```

  More interestingly, an each expression can be part of a larger expression:

```
put "Z" & each character of "Cat" -- ("ZC","Za","Zt")
put 2 + each item of "1,2,5,6" -- (3,4,7,8)
put the length of each word in "Wisdom begins in wonder"
          -- (6,6,2,6)
put each word of "Wisdom begins in wonder" begins with "w"
          -- (true,false,false,true)
```

  An each expression can also include a **where** clause to select a subset of the items in the list. The word **each** can be used within the where clause to refer to each source item:

```
put each word of "Wisdom begins in wonder" \
          where each begins with "w" -- ("Wisdom","wonder")
put each item of (1,2,3,4,5,6,7,8,9) where the square root \
          of each is an integer -- (1,4,9)
```

- Added the ability to open an external process and communicate with it bidirectionally. This is done through the **open process**, **write to process**, **read from process**, **close process** and **close all processes** commands. The **openProcesses()** function can be used to obtain a list of currently open processes. These commands are similar to the corresponding commands for opening, reading, and writing files or sockets, but require a process identifier rather than a file name or socket identifier.

  A process identifier should be the full path name to a unix command or (most often) a shell, such as "/bin/sh". The identifier may also include a hash mark "#" and a unique id string; this allows multiple process streams to be open to the same command at once.

```
put "/bin/sh#id2" into myProcess -- "id2" can be anything
open process myProcess
if the result is not empty then throw the result -- error
put openProcesses() -- ("/bin/sh#id2")
```

```
write "cd /tmp/processTest" & return to process myProcess
if the result is not empty then throw the result -- error

read from process myProcess until end
put it is empty -- true (there is nothing to read yet)
write "ls" & return to process myProcess
wait 5 milliseconds -- allow time for command to execute
read from process myProcess until end
put it into filesInProcessTestFolder

close process myProcess
```

- The **open process** command also includes a **with options** option. This allows you to pass a property list specifying any of these options: **folder** or **directory** (the current directory where the process will be run); **environment** (a property list specifying environment variables and their values); **parameters** (a list of values to be passed as parameters to the process when it is launched).

- Made the "." and "'s" syntax more consistent for calling functions vs. accessing a property of an object with optional parameters. Use of parentheses immediately following an identifier will now always signify a function call. To specify parameters with a property access, the word **with** must be used. This is a potential **COMPATIBILITY ISSUE** although it is not expected that many scripts will be impacted.
```
get anObject.someFunc(param1) -- this is now only a function call
get anObject.someProp with param1 -- property access with parameter
```

- Calling a specific handler of an object as a command (using the dot or apostrophe-s syntax) formerly required using the **run** command. The word "run" can now be omitted, calling the script's handler directly in a more natural way:
```
myScript's commandName -- the new way
run myScript's commandName -- the old way (still works fine)
```

- Using either the **run** command or by invoking a script name directly, the name of the handler to be run can now optionally be given as a quoted literal or as an expression enclosed in parentheses:
```
run anObject's ("some" & "Command") param1,param2
anObject's "tweedle" deedle, dum
```

- The **RunWithNewResults** command can now also be called as a function. The return value of the function will be the results property list. The following are equivalent:
```
RunWithNewResults "Test1","param1" -- the old way, as a command
put the result into test1Result

set test1Result to RunWithNewResults("Test1","param1") -- as a function
```

- Added **the strictFiles** global property to provide stricter usage of files at runtime. When the strictFiles property is set to true, reading a nonexistent file as a container will throw an exception rather than simply returning empty. This property is initially set to false. Setting it to true can aid debugging and catch some errors such as file names that were inadvertently misspelled, for example.

- Added **the URLErrorLevel** global property. Its value is an integer indicating the lowest URL status value that is treated as an error when fetching the contents of a URL. The default value is 400, so a returned status value of 400 or above will either throw an exception or set the result() function to an exception (depending on the current setting of **the throwExceptionResults** global property). You may set the URLErrorLevel to 0 (or to any sufficiently large number) to prevent this type of exception from being generated.

- When a URL is fetched with a status less than the URLErrorLevel, the result() function will now return the status value that was returned.

- The **CallStack()** function now includes MeObjectID and TargetObjectID properties for each frame, to identify the "me" object (the object being helped) and the target object (the object to which the message was sent). The ScriptObjectID property now always identifies the script (that is, "this object" -- previously it was identifying the "me" object, which is different when the script is running as a helper).

- Added the **tryDepth()** function to return the current level of nesting of try/catch blocks. This can be used to quickly determine whether a thrown exception will be caught at some higher level of the call stack: a return value of zero indicates there are no try/catch blocks in effect.

- Changed the **asText()** function (and synonymous **as text** operator) to force textual comparisons between values. Normally, if two values can be treated as numbers, SenseTalk will compare them numerically. The as text operator (applied to either value) can now override this behavior:

```
put "007" is equal to "7.0" -- true
put "007" is equal to "7.0" as text -- false
put "007".asText is equal to "7.0" -- false
```

- Enabled the **there is a** operator (and its synonyms) to be used to check whether a variable "exists" (has been assigned a value):

```
put variable foo exists -- false
set foo to 12
put there is a variable foo -- true
```

  This can also be used to check for global and universal variables:

```
if there is not a universal truth then set universal truth to "???"
```

- Changed the **makeNewObject** function (invoked internally by the **new** operator) to copy all standard (non-special) properties of the prototype object into the newly-created object. This gives a more useful result in most cases, but may cause compatibility issues for some scripts. A new object can be created that doesn't inherit properties from the prototype object, by including the word **empty** after the word **new**:

```
set car to a new empty Automobile with (make:"Ford")
```

- Added **the timeInputFormat** global property. This property is a list of date and/or time formats that specifies the possible formats in which text strings can be interpreted as dates or times. The value of this property is initially linked dynamically to the values of **the timeFormat** global property, so any changes to the formats available there will be reflected in the timeInputFormat as well. To restore this dynamic relationship after setting the timeInputFormat to some other value, simply set the timeInputFormat to empty. COMPATIBILITY ALERT -- SenseTalk no longer does natural date recognition by default (set the timeInputFormat to "Natural" to restore the old behavior).

  When setting the value of the timeInputFormat, you may set it to a single format value, to a list of format values, or to a property list containing format values. In the latter case, the values from the property list (in sorted key order) will be used. This makes it easy to allow all of the specific formats defined in the timeFormat global property:

```
set the timeInputFormat to the timeFormat -- many specific formats
put "g4" is a date -- false (doesn't match any of the formats)
put "g4" + 3 minutes -- throws an exception
```

  The value "Natural" is a special setting which specifies that multiple input formats as well as natural language processing will be used to arrive at a "best guess" of the value. This was the old behavior, which in some cases may be overly aggressive about treating values as dates/times:

```
set the timeInputFormat to "Natural" -- this is the old behavior
put "g4" is a date -- true
put "g4" + 3 minutes -- 2007-05-04 12:03:00 -0600
```

  Because the timeInputFormat is a list, you can easily add additional formats using the insert command:

```
insert "%d/%m/%y" before the timeInputFormat -- assume d/m/y order
insert "Natural" into the timeInputFormat -- allow natural, last
```

- All of the verbose date and time functions can now accept an optional parameter using "of" to specify a date/time other than the present. When called with an adjective ("long", "short", or "abbreviated") the word "the" can now be omitted:

```
put the internet date of "6/23" -- 23 Jun 2007 12:00:00 -0600
put long date of "12/21/04" -- Tuesday, December 21, 2004
```

- Added a **textDifference()** function to calculate the "Levenshtein Distance" between two words or phrases. This is one measure of how different two text strings are from each other. If the two values are equal, their textDifference will be zero. If one character must be changed to a different character, or one character

inserted or deleted, in order to turn one string into the other, then the difference will be 1. A greater number indicates a greater difference between the two strings. The largest number that will be returned is the length of the longer string, which would indicate that the two strings are completely different. This function is case insensitive, unless the caseSensitive global property is set to true.

```
put textDifference("hobbit", "hobo") -- 3
```

- Changed socket identifiers to use "#" rather than "|" as the separator for a unique id string (socket identifiers using "|" will still work, but "#" is recommended in the future).

- The behavior of the "read ... until end" command has been changed slightly for sockets and processes. If no input is immediately available from that socket or process, this command will now continue to wait until either some input is available, or the duration of the readTimeout has elapsed. This greatly simplifies reading when some input is expected:

```
read from process myProc until end -- waits for some response
```

- Changed references to not carry through on assignment. If x is a reference to var1, resetting x as a reference to var2 will now redirect x, it will not affect var1. To explain this change in more detail: previously, setting x to refer to var2 would actually set var1 to refer to var2, with x still referring to var1 and therefore (indirectly) to var2; now, x will be reset to refer directly to var2, without affecting var1.

  If the old behavior is needed, a new syntax allows explicitly setting the value referred to: (**the**) [**value | container**] [**referred to | referenced**] **by**. For the example above, the old behavior could be achieved as follows:

```
set x to be a reference to var1
set the container referred to by x to refer to var2
```

- Added a single-line variation of **try** that automatically catches any exceptions thrown by a single statement. If an exception is caught, it is stored in the global property **the exception** which is otherwise set to empty:

```
try to set x to 12*y -- (will fail if y is not a number)
if the exception is not empty then LogError the exception
```

- The new global property **the exception** is also available in the catch portion of a try/catch block (or at any time until the next **try**, which will reset it to empty):

```
try
    set x to 12*y -- (will fail if y is not a number)
catch
    LogError the exception
end try
```

- Added a **frac()** function to return the fractional part of a value, such that frac(x) + trunc(x) is always equal to x.

- Added **positive number**, **negative number**, **positive integer**, and **negative integer** as types that can be checked for with the **is a** operator. They return true if the value is both a number and is greater than -- or less than -- zero. For positive and negative integers, the number must also be an integer:

```
put -8.7 is a negative number -- true
put -8.7 is a negative integer -- false
```

- Added **the duplicatePropertyKeyMode** global property to control the behavior when a property list is specified that contains duplicate keys. The default setting of this property is "error", which will throw an exception if a duplicate key is encountered. Other options are "first" or "last" which will take only the first or last value given for a particular key, ignoring all others; or "list" which will accept all values for a duplicated key as a list.

- Added **rounded to** and **rounded to nearest** operators to provide a natural syntax for calling the round() and roundToNearest() functions.

Bug Fixes / Tweaks:

- Eliminated "Zombie" connection windows.

- Fixed some memory leaks on remote connections.
- Fixed a bug when opening Eggplant by double-clicking a script or dropping an Eggplant document on the icon.
- Fixed a bug where Text Images that were set to Trim:No didn't generate a new version if they were cached.
- Made a fix to properly terminate processes when closed, and to close all processes at the end of a script run.
- Fixed a problem with the cache clearing mechanism that could sometimes fail to reload an object's properties.
- Fixed a bug with abs() when called with a value of negative zero (really!).
- Fixed a bug with script error exceptions.
- Fixed a bug in which predefined literals would be passed as parameters by reference.
- Fixed a bug in which a property list turning into a list could cause a hang or crash.
- Fixed a bug with deleting a customized setting for the results directory on the Settings tab.
- Fixed some problems with attempting to update toolbar items improperly from a secondary thread.
- Fixed a bug with image captures.
- When generating a TypeText command, the sheet on the Remote window now properly handle entries with a return in them (entered using Option-Return).
- Fixed a crashing bug that could happen on occasion when hitting a dynamic breakpoint.
- Fixed context menus in the Connection List.
- Fixed a problem with the use of pure black or white in generated text images that would throw an exception.
- Fixed several clipboard transfer bugs relating to the active window.
- Fixed a bug where clipboard transfers were not registered.
- Fixed a bug where clipboard transfer could be blocked by cursor updates.
- Fixed a bug where dead connections could crash eggplant.
- Fixed a bug where Eggplant would lock up on a clipboard "Cycle" between SUTs.
- Fixed a logging bug where scripts started by runscript from the command line didn't record a "START" line.
- The word "from" is now allowed as an alternate instead of "in" or "within" in **delete** commands of the form "delete every x from y".
- Fixed the **folder()** function to return empty and set result() when called on an object or property list other than a script file object or a fileDescription property list.
- Fixed the **value()** function to return empty when called with a parameter that is empty or all whitespace, rather than throwing an error.
- Improved the performance and fixed some resource problems when reading from an open file or socket.
- Fixed occasional problems with connecting to a socket.
- Fixed a small but significant memory leak that occurred in every message send.
- Fixed some problems with the "as parameters" construct for passing a list as individual parameters in certain contexts.
- Improved parsing of RTF text, when running in a host application that uses AppKit. This applies both to rich text script files, and to the rtfToText() function. Some issues may still exist when running SenseTalk outside of an AppKit-based application.
- Removed some debugging console logs.
- Fixed a bug with the Connection List not always showing all available Bonjour servers.
- Fixed a problem with entries in the Connection List not always indicating the proper state after a connection is lost.
- Fixed a connection state bug when using Connect/Disconnect commands together with RunWithNewResults.
- Fixed a problem when changing color depth would disconnect a connection.
- Fixed a problem with restoring a connection when re-launching if it's a Bonjour entry.
- Settings for a Bonjour server (e.g. Scale To Fit) are now properly saved when customized.
- Generated TypeText commands now properly handle entries with a return in them (entered with Option-Return).
- Added billion as a predefined variable (value: 1000000000).
- Enabled the use of "as parameters" to pass a list of parameters when passing parameters while accessing a property of an object.
- Fixed a bug with the **split by** operator that would incorrectly alter the source container.
- Corrected a small issue with **the numberFormat** property. Setting a numberFormat that begins with a

decimal point (such as ".##") would previously yield a displayed value with a leading 0 before the decimal point. It will now correctly yield a formatted value with no leading zero.

- Improved the asText message of some caught exceptions to better report the underlying error.
- Changed the STUnknownCommand error to STUnknownMessage and improved the error message to better describe the problem.

Release v3.31 (12-February-2007)

Highlights:

- Added command colorization and command history in the Ad Hoc Do Box.
- Added ability to try some Purple features and learn more about Purple in Eggplant Green.
- Corrected a number of bugs and minor issues.

User Interface:

- Two new features were added to the Ad Hoc Do Box (AHDB), the command field at the bottom of the Run window. If the "Update colors continuously while typing" option is enabled on the Script Colorization preferences pane, it will now apply to commands entered in the AHDB as well as in scripts. This provides immediate feedback that will let you know when you have a well-formed command.

- The other feature added to the Ad Hoc Do Box is the ability to retrieve and reuse earlier commands. Press the Up Arrow key on your keyboard to step back through a history of all previous commands. The Down Arrow key will cycle forward through the list. When the desired command is shown, press Return to execute that command. You may edit the displayed command to make changes before pressing Return.

- Eggplant Green now includes an About Eggplant Purple menu item that displays a panel with information on the advanced features of Eggplant Purple. A time-limited trial of some features is available directly from this panel.

Scripting:

- Added a **runningFromCommandLine()** function that returns true when the current script run was initiated from the Unix command line, or false when it was started from within the Eggplant user interface.

Bug Fixes / Tweaks:

- Fix to problem requesting a Green Trial License.
- Fix to Green bringing up "No Suite For Script" repeatedly for scripts that couldn't be opened.
- Fix to problem submitting bugs from the Support Request form.
- Fix to a bug where passwords with special characters couldn't be used for SSH connections.
- Fixed a bug where Color Sync settings on a machine could prevent text images generated by a TIG (text image generator) from being trimmed correctly.
- Fix to a bug setting a dynamic breakpoint on an empty script, or when deleting all text when a breakpoint is set.
- Fixed a bug running on Mac OS X 10.3 with a minimized Remote window.
- Fixed a bug saving full screen captures.

Release v3.3 (18-January-2007)

Highlights:

- New Connection List panel for managing connection (SUT) information.
- Multiple connections can now be open simultaneously.
- Remote screen view can be scaled to fit in the window.
- Go to line and go to handler controls available in script editor toolbar.
- VNC support for 16-bit and 8-bit connections, various color formats.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- Scripts run from the command line using the runscript command no longer reopen a previously open connection. Scripts that expect a connection to already be open should be called using the appropriate options on the runscript command to specify a connection, or modify the script to include a Connect command.

- Reverse Connections do not become active automatically. When a reverse connection is received during script execution, in previous versions it would become the active connection (since only one connection could exist at a time). Now, if a script waits for a reverse connection to be made, it will need to check for it to appear among the connections listed by the allConnectionInfo() function (a reverse connection in the list will have its Reverse property set to true).

- Connect commands now always show the Viewer window by default. To prevent the Viewer window from being shown, include a Visible property with a value of false in the connection property list. Previously, it would default to not visible whenever the Remote window was not open.

- Schedule items that specify a SUT should now use the name or IP address of a connection in the Connection List. Existing items that specify a SUT will continue to work.

- The vncMaxTurnaroundDelay has been deprecated. This setting is now obsolete, and will be ignored.

Connectivity:

- The Establish Connection panel has been replaced by the Connection List. This panel clearly displays and allows you to manage a list of known VNC servers. Connections can be added or deleted, and information about existing connections can be edited. The information recorded for each connection includes the host name or IP address and port, and also a display name. In addition, you can specify a color depth different from Eggplant's standard (Millions of colors).

- A new feature of this version is the ability for Eggplant to connect to servers using their native format (including color depth and big- or little-endianness). This supports connecting to a wider variety of VNC servers, including some that don't fully implement the RFB protocol (such as the VNC server available on WindowsCE). If a connection cannot be established using Eggplant's standard settings, you may edit the connection and set the color depth to "Default" to try connecting using the server's native format.

Important: Note that connecting at a color depth other than "Millions" (32-bit) may alter the colors received by Eggplant such that previously captured images can no longer be found. For best results, always connect to a SUT at the same color depth that was used when images in your script were captured.

User Interface:

- **Connection List.** The connection panel has now become a list of known hosts, including both previously defined server connections and servers discovered dynamically through the Bonjour service.

- **Multiple Open Connections.** Eggplant now allows simultaneous connections to be open to multiple remote machines (SUTs) at once. Only one open connection can be the "Active Connection" to which script actions apply at any given time. The active connection is indicated by asterisks surrounding the title in the Viewer window's title bar. A different connection can be made the active connection by selecting its viewer window when no script is running, or with a Connect command (Purple only) within a script. During a script run, only the active connection is blocked -- you may continue to interact with non-active windows in Live Mode.

- **New Preferences.** The Preference panes now include new settings for:
  • *Update connection availability* (General) -- specify how often Eggplant should poll the known connections in the Connection List to see if those SUTs are available.
  • *Maximum Open Connections* (General) -- specify the maximum number of connections that can be open at once. This applies both to connections being used in Live and Capture modes and to those opened during a script run (this corresponds to the MaxOpenConnections global property).
  • *Close Connection List upon opening a connection* (General) -- whether the Connection List window should be closed when a connection is established, or should remain onscreen.
  • *Mouse Right Click Key* (Viewer Window) -- specify a modifier key that can be used in Live Mode in conjunction with a left mouse click to transmit a right click to the remote system.
  • *Always scale remote display proportionally* (Viewer Window) -- specify whether viewer windows that are set to Scale To Fit will always maintain their native proportions, or whether their horizontal and vertical dimensions can be scaled differently.
  • *Reverse Connections Port* (VNC) -- specify the port number on which to listen for incoming connections.

- Added a "Line" field on the Script Editor toolbar to show the current line number within the script or handler, and allow selecting a specific line.

- Added a "Handler" popup on the Script Editor toolbar to show the current handler within the script, and allow selecting the first line of a specific handler.

- A connection item can be dragged from the Connection List into a script to insert a Connect command in the script for that connection. Hold down the Option key while dropping to insert the full property list for that connection rather than a simple identifier.

- Added a "Scale To Fit" / "Show Full Size" button on the Viewer window toolbar to toggle between showing the remote display at full size or scaled to fit within the window. Scaled windows are fully functional.

- When a Viewer window is minimized, its icon in the Dock will now show live updates of the remote screen.

- A new "Settings" tab has been added in Suite windows. This tab contains two items: a Suite Description, and a Results Directory. The Suite Description field lets you enter any descriptive information you would like to record for that suite. This field is purely for you own use -- it is not used by Eggplant in any way.

  The Results Directory field shows the path to the folder where script results for the suite will be stored. Usually you will not need to change this from the default location within the suite, but in some situations it may be desirable to have results stored somewhere outside of the suite.

- Simplified specifying a SUT to connect to for scheduled scripts (in the Schedules tab of a Suite window). A SUT is now specified in the schedule by simply giving its Display name, IP Address, or Host name. All other details about the connection are obtained when the schedule is run, by locating that connection in the Connection List.

- Eggplant now provides rich clipboard support when working in Live Mode with a Mac SUT that is running

the Vine Server VNC server. This allows any type of data (not just text) to be copied between the Eggplant machine and the SUT, including rich text, images, and even files in the Finder.

- Eggplant will now ask the user before displaying the Release Notes or Getting Started manual the first time that user launches a new version.

- An "Answers Key" item has been added to the Help menu. This menu item opens the "Answer Key" suite, containing completed scripts that correspond to the Eggplant Tutorials for you to examine and run.

- The License Registry panel can now automatically install trial keys.

Scripting:

- The **Connect** command can now be called with the identifier of an item in the Connection List. The Connection List will be searched for an item whose display name or host entry matches this identifier, and will open that connection.

- When a connection is established by the **Connect** command, it becomes the "Active" connection. This is the connection to which all SUT-related actions will be directed. If the connection is already open, the Connect command will simply make it the active connection. This mechanism can be used by a script to work with multiple SUTs and quickly switch between them.

- The **connectionInfo()** function always returns information about the active connection. The **allConnectionInfo()** function (formerly *knownHosts()*) returns information about every connection in the Connection List. Connections that are currently open will have a Status of "Connected".

- A connection property list passed to the **Connect** command may now include a "colorDepth" property to request connection using a specific color bit-depth. The value of this property should be 8, 16, or 32, or 0 to request the native format of the SUT.

- Similarly, the property lists returned by the **connectionInfo()** and **allConnectionInfo()** functions will contain a colorDepth property indicating the bit-depth of the connection.

- The **Disconnect** command can now be given one or more parameters to indicate specific connections that should be disconnected. Each parameter may either be a connection identifier (name or host) or a connection property list, such as the one returned by the connectionInfo() function. If no parameters are given, the disconnect command will disconnect the current active connection.

- Added **the MaxOpenConnections** global property to allow a script to access and change the maximum number of connections that can be open at once. By setting this to 1, Eggplant will behave as it did in previous versions, closing the previous open connection whenever a new connection is established. Setting the MaxOpenConnections to zero (the default) will allow an unlimited number of connections to be open at once. This could lead to performance problems if your script connects to a lot of different SUTs.

- The property list provided to the **Sendmail** command can now include an "SMTP_Port" property to specify the port number on which to contact the SMTP mail server. If not specified, the standard mail port will be used.

- Added **the watchForScriptChanges** global property. Setting this property to true will cause SenseTalk to check script files for changes each time a message is sent to an object that was loaded from a script file. The default value is false.

- Added a **filesAndFolders()** (or **the files and folders**) function, to return a list of fileDescription objects for all of the files and folders in a specified folder, or in the current folder.

- The verbose version of the **offset** function has been enhanced to include **before** and **considering case** options. If you specify "before the end" or "before *location*" SenseTalk will search backward from the end of the text or from the indicated location to find the previous occurrence of the target string before that point in

the string. If "considering case" is specified, the search will be case sensitive (the default is not case sensitive):

```
get the offset of "Error" in myText before the end considering case
```

- The **contains** and **is in** operators have been enhanced for cases where the container being examined is an object. As before, the object is sent a **containsItem** function message with the search value as a parameter. The default implementation of the containsItem function can now behave in any of three ways, according to the setting of a new global property: **the objectContainsItemDefinition**. If this property is set to "AsTextContains" (the new default behavior) the expression is true if the object's text value contains the search value. When set to "NestedValueContains", the operator is applied to each of the object's values and the expression will yield true if any them contains the search value. Finally, a setting of "KeyOrValueEquals" will give the former default behavior: the operator returns true if the search value is equal to one of the object's keys or to one of its values.

- The **catch** portion of a **try** block is now optional. If omitted, any exceptions encountered while executing the statements in the block will prevent the remainder of statements within that try block from being executed, but will otherwise be ignored, and execution will continue following the try block.

- A specific handler of a script can now be called as a command without using the word **run**. Simply give the name of the script followed by either a dot (.) or apostrophe-s ('s) followed by the command to send to that script:

  ```
  myScript's commandName param1,param2
  ```

  This is equivalent to:

  ```
  run myScript's commandName param1,param2
  ```

  You can also use a quoted string to specify a full path or name containing special characters:

  ```
  "/path/to/my other script".commandName param1,param2
  ```

- Parentheses are now optional around a property list at the end of a parameter list. This allows any number of individual ordered parameters to be followed by any number of "named parameters" which are sent as a single property list. So the following are equivalent:

  ```
  colorize 12, 27, (inside:blue, outside:yellow)  -- the old way
  colorize 12, 27, inside:blue, outside:yellow    -- the new way
  ```

  Named properties can also be included at the end of any ordinary list, resulting in a nested property list as the last list item. So these are also equivalent:

  ```
  put (1,2,3,(name:Rik, age:42)) into data
  put (1,2,3,name:Rik,age:42) into data
  ```

- A folder can now be treated as an object. Every script in that folder can be called as a handler of the folder object. In addition, if a script file named "<initialHandler>.st" exists in that folder, it will be loaded and used as the object's initial handler, and any properties defined in that script will be used as the initial property values of the folder object.

- Scripts stored in a wider variety of encodings can now be read appropriately. Scripts saved using the internal SenseTalk mechanisms will now use UTF8 encoding.

- Added some predefined variables that can be used like unquoted literals even when the strictVariables is set to true (including Yes, No, On, and Off).

- **The MaxVNCTurnaroundDelay** global property has been deprecated, as it no longer serves any useful purpose.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Fixed a bug with comparison of values containing 16-bit Unicode characters.
- Improved the handling of scripts and logs containing two-byte Unicode characters. By default, plain text scripts and logs are now stored using UTF8 encoding, but can be read in a number of encodings.
- Fixed a number of issues with dynamic breakpoints when using cut, paste, undo, and redo operations.
- Fixed a problem with the ColorAtLocation() function on Intel processors.

- Fixed a problem in which images would sometimes be captured incorrectly when the Mac OS X system Zoom feature was used to magnify the screen (now that this works reliably, it can be helpful when capturing small images -- go to the System Preferences -> Universal Access panel to enable zooming).
- Fixed a problem with trimming of generated text images on Intel processors.
- Added spaces in log file wherever control characters were previously being removed.
- Fixed the value recorded in log file when using the ScrollWheelDown or ScrollWheelUp command.
- Fixed a bug that could cause a crash when deleting image files that lacked a thumbnail icon.
- Fixed a bug with site-wide license keys being properly recognized.
- Fixed a bug that could cause Eggplant to hang (when Script Animation is off).
- Fixed a bug with generating text images when running a script from the Unix command line.
- Fixed a bug with recording movies when running a script from the Unix command line.
- Fixed a bug with the **run** command that could cause problems if the script being run threw an exception.
- Fixed a problem with some date- and time-related functions ignoring the time zone information in a date value. Affected functions included the formattedTime(), hour(), and day() functions, and possibly others as well.
- Fixed a bug with accessing a property of an item in an empty value -- now returns empty as it should, rather than throwing an exception.
- Fixed a bug in which setting **the folder** to a folder identified by a fileDescription object would not work properly.

Release v3.21 (11-September-2006)

Highlights:

- Fixed a bug calling scripts in helper suites and open suites.

- Updates to Eggplant Reference to reflect new functionality introduced in 3.2.

Release v3.2 (30-August-2006)

Highlights:

- Improved support for testing applications running on pre-release versions of Mac OS X 10.5 (Leopard).
- Improved Licenses panel interface and license-related messages.
- Added a trial request sheet in the Licenses panel to directly request a trial license.
- Fixed a number of bugs and minor issues.

User Interface:

- Enhanced License Panel.

- The appearance of the Licenses panel and the wording of a number of license-related messages has been improved, to make them clearer and friendlier for the user.

Scripting:

- The **TypeCommand** command has been made more flexible and easier to use for working with modifier keys. Previously, to type Option-Shift-Command-X for example, would require you to do something like this:

```
TypeCommand "Option Shift X"
```

  A few slight variations on the above were also possible, such as changing the order of the modifier key names and eliminating the spaces. Now, you can also use multiple parameters, with some of the parameters being names of modifier keys. So the above example can now also be written like this (assuming that Option and Shift are not used as variables in your script):

```
TypeCommand Option,Shift,"X"
```

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Properly disable the Text Image (Purple) and Use Image (Green and Purple) menu items as appropriate.
- Fixed a small issue with the License Agreement menu item at startup.
- Fixed an issue with calling a script from the Ad Hoc Do Box before any script has been run.
- Fixed the context menu in the Script Editor to include the correct items (Green).
- Fix to correct a problem connecting to the embedded VNC server in Leopard (Mac OS X 10.5) - (Green and Purple).
- Fix to correct a compatibility problem with standard output on 10.3.9 (Purple only).
- Fix to correct a problem with the Run window toolbar (Purple only).
- Fix to enable a performance optimization for file changes (KQueue handling) that was inadvertently disabled (Green and Purple).
- Fixed a problem with being unable to click the text of the 'Connect Securely (SSH)' button on the Connection panel to turn SSH off (Green and Purple).
- Correction to the tiny sized (unreadable) License Agreement shown during installation (Green and Purple).

Release v3.10 (9-August-2006)

Highlights:

- Support for multiple "Colors" of Eggplant: Purple and Green.
- Added the ability to send command line output to standard out (stdout).
- Improved bookmarks for better navigation of the Eggplant documentation.
- Added a newer release of the Windows Text Image Generator that handles Unicode characters.
- Fixed a number of bugs and minor issues.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- ScrollWheelUp and ScrollWheelDown. A bug in these commands was causing them to tell the SUT to scroll by an amount one greater than requested. For example, `ScrollWheelUp 2` was scrolling by 3 units instead of 2. This has now been corrected, with the result that existing scripts that use these commands will

now cause slightly less scrolling than they did previously.

User Interface:

- Improved the behavior when a generated text image is not found on the screen. The alert panel that is displayed now provides an additional option to return you to the Text Image Sheet in order to modify the text or attributes and try again.

Scripting:

- Added **the CommandLineOutput** global property to control whether output is sent to standard output while running a script from the command line. This can also be set using an option on the command line. When set to 'true' or 'on', all of the output that would usually be displayed in the Log Area of the Run window will be sent to standard output (stdout). Using this property, you can control exactly which sections of a script produce output and which sections don't:

```
set the CommandLineOutput to ON
-- do stuff whose output you want to see
set the CommandLineOutput to OFF
```

- **Command Line Interface**:

  Can now turn CommandLineOutput on for scripts using `-CommandLineOutput YES` argument when starting scripts from the command line.

  Removed several scary looking warnings that would come up when running CLI.
  Fixed a problem with alert panels being shown when multiple parameters were passed via CLI.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- License Panel now reports the license type (Purple, Green, CLI).
- Fixed a bug with the ScrollWheelUp and ScrollWheelDown commands, which were sending one more scroll event than requested.
- Improved Eggplant to have a MaximumMovieTime (15 minutes default) and improved MovieStatus() function.
- Improved Eggplant to ONLY send a mouse event to release the mouse button(s) when closing a connection if we have buttons held down that need to be released.
- Fixed the file name used in the text image cache for text containing characters high in the Unicode range.
- Resolved a class of lockup problems that could occur if there was an error when starting or ending recording of a movie.
- Fixed a bug where "add image" generated an additional set of quotes around the image name.
- Fixed a bug that setting the ScriptLogging global property on would log that change.
- Fixed a deadlock condition with redrawing dynamic breakpoints.
- Fixed a crashing bug that could occur when a new connection was made using a Connect command to a system with a different screen size.
- Fixed a bug on Intel machines that prevents reliable connection to a Windows TIG.
- Fixed a bug on Intel machines that would lock up a connection when using the SetRemoteClipboard command.
- Fixed several exceptions that could be thrown while executing scripts repeatedly.

Release v3.01 (14-June-2006)

Highlights:

- A fix for Eggplant running on Intel macs solves a problem connecting to some VNC servers running with Authentication mode enabled.
- Added a read me and newer release of the Windows Text Image Generator.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- imageInfo(). Since 3.0, the **imageInfo()** function returns slightly different information than it did in earlier versions of Eggplant. The value of the imageName property is usually the relative name of the image, or, for text images, the word "Text: " followed by the text. A new imagePath property contains the full path of the image file.

Release v3.0 (30-May-2006)

Highlights:

- Text Image Generator creates images on-the-fly for any text with a given set of characteristics. This allows scripts to find and work with textual elements on the remote screen without the need for pre-captured images. This feature enables scripts to work with dynamic names and dates, makes scripts easier to create and more robust against font changes in an application under test, and greatly simplifies testing of cross-platform and multi-language applications.
- Dynamic breakpoints can be set and cleared at any time, even while a script is running, make debugging easier.
- Eggplant now accepts "reverse" (incoming) connections initiated from a VNC server.
- A new Support Request panel makes it even easier to submit questions or bug reports from within Eggplant.
- Support for working with colors in scripts, and many other scripting enhancements.
- Numerous documentation improvements.
- This release is a Universal Binary that runs natively on both PowerPC and the new Intel-based Macs.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- Folder Names. All SenseTalk commands, functions, and properties that return the full path of a folder now return the folder name with a slash at the end. This simplifies the task of creating file paths, by eliminating the need for your code to insert a slash between a folder name and the name of a file in that folder. If the old behavior is desired, simply set the new global property **the folderNamesEndWithSlash** to false.

- Exit Command. The **exit** command has been fixed when used in an initial handler to properly require

either the name of the handler or the word "handler" or "script". **Exit script** is a new synonym for **exit handler**. Any scripts with exit commands which use something other than the name of the handler or the words "handler" or "script" will need to be corrected.

- The word "the". The word "**the**" is now a restricted word. It may not be used as the name of a command or function.

- New exceptions. Many commands and functions that formerly set "the result" to indicate an error condition will now throw an exception instead. A new global property, **the ThrowExceptionResults**, can be set to false to restore the old behavior, if preferred.

- Platform support. This release of Eggplant does not run on Mac OS X 10.2 (Jaguar) – Mac OS X 10.3 (Panther) or later is required.

User Interface:

- **Dynamic Breakpoints**. Script Editor windows and the script display area of the Run window now include a breakpoint column to the left of the script. Clicking in this column adjacent to a command in your script will set a breakpoint on that command, indicated by a triangle in the breakpoint column. Breakpoints can be dragged to a different line in the script, or may be dragged completely out of the breakpoint column to remove them.

  When a script containing dynamic breakpoints is run, execution will be paused before executing any line where a breakpoint is set. Dynamic breakpoints can be added, moved, or removed before a script run, while a script is paused, or even while a script is running. Dynamic breakpoints remain while a script is open during an Eggplant session. They are not retained when a script is closed.

- A new Text Image item has been added to the Remote Screen toolbar for generating text image commands (*SUGGESTION*: select Customize Toolbar from the Window menu and drag the new default toolbar into place). Clicking this item, or holding the Shift key down while clicking any of the command generation items (Click, DoubleClick, etc.) will bring up the new Text Image Sheet. This sheet allows you to select a text platform and style, and to customize any of the individual text attributes. It then generates and executes the selected command.

- A new Text tab has been added to the Run Options panel in Preferences. This new tab lets you define new text platforms and styles, or modify existing ones. You can also set the default platform.

- When quitting, Eggplant now displays a confirmation panel with three options: "Cancel Quit", "Quit After Next Script Run", and "Quit Now". If "Quit After Next Script Run" is selected, the next time a script is run, Eggplant will automatically quit (without asking) when that script finishes running. If you change your mind and want to cancel this request, choose "Quit Eggplant" from the Eggplant menu again, and click "Cancel Quit".

  If a script is currently running when the "Quit Eggplant" command is given, the last two options will be titled "Quit When Run Finishes", and "Abort and Quit". Selecting "Quit When Run Finishes" will cause Eggplant to finish the script run and then quit automatically. Selecting "Abort and Quit" will abort the script execution and quit Eggplant immediately.

  You may hold down the Option key (type Command-Option-Q instead of Command-Q) to have Eggplant quit immediately without asking.

- A new Support Request panel has been added. The "Submit Question..." , "Report Bug..." , and "Request Feature..." items on the Help menu have all been changed to use this panel. Also, if Eggplant quits unexpectedly or experiences an internal error, you will be presented with an option to submit information about the problem to the the support team through this panel. This panel sends information directly to the TestPlant website, so you don't need email to be configured on your system in order to use it.

- Eggplant now accepts "reverse" (incoming) connections initiated from a VNC server. Whenever Eggplant does not have an open connection to a SUT, it is now listening for such incoming requests. If one is

received, the connection is opened and the screen of the SUT is displayed. This can be useful for working around firewall restrictions. This feature can be disabled, or the port number changed, from the Eggplant Preferences panel.

- The Find panel has been updated, with new options for ignoring case, wraparound, and contains / start with / full word searching.

- The "Add Image" item on the Remote Screen toolbar can now be used when a script is running.

Scripting:

- **Text Image Generation**. At any point in a script where an image is needed, you can now supply a property list containing a "text" property, and optionally additional properties that describe the text characteristics. When such a text image is called for, Eggplant will check its cache of text images to see if it already has the needed image and, if not, it will immediately generate an image of the requested text. So for example, to bring up the window menu, instead of capturing an image of the word "Window", saving it as "WindowMenu" and using this click command:

```
click "WindowMenu"
```

  you can now give this command:

```
click (text:"Window", textSize:14, bold:YES)
```

  This approach provides some very powerful benefits: no image has to be captured manually; and the text can just as easily come from a variable or expression as from the literal text "Window". This makes it easy to look for text that may change dynamically. For example, if the next step is to click the name of a particular window in the window menu, but that window's name is different each time the script is run, you might do it like this (assuming that windowName is a variable containing the window's name, and all of the default text attributes are set properly):

```
click (text:windowName)
```

  The text attribute properties that can be specified include:

text	the text string to generate (required)
textFont	the name of the font to use
textSize	the size of the font, in points
textColor	the color of the text
textBackgroundColor	the color of the background
bold	whether the text is bold
italic	whether the text is italic
underline	whether the text is underlined
textPlatform	identifier for the text rendering platform

- Added **the currentTextPlatform** global property. At the beginning of a script run this will be set to the default platform as selected in the Text section of the Preferences.

- Added **the textPlatforms** global property. At the beginning of a script run this will be set to be a property list containing all of the information about text platforms as defined in the Text section of the Preferences panel. This may be modified by your script to set the generator for a platform, change the available text styles for a platform, or even to define entire new platforms.

- Added **the defaultTextStyle** global property. This is actually a shortcut to access the "Default" text style for the current text platform. If your script makes any changes to the attributes defined in the defaultTextStyle, those changes will be reflected in the corresponding properties within the textPlatforms global property. Here's a simple way to set the default text size to use to be 12 whenever the textSize property is not specified explicitly:

```
set the defaultTextStyle.textSize to 12
```

  To set several attributes at once, you might use a command like this:

```
replace properties (textFont:"Times", textSize:14) in the defaultTextStyle
```

- Enhanced the **imageFound()** and **anyImageFound()** functions to treat an initial timing parameter of zero to indicate that Eggplant should perform an "immediate" search, looking only once at the SUT screen, without

performing a full screen refresh or repositioning the mouse. This gives the fastest possible times for checking for an image that may not be visible -- if it's not showing, these functions will look once and then report that it's not found.

- Added **the searchRectangle** as a global property, allowing it to be set using put or set commands. The old **SetSearchRectangle** command will continue to function, but is now deprecated in favor of using the new property. To restore the search rectangle to search the full screen after restricting it to a smaller area, you can do either of the following:

```
set the searchRectangle to "" -- (or '... to empty')
set the searchRectangle to the remoteScreenRectangle
```

- The new **captureTextImage** command is typically only used within a text image generation script, although it can be called at any time. It takes a single parameter which is a property list that must include at least a 'text' property and a 'rectangle' property, and should also include properties for the various attributes of the text image being captured.

  The 'rectangle' property specifies the region on the SUT screen where the text indicated by the 'text' property is visible, ready for Eggplant to capture. The captureTextImage command will capture this image, trim off any excess background color from the four sides, and store the resulting image in its cache, identified by all of the supplied text attributes so that it can be retrieved again later. If any attributes are not supplied, the corresponding values from the defaultTextStyle global property will be used.

- **Color Support.** Added full support for colors in SenseTalk, which can be used to specify the text color and background color when generating text images. For complete information, see the new chapter "Working with Color" in the SenseTalk Reference manual. Briefly, the color capability includes the following new features:

  The ColorFormat global property can be set to Basic, Alpha, W, WA, RGB, RGBA, HSB, HSBA, CMYK, CMYKA, or HTML to specify the format in which a color value will be displayed. Basic is a list of 3 integers from 0-255 representing the levels of red, green, and blue in the color. The Alpha format adds a fourth integer representing the opacity of the color. The HTML format presents a color as a hexadecimal number preceded by "#" (such as "#FF0000" for red). The other formats are text lists beginning with the format identifier, followed by one or more component values from 0.0-1.0 indicating the level of each color component. The colorFormat is initially set to Basic.

  The **is a color** operator can be used to test whether a value is a color, or is in a format that SenseTalk can interpret as a color.

  The **color()** function returns a color value from a string or array in one of the recognized color formats. Any of the 11 formats described for the colorFormat are recognized, as either text lists (with items separated by commas) or as true lists of values. In addition, color names can be used. One use of the color() function is to compare two colors that may be in different formats to see if they represent the same color, for example:

```
if color of stripe is color("red") then ...
```

  The namedColors global property is a property list whose keys are names of colors, with the corresponding values being the definition of those colors. New colors can be defined in a script, or the values of existing colors can be changed. Predefined named colors include: Aqua, Black, Blue, Dark Blue, Brown, Clear, Cyan, Fuchsia, Gray, Dark Gray, Light Gray, Green, Dark Green, Light Green, Lime, Magenta, Maroon, Navy, Olive, Orange, Purple, Red, Dark Red, Silver, Teal, White, and Yellow.

  Functions are provided to retrieve individual color components from a color. The **red(), green(), blue(), hue(), saturation(), brightness(), cyan(), magenta(), yellow(), black(), white()**, and **alpha()** functions return the level of the indicated color component, in the range 0.0-1.0.

- Added a **colorAtLocation()** function that accepts an x,y coordinate on the screen as a parameter and returns the color of the pixel at that location on the SUT screen. The origin (0,0) is the top-left corner of the screen.

- Added an **imageColorAtLocation()** function, to enable a script to determine the color of a particular pixel within any captured or text image. The function requires two parameters: the image, and the location (x,y coordinates) within the image (with 0,0 being the top left corner of the image).

- The **imageInfo()** function returns slightly different information than it did in earlier versions of Eggplant. The

value of the imageName property is usually the relative name of the image, or, for text images, the word "Text: " followed by the text. A new imagePath property contains the full path of the image file.

- Added a **foundImageInfo()** function, which returns a property list containing complete information about the last image that was found on the remote screen (this is the same information that is returned by the imageInfo() function).

- Image property lists may now include a **ClipRectangle** property. If present, its value should be a rectangle. The image will be clipped to the intersection of this rectangle with the image's rectangle before searching.

- Added **the ScriptLogging** global property, which may be set to "On" (or true), "Off" (or false), "Minimal", or "Silent" for fine control over logging behavior. The SetLogging command and the LoggingIsOn() function have been deprecated.

- Added a **standardFormat()** function to return a text representation of any value in a format suitable for archiving. For any type of value, this function should return a text value that can be suplied as the parameter to the value() function to retrieve the original value. In general, you shouldn't rely on the returned value to always be in a particular form -- the exact format may change from one version of SenseTalk to another, and may also depend on other factors, including the current internal representation of the value. However, value(standardFormat(someValue)) should always be equal to someValue.

- Added **the defaultQuoteFormat** global property to provide control over how values are quoted when they are displayed within a list or property list. The default setting of this property is "None" which will cause text values to display without any special quoting. Setting it to "Standard" will cause values in lists and property lists to be quoted according to their contents. The defaultQuoteFormat may also be set to a quote string (such as ' or ") or a list of two delimiters (such as ("<<" ,">>") for example, to set how text values will be quoted.

- Added an optional **quotes** property within both **the listFormat** and **the propertyListFormat** global properties. If set, its value will override the defaultTextFormat for lists or property lists respectively.

- Added an optional **asTextEnabled** property within **the propertyListFormat** global property. If set to false or NO, the "asText" mechanism will be disabled for displaying property lists.

- Added **the defaultStringEncoding** global property to provide control over how text values are encoded. The encoding affects how text values are interpreted when reading and writing text files, URLs, and sockets.

- Added **the URLCacheEnabled** global property to control caching of remote URL accesses. When the URLCacheEnabled is set to 'true' (the default), SenseTalk may cache the contents of a remote URL once it has been fetched to speed up later access. To force a fresh copy of the URL to be loaded, set this property to 'false', as in this example:

```
set the URLCacheEnabled to false
put url "http://www.apple.com" into appleHomePage -- get fresh copy
set the URLCacheEnabled to true -- restore caching
```

- Extended the property access syntax to allow accessing a list of properties at once:

```
set point3 to (x:100,y:200,z:300)
put point3.x -- 100
put point3.(x,z) -- (100,300)
```

- The multiple property access syntax can be used to assign to multiple properties as well:

```
set point3's (x,y) to 0
put point3.x -- 100
put point3.(x,z) -- (100,300)
```

- Extended the chunk access syntax to allow accessing a list of chunks at once:

```
set pets to ("Sparky","Snoopy","Whiskers")
put items (3,1) of pets -- ("Whiskers","Sparky")
put characters (1,3,5) of item 3 of pets -- ("W","i","k")
```

- The new multiple-chunk syntax can also be used as the destination container in a **set** or **put** command:

```
set notes to "c,d,e,f,g"
```

```
put "#" after items (1,4) of notes
put notes --"c#,d,e,f#,g"
set items (1,3) of notes to ("a","c")
put notes -- "a,d,c,f#,g"
```

- Added a **retain properties** command and **retaining properties** operator to remove properties from a property list, keeping only the properties that are specifically retained:
```
retain properties ("black", "white", "gray") in the namedColors
```

- Added **exists** (and **does not exist** and **doesn't exist**) as an alternate syntax for the **there is a** function. The following are equivalent:
```
if there is no file "xyz" then create file "xyz"
if file "xyz" doesn't exist then create file "xyz"
```

- Added **the ThrowExceptionResults** global property to specify whether any command that sets "the result" to an exception object should automatically throw that exception. Defaults to true.

- The **post** command now sets the result to an exception object if the post operation fails.

- The **open, read, write, close** commands now set the result to an exception object if the operation fails.

- The word "**the**" is now a restricted word. It may not be used as the name of a command or function.

- Added **the folderNamesEndWithSlash** global property, which defaults to true. All SenseTalk commands, functions, and properties that return the full path of a folder now return the folder name with a slash at the end. This simplifies the task of creating file paths, by eliminating the need for your code to insert a slash between a folder name and the name of a file in that folder.

- Added a **folder()** function (and synonymous **directory()** function) that accepts a file path as a parameter and returns that path without the final path component.

- Added a **lastPathComponent()** function that accepts a file path as a parameter and returns the final component of that path.

- Added a **fileExtension()** function that accepts a file path as a parameter and returns just the file extension (the part of the name after the final '.').

- Added a **fileDescription()** function that accepts a file path as a parameter and returns a fileDescription object containing lots of information about the file at that path.

- The **plus properties** and **minus properties** operators have been changed to **adding properties** and **removing properties**, which can be abbreviated as **adding** and **removing**:
```
put the timeFormat removing (shortDate,longDate) into myFormats
put myFormats adding (quick:"%j of %Y") into extendedFormats
```

- Added a **replace properties** command and **replacing properties** operator to add properties to a property list, replacing any matching properties with new values:
```
replace properties (grey:(220,220,220)) in the namedColors
```

- An empty value is now treated as a valid boolean value, equivalent to false.

- Added **with headers** clause to **url** expressions, to allow setting custom http headers (in the form of a property list). Currently these headers are only used by the **post** command, but this may change in the future.

- Added eight international date and time formats, accessible both in the convert command and through the following new functions: **international date, long international date, abbreviated international date, short international date, international time, long international time, abbreviated international time,** and **short international time**. These formats present the full date, the year and month, or the full date and time, in a manner that complies with the international ISO 8601 standard (see http://www.ietf.org/rfc/rfc3339.txt) except for the abbreviated international time, which is a widely used format on the internet that varies

slightly from the standard.

- Added four internet date formats, accessible both in the convert command and through the following new functions: **internet date, long internet date, abbreviated internet date,** and **short internet date**. These formats present a date and time in a manner that complies with the date and time specification of the internet message format as defined in RFC 2822 (http://www.ietf.org/rfc/rfc2822.txt), except that the "long internet date" shows full weekday and month names.

- Added four functions corresponding to the dateItems formats available in the convert command: **dateItems, long dateItems, abbreviated dateItems,** and **short dateItems**. These formats present a date and time as a comma-separated text list. The short dateitems returns 6 items: the year, month, day, hour, minute, and second. The dateitems (without an adjective) returns 7 items, with the seventh being the day of the week (0-6, where Sunday is 0). The abbreviated dateitems adds the timezone offset in HHMM format, and the long dateitems returns that same information, but with the timezone name rather than offset.

- Added **the timeFormat** global property to provide access to all of SenseTalk's date and time formats and allow them to be modified. The timeFormat is a property list containing all of the date and time format strings used by the various date and time functions and by the convert command. The formats included in the timeFormat are: AbbreviatedDate, AbbreviatedDateItems, AbbreviatedInternationalDate, AbbreviatedInternationalTime, AbbreviatedInternetDate, AbbreviatedLocalTime12, AbbreviatedLocalTime24, AbbreviatedTime12, AbbreviatedTime24, Date, DateItems, InternationalDate, InternationalTime, InternetDate, LocalTime12, LocalTime24, LongDate, LongDateItems, LongInternationalDate, LongInternationalTime, LongInternetDate, LongLocalTime12, LongLocalTime24, LongTime12, LongTime24, ShortDate, ShortDateItems, ShortInternationalDate, ShortInternationalTime, ShortInternetDate, ShortLocalTime12, ShortLocalTime24, ShortTime12, ShortTime24, Time12, and Time24.

  Any of the formats can be examined or changed. The formats may contain any characters, and make use of the following special placeholders which indicate the parts of the date that will appear in the formatted string:

%%	a percent sign (%)
%a	abbreviated weekday name
%A	full name of the day of the week
%b	abbreviated month name
%B	full month name
%c	complete localized date and time
%d	day of the month as two-digit number (01-31)
%e	day of the month as one- or two-digit number (1-31)
%F	fraction of a second to three places (000-999)
%H	hour based on a 24-hour clock as two-digit number (00-23)
%I	hour based on a 12-hour clock as two-digit number (01-12)
%j	day number within the year as three-digit number (001-366)
%m	month as a two-digit number (01-12)
%M	minute of the hour as a two-digit number (00-59)
%p	AM or PM
%S	second of the minute as a two-digit number (00-59)
%w	weekday number (0-6), where Sunday is 0
%x	date using the date representation for the locale
%X	time using the time representation for the locale
%y	year without century as a two-digit number (00-99)
%Y	full year number (with century -- currently a 4-digit number)
%z	time zone offset in hours and minutes from GMT (HHMM)
%Z	time zone name

- Added a **formattedTime()** function to return the current time (or any date or time) using a custom format.

- Changed the **convert** command to return a formatted date/time value rather than a simple string (except for "seconds" formats). This preserves formatting information through date arithmetic operations. The seconds formats have no corresponding format, so converting to seconds, long seconds, etc. will still result in a fixed string rather than a formatted time value.

- Added **monthNames()** and **weekDayNames()** functions to return the list of strings used in formatting dates. The adjectives long, short, and abbreviated may be used with these functions to return variations on the

names. In each case, the long form is the same as not specifying any adjective, the abbreviated form returns a list of three-letter abbreviations rather than the full name, and the short form returns numeric representations.

- Added **ago** and **hence** operators (with same precedence as squared and percent). When used following a time interval, these produce a date/time value that is the specified length of time in the past or future, respectively (using the AbbreviatedInternationalTime format). This is particularly useful when comparing another date/time value to the present:
```
if the modification date of file "xyz" is later than 1 hour ago...
```

- Added **now** as a predefined "literal" whose value is always the current time, formatted with the AbbreviatedInternationalTime format.

- Added **today** as a predefined "literal" whose value is always the current date, formatted using the InternationalDate format.

- Added **the listFormat** and **the propertyListFormat** global properties to replace the old individual global properties for list and propertyList formatting as text: **listPrefix, listSeparator, listSuffix, plistPrefix, plistEntrySeparator, plistKeySeparator, plistSuffix,** and **emptyPlistAsText**. The listFormat property is a property list containing "prefix", "separator", and "suffix" properties. The propertyListFormat is also a property list, containing "prefix", "entrySeparator", "keySeparator", "suffix", and "emptyRepresentation" properties.

- Added a variation of the "<chunk> **number of** <target> **within** <source>" expression -- by using the word **containing** instead of the word **of** it will return the index number of the first chunk that *contains* the target rather than the first that is *equal* to the target.
```
put the line number containing "Rogers" within addresses into recNum
```

- Enhanced the **random()** function to accept two parameters, indicating the lowest and highest integer values of a range, and return a random number in that range, inclusive:
```
get random(100,200) -- get a number from 100 to 200 inclusive
```

- Added "TryDepth" to the information returned for each frame by the **callStack()** function.

- Added **isn't**, **aren't**, and **doesn't** as synonyms for "is not", "are not", and "does not":
```
put 7 isn't 11 -- true
put "R" doesn't come after "X" -- true
put the first two characters of "Apple" aren't "Ap" -- false
put 12 isn't a date -- true
put "Q" isn't a number -- true
```

- Added **reversed** and **in reverse order** as synonyms for **descending** in the sort command.

- Added **penultimate** as an ordinal (meaning "next to last").

- The message name (or the word "message") is now optional in the **pass** command. Similarly, the **return** command can be used without supplying a return value (will return **empty** by default).

- Enabled direct access to the individual properties of a global property which is a property list, such as:
```
set the listFormat's separator to ";"
```

- Added a **loadedModules()** function that returns a list of all of SenseTalk's internal back-end modules that have been loaded into the system. This can be used to determine if a particular set of functionality is available. For example, the STColor XModule shows up in this list as "STBackEndColor", so you can test whether the color functions are available like this:
```
if "STBackEndColor" is in the loadedModules then ...
```

- Extended the verbose style of the offset function to allow specifying "after" some location:
```
put the offset of "a" in "Banana" after 2 -- 4
```

- You can now access any file properties of a fileDescription object, not just those that are explicitly stored in it.

- The "is a file" and "is a folder" operators now properly treat fileDescription objects as referring to the file using their long name, not their "asText" name.

- The **text** or **asText** function returns a text representation of any value. The **number** or **asNumber** function returns a numeric representation of any value that can be evaluated as a number. Also added **asDate**, **asTime**, and **asColor** synonyms for the date, time, and color functions.

- Added **as text**, **as number**, **as date**, **as time**, and **as color** operators as special syntax for calling the asText, asNumber, asDate, asTime, and asColor functions.

- Any **url** expression can now be expressed using a property list instead of a string, without the need to explicitly call makeURL().

- Added **the URLTimeout** global property to specify the maximum timeout for URL operations (currently used only by the post command).

- An at sign is now allowed before quoted string literals (@"text" is equivalent to "text").

- Allow "end" as an ordinal to mean the same as "last":
```
put chars 6 to end of "Jennifer Juniper" -- "fer Juniper"
```

- Allow "empty property list" as well as "empty object":
```
set configuration to be an empty property list
```

- Allow either curly braces or parentheses around list of containers in a multiple assignment statement.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Improved support for working with Read-Only suites.
- Enhanced the functioning of screen tracing to more accurately capture images of the screen as it looked exactly at the time when an image was found (or failed to be found) prior to most SUT interactions.
- Fixed a Lock Focus exception when pausing.
- Fixed a Minor menu validation issue when we haven't connected Successfully.
- Fixed an NSRangeException on Auto-complete on first word of a script.
- Fixed a problem with deleting files (suites, scripts, images, results) when more than one user is running Eggplant on a machine.
- HotSpots are written correctly for generated Text Images.
- Fixed a deadlock that could occur when an exception is raised during a run, with script tracing set to "All".
- Fixed a bug with logging of text that included some unexpected characters.
- Fixed a bug that could cause Eggplant to hang under certain circumstances (having to do with edited scripts) when doing a delayed quit.
- Fixed a performance issue with the URLEncode() function, also used by MakeURL() and the post command, that could have a severe impact when working with very large strings.
- Fixed an issue with modifying a sublist (or single item) within a global property that's a list in some cases (e.g. "add 100 to item 3 of the searchRectangle").
- Fixed a bug with the square bracket messaging syntax when exactly one parameter is passed.
- Corrected the **params()** function to show lists and property lists properly without extraneous quotes.
- Enhanced syntax colorization of lists and function calls to help differentiate them.
- Improved the error message reported by the **run** command when attempting to run an object that has no script.
- Fixed the **run** command to work properly with a list passed using "as parameters".
- Fixed a crashing bug that could occur when creating a list from a property list.
- Fixed a bug with setting a property of an existing property of a property list in some situations.
- Fixed a bug with vector addition involving global properties.

- Changed the name of the "SRUN_UnknownCommand" exception to "STUnknownCommand".
- Fixed a problem with the compiler that prevented it from being able to parse lists containing more than 246 items or other very long or complex expressions. There is now no practical limit -- it should be able to handle lists of essentially any size.
- Fixed a bug with setting of initial properties (from a SenseTalk "properties" declaration) when a script is loaded.
- Fixed a bug with command (but not function) messages being sent to the target of a send command rather than to 'me' within handlers called indirectly by the send.
- Fixed a bug that could result in "NSInvalidArgumentException *** -[NSCFDictionary intValue]: selector not recognized" when working with a script containing a properties declaration.
- Fixed crashing problems when resizing the Remote Screen window after a screen change.
- Fixed a problem in the "hide" commands which could crash Eggplant.
- Fixed a rare crash that could sometimes occur when searching for an image after a screen resize.
- Fixed a crash when Eggplant attempted to draw the cursor into a movie immediately before a screen resize.
- Fixed a problem with certain mouse events that involved moving the mouse by an amount less than the mouseMoveSpeed, particularly when shouldRepositionMouse was true. This bug led to some strange behavior on SUTs running Mac OS 9 including unexpectedly switching focus to another application or failing to perform clicks. It is unknown whether it caused problems on any other systems.
- Fixed a small problem with date/time arithmetic involving properties of objects.
- Fixed a bug with the callStack() function, when called from a script called out to by a built-in command or function.
- Fixed a bug with the median() function, when called with date/time values.
- Improved the error message displayed if an attempt is made to set a property of a text chunk.
- Fixed a problem that accessing properties using the "property" keyword would improperly call a function if the property was not found.
- Fixed a small problem that would raise an exception rather than returning empty when accessing a non-existent property of an object when the property name was the name of one of the color functions.
- Statements containing multi-line block quotes are now properly recognized as a single statement during continuous colorization formatting.
- The put statement now uses UTF8 encoding when sending data to standard output. This fixes a problem displaying non-ASCII characters.
- Fixed the add and subtract commands to properly preserve date/time formatting of the destination.
- Fixed or improved a few small issues with colorization. For example, in "`put foo's bar`" the word "bar" is now colored as a literal value.
- Fixed a problem with implicit concatenation being a little too aggressive in some situations (such as with this command that now works properly: `run "some.script"  "some parameter"`).
- Fixed a bug when accessing an inverted range of items in a list in which the specified start of the range is higher than the end of the range.
- Fixed a bug that would throw an exception with the message "STUnknownOperation: Attempt to perform unknown operation: accessProperty" in some situations when trying to access nested properties of a property list.
- Creating a property list with duplicate keys will now throw an exception, rather than silently ignoring the duplicates (in a seemingly non-deterministic manner).
- Property lists will now properly accept references as property values at creation time.

Release v2.22 (13-September-2005)

Highlights:

- Fixed a bug which prevented connecting to OS9vnc servers.

Release v2.21 (30-June-2005)

Highlights:

- Fully compatible with Mac OS X 10.2 (Jaguar) as well as 10.3 (Panther) and 10.4 (Tiger).
- Added ability to have Eggplant call an attemptRecovery handler that you supply before a script fails due to inability to find an image on the SUT.
- A number of interface enhancements, particularly on the Results tab in a Suite Editor window.
- Fixed some bugs and compatibility issues.

User Interface:

- Resources such as images and scripts can now be opened by option-clicking them in the Run window as well as from a Script Editor window. Image names containing spaces are now displayed quoted in the Run window's log view to support this feature.

- The Results tab now includes "Link" icons in both the Script Name/Run Date list and on relevant lines in the Log area. Clicking a link icon next to a script name will open the RunHistory.csv file which contains the record of all runs for that script. Clicking a link icon next to a run date opens the LogFile.txt file containing the detailed results log for that run. In the detailed log area, a link icon will select an Eggplant image in the Images tab of its suite, open a captured screen image or Screen_Error.tiff file in an external image viewing application, or open the movie file created by a StartMovie command in a movie viewer.

- The Results tab now includes a Duration column showing the total time taken for each script run.

- The Results tab now displays images from Helper suites as well as those from the local suite for the selected line in the detailed results log.

Scripting:

- Omega13: This is a feature that allows you to supply an attemptRecovery handler; before a script would fail due to inability to find an image on the SUT, Eggplant allows an opportunity for your attemptRecovery handler that to correct the problem. To view these notes within Eggplant, simply type "Omega13" in the Ad-Hoc Do Box at the bottom of the Run window:

  Omega13 is a script providing alternate versions of many Eggplant commands and functions that will give your script an opportunity to try to recover from an "Image Not Found" condition and proceed with execution.

  To use Omega13, add the following command to your script:

  ```
  start using Omega13
  ```

  To return to standard operation, execute this command:

  ```
  stop using Omega13
  ```

  These handlers override Eggplant's built-in commands and functions that search for an image on the SUT screen, in order to allow a custom recovery routine to be written which can attempt to deal with any unexpected events such as pop-up panels that may be interfering with finding the image. Your attemptRecovery function will be called once after a given failure to find an image.

  If you supply an attemptRecovery handler, it should return "yes" or "no" to indicate whether or not it took any actions that might enable the image to be found. If "yes" is returned, the Omega13 script will try a second time to carry out the original action. The attemptRecovery handler is only called once when an image is not found, so it should attempt to deal with all exceptional conditions that it knows how to recognize and address.

When your attemptRecovery handler is called, it will receive three parameters:
- the exception that occurred in trying the original command;
- the name of the command or function; and
- a list containing the parameters that were passed to that command or function.

Your handler can make use of this information in any way it chooses. The command name and list of parameters are passed by reference, allowing the attemptRecovery handler to change their values prior to Omega13's second attempt to carry out the command.

- Added **x**, **y**, **width**, **height**, **origin**, and **size** functions to return the x and y coordinates of a point or rectangle, and the width, height, origin point, and size of a rectangle. When accessed using the "of", "." or "'s" syntax these functions have the feel of properties, making them very convenient to use (although they are not actually properties, so they can't be used to modify values):

```
put foundImageLocation().x -- displays the x coordinate of the hotspot of
the last image found
put ((20,10),(60,100)) into aRect
put the width of aRect -- 40
put aRect's origin -- (20,10)
put aRect.y -- 10
put size of aRect -- (40,90)
```

- The **version()** function and **the long version** now return more complete information about the current versions of both Eggplant and SenseTalk.

- Enhanced the result returned following a **runWithNewResults** command to include an ErrorMessage property when the called script fails.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Fixed a bug with monitoring file system updates after capturing images that would cause a crash under Mac OS X 10.2, and could generate error logs in the console under some circumstances on other platforms.
- Fixed a bug in which Eggplant could spend time repeatedly recording a thumbnail for an image in certain situations.
- Fixed a recent bug with re-throwing caught exceptions.
- Fixed a bug when clicking "Show Results" in the Script Editor toolbar for a script that has no results.
- Fixed a bug with the "as parameters" construct when calling functions on a specific object.
- Fixed a problem with opening Eggplant scripts from the Finder by double-clicking them or dragging them to the Eggplant icon.
- Fixed an intermittent problem with opening an image in an external image viewing application by double-clicking or dragging the image in the Images tab.
- Fixed the CapsLockDown constant for use in a TypeText command, which had inadvertently been spelled CaspLockDown.
- Enhanced **send** to allow "as parameters" and named parameters (property list without enclosing parentheses).
- Fixed a bug where Eggplant would raise an [NSConcreteFileHandle fileDescriptor] exception

---

Release v2.2 (9-June-2005)

Highlights:

- Fully compatible with Mac OS X 10.4 (Tiger).
- Suite synchronization and integrity has been significantly improved when running multiple instances of

Eggplant.

- Improved responsiveness when using Suites with a large number of images.
- New script capabilities including multiple variable assignment, a format() function, and more.
- Improved handling of lists in scripts.
- Fixed a number of bugs and compatibility issues.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.

Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- **insert** command: When inserting a list of values into another, the **insert** command now inserts the source items directly into the main destination list rather than making a sublist. The **put ... into** and **set** commands have been changed in corresponding ways when working with lists. These changes make working with lists more natural and intuitive. If you have scripts which rely on the old behavior, you can either update the individual commands that were affected (by changing **into** to **nested into** for example) or set the local **listInsertionMode** or global **defaultListInsertionMode** property to "nested". To revert to the old behavior for all scripts, issue the following command in a Terminal window:

  defaults write Eggplant STDefaultListInsertionMode nested

- Unquoted Literals: SenseTalk is now somewhat more restrictive about its use of unquoted literals. When assigning to or modifying the value of an undefined variable, its value is considered to be empty rather than being treated as already having the value of its name.
- **Contains** and **is in** operators**:** The **contains** and **is in** operators have been changed for cases where the container being examined is an object or property list.
- RemoteWorkInterval: Changing the RemoteWorkInterval during script execution now changes the timing for the very next event on the SUT. Previously the change was delayed until the following event, making it very difficult to adjust timing precisely for specific events. This may impact scripts that change the RemoteWorkInterval dynamically.

User Interface:

- The shortcut for displaying a resource from a script has been changed from clicking with the Command key, to clicking with the Option key. This change was necessary because Mac OS X 10.4 (Tiger) uses the command key when clicking in text to allow noncontiguous selection of text.
- Option-clicking the name of a resource is now also available in the Run window to show the image or script that was clicked on.
- A new preference option on the General Preferences pane – "Bring Run Window forward when a script is run" – allows you to control whether you would like the Run window to always be brought to the front when a script run begins.
- The Run window is now brought to the front whenever a running script becomes paused in the debugger.
- Improved the auto-completion feature in the Script Editor for image names of images that are located in subfolders within the Images folder.
- Folders are now listed first in the Images tab, before any image names.
- The Delete button for deleting logs on the Results tab was moved to a more suitable location at the lower left of the log list.

- Pressing Shift-Tab while in Live Mode now passes that key sequence properly to a Windows SUT.

- Improved the performance of listing images in the Images tab and in the Capture Image panel when a suite contains a large number of images.

- Added suite record locking and synchronization for much more reliable behavior when running the same script or multiple scripts in the same suite from many instances of Eggplant at once (whether from the full Eggplant application, or scripts run from the command line).

- Attempting to quit Eggplant while a script is running presents a panel asking if you would like to abort the script. If an attempt has already been made to abort a script, this option is now presented as "Kill and Quit". This provides a clean way to exit Eggplant even when a script can not be easily aborted.

- Added striped backgrounds to tables in the Suite window (on Mac OS X 10.3 and later).

- The RunHistory.csv file for each script in the Results folder has been enhanced by adding an Error Message item for each run.

Scripting:

- The key mappings for ApplicationKey, WindowsDown and WindowsUp in the TypeText command have been changed to conform with RealVNC 4.0 so it is now possible to send these keystrokes to Windows SUTs.

- Added TypeText support for these additional keys: PrintScreen, PauseKey, ScrollLock, and NumberLock (the "num lock" key -- the previously implemented NumLock corresponds to the "num lock" key on the keypad, which is still supported as well).

- Added **show runWindow** and **hide runWindow** commands to control visibility of the Run window under script control.

- Added a **KnownHosts()** function to get a list of all VNC hosts for which information has been saved, or that are currently visible through a *Bonjour* connection.

- The **SendMail** command will now log a message when attachments are included if it can't find the specified attachment files.

- The **insert** command has been changed to work in a more natural fashion, be more flexible, and provide some additional capabilities. Previously, inserting a list into another list would create a nested list. Now the default behavior is to insert all of the items of the inserted list into the destination list:

```
put (1,2,3) into aList
insert (4,5) after aList -- (1,2,3,4,5) [ formerly: (1,2,3,(4,5)) ]
```

- The old behavior can be achieved on a single command by using the new "nested" keyword. The corresponding "item by item" keyword ensures the new behavior:

```
put (1,2,3) into aList
insert (4,5) item by item after aList -- (1,2,3,4,5)
insert (6,7) nested after item 2 of aList -- (1,2,(6,7),3,4,5)
```

- The new **listInsertionMode** local property, and the corresponding **defaultListInsertionMode** global property, can be set to "nested" or "item by item" to set the standard behavior within the local handler, or globally for all handlers, respectively:

```
put (1,2,3) into aList
set the listInsertionMode to "nested"
insert (4,5) after aList -- (1,2,3,(4,5))
```

- The **put ... into** and **set** commands have been changed somewhat when the source is a list and the destination is an item or range of items within a list. The new behavior is similar to the new behavior of the **insert** command, and is intended to allow you to work with lists in a more natural and flexible fashion. Previously, putting a list into a sublist of another list would create a nested list. Now the default behavior is to create a nested list if the destination is a single item, or to replace all of the items with the items from the source list if the destination is a range of items. In both cases, the "nested" or "item by item" options can be specified to achieve a particular result:

```
put (1,2,3) into aList
put (4,5) after aList -- (1,2,3,4,5) [ formerly: (1,2,3,(4,5)) ]
```

- Added the ability to assign values to multiple variables at once with the **set** or **put** commands. With this capability, you can assign the same value to several variables at once, or assign different values from a list to a list of variables.

```
set (a,b,c) to (1,2,3) -- sets a to 1, b to 2, and c to 3
set (a,b,c) to (4,5) -- sets a to 4, b to 5, and c to empty
set (a,b,c) to (6,7,8,9) -- sets a to 6, b to 7, and c to 8 (the value 9 is
ignored)
set (a,b,c...) to (6,7,8,9) -- sets a to 6, b to 7, and c to (8,9)
set (a,b,c) to 10 -- sets a, b, and c to all be 10
set (a,,c) to (1,2,3) -- sets a to 1 and c to 3
set (a,b) to (b,a) -- swaps the values of a and b
set (prot,,domain) to "http://www.apple.com/dev/xyz" split by "/"
          -- sets prot to "http:" and domain to "www.apple.com"
put (1,2,3) into (x,y,z) -- x=1 y=2 z=3
put 0 after (x,y,z) -- x=10 y=20 z=30
put ("a","b","c") before (x,y,z) -- x="a10" y="b20" z="c30"
put (2,4,6,8) into (first, rest...) -- first=2 rest=(4,6,8)
```

- Handlers can now use '...' to indicate that the last declared parameter should receive all of the remaining passed parameters as a list:

```
catchAll 1,2,3,4,5,6

to catchAll first, second, rest...
          put first -- 1
          put second -- 2
          put rest -- (3,4,5,6)
end catchAll
```

- Added a **format()** function to produce a formatted string. The first parameter to the function is a format string conforming to the C printf() formatting standard. Additional parameters are values to be included into the resulting string:

```
put format("Rate: %3.2f", interestRate) into rateDisplay
put format("%x", mask) -- display mask's value in hexadecimal
```

- Enhanced the **split** and **combine** commands to allow them to be called as functions as well as commands. **Join** may now be used as a synonym for **combine**. Having these available as functions may be handy in some cases:

```
set path to "/Users/doug/Documents"
put split(path, "/") into components
insert "Eggplant" after components
set newPath to join(components, "/")
put newPath -- /Users/doug/Documents/Eggplant
```

  The new functions may also be called using an operator-like syntax: "*source* **split by** *itemDelim* {**and** *keyDelim*}" and "*source* **joined by** *itemDelim* {**and** *keyDelim*}" (the words "using" or "with" may be used in place of "by", and "combined" in place of "joined"):

```
put path split by "/" into components
set newPath to components joined using "/"
```

- The **delete** command has been enhanced to enable deleting one or more occurrences of a particular text string within a container, in a very flexible manner similar to the **replace** command:

```
delete the third tab in combinedRecord
delete every occurrence of "vulgar" in cleanedUpEntry
```

- Added a **median()** function, to calculate the statistical median of a list of numbers.

- Added **millisecond()** and **microsecond()** functions to provide convenient access to the millisecond (0-999) and microsecond (0-999999), respectively, within the current second or within the time value passed as a parameter, if any.

- The **contains** and **is in** operators have been changed for cases where the container being examined is an object. Previously, in this case, the object's string value would be obtained and a string "contains" operation performed. Now, the object is sent a **containsItem** function message with the search value as a parameter. A second parameter indicates whether the test should be case sensitive. If the object (or one of its helpers or another object in the message path) implements a handler for this message, the return value from that handler is used as the value of the operator. If the **containsItem** message is not handled, the operator returns true if the search value is equal to one of the object's keys or to one of its values:

```
set comic to (title:"Peanuts", author:"Schultz")
put comic contains "title" -- true
put "Peanuts" is in comic -- true
put "nuts" is in comic -- false
```

- Added a **callStack()** function, to return a list of SenseTalkFrame objects providing information about the current execution frames.

- Added more information in the exception objects caught by a **try / catch** construct, including a CallStack property that lists the stack frames in effect at the time the exception occurred.

- Updated the **messageType()** function to return "GetProp" or "SetProp" when appropriate, in addition to "Command" or "Function".

- Initial values for all SenseTalk global properties can now be set in the user's defaults database. The keys are the same as the name of the global property, prefixed by the letters "ST" and using appropriate capitalization. So the key for the defaultItemDelimiter property is "STDefaultItemDelimiter". The complete list of keys is: `STDefaultItemDelimiter, STClockFormat, STDefaultNumberFormat, STShellCommand, STReadTimeout, STStrictVariables, STStrictProperties, STExplicitVariables, STBreakpointsEnabled, STListPrefix, STListSeparator, STListSuffix, STPlistPrefix, STPlistEntrySeparator, STPlistKeySeparator, STPlistSuffix, STEmptyPlistAsText, STDefaultListInsertionMode`

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Improved stability by serializing calls to generate thumbnails of images.
- Fixed a problem opening scripts by double-clicking in the results log when the suite was moved since the log was generated.
- Fixed a problem getting notification of changes made externally to the imageinfo file when the image itself was not changed.
- Fixed a bug where the list of open documents to be reopened on the next run was being modified even when set to not reopen documents.
- Fixed some error messages so they will properly report to the user (instead of being blocked by the document loading mechanism).
- Fixed a bug with sending binary files as attachments using the **SendMail** command.
- Fixed a bug in which the automatic full screen refresh before a final image search could get skipped in rare circumstances, causing images to not be found when they were on screen but were obscured by a VNC artifact. This happened mostly on multi-processor machines or over slow networks.
- Fixed a bug with the "Show Resource" mechanism logging an exception to the console when a user double clicked.
- Fixed a bug with the first invocation of auto-completion in the Script Editor.
- Fixed a bug (introduced by an earlier optimization) in which putting a variable into a list could, in essence, put a reference to the variable into the list instead of its value.

- Fixed a bug with accessing "any item" of an empty list or "any" text chunk of an empty value.
- Fixed a bug in which an unquoted literal passed as a parameter may lose its value in the called handler, depending on how it is used.
- Fixed a bug with detection of circular references through object properties, that could lead to a crash.
- Fixed a bug that could cause a crash when returning a value from a handler called through **send** or **do** under certain conditions.
- The **add properties** command can now accept an empty value without complaining (simply does nothing).
- The **wait** command can now accept not only a number of seconds, but also any value that can be evaluated to a number of seconds (or that can form a valid wait command).
- TimeInterval and ByteSize expressions can now use values stored in variables as well as literal numbers.
- Added support for 'true' and 'fals' return values from AppleScript.
- Fixed a bug with passing parameters to a command from a list using **as parameters**. This was working properly for function calls, but not commands.
- Fixed a bug with the **sort** command when sorting a subrange of a list -- it will now sort properly in place rather than creating a nested list.
- Fixed a bug with the **insert** command when inserting into a range that spans beyond the end of the destination list.
- Fixed a bug with accessing a range of a list spanning before the beginning of the list (using negative indexes larger than the length of the list).
- Fixed a bug with exceptions raised within the context of a **pass ... and continue** command, which were being ignored if they were not explicitly caught by a try/catch block.
- Fixed an error in the way lists were compared for equality that would evaluate two lists as equal in the case where the contents of one list was a single nested list that was equal to the other list. Two lists will now be treated as equal only if they have the same number of items and all of the corresponding items of the two lists are equal.
- Fixed a small bug with inserting an item into an empty or non-existent item of a list -- the previously empty item now properly becomes a list containing the newly-inserted item, rather than simply being that value.
- Script indenting is no longer applied inside multi-line block quotes (using {{ and }} or << and >> etc.).
- Fixed some issues with accessing or putting things into nonexistent list items specified by negative indexes larger than the size of the list. In some cases these actions will now throw exceptions.
- Fixed some issues with deleting list items or ranges using negative indexes.
- Fixed the **open** *application* command to work with file and application names containing spaces and other non-alphanumeric characters.
- Fixed a bug in which text chunks could evaluate incorrectly in certain situations (in particular, when accessing chunks of a variable used as a loop variable in a repeat with each to iterate over a series of property lists).
- Fixed a rare problem where the Continue button would be ignored if it was clicked at just the wrong time.
- Fixed a problem with the title displayed in the Run window before any scripts had been run.
- Fixed a problem with the SendShiftForCaps preference setting being ignored in Live Mode in the Remote Screen window.
- Fixed a problem with the **openSuite** command checking in strange places when an absolute path was specified.

Release v2.1 (2-February-2005)

Highlights:

- Improved the image save dialog that is used when capturing images.
- New year(), month(), day(), hour(), minute(), second(), dayOfWeek(), dayOfYear(), and dayOfCommonEra() functions simplify some date and time operations.
- The files() and folders() functions now return property lists containing additional information about each file or folder.
- New OSXvnc 1.5 with improved startup item control and clipboard behavior.

- Fixed a number of bugs and compatibility issues.

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- OSXvnc 1.5 is the supported VNC for use with Eggplant we recommend that all Mac OS X systems be upgraded to the latest OSXvnc.

- Accessing a property of an object or property list using the form "property x of y" or "my direct x" will now treat x as the actual property name rather than as a variable. To retain the old behavior, you must insert parentheses around the variable name (e.g. "property (x) of y"). A script is available to help locate potential problems in your scripts.

- When using the setRemoteClipboard command or remoteClipboard() function with a Mac SUT running an OSXvnc(1.5) server that does NOT have pasteboard access (usually when set as a System Startup Item), Eggplant will now throw an exception. Previously, the results were undetermined. Also, the remoteClipboard () value is now immediately set when using the setRemoteClipboard command. This should provide for greater reliability in scripts working with clipboards.

User Interface:

- The image save dialog panel has been redesigned to simplify and streamline its use. The file system browser at the top of the panel now shows only relevant files and folders within the Images directory of the current suite. The new panel also resolves a problem which sometimes prevented saving images when using Eggplant on systems running Mac OS X version 10.2.

- The "Run Script" and "Run Selection" actions in the Run window (accessible from the corresponding items on the Run window's toolbar, and on the Run menu when the Run window is the main window) have been refined. Run Script will run the script whose name is shown as the "Last Run" script in the Run window's title bar. This is the same as the "Run Script" action of the Script Editor window for that script, even if the last run was only a selection from that script.

  The Run window's "Run Selection" action is now enabled when nothing is selected in the Run window (previously, it was disabled in this case). When there is no text selected in the Run window's script view, Run Selection will treat the entire contents of the script view as the selection and run it again. When a portion of that text is selected, Run Selection will run only the selected text.

  Following an error, more than one frame (script or handler called by another script) may be available from the popup menu near the top of the Run window. In this case, Run Script will always run the original script that was last run, regardless of which frame is selected. Run Selection, however, will run whatever is shown in the Run window's script view (that is, the script of the selected frame), or whatever portion of it is selected, if there is a text selection.

- Eggplant now stops moving the mouse on the remote system in Live Mode when the local mouse is outside of the Remote Screen window. A new preference option ("Track mouse outside of window in Live Mode" on the Remote Window preferences tab) allows you to restore the old behavior if preferred.

- Eggplant now checks additional locations for UserKeywords.txt files to use in extending the list of words available for auto-completion while editing a script. As in the previous version, the UserKeywords.txt file in the Contents/Resources folder inside the Eggplant application is used. In addition, Eggplant will look in the /Library/Eggplant folder at the root level of your system, and in the ~/Library/Eggplant folder within your home folder for files named UserKeywords.txt. These new locations are the recommended places to store such a file, as they won't be disrupted when upgrading to a new version of Eggplant. All such files that are

found are used to extend the list of available completions. The files should contain one word or phrase per line. Lines beginning with "//" or "#" will be ignored.

- The list of auto-completions available by pressing Option-Escape in a Script Editor window has been expanded to include predefined constants useful in the TypeText command (such as WindowsAltDown).

- The Remote Screen window title bar now shows the name of the currently running script.

- The Run window displays the name of the last run script.

- Eggplant license files may now be double-clicked in the Finder to automatically launch Eggplant (if it is not already running) and install that license.

- Floating licenses now validate through a new EggplantFloatingLicenseServer which must be running on the local network. Customers with floating licenses will be provided with instructions on how to obtain and use a new license.

Scripting:

- The **SetRemoteClipboard** command will now directly update the contents of the buffer returned by the **remoteClipboard()** function.

- When connected to a SUT running OSXvnc 1.5 or later, calling either the **remoteClipboard()** function or the **SetRemoteClipboard** command will now throw an exception when the clipboard on the SUT is disabled (because the VNC server was run as a system startup item).

- When connected to a SUT running OSXvnc 1.5 or later, the **remoteClipboard()** function can now return the contents of the remote clipboard as soon as a connection is established (previously this would only be available after the first cut or copy operation on the SUT).

- The remote clipboard is now also available as a global property (a "container" in SenseTalk terminology), making scripts even simpler -- to access the remote clipboard either for retrieving or setting its value, simply refer to **the remoteClipboard** anywhere in a script, treating it just as you would a variable.

- Improved the **connect** command to accept Rendezvous names for the ServerID property, or any host name that appears in the list of previously connected remote hosts in the Connection Panel. In this case, Eggplant will automatically use property values from the last successful connection to that host for any properties that are not supplied in the connect command.

- Improved the **connectionInfo()** function to include a "Connected" property. Its value is either "true" or "false" so that it can be tested directly in a conditional statement. Fixed the value of connectionInfo()'s "Rendezvous" property to also have a value of "true" or "false" to indicate whether the connection was made through Rendezvous.

- Added support for typing function keys beyond F12 on the SUT. Typing function keys can be done in two ways, both using the TypeText command. The simplest way is to use one of the predefined constants F1 through F35, like this:

```
typeText F13
```

  Or, you can use an "escape sequence" within the text string being typed, by using a backslash followed by the uppercase letter "F" followed by the number of the function key. This can be part of a longer string, as shown here:

```
typeText "\F9bri\n\F9"
```

  If the next key to be typed following a function key will be a digit key, then you must include a leading zero if necessary to make the function key number 2 digits long when using an escape sequence. Or simply use the predefined constants. The following two examples will both correctly type the F3 function key followed

by the number 6:

```
typeText "\F036"
typeText F3,"6"
```

- Fixed property-access expressions of the form **property x of y** and **my direct x** to treat x as a direct property name, rather than a variable. This makes all forms of property access consistent in this regard. To use x as a variable, enclose it in parentheses: "property (x) of y", or "my direct (x)".
  NOTE: THIS CHANGE MAY CAUSE EARLIER SCRIPTS TO BREAK. A utility script is available on request which will search your existing scripts for possible instances that may need to be updated.

- Added **month(), day(),** and **year()** functions to provide convenient access to the month number (1-12), the day of the month (1-31) and the year number, respectively, of a given date or of the current date if no parameter is given.

- Added a **dayOfWeek()** function which returns a number from 0 to 6 indicating the day of the week, with 0 being Sunday, a **dayOfYear()** function which returns a number from 1 to 366, and a **dayOfCommonEra()** function which returns the day number since the beginning of the common era (Jan. 1, 1 AD). Each of these functions can be passed a date value as a parameter, or will operate on the current date if no parameter is supplied.

- Added **hour(), minute(),** and **second()** functions to provide convenient access to the hour of day (0-23), the minute of the hour (0-59) and the second within the minute (0-59), respectively, of a given time value or of the current time if no parameter is given.

- Added **is later than** and **is earlier than** operators (and appropriate permutations and negations of them) as synonyms for the **comes after** and **comes before** operators. The new synonyms provide a natural alternative for comparing dates and times.

- Changed the **files()**, and **folders()** functions to return new *fileDescription* objects. These objects are property lists containing a number of different properties of the file or folder. For backward compatibility, the *asText* property of each fileDescription is set to the short name of the file or folder, so that is how the object will be displayed.

- Improved the **files()**, **folders()** and **diskSpace()** functions to accept a path parameter beginning with "~". Also, fixed a problem with **folders()** being called with an empty parameter, which would previously throw an exception.

- The **create file** and **create folder** commands have been enhanced to accept some initial properties for the file or folder, including setting the owner, group, permissions, modification date, creation date, type code, creator code, and locked properties.

```
create file "/tmp/xyz" \
          with (groupName:"admin", modificationDate:"5am tomorrow")
```

- Setting a property of an empty or undefined variable (or of an empty item in a list), is now allowed and will automatically turn that value into a property list. This simplifies scripting, and allows a list to be treated as a list of objects even though some items have not yet been set.

- Extended the **is a** operator to be able to test for even and odd numbers:

```
put 7 is an odd number -- true
put 1234 is an even number -- true
```

- For completeness, **does** is now allowed in all operators where **does not** can be used, with opposite meaning (e.g., "7 does come between 3 and 9").

- The short form of "repeat with each" style of repeat loop may now specify "by reference" just like other forms of "repeat with each":

```
repeat with each stamp in stampCollection by reference
     if stamp's status is "sold" then delete stamp
end repeat
```

- The "repeat with n=1 to 10" style of repeat loop may now optionally include the word "up" before "to" (e.g.

"repeat with n from 1 up to 10").

- SetProp handlers may now be called with extra parameters (in addition to the new property value) when the property is set using a syntax that supports passing extra parameters, such as:

```
put "Hello" into Bob's greeting(5)
```

  The command above will call the **setProp greeting** handler (if there is one) in the object Bob, passing two parameters, "Hello" and 5.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Fixed a crashing bug when deleting the results folders for many scripts (20 or more) at once.
- Fixed a crashing bug when searching for images larger than the current searchRectangle.
- Fixed a crashing bug when dragging an open script to another suite from the script's title bar.
- Fixed a bug where image thumbnail generation code could enter protected logic, in some cases causing a crash.
- Fixed a bug where occasionally Eggplant would try to restore a connection on startup that had been closed when Eggplant was last quit.
- Fixed a bug involving comparisons between an object or property list that is stored in a list and one that is not.
- Fixed a bug that prevented deleting empty text chunks.
- Fixed a bug with testing whether an item in a list is a reference, using **is a reference** (would erroneously report that it was not a reference when it was).
- Fixed a bug with writing a list directly (to a file or output stream) using the write command.
- Fixed a bug that could sometimes prevent storing a value into an empty item in a list.
- Fixed a bug that could cause a crash sometimes in rare situations when assigning part of a value to the variable holding that value (along the lines of "put item 3 of x into x").
- Fixed an obscure crashing bug that could occur when accessing "the params" by reference.
- Fixed the run() function to only run the initial handler of an object's script, without going through the message path or even calling any helpers. This also fixed a problem when running a selection that began with a handler declaration, which previously would run the entire script.
- Fixed a bug with text chunks using negative or special indexes (including such things as "last char of ...") not being re-evaluated each time when accessed through a reference.
- Fixed some problems working with items in a list when the list is an object property.
- Fixed the **create link** command, which wasn't working.
- Improved the error message reported by **the result** function following a **copy file**, **move file**, **rename file**, or **delete file** command that fails.
- Improved the efficiency of the **wait while** and **wait until** commands.
- Made a minor correction to allow expressions such as "three and a half minutes" to parse correctly without requiring them to be written as "(three and a half) minutes".
- Fixed the **read** command when reading from standard input, to enable reading a single character at a time without waiting until the user presses the Return key.
- Attempting to set a text chunk to be a reference to something will now properly throw an exception.
- Fixed crashing bug on OS X 10.2 when running commands from the Ad-hoc Do Box without any files in the Run window.
- Changed Eggplant so that thumbnails are NOT recorded when running a script from the command line.
- Fixed Eggplant to terminate immediately when interrupted (because of error or break) when running a script from the command line.
- Empty cursors sent to Eggplant by the SUT are now ignored (so the cursor won't disappear).
- Fixed resolution of Rendezvous names so that it will stop resolving once service information has been received (and it will no longer complain about being a burden on OS X 10.3.6 and below).
- Added explicit alphabetic sorting of Image names, primarily for UFS volumes which do ASCII sort by default.

Release v2.0 (16-November-2004)

Highlights:

- New Run window provides for comprehensive control and interaction with a running script.
- Enhanced script animation and tracing features for tracking a script's progress.
- Rendezvous discovery of OSXvnc servers on your local network.
- Convenient navigation features – a single click will bring up images, scripts, or the latest results for a script.
- User-extensible keyword completion within scripts.
- Enhanced ability to send email from within Eggplant scripts.
- Updated and improved documentation, including expanded articles in "Using Eggplant"
- Extremely stable and reliable

Important Changes (Compatibility):

These changes may impact the behavior of your existing scripts.
Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- Renamed the FindImageRetryDelay and FindImageRetryCount options to ImageSearchDelay and ImageSearchCount. The old names are still supported for backward compatibility.

- Improved the default ImageSearchTime to be 1.8 seconds, by having ImageSearchCount set to 7 and ImageSearchDelay at 0.3 seconds. The value of the imageSearchTime now includes only the delay interval between searches (e.g. 6 x 0.3 = 1.8 in the new default setting), which more accurately reflects the minimum time Eggplant will spend searching before it concludes an image is not found. To restore the previous timing behavior, set ImageSearchCount to 6 and ImageSearchDelay to 0.3.

User Interface:

- Added a new Run window providing extensive control over scripts as they run.
  - Scripts can be paused at any time while running
  - Continue, Step Over, Step Into, and Step Out capabilities give precise control over execution
  - Ad-hoc command execution to interact with scripts while paused or running, or to test commands at any time
  - Interaction with the SUT in Live Mode is possible while a script is paused
- Holding down the Option key while clicking Run Script or Run Selection will start the script in a paused state in the Run window to facilitate debugging. The Debug Script and Debug Selection commands on the Run menu also do this.
- Script Animation can now be set to only show the handler being run without highlighting each line, and can be turned on and off while a script is running.
- A new Script Tracing feature will echo the lines of your code as they are executed, or can be set to just trace calls into and out of other scripts or handlers.
- A new Run menu organizes access to all of the script running and debugging features.

- Added a new Show Resource capability -- simply command-click on the name of an image within a Script

Editor to immediately display that image in the Images tab of the Suite Editor. Similarly, command-clicking the name of another script will open that script in a new Script Editor. This feature is also available from the Edit menu.

- A new Complete command on the Edit menu provides auto-completion of words while editing a script. Type the first 2 or more characters of a word and press the F5 key to pop up a list of possible completions. The words and phrases offered for completion include all Eggplant and SenseTalk commands, functions, properties and run options, plus the names of images and scripts within the suite, and even variable names that already appear in that script. In addition, a customizable file is available for adding your own standardized variable names or keywords for use throughout your scripts.
- A Show Results command, available as both menu and toolbar items, will quickly access the latest results for a script.

- Added Rendezvous servers to the Connection panel. The pulldown list of SUTs will now include auto-detected servers (shown in blue) to simplify initial connection without needing to know the exact name or IP address of that machine.
- Improved user control over the list of remote servers shown in the Connection panel. Simply check or uncheck Remember This Connection to add or remove items from the list.

- Improved the logic establishing the relationship of the ImageSearchTime to the ImageSearchCount and ImageSearchDelay values. The value of the imageSearchTime now includes only the total delay interval between searches. So, if the ImageSearchCount is 7, then there are 6 delay intervals between the individual searches that are performed. If the ImageSearchDelay is 0.3 seconds, then the ImageSearchTime will be 1.8 (i.e. 6 x 0.3 = 1.8). This more accurately reflects the minimum time Eggplant will spend searching before it concludes an image is not found.

- An Abort Script button is now available for inclusion on the toolbar of the Remote Screen window.
- When quitting Eggplant while a script is running the user is offered choices about whether to abort the script or cancel the quit.
- Aborting a script is no longer treated as a script failure, and will not bring up a "script has failed" dialogue.
- A new preference option allows you to choose whether Eggplant should display an alert when a script fails.
- Added a Mail pane in the Preferences panel for specifying the default account to use for sending email from a script using the SendMail command.
- The Script Editor Preference pane now includes additional choices about what to do if there are unsaved changes when starting a script run, including options to save all scripts, or to ask you.
- The Preferences Panel now has tool tips for all panes, and improvements to tabbing between fields.
- The "Add..." buttons on the Schedules and Helpers tabs now display a sheet for file selection rather than a modal panel.
- If a VNC connection can't be reestablished, Eggplant won't automatically try to reopen that connection the next time Eggplant starts up.

Scripting:

- The **SendMail** command has been improved to support a wider array of authentication protocols (configured through the new Mail pane of the Preferences panel). In addition, the underlying mail framework has been replaced, addressing previous issues with robustness and reliablility.

- The **ScriptAnimation** global property can now be set by a script to "All", "Calls", or "Off" to control script animation while a script is running. For backward compatibility, "on", "true", or "yes" are treated as synonyms for "All" and "false" or "no" as synonyms for "Off".

- Added a **ScriptTracing** global property to allow controlling script tracing from within a script. Set it to "All" to trace every line before it is executed, to "Calls" to trace only the entries and exits to/from a script or handler, or to "Off" to turn tracing off. The following example ensures that tracing is set to "All" when calling the commandThatShouldBeTraced:

```
put the scriptTracing into formerSetting
set the scriptTracing to "All"
commandThatShouldBeTraced
set the scriptTracing to formerSetting
```

- The **CaptureScreen** command has been enhanced to allow capturing part of the screen as an image that can be used later in the script, including setting a HotSpot location and other properties. See the documentation for full details.

- Both **PauseScript** and **Breakpoint** commands may be used to cause a script to stop in a paused state during execution. You may then interact live with the remote system, or query the state of the script through the Run window before continuing execution. Breakpoint commands can be disabled by setting the **BreakpointsEnabled** global property to false.

- Added the ability to create *references* to any container, using the words **container** or **reference** or **reference to**, or using the shorthand symbol '**@**'. This can be a powerful feature for advanced scripters to use, particularly for passing or returning values "by reference". See the documentation in Chapter 4 of the SenseTalk Reference manual for full details.

- Handlers can now be specified using **to handle *messageName*** (the word **handle** is optional). This form can handle either function messages or command messages. The **on** and **function** type of handlers will continue to work, but are only called for their specific type of message. If a script has both **to** and **on** or **function** handlers for the same message, the specialized handler (**on** or **function**) will be used.
```
to handle setupForRun with initialValues
     -- do interesting stuff
end handler
```

- The initial handler in a script is now treated as a **to** handler, so a script may be called as either a command or a function.

- All types of handlers can now optionally include the word **with** in their declaration, before the list of incoming parameter names, for added readability.

- Added a **run()** function which will call the target object's initial handler as a function:
```
set doubler to (script:"return 2*param(1)")
put doubler's run(3) -- 6
```

- Added a **delete variable** command, along with its more specific variants: **delete local**, **delete global**, and **delete universal**. Deleting a local variable returns it to an "undefined" state, so subsequent uses of that variable will be treated as unquoted literals until something else is stored in it. Deleting a global or universal variable removes it from the list of defined global or universal variables, until something else is stored in it.

- Added a **readTimeout** global property to control the maximum length of time the **read** command will wait for the requested data to become available. Note that data read from a file should be available immediately, so this only affects reading from sockets or standard input.

- The **read** command can now use chunk terms to read in specified quantities of words, items, and lines, in addition to the characters, integers, floats, and doubles that were previously supported:
```
read 3 words from file poem
read from file toDoList for 1 line
```
Reading chunks in this way is different from using the **read ... until** form to read until a delimiter, as the chunk form will discard the final delimiter from the value returned (and, in the case of words, may discard leading whitespace as well).

- The **read** command now supports flexible syntax, allowing the various parts of the command to appear in any order. In addition, you can now specify **into** *container* to read into a specific container other than `it` :
```
read 3 words from file poem at 27 into preface
read into preface at 27 3 words from file poem
```

- The **read** command can now be called without a **for** or **until** clause to read a single character.

- Enhanced **read from socket...for** to return any available data without blocking. If less data is read than requested, **the result** is set to a value containing 'eof' or 'time out' depending on the reason.

- The **open socket** command can now be aborted. In addition it will now time out and stop trying to connect if it fails to establish a connection within **the readTimeout** seconds.

- Added **this object** as a way to identify the object whose code is currently executing, since this may sometimes be an object distinct from **me** or **the target**. Specifically, **this object** is always the object that owns the script, which is the same as **me** except when this object is acting as a helper for another object. In that case, **me** is the object being helped and **this object** is the object doing the helping. **The target** refers to the object that the message was sent to, which may be different from **me** (and **this object**) if **me** is another object located somewhere along the message passing path.

- Added a **globalNames()** function which returns a list of the names of all currently-defined global variables. This includes all globals which have had values stored into them, unless they have been subsequently deleted.

- Added a **universalNames()** function which returns a list of the names of all currently-defined universal variables. This includes all universals which have had values stored into them, unless they have been subsequently deleted.

- The **date()** function has been changed slightly such that when converted to another format or evaluated as a number it will always represent a time of noon on the given date. In addition, it can take a parameter representing any date or time, and return a value of noon on that day.

- The **time()** function now optionally accepts a parameter representing any date or time, and returns that value in the form of a time, to facilitate time comparisons (otherwise, SenseTalk would perform a text comparison).

- Can now test whether a value **is a time** as well as whether it **is a date** (however there is no distinction between the two -- any date/time value will yield true for both).

- Added **folder** (or **directory**) property of a file-based script object. So, any script that is stored in a disk file can now access the folder where it is located as 'my folder', as in this example:
```
set the folder to my folder
```

- Exceptions raised while working with an image, like Image Not Found now have an **ImageName** property to determine what image was being searched for.

- "On" and "off" are now accepted as valid boolean values, along with "yes", "no", "true", and "false".

- Extended the **URLEncode()** function to accept a property list and encode it in the format used in the query portion of a URL. This is a convenience for producing the same format already created for a query property-list by the makeURL() function. It may be useful for formatting data to be sent to certain web services using the **post** command:
```
post URLEncode(queryValues) to serviceURL
```
- Scripts will now be much more consistent about updating to and running the very latest version when they are modified while scripts are running.

- Enhanced the **doAppleScript** functionality to enable the AppleScript code to return lists or property lists ('records' in AppleScript terminology) as well as simple values. When the return type is not known, a property list is returned containing the type and the data.

- The **number** function can now be used to determine the number of properties (or values) or keys in an object or property list, or the number of occurrences of a particular value among all of its values or keys:
```
put the number of properties in (x:27, y:19) -- 2
put the number of occurrences of 27 \
        among the values of (x:27, y:19) -- 1
put the number of keys in (x:27, y:19) -- 2
put the number of occurrences of 27 \
        among the keys of (x:27, y:19) -- 0
```

- Similarly, the **is among** function can now be used with property keys and values:
```
if "Frank" is among the values of person then ...
if "name" is among the keys of person then ...
```

- Added a facility for passing the items of a list as individual parameters to a command or function, using **as parameters**:
```
calculateVariance bigListOfNumbers as parameters
```
This can be especially useful for passing along to some other handler the parameters that were received by the current handler (obtained using the **parameterList** function):
```
return otherFunction(the parameterList as parameters)
```

- Invocations of the **open file** command that create a file will now create the full path to that file if needed.

- The **average()**, **maximum()**, **minimum()**, and **sum()** functions will now accept nested lists -- to any depth -- of numbers and/or strings containing numbers separated by commas. Empty values within a list are now ignored.

- Changed **min()** and **max()** functions to **minimum()** and **maximum()**, with synonyms for the old names **min** and **max**.

- Added **and if** and **or if** operators, which are "short-circuit" versions of the **and** and **or** logical operators. Using these new operators, the expression after the operator will not even be evaluated if the resulting value can be determined from the expression before the operator. That is, if the expression before an **and if** operator is false or the expression before an **or if** operator is true, then the entire expression will evaluate to that value.

- Fixed dates and times to be available as numbers (so "the date is a number" is now true). To force a date or time to appear as a number, you can multiply it by 1:
```
put the date -- 09/16/04
put the date * 1 -- 117050400
put the time -- 12:09 PM
put the time * 1 -- 117050933.606729
```

- Date/time arithmetic now preserves formatting, so, for example:
```
put the date + 24 hours -- 09/17/04
put the long date + 24 hours -- Friday, September 17, 2004
```

- Improved the addition of a number and a date or time so that adding in either order will now produce a date/time result:
```
put 24 hours + the date -- 09/17/04
```

- Added **the clockFormat** global property to specify whether time formats should use a "12 hour" (with "AM" and "PM" -- the default) or "24 hour" format. This affects formats used by all of the time functions (the time, the long time, the local time, etc.), and by the convert command.
```
put the time -- "06:18 PM"
set the clockFormat to "24hr" -- any value starting with "24" works
put the time -- "18:18"
```

- Properties declarations in a script are now properly indented and formatted. Lists and property lists that are enclosed in braces { } are also indented if they include several lines. Properties declarations may now appear *before* the initial handler in a script.

- Improved handling of invalid properties -- specifically helpers -- listed in properties declarations, so only the invalid ones are ignored instead of the entire group of properties.

- A script can now check whether an object has a particular property using the **there is a** function:
```
if there is a property "size" of myObject then ...
```

- Added a **messageType()** function that returns the type of the current message, either "Command" or "Function".

- A property list can now be passed as a parameter to a command without needing to enclose it in parentheses, which gives the effect of having named parameters (the called command receives a single property list parameter):
```
connect host:"10.0.0.3", port:5901, password:"frumple"
```

- Objects without an **asText** property may now contain an **asTextFormat** property instead. Its value will be evaluated as a merge() expression to produce the formatted text value of the object.

```
set account to {balance:1234.25, type:"Savings",
    asTextFormat:"[[my type]] Account: [[my balance]]"}
put account -- "Savings Account: 1234.25"
add 5050.50 to account's balance
put account -- "Savings Account: 6284.75"
```

- Expanded the implicit concatenation literals to include **comma**, **slash**, **backslash**, **cr**, **lf**, and **crlf**.
- Can now use **error** as a synonym for **stderr** in a write command.
- An else block may now end with **end else** instead of **end if**. May now use **exit to top** as a synonym for **exit all**. **End handler** may now optionally be followed by the handler name.
- The word **then** is now optional in **if ... then ...** constructs in situations where **then** would be at the end of a line.
- Removed the old limitation on length of a handler. The only limit on script size imposed by the compiler now is that an individual handler cannot be more than 32,767 lines long.
- Made some small enhancements to the ability to express fractions as words.
- Significantly improved performance when establishing a remote connection from a script.
- **CaptureScreen** now supports an imageInfo property that is itself a property list (see imageInfo() and its property list for accepted keys).
- Added property (scriptTracing) To Control the new Script Tracing behavior.
- Added property (scriptAnimation) To Control the new Script Animation behavior.

Bug Fixes / Tweaks:

The following bugs and minor issues were resolved for this release:

- Eggplant now notices changes to images made through the file system and reloads the image when needed.
- Automatic scrolling in the Remote Screen window has been improved for selections that are very large.
- Increased maximum size of the Image Name field in suite Results pane
- Improved writing of Connection process during command line execution
- ConnectionInfo() no longer reports internal status variables
- Fixed a bug with "the last line" or "the last item" (or "line -1" or "item -1"). When the source ended with the delimiter character, the delimiter was incorrectly being returned as part of the chunk.
- Fixed crashing when Starting A New VNC Connection with an existing one up.
- Fixed lockup when Starting A New VNC Connection with an existing one up.
- Fixed a crashing bug that could occur when switching an Image to a TextMatching Type (including from within a script)
- Fixed a crashing bug (findSingleImage/findAnyImage) which could occur when running a script after failing to find a prior image.
- Fixed a crash with changing Remote Window Title during reconnection attempts.
- Fixed a rare crash with references to scripts/suites from SenseTalk when they were already open inEggplant
- Fixed a rare crash when running schedules of a suite sometimes
- In the Schedules tab's info drawer, the password field in this panel is now a real password field with masking.
- Fixed a problem with Save Panel refusing to move to recently moved/deleted folders.
- Fixed a bug with accessing files through aliases. If the local name of the alias was longer than the local name of the file it referred to, SenseTalk was adding the extra characters from the alias to the actual file name, making it unable to find the file.
- When accessing a non-existent or inaccessible file as a container, the value will still be returned as empty, but SenseTalk will now also set **the result** function to a message indicating that it was unable to read the file.
- Fixed a problem with auto-indenting and colorizing of a **params** declaration as the first line of a named handler.
- Fixed the **local time** functions to work properly and maintain formatting with date/time arithmetic.

- Fixed the "local time" and "short local time" formats, which had been reversed.
- Fixed a problem with the **answer** command when specifying a panel type.
- Fixed a bug with accessing the **parameterList** function using the syntax **the parameterList** (could cause a crash in some cases).
- Fixed a problem with "Exception Raised During Posting of Notification"
- Fix to a problem setting hot spot in image info (either on override or for saving an image).
- Fixed a problem accessing ScriptAnimation and ScriptTracing with SetOption and GetOption.
- Fixed a problem with 'me' and 'my' within expressions evaluated by the **merge()** function.
- Fixed reporting of errors that occur within a merge().
- Fixed a bug with compiling any script with a backslash as the last character of a quoted string within a properties declaration or a list or property list enclosed in curly braces { }.
- Fixed a bug with script indentation that could lead to a crash in extreme cases.
- Fixed a bug displaying a property list with an empty property name (a property with a key of "").
- The error thrown by the **value()** function is now an exception object rather than a simple string when an error occurs in evaluating the **value()** function.
- Fixed a problem with setting the properties of a folder, and with accessing file or folder properties with two-word names.
- Fixed comparison of long ID's of objects, so you can meaningfully perform tests such as "if this object is the target".
- Fixed **read from socket...until** to work properly.
- Fixed a bug that would throw an exception when setting a property of a script object to a property list value.
- Fixed a bug with sending messages to objects that are stored as properties of another object.
- Fixed a bug in which "property x of y" would improperly call function x instead of accessing property x directly.
- Fixed a crash that could occur while clearing the object cache.
- Fixed a bug with inserting a list consisting of chunk expressions into a list (was resulting in the same value multiple times in the final list).
- Fixed a problem with properties passed as parameters being treated incorrectly in the called handler.
- Fixed a problem with properties declarations in scripts.
- Fixed a problem with renaming a script and/or renaming a suite.
- Fixed a problem running **captureScreen** in a selection causing SenseTalk thread to lock up and report bad exception.
- Fixed a problem of not able to cancel a connection during certain parts of a connect.
- Fixed a problem passing in numbers (not strings) as imageTolerance to image property lists.
- Fixed a problem sometimes saving correct state of Known VNC Servers.

Release v1.5 (21-April-2004)

Highlights:

- Improved mechanism for selecting images to be captured by dragging out a selection rectangle.
- More flexible and dynamic image capture dialogue using a sheet rather than a modal panel.
- New Search Time option on ImageFound() and AnyImageFound() functions for improved usability – just include a time as the first parameter.
- Ability to store scripts as plain text (but still have them appear formatted in Eggplant), for better use with version control systems.
- Ability to run AppleScript code directly from within SenseTalk.
- Significant upgrades to SenseTalk, including easier ways to call functions in other scripts (see the section on Scripting below)
- Upgraded to OSXvnc 1.33 for use on Mac SUTs

- Updated documentation, including improved description of SenseTalk property lists and objects in the SenseTalk Reference Manual
- We have corrected a number of bugs, inconsistencies and annoyances
- Beta Feature: Movie Recording functionality is now included for trial use and feedback.

Important Changes (Compatibility):

These changes may impact the functioning of your existing scripts. Please read these descriptions, plus any further explanations later in the release notes, to be sure you are aware of any changes you may need to make in your scripts.

- We have fixed an issue with "NextKeyDelay" being ignored, so anyone who set "KeyDownDelay" to a higher value to compensate should return it to its regular value of 0.001.

- For improved reliability we have modified Eggplant so that after any script or selection is executed (including scripts from the scheduler) the SUT will have its mouse buttons and modifier keys RELEASED if they had been down (previously this was not always the case). We found that most people unintentionally were impacted by this. If you have scripts that depend on earlier scripts leaving keys or mouse buttons down, you will need to use Universal variables or some other mechanism to restore down states on modifier keys and mouse buttons.

- If you have a TypeText command with multiple arguments: Eggplant used to wait the entire Remote Work Interval between sending the strings; now it only waits the Next Key Delay. Example:

```
TypeText "Hello", "World"
```

- The Set command has been enhanced to be more flexible. As a result, any scripts which use the set command to set the value of a global property *must* now use the word 'the' before the property name (previously, 'the' was optional -- see longer description below). Example:

```
set the numberFormat to "00.00"
```

- A pair of empty curly braces { } in a script will now be interpreted as an empty list, rather than an empty property list.

- Because the AnyImageFound() function now takes an optional Search Time for its first argument, if you were using this function and the first argument looks like a time value (e.g. "5") it will now delay and not look for such an image. Please refer to images that have only numbers for a name with their .tiff extension (e.g. "5.tiff").

- Unquoted literals in SenseTalk are now case-sensitive (previously they would evaluate to all upper-case). Any scripts which relied on the old behavior may need to be changed.

- This version of Eggplant will not run on versions of Mac OS X earlier than 10.2 (Jaguar). Any version of the Mac OS with a supported VNC server (including Mac OS 9) may still be controlled as a SUT, of course.

User Interface:

- The Remote Window now supports Panning (automatic scrolling) when it is running in a window smaller than the remote screen. A preference allows you to turn this behavior on or off.
- Improved Capture Image Behavior:
  - You can now drag out a selection to define the capture area. This can be much faster when capturing images of varying sizes and locations.
  - When working with the capture area Eggplant will now Autoscroll as needed when the remote screen is larger than the window.

The image save panel for new images is now displayed as a sheet rather than a modal panel, and updates dynamically if a different script or suite window is selected while it is up. The fact that it is a sheet also allows you to change the insertion point in the script before clicking Save.

The image save panel now has a popup list to allow switching the command to be generated after the sheet is brought up. This popup also lists functions as well as commands.

The image save panel shows a preview of the command or function that will be inserted into the script when you click Save (this feature is somewhat limited under Mac OS X versions earlier than 10.3 (Panther)).

- The Open panels displayed by the items on the "Insert..." pulldown on the Script Editor toolbar and by Option-clicking many of the icons on the Remote Screen toolbar now display a thumbnail of the selected image rather than a generic icon. This can be turned off using a preference if you don't want to increase the extra size or you can't have images with resource forks.
- All Screen Shots are now compressed using LZW compression. This should make Results folders much smaller.
- Added a preference to disable auto detection of image types for use when connected via low-bandwidth or if capturing a bunch of images which need to be set to a particular type.
- Script Animation may now be turned on or off during script execution. It can also be changed from within a script, by setting 'the ScriptAnimation' global property.
- All SUT-related commands are now logged (including mouseButtonUp/Down, setSearchRectangle, openSuite, closeSuite).
- Added support for keys on the numeric keypad throughout Eggplant (Live Mode, Script Mode, and OSXvnc server).
- Added a preference to specify if script files are stored as Rich Text (RTF) or Plain Text.
- Added a menu command to toggle the format of a script (from Rich Text to Plain Text or vice versa).
- You may now double click on scripts in the Schedules tab to bring up that script's editor window.
- The Live Remote Window screen will now attempt to release any mouse buttons or modifier keys that are held down when the window loses its focus. Previously actions like Command-Tab and Command-Space could leave the SUT in a state where its command key was held down.
- Tweak Remote Window Capture view to only engaged double click on a mouse up.

Scripting

- Eggplant's runtime options (everything that can be set from the Run Options pane of the Preferences panel, or using the SetOption command) are now accessible as global properties -- just use the word 'the' in front of the option name in a script, to access or set its value:

```
set the remoteWorkInterval to 0.8
```

  Because SenseTalk treats properties as containers (just like variables) you can even do things like this:

```
add 1 to the standardImageTolerance
```

  Previously, to do the same thing, you would have needed something like this:

```
SetOption "standardImageTolerance", getOption("standardImageTolerance")+1
```

- Added the following script commands, described in the Eggplant Reference Manual:

  `show RemoteWindow` -- To Bring the Remote Window to the foreground in the GUI
  `hide RemoteWindow` -- To Hide the Remote Window in the GUI

- Added a number of predefined constants as alternatives to escape sequences for special keys. So, for example, typing *Control - PageDown* can now be done like this:

```
TypeText ControlDown, PageDown, ControlUp
```

  Please see the TypeText command on pages 112-113 of the Eggplant Reference manual for a complete list of these constants.

- When passing a property list in place of an image name, a SearchRectangle property may be included to restrict the search for that image to a particular region.

- The **CaptureScreen** command can now take a property list that allows you to specify Name, Rectangle and Increment values. Although the old syntax is still supported, the new one allows you to easily specify just the options that you need.

- As part of an overall effort to make SenseTalk more consistent and easier to use, the **set** command has been enhanced to allow it to set the value of variables (or any container), not just properties. This makes its functionality identical to that of the **put ... into** command. **Set** also allows some optional words (be, equal to) for more English-like syntax:

```
set x to be 7
set y to x - 1 -- (y is now 6)
set z to be equal to x times y -- (z is now 42)
```

  COMPATIBILITY WARNING: Due to this change, it is now necessary to use the word 'the' whenever referring to a global or local property (one that is not a property of some specific object). This was already necessary everywhere except in the **set** command. **THIS CHANGE MAY BREAK SOME SCRIPTS** if they use the set command to set one of the global or local properties without using 'the'. The only change needed to comply with the new syntax is to insert the word 'the' as needed. For example, take the following command:

```
set numberFormat to "0.00"
```

  In previous versions of SenseTalk, this would change the local numberFormat property. Starting with this release, the above command treats numberFormat as a variable name and sets the value of that variable, not the property as in the past. To restore the previous behavior of setting the numberFormat property, you must add the word 'the', like this:

```
set the numberFormat to "0.00"
```

- In another major step toward greater consistency, accessing a property of an object or calling a function in a specific object or script are now both done in a unified fashion using the dot (**.**), apostrophe-S (**'s**) or **of** syntaxes. To see how this works, consider the following three examples (which are all equivalent):

```
put gumdrop.color
put gumdrop's color
put the color of gumdrop
```

  If `gumdrop` is an object, any of the examples above will call its color function if it (or one of its helpers) has one. If it doesn't have a color function, its color property will be accessed. If `gumdrop` is not an object, or if it doesn't have a color function or property, the color(gumdrop) function will be called.

  Parameters may also be passed:

```
put gumdrop.color("sprinkles")
put gumdrop's color("sprinkles")
put the color of gumdrop with "sprinkles"
```

  Here, the parameter (the word "sprinkles") will be passed as a parameter to the color function, and also to the getProp handler if it is called. If the final step of sending a 'color' function message to `me` is invoked, both `gumdrop` and "sprinkles" will be passed as parameters (that is, it will be the same as calling color(gumdrop, "sprinkles")).

  Note that these changes not only make it easy to call a specific function within another script, but also allow the dot and apostrophe-S syntaxes to be used to call any function that takes a single parameter:

```
put "ambrosia".length -- 8
put (1,3,5,7)'s average -- 4
```

- Added **getProp** and **setProp** handlers. These allow an object to respond to requests to access or set one of its properties. See chapter 9 of the SenseTalk Reference Manual for details.

- A new property access syntax (**my direct** <propertyName>, or **my direct property** <propertyName>) can be used to avoid sending any function or getProp/setProp messages.

- Added a **doAppleScript** command (and function) to execute a string as AppleScript code. Use this along with the merge function to insert values into the AppleScript. For example, the following will count the number of words in a Microsoft Word document whose name is in the local variable wordDocName:

```
put doAppleScript of merge of {{
    tell application "Microsoft Word"
          open "[[wordDocName]]"
          count words in window 1
    end tell
}}
```

This can also be called as a command using any of these formats:

```
do AppleScript aScript
do aScript as AppleScript
doAppleScript aScript
```

- Added **strictVariables** and **strictProperties** global properties to provide stricter usage of variables and properties at runtime. When the strictVariables property is set to true, accessing an uninitialized variable will throw an exception rather than evaluating as an unquoted literal. Similarly, when strictProperties is true, accessing an undefined property of an object (one that has never had a value set) will throw an exception rather than returning empty. Both properties are initially set to false. Setting them to true can aid debugging by helping to find variable or property names that were inadvertently misspelled, for example.

- Extended the **is a** operator to be able to test the **objectType** of any object. When testing whether an object is of a particular type other than the standard types, the object is sent an **isObjectType** function message, with the type as its only parameter. The built-in default implementation of the isObjectType function checks whether the requested type is among the items of the object's objectType property (which may be a single type or may be a list of several types).

- Added **propertyList** as a synonym for **object** for the **is a** operator (so you can now say "`if someVar is a propertyList ...`").

- Added **add properties** and **remove properties** commands, for modifying an existing object by adding and removing properties.

- Added **plus properties** and **minus properties** operators, for generating modified property lists on the fly by starting with a property list and adding and removing properties.

- Added the ability to write handlers that can be called on receiving any message of a particular type (on, function, getProp, or setProp) by specifying **<any>** as the handler name.

- Added a **sum** function to return the sum of all of its parameters or all of the items in a list.

- Allowed the **end** line of a handler to be somewhat more flexible and consistent with other uses of **end**. An **on** handler may now end with **end on** or **end on *handlerName***, and similar **end** lines may be used for **function**, **getProp** and **setProp** handlers, using the corresponding key word.

- Changed the **exit** and **pass** commands to be somewhat more flexible. They will now optionally accept the handler type (**on**, **function**, **getProp** or **setProp**) in place of the handler name.

- Made the **sort** command smarter about understanding things like:

  ```
  sort the last 5 items of bigList
  ```

  which previously would have to be written as:

  ```
  sort the items of the last 5 items of bigList
  ```

- Changed unquoted literals to evaluate case-sensitive using the case of their first use in the script (not counting declarations), instead of evaluating to all UPPERCASE as in previous versions. **WARNING: This change might impact some existing scripts.**

  ```
  put Hello -- displays "Hello" rather than "HELLO" as in the past
  ```

- Added a **parameterList()** function that returns a list containing all of the parameters that were passed to the current handler. This can sometimes be more convenient when working with a variable number of parameters than using the **paramCount** and **param** functions.

- Added an **env()** function to access environment variables. When called without any arguments, it returns a property list containing all environment variables and their values. If called with a variable name as an argument, it returns just the value of that environment variable.

- The **properties** declaration in a script can now be formatted like a standard SenseTalk property list. The old format is still permitted, but this is the preferred format:

```
properties
    name:"Johnson", -- comments are allowed in this format
    phone:"555-1212",
    age: 32  -- comma is optional following the last property
end properties
```

- The **shell()** function now sets 'the result' to the shell command's exit code.

- The **pass message and continue** command now sets 'the result' to the value returned by the passed message.

- Double curly braces **{{ }}** can now be used to enclose text strings spanning multiple lines. These block quotes can be tagged for uniqueness.

- Curly braces **{ }** can now be used instead of parentheses to enclose lists or property lists. When curly braces are used, the list or property list can span several lines without the need for line continuation characters. If braces are used to create nested lists (or lists containing property lists), care must be taken to include at least one space between one opening brace and the next. Otherwise the two opening braces will be treated as a double-brace quote.

  Note: One CHANGE as a result of this is that { } now represents an empty list, not an empty property list / object.

- Added a **values** function to return a list containing all of the values of a property list, or the values specified by its parameters:

```
put favorites.values("color", "book", "movie") into bestList
```

- Added a simplified form of **repeat with each** using a variable, that allows you to omit the chunk type ("item of") when iterating over the items in a list. Basically, instead of saying "repeat with each item of" you say "repeat with each <varName> of" (the only limitation is that the variable name can't be "char", "character", "word", "item", or "line"):

```
repeat with each name in nameList
    put name
end repeat
```

- Added a global property **umask** for retrieving or setting the posix permissions mask for newly created files.

- Working with files as containers has been enhanced. Putting something into a file in a folder that doesn't exist will now create the folder and all parent folders as needed, as well as the file. Also, **the result** is now set to an error message if the file operation fails.

- Changed the **short name** and **abbreviated name** properties of script file objects slightly.

- Added a **long id** property of objects.

- Added a **handlerNames** property of any object, which returns a list of the names of all of the handlers in that object. This property is read-only.

- Added an **exceptionLocation()** function for use in a **catch** block to obtain a description of the line number and handler name where the current exception was thrown. Additional lines are returned describing each calling location, out to the level of the **try** block itself.

- The value returned in a **catch** statement is now an exception object (a property list with its objectType set to "exception", and "name", "reason", and "asText" properties), rather than just the exception name. The name

and reason properties of this value are the same as the values returned by the **exceptionName()** and **exceptionReason()** functions, which are now obsolete (but will continue to be supported for the foreseeable future). The asText property is set to the same value as the name, so the exception will still appear to be the same text as in previous versions.

The **throw** command can now accept a property list containing "name" and/or "reason" properties, passed as a single parameter, instead of one or two parameters containing the name and reason, to facilitate re-throwing a caught exception. Additional custom properties may also be included.

- Changed **the frontScripts** and **the backScripts** to act as global properties, allowing their contents to be manipulated directly as lists.

- The **begins with**, **ends with**, **is in**, and **contains** operators now look deeply into nested lists, as they should.

- Added the ability to express some fractions as words:

```
wait one half second
put 123 thousandths -- 0.123
```

- List expressions in parentheses and lists of parameters to commands and functions may now include two or more sequential commas to represent empty values.
- Empty parameters to offset() will now be treated as default values.
- An object may store a string representation in its **asText** property now, in addition to implementing an asText function handler.
- Added a **buildNumber()** function which returns the current build number of SenseTalk. Newer builds of the SenseTalk engine will have higher build numbers than older builds.
- Added a **SenseTalkVersion()** function which returns a number indicating the current version of SenseTalk. The **version()** function returns the same number, unless it is overridden by the host application environment. **The long version** returns a longer string containing both the version number and the build number.

Bug Fixes / Tweaks:

- Improved display control in Remote Window should result in fewer visual artifacts.
- Added automatic filtering of the list returned by the EveryImageLocation() function when searching for images with the "Text" search type, to eliminate most cases of "near-duplicate" clustered locations being returned.
- Improved SSH support by using loopback address (127.0.0.1) when appropriate saving many DNS related issues.
- Improved behavior where image captures are always refreshed from Remote Window to prevent capturing VNC artifacts inadvertently.
- Improved support for Eggplant's updating of the CSV files
- Global properties are now reset for each script run.
- Fixed a crashing bug on Mac OS 10.3 if you try to load a script outside of a suite
- Fixed a crashing bug on Mac OS 10.3 when minimizing the remote screen
- Fixed a problem with Screen Error being recorded a split second after actual error occurred
- Fixed a problem with generating a WaitFor command with images that were located in subfolders
- Fixed a problem using RUN on a script outside of a suite
- Fixed a problem using RUN on a script with a .st extension
- Fixed a bug where the Generate Script Menu (and Context Menu) were always disabled
- Fixed a bug which could slow down image captures for the top left corner (or sometimes top edge)
- Fixed a bug when trying to capture a pulsing image with "no searchable pixels"
- Fixed a script bug when trying to coerce an image with "no searchable pixels" to pulsing
- Fixed a problem with deadlocks/crashing when Remote Window was up and you loaded a non-string onto the PasteBoard.
- Fixed a problem with NextKeyDelay not being used between keys.
- Fixed a bug where you couldn't rename a script that hadn't been run (didn't have at least one result).
- Fixed a bug where you couldn't add a script description if it hadn't been run (didn't have at least one result).

- Fixed a bug where scripts that were reformatted on save appeared as dirty.
- Fixed a bug where the suite window would dirty as the result of editing the image description.
- Fixed a problem where certain values would evaluate as dates when queried with the 'is a date' operator but give an 'invalid date' error if you attempted to convert them to a different format with the 'convert' command. Also fixed a related problem in which SenseTalk would erroneously try to perform date arithmetic on such values.
- Fixed a bug that would raise an error if the convert command was used with two formats.
- Fixed an issue with creating a property list containing a script property, so it now actually sets the script.
- Cleaned up some issues with setting properties, including the "name" property, in a properties declaration.
- Fixed a nasty bug with the merge() function that would cause a script to hang forever if the merge template contained just the first character of a multi-character opening merge delimiter.
- Fixed a bug with the **sort** command that would cause an error when sorting an empty list.
- Fixed undefined global and universal variables to be empty, not treated like unquoted literals.
- Fixed a problem with 'properties' constructs not always being recognized in script files.
- Added locks to make the SenseTalk compiler (but not the interpreter yet) thread-safe.
- Fixed a crashing bug when running on Panther (Mac OS X version 10.3).
- Fixed a bug that could potentially cause a crash when running a script that began with one or more blank lines followed by a handler definition.
- Fixed a bug that would pass a formal parameter (one declared in the declaration of a handler, or in a 'params' statement) by reference instead of by value when it was passed directly to a nested handler.
- Fixed the initial handler of a named script to have the name of the script. In some cases it was being given the generic name "initialize". The default name for the initial handler of an anonymous object (an object without a name) is now "<initialHandler>" (Note: **This is a CHANGE from previously documented behavior**).
- Fixed a problem with setting **the folder** or **the directory** property to a path beginning with a tilde (~) to indicate a user's home directory. This should now work properly.
- Fixed a bug with **answer file of type** if more than one type was given, which could result in a variety of problems.
- Fixed a small problem with the **copy** and **move** commands, which would not always create the destination directory as they should.
- Fixed a bug with accessing a chunk of a nonexistent list item. An out-of-range item in a list should now be treated as empty in all cases, rather than raising an error.
- Fixed a problem that prevented 'selection' from being used as a variable name.
- Fixed a bug when accessing object properties under certain circumstances.
- Fixed a problem with inserting dynamically-created objects into an object's helpers.
- Fixed a problem with a comma at the end of a parameter list sending an extra empty value.
- Fixed several script formatting problems, including incorrect coloring of block comments, incorrect indenting of a handler following immediately after a multi-line block comment or another handler, and incorrect insertion of indentation at the end of a comment.

Release v1.42 (24-November-2003)

Highlights:

Runs on Mac OS X 10.3 (Panther). Mac OS X versions 10.1 and 10.2 are still supported as well. Numerous minor enhancements and bug fixes.

User Interface:

An **Eggplant Examples** item was added to the Help menu. This menu item takes you to the Community Forum page of the TestPlant website. All Eggplant users are encouraged to use the forums, to access the example scripts found there, and to participate in ongoing discussions and sharing of ideas related to Eggplant.

Logs shown under the **Results** tab in the Suite window now display the time of each event to the millisecond.

Bug Fixes/Tweaks:

The following bugs and minor issues were resolved for this release:

Fixed a bug which would cause a crash when editing or running a script on Panther (Mac OS X 10.3).
Fixed a bug in which the **Add Image** feature in Capture Mode would truncate the name of the image being added following a period in the name.
Fixed a problem with the Run Selection menu item remaining enabled while a script was running.
Updated the items on the Help menu which send email messages to include the license name and serial number.
Fixed a bug where Eggplant didn't always honor VNC Encoding preferences as set from the Preferences pane.
Fixed a bug in which commands inserted in a script from Capture mode or from the Insert pulldown menu would be inserted at the wrong location (skipping lines) in an indented portion of a script.
Fixed a problem with colorization of a newly-inserted command in a script.
Fixed a bug where clicking the Cancel button on the Establish Connection panel could sometimes cause Eggplant to lock up.

Release v1.41 (23-October-2003)

Highlights:

Significant changes to some timing parameters, for improved behavior working with a variety of remote platforms.

Numerous minor enhancements and bug fixes.

Important Changes (Compatibility Issues):

Some changes made in Eggplant's timing parameters may impact existing scripts. The most significant change involves the way the **RemoteWorkInterval** works. The purpose of the RemoteWorkInterval is to allow some time for the remote system to process one event before the next mouse or keyboard action is sent to it. Previously, this interval was a fixed delay whose behavior was equivalent to including a **wait** command before each remote action command. Beginning with this release, the RemoteWorkInterval is treated as a minimum elapsed time between one event and the next. The difference is subtle but significant in some cases. For example, take the following script excerpt:

```
Click "AnImage"
Wait 0.5
TypeText "Some text"
```

In previous versions of Eggplant, with a default RemoteWorkInterval of 0.3 seconds, the behavior of this example would be to click the image, wait 0.3 seconds due to the remote work interval, then wait an additional 0.5 seconds for the wait command (for a total delay in this case of 0.8 seconds) before typing the text. The new behavior, with a default RemoteWorkInterval of 0.7 seconds, is to click the image, wait 0.5 seconds, then wait only an additional 0.2 seconds (bringing the total delay up to the desired minimum of 0.7 seconds) before typing the text. In this case, the wait command could be removed without changing the overall timing.

The net result is to shorten the delay before some operations and lengthen the delay before others, depending on the actual processing that is being done in the script between actions on the remote system. If it is important for your script to run as quickly as possible, you may set the RemoteWorkInterval to a smaller value, but it is recommended that you begin with the default value, and reduce it only as needed, as shorter delays may not allow sufficient time for the SUT to process an event and be ready to process the next on some systems.

Timing Parameters:

Several changes were made in this release to fine-tune the way Eggplant interacts with the remote system. These changes should result in a better out-of-the-box experience for Eggplant users when controlling a

variety of different systems under test. The Eggplant timing parameters are needed in order to slow down the rate at which events are sent to the remote system, since otherwise Eggplant would drive most SUTs faster than they are able to respond.

As described above, the most significant timing change is in the way the **RemoteWorkInterval** is handled. In certain situations (such as scripts that time the performance of some process on the SUT) it may be desirable to use the old approach of a fixed additional delay between remote actions rather than the new "minimum elapsed time" approach. This can be accomplished by setting the **RemoteWorkInterval** to 0, and setting the new **RemoteWorkDelay** option to the desired delay amount (such as 0.3 for the old behavior).

The new **MouseMoveSpeed** and **MouseDragSpeed** options replace the former **MouseMoveSegments** setting. The old MouseMoveSegments specified the number of steps Eggplant would take in moving the remote mouse from one point to another. Because the distance being moved might vary widely, this was an awkward and imprecise method of controlling the manner in which the mouse was moved. The new MouseMoveSpeed specifies the distance (in pixels) that the mouse will be moved in each step, with the number of steps depending on the total distance being moved.

A separate option, MouseDragSpeed is used to control the mouse speed during drag operations, since experience has shown that many SUTs require slower mouse movements during a drag in order to respond properly. Setting either speed to zero will cause the mouse to move to its destination in a single step. For backward compatibility, if the speed is set to zero the old MouseMoveSegments may be set to control the movement as it did in the past.

User Interface:

Preferences Window:

The Run Options pane of the Preferences window has been changed to reflect the new timing defaults. MouseMoveSegments has been removed from the panel and replaced by the new MouseMoveSpeed and MouseDragSpeed. In addition, a **Restore Defaults** button has been added which resets all of the Run Options to their default settings.

Suite Window:

Items in the Images and Results tabs of the Suite window that were selected by the user now remain selected following various actions which in earlier versions would cause them to be deselected. These actions included anything which would change the contents of these tabs, including running a script or capturing a new image.

Scripting

- The **getOption()** function has been enhanced to accept a variety of different parameters. Passing the name of a single option as the sole parameter still returns the value of that option:

  ```
  put getOption("MouseDragSpeed") -- 10
  ```

  Passing multiple option names (or a list of options) will return a property list containing all of the requested options:

  ```
  put getOption("MouseDragSpeed", "MouseMoveDelay") -- (MouseDragSpeed:10,
  MouseMoveDelay:0.01)
  ```

  Calling the getOption() function without supplying any option names will return a property list containing the full set of options and their current settings.

- A new **getOptions()** function (note the 's') has been added. It is identical to getOption() except that it always returns a property list, even when just a single option name is passed:

  ```
  put getOptions("MouseDragSpeed") -- (MouseDragSpeed:10)
  ```

- The **setOption** command has been similarly enhanced to allow you to supply a property list rather than just a single option and its value. This allows you to set a number of options with a single command:

  ```
  SetOption (MouseDragSpeed:20, MouseMoveDelay:0.05)
  ```

This capability may be especially useful for storing a number of related settings in a variable so they can be switched and restored as a group, as shown in this example:

```
put (MouseDragSpeed:20, MouseMoveDelay:0.05) into specialDragSettings
-- then, possibly sometime later...
put getOptions(keys of specialDragSettings) into originalDragSettings
setOptions specialDragSettings -- set the specialized options
-- do some dragging with the special settings here
setOptions originalDragSettings -- restore the previous settings
```

- For convenience, **setOption** may also be called as **setOptions**.

- Added a **number of occurrences** (or **number of instances**) function to determine the number of times that a value of a particular chunk type appears in some value:

```
put number of occurrences of "a" among the chars of "banana" -- 3
put the number of instances of "be" \
          among the words of "to be or not to be" -- 2
put the number of occurrences of 15 among \
          the items delimited by "-" of "315-15-4152" -- 1
```

  If a specific chunk type is not named, **items** are assumed:

```
put number of occurrences of "a" in "banana" -- 0
put the number of occurrences of 3 in (1,3,5,4,3) -- 2
```

  Comparisons are normally case-insensitive (or follow the setting of the **caseSensitive** property), but **considering case** may be used to change this:

```
put number of occurrences of "a" among the chars of "baNAna" -- 3
put number of occurrences of "a" among the chars of "baNAna" considering
case -- 2
put number of occurrences of "Do" in "Do,re,mi,do" -- 2
put number of occurrences of "Do" in "Do,re,mi,do" considering case -- 1
```

  As a special case, "among the characters of" can be used not only to count occurrences of a single character, but of any sequence of characters:

```
put number of occurrences of "na" among the chars of "banana" -- 2
```

- Added an **is among** operator to simplify testing whether a value appears as one chunk of another value, when you only care that it is present, not where it appears or how many times. The expression "x is among the chunkTypes of y" is equivalent to "(the index number of chunkType x within y) is not zero" or "(the chunkType number of x within y) is not zero". All of the following will display 'true':

```
put "a" is among the characters of "heritage"
put "ace" is among the items of "king,queen,ace"
put "ace" is not among the items of "strength,poise,grace"
put "ACE" is not among the items of "ace,two" considering case
put 7 is among the items of (5,5+1,5+2,5+3)
put "be" is among the words of "to be or not to be"
put "be" is not among the words of "busy as a bee"
```

- Added a **between** operator to simplify testing whether a value is in a given range. The expression "x between a and b" is essentially equivalent to "a <= x and x <= b". All of the following will display 'true':

```
put 2 is between 1 and 3
put 7 is not between -3 and 5
put "ace" comes between "able" and "actor"
put "ACE" does not come between "able" and "actor" considering case
put 5 is between 5 and 10 -- the range is inclusive
put 8 is between 10 and 5 -- the range can be specified in reverse
```

- Enhanced the **throw** command to accept any number of parameters. If no parameters are given, it throws an exception with the name "ThrownException" and reason "no reason given". If more than two parameters are given, the second and subsequent parameters are listed as separate lines in the reason.
- Added **instance** (and **instances)** as synonyms for **occurrence** (and **occurrences**) in the replace command as well as the new number of occurrences function.
- The compiler now requires a "factor" in **delimited by** expressions, rather than a full expression. In the extremely unlikely case of having used a complex expression in the past, it will need to be enclosed in

parentheses to continue to function as before.

- The **remove** command now accepts **frontscripts** and **backscripts** as synonyms for **front** and **back**, just like the **insert** command.

Miscellaneous:

Changed the list of images displayed under the Images tab of the Suite window to not display folders named "CVS", in order to facilitate suite management using the cvs revision tracking tool.

When capturing images, the destination suite (and script) is now shown in the title bar of the save panel.

Parameters may now be passed to scripts called from the command line. This is accomplished by including a '-params' option after all other command line options have been specified, followed by the parameters that are to be passed to the script. The following example passes two parameters, "dataset2" and 19:

```
% /Applications/Eggplant.app/runscript /EggplantSuites/master.suite/Scripts/
   test1.script -params dataset2 19
```

Bug Fixes/Tweaks:

The following bugs and minor issues were resolved for this release:

Fixed Undo and Redo to work properly in the script editor.
Fixed a performance problem with suites containing many scripts, which was particularly noticeable with suites located across a network.
Fixed a problem with deleting of images, scripts, etc. which were located across a network on a volume without Trash.
Fixed a problem in which response to events in Live Mode could get increasingly out of sync with current actions.
Fixed a small bug with display of the RemoteWindow cursor over the scrollbars and toolbar.
Fixed a problem with generating commands in which the captured image could be saved in the wrong suite if a Suite window for a different suite was clicked on before generating the command.
Fixed a bug where the RemoteWindow cursor stayed off when connection was closed.
Errors generated by the **Throw** command are now logged in purple and included in the exception count for a script run.
Fixed a new 1.4 crash that was particularly likely on multi-processor machines.

Fixed a bug that would occasionally cause Eggplant to crash when moving images between folders.
Fixed a persistent bug that could still crash Eggplant if closing a script that was aborted immediately before being closed.
Fixed a problem with saving Schedules when scripts were dragged in.

SenseTalk:
Fixed the **pass ... and continue** command which was severely broken to the point where it could cause a crash.
Fixed a new bug with certain uses of **me** or **my** that could manifest in various ways, including a crash.
Fixed a bug with **read ... until** that could cause it to stop reading and return data before the desired terminating character was read.
Fixed a relatively recent problem with reading from standard **input** so that a script run from the command line can receive input whenever return is pressed.
Fixed a small problem with the **send** command which could cause script setup and cleanup (such as closing all open files) to be done at inappropriate times.
The compiler now detects and reports null characters in the middle of a script as errors, instead of quietly treating them as the end of the script.

Release v1.4 (15-September-2003)

Highlights:

Brand New TEXT Search Type that deals with font smoothed text.
More connectivity features including the ability to Send Email, Read and Post Data to URLs and initiate phone calls using iVeena
Enhanced Results and Statistics - including the ability to customize results from script control
Faster and smoother display of the Remote Screen
Improved Image searching, including the ability to change an image's searching tolerance.
Ability to change image attributes on-the-fly as they are used in a script.
New OSXvnc 1.3 (with Cursor support)
Numerous Bug Fixes and Smaller Enhancements
and...
A BETA release of capability for recording Quicktime Movies either under script control or in Live mode (See Below)

Important Changes (Compatibility Issues):

OSXvnc 1.3 is the supported VNC for use with Eggplant we recommend that all Mac OS X systems be upgraded to the latest OSXvnc.

LogError command now causes a script to be reported as a Failure at the end of the script execution. This allows you to detect and fail a script but still run to the end, you can obtain the previous functionality of **LogError** by using the new **LogWarning** command.

EveryImageLocation() now returns an empty list if the image is not found. Previously it threw an exception - if you rely on getting at least one image please check the return value of **EveryImageLocation()**.

Movies

We have added a BETA release for Recording QuickTime Movies.

A few notes about Movies:

Movies will record everything that occurs on the Remote Screen. If you are looking to demonstrate something using a script you will probably want to slow things down a little by using the **setOption** commands or changing the Preferences for Script Defaults. In particular **MouseMoveSegment, MouseMoveDelay,** and **NextKeyDelay** should all be increased.

We don't offer much in the way of editing control for this first Beta release. We recommend just recording movie segments and then using an editing tool like Final Cut Pro if you have it or Quicktime Pro if you just want some basic control.

You can only start a movie when you already have a valid connection and when your VNC connection changes the recording will end.

You can record movies under script control using...
startMovie FILENAME - begins a movie record of the SUT, if not an absolute path, filename is stored relative to the current results directory.
stopMovie - stops the current movie

Or you can start and stop live movies using the Menu or the Remote Window Toolbar. These movies will continue to record if you run a script (as long as the Remote Connection stays the same).

At this time movies recording in one mode prevent movies in the other mode from starting.

Image Searching:

We have added two new Image Search Types, "Text" and "Text&Pulsing". These types are particularly designed for use with Text which has been Font Smoothed. By default Mac OS X's Aqua interface can be set with a particular Font Smoothing mode and for fonts larger than a user assigned threshold it will anti-alias the text with the background. These new Image Search Types should allow you to find text even if the current SUT is using a different smoothing mode or if the text is anti-aliased differently, because the text landed slightly differently on the screen or on a slightly different background.

Caution: The **EveryImageLocation()** can occasionally find multiple "Text" type images in virtually the same point on the screen. For clicking this isn't a large concern but it can cause the number of returned values of an **EveryImageLocation()** command to seem high.

Image searches in Eggplant have some built in tolerance, previously determined by the **standardImageTolerance** and **preciseImageTolerance** options. The tolerance of an image can now be set on a per image basis using a new element in the Image Info drawer.

We have also made some improvements to the Pulsing Image Search Type. We found after much usage that if you captured the extreme dark end of a an animation that you could fail to find that pulsing image approximately 1.5% of the time.

Scripting Enhancements:

We have added an **ImageInfo(**ImageName) function that will return a property list with various attributes about an Image. Properties include:

ImageName - Name of the Image
HotSpot - A point which is the images hot spot (relative to the upper left corner of the image)
ImageSize - The image's size (width, height)
SearchType - A string describing what kind of Searching will be used to find the image on the remote screen **ImageTolerance** - The tolerance that will be used when searching for the image
Description - The image's description as entered in the Image Info Drawer
CaptureLocation - The original point on screen that the image was captured

Temporary Images using a Property List :

You can now override an Image's stored behavior. Any command that accepts an Image Name can now optionally be given a property list, like the one return by the **ImageInfo()** function or one you build yourself. The command will use the image specified by the **ImageName** parameter. But, if supplied Eggplant will use the property passed in for **SearchType, HotSpot,** or **Tolerance** to override the default values found in the image. So for example the following command will click in the upper left corner of "MyImage" on screen regardless of what that images regular Hot Spot is:

Click (ImageName:"MyImage", HotSpot: (0,0))

We have added a **ScriptResults(**ScriptName**)** function to get information about a scripts results while running a script. This allows you take take some further action such as post results to a website (perhaps using the **merge** command) or do something based on a scripts history of results. If you do not specify ScriptName it will default to the script that is currently generating results. The **ScriptResults()** function returns a List of each run of that script. Each run has a Results Property List, with the following properties:

LogFile - The file where the log was stored
Errors - A count of errors logged in the script,
Warnings - A count of the warnings logged in the script,
Exceptions - A count of the Eggplant exceptions thrown(caught or uncaught)
Duration - The length of time (in seconds) the script ran (or has been running)
RunDate - The Date and Time the run was started
Status - The status of the run **Success, Failure,** or **Running**
ReturnValue - If the Script had a RETURN statement, the returned value.

As an example this command would print the current log file name:

```
put "LogFile" of last item of ScriptResults()
```

We have created a new command which allows you to run a script but generate results as if that script had been run independently. Eggplant will store the scripts Results in the file system as if you had clicked the "Run Script" button for that script or specified it in the Scheduler tab.

RunWithNewResults. ScriptName, [parameters]
This command requires a script name as the first argument, any other arguments will passed to the new script. The command returns a Results Property List just like those returned in the **ScriptResults()** function. The results will also be stored in the Results directory for that Script. This command will NOT raise an exception or fail the script if the script you called failed. Although you may do so yourself after examining the Status of the return value.

SetLogging On | **Off**
This command allows you to temporarily turn logging off and then back on for the results which are currently being generated. This is valuable if you do not wish to log certain instructions (perhaps entering a password) or if you aren't interested in the detail for certain lower level scripts.

You can check the current state of logging using the new **LoggingIsOn()** function to check whether logging enabled.

Changed **LogError** command to now causes a script to be reported as a Failure at the end of the script execution. This allows you to detect and fail a script but still continue running. Errors are tallied by the script and will show up in **RED** in the Log View and in the Script's output display.

Added **LogWarning** to take the place of the old **LogError** command. In addition, warnings are tallied by the script and will show up in **ORANGE** in the Log View and in the Script's output display.

Eggplant can now send Email. To specify your SMTP server and settings use the Eggplant->Preferences->Mail panel.
The **SendMail** command takes a property list with the following special properties. Additional properties will be treated as regular e-mail headers:

To - An e-mail address to send to (or multiples separated by commas)
Subject - The emails subject
Body (or Message) - The text of the Email
Attachment - A filename or list of filenames to attach to the email.
ReplyTo - To set the Emails Reply
CC - E-mail carbon copy to this adresss

So the following command could send a bug report with an attached file
SendMail (to:"bugreports@testplant.com", subject:"Bug Report", body:"I had Eggplant crash today. Log Attached",Attachment:"~/ScreenShot.jpg"**)**

A new function, **ConnectionInfo()** returns a property list with information about the current connection property list. This list has the same properties that can be passed with the **connect** command plus it returns...

Status - Connected or Not Connected

For **SetOption** and **GetOption()** we have reworded the options named **preciseColorVariance** and **tolerantColorVariance** with **preciseImageTolerance** and **standardImageTolerance**, although the previous names will continue to be supported.

EveryImageLocation() now returns an empty list instead of raising if no images are found.

CaptureScreen now works with absolute paths, also allows for a third **YES** argument to have Eggplant make the capture unique (by adding a number to it).

User Interface:

Remote Window:

Remote Window updates are now performed in a dedicated thread. This results in smoother display performance, particularly while watching a script run in the Remote Window. This also allows for a shorter timeout when trying to establish VNC connection and allows the user to Cancel while establishing a

connection to a VNC server.

Eggplant now works with VNC servers that do local cursor rendering (OSXvnc 1.3, RealVNC4.0, TightVNC 1.2.9). This allows the cursor in the Remote Window to be much faster and also allows it to change with respect to what is occurring on the Remote machine.

Added a "None" cursor preference which works well for VNC servers which allow for local cursor handling

The Remote window now comes up visually scrolled to the top left, which is generally better than the bottom-left for most situations.

Users can now refresh the Remote Window by using either the Refresh Screen Toolbar Item or the Refresh Screen Menu Command (Option-Cmd-L)

Suite:

Added Milliseconds to Log Files - thus preventing overlaps of results on scripts which execute very quickly

Errors, Warnings and Exceptions are now tallied on the Results Section (and recorded in the .CSV files)
Users can now drag scripts into the schedules tab and drop them at a particular point in schedule
Dramatically improved performance opening Suites with lots of Files

Script:

Errors, Warnings and Exceptions are now colorized in the Log Area at the bottom of the Script Window

Eggplant Now remembers which Sub Directory you have been saving images to and defaults you to that location

---

OSXvnc 1.3:

A new version of OSXvnc, version 1.3, is included with this release of Eggplant. Eggplant users are strongly encouraged to use this latest release of OSXvnc. This new version includes the following features/fixes:

OSXvnc GUI has been broken out to allow for more options and to match Preference Panes more closely.
Added GUI controls to set for System Startup Up (Including Authentication)
WARNING: OSXvnc run at StartUp Does NOT have pasteboard synchronization
Added Mac OS X Cursor Support
Added RichCursorSupport (RFB 3.7 + Tight)
Added CursorPositionUpdate Support (Tight)
Supports Dynamic Screen Sizing (Clients need no longer reconnect after a screen size change - RFB 3.7)

Fixed crashing bug where certain unknown keys would crash OSXvnc
Fixed problem with Shift-Tab (sent from OS X) being ignored as an End Of Medium character, when it didn't crash the whole server
Fixed problem with OSXvnc causing machine interruptions when no clients connected by disabling poll for ScreenUpdates (Bug-770661)
Now Attempting to pass the CGCharCode, reports that this improves stability (no crashing during key events like Return).
Added Bundle Loading to hopefully allow OSXvnc to run on 10.0 and use 10.2 and 10.3 specific enhancements
Fixed an occasional problem sending a rectangle without having accounted for it
Fixed a rare bug where it was occasionally possible for clients to miss a copy event which occurred on the server.
Fixed so that scroll wheel events don't leak by adding NSAutoreleasePool (Bug-760383)
Fixed some pasteboard problems with non text put to pasteboard
Minor Performance improvements for delay between thread notifications

---

SenseTalk™

- Fixed the **insert** command so that it never removes or replaces anything: the **into** option has been altered to be the same as **after** when the destination is a reference directly to a container. When the destination is an item within a list, **into** will treat that item as a list (whether or not it already was one) and insert the source after that nested list. The behavior of the **before** and **after** options is unchanged.

- Substantially improved the performance of the **replace** command when performing many replacements on a large text string.

- Fixed a compiler bug involving multi-line block comments enclosed in (* *). Such comments at the beginning of a handler (and possibly elsewhere) caused the SenseTalk compiler to complain about a variety of syntax errors in scripts that should have been perfectly fine. The bug only occurred when the block comment spanned more than one line, and only in certain situations.

- Fixed a problem with function calls similar to "(sqrt of 9)" -- when they were enclosed in parentheses the compiler would treat this as a property reference and complain that '9' was not an object identifier.

- Added first implementation of **url** containers. The new keyword **url** followed by an expression which evaluates to a valid URL, can be used as an expression representing the contents of that resource.

```
put url "http://www.apple.com" into appleHomePageSource
```

  When the URL represents a resource that can be written to (such as a "file:" URL) you can change its contents just like any other container.

```
put appleHomePageSource into url "file://localhost/appleCopy.html"
```

- Added a **makeURL()** function which takes a property list and returns a URL string by combining any of the properties that are included: scheme, host (or baseURL or url if host is not given), port, user (ftp), password (ftp), parameters (ftp), path (http), query (http), and fragment. Spaces and special characters in each value are properly encoded as needed.

```
put makeURL(host:"www.apple.com", path:"developer") into appleDevURL
```

- Added an **extractURL()** function which takes a URL string and returns a property list of the components of that URL: scheme, host, port, user, password, path, query, parameters, and fragment. Encoded spaces and special characters in each value are properly decoded as needed.

```
put extractURL(appleDevURL) into urlPropertyList

get extractURL("http://xyz.net:8100/get/this?a=1&b=2#mark")
put it -- (fragment:"mark",host:"xyz.net",path:"/get/this",
          port:"8100",query:(a:"1",b:"2"),scheme:"http")
```

- Added a **split** command which takes a text string and turns it into a list or property list (object) based on the contents of the string. If the source text is a container (a variable), that variable becomes a list or a property list. Otherwise the resulting list or property list is returned in the variable `it`.

```
split "apple;banana;orange;cherry" by ";"
put it -- (apple,banana,orange,cherry)
put "x=7;y=2" into coordinate
split coordinate by ";" and "="
put coordinate -- (x:7, y:2)
```

  If the same key appears more than once, the resulting associated value will be a list containing all of the values.

```
split "a/2,b/15,a/9" by "," and "/"
put it -- (a:(2,9), b:15)
```

- Added a **combine** command which takes a list or property list (object) and combines its items or elements into a text string. If the source is a container (a variable), that variable's contents are converted into a text string. Otherwise the resulting string is returned in the variable `it`.

```
combine (1,2,3) using ":"
```

```
put it -- 1:2:3
```

Values can be quoted using the **with quotes** option:

```
combine (1,2,3) using ":" with quotes
put it -- "1":"2":"3"
```

Alternative quote characters can also be provided:

```
combine (1,2,3) using ":" with quotes ("<<",">>")
put it -- <<1>>:<<2>>:<<3>>
```

For property lists, provide two separators to preserve both the keys and values:

```
combine (a:12,b:5) using ":" and "/"
put it -- a/12:b/5
```

Or use a single separator for the values only, discarding the keys:

```
combine (a:12,b:5) using ":" with quotes
put it -- "12":"5"
```

- Added a **URLEncode()** function which returns a "URL safe" version of a string by substituting "+" for any spaces, and encoding other special characters using %xx where xx is two hexadecimal digits representing the character's value.

```
get urlEncode of "Good Times & Bad"
put it -- "Good+Times+%26+Bad"
```

- Added a **URLDecode()** function which returns a user-readable version of a string that was in a URL format by substituting spaces for any "+" characters, and decoding other special characters from the %xx format to their usual appearance.

```
get urlDecode of \
 "http%3A%2F%2Fwww.google.com%2Fsearch%3Fq%3Dice%2Bcream%26ie%3DUTF-8")
put it -- "http://www.google.com/search?q=ice+cream&ie=UTF-8"
```

- Added a **post** command to post data to a URL. If the data to be posted is a property list, it will be properly encoded before being posted.

```
post "search?q=ice+cream&ie=UTF-8" to url "http://www.google.com"
post (name:"Ginger", pwd:"hinky2") to url "http://login.snuffle.com"
```

- Added **plistPrefix**, **plistEntrySeparator**, **plistKeySeparator**, **plistSuffix**, and **emptyPlistAsText** global properties. The initial default values for these are "**(**", "**,**", "**:**", "**)**", and "**(:)**" respectively. These properties now control the format in which property lists (objects) are displayed by default, if they don't handle the **asText** function message to return a display string for themselves.

  If the **plistKeySeparator** is set to empty, converting a property list to text will result in a list of the values of that property list only, without any keys.

  IMPORTANT NOTE: Due to these changes, property lists converted to text will appear in A DIFFERENT FORMAT than they did in previous versions. This should not cause problems in most cases, since both the old and new text formats can be converted back into property lists using the **value()** function. However, if you have scripts which display property lists you will notice a change. If you would prefer to use the old format, set both the **plistEntrySeparator** and the **plistKeySeparator** to empty.

- Added the ability to easily create or refer to an empty property list, using either a colon enclosed in parentheses **(:)** or the phrase **empty object**:

```
put (:) into myObj
put 12.88 into property "cost" of myObj
if sut1 is an empty object then put "black" into sut1's color
```

- Made the **ask** and **answer** commands more flexible regarding the order in which their options are specified.

- The **repeat** control structure now accepts a **step** option to step by an increment other than 1. In addition, the starting and ending values, and the step increment, may now be non-integer values.

- Added a **repeatIndex()** function that returns the current iteration count of the innermost executing repeat

loop.

- Added an **rtfToText()** function that takes rich text in RTF format and returns its contents as plain text. This can be useful for reading a file in RTF format and working with its contents.

- It used to be possible (although this was undocumented and unsupported) to use dots within a property name to specify a "property path" -- that is, for descending a hierarchy of properties of properties, where the dots separated the individual property names. By removing that capability, it is now possible to use property names containing periods. The old behavior can be restored by setting the new global property **treatDotsAsPropertyPaths** to true, if desired.

- Lists and property lists now allow a trailing comma at the end, such as (1,2,3,) or (a:31, b:25,). This simplifies the process of computer-generating a list, if desired.

- “Curly” double quotes may now be used around strings instead of "straight" double quotes. They must be paired with the left curly quote before the string and the right curly quote following it:
```
replace every occurrence of “ he ” in manual with “ she ”
```

- Added several new predefined variables: **cr** as a synonym for **creturn** (a carriage return character, ASCII 13); **lf** as a synonym for **linefeed** (ASCII 10); **crlf** for the two-character combination of the a carriage return followed by a linefeed; **slash** for the "/" character; **backslash** for the "\" character; and **null** for a null character (ASCII 0).

- The **ask file** command may also be specified as **ask folder** or **ask directory**. Since it doesn't actually create anything, but only returns the new name and path selected by the user, there is no difference between the three commands. The new versions seem more natural in a script which will be creating a folder, though.

- As a convenience, a property list may now be passed as the only parameter to a function using only a single set of parentheses. For example:
```
put selectOptions(color:"red", size:42) into selection
```
  is the same as:
```
put selectOptions((color:"red", size:42)) into selection
```

- Added **frontScripts()** and **backScripts()** functions to return lists of the objects that are currently in use as front and back scripts.

- Added support for accessing the **helpers** property of an object to get a list of that object's helpers, or to change its helpers.
```
insert "usefulLibrary.script" after my helpers
```
  You can also access an object's **early helpers** property to get or change the list of objects that can handle messages before the object itself gets a chance to:
```
insert "bossy.script" before my early helpers
```

- Improved memory management. Temporary objects are now released on each iteration through a repeat loop. This can be turned off by setting the new **releasePoolsForEachRepeat** global property to false. There is also a **releasePoolsForEachHandler** global property that governs memory management at the handler level. Unless you have an exceptional situation, however, it is recommended that you leave both of these set to true.
- Updated the internal object cacheing scheme for better performance -- disk-based objects no longer check for newly-saved changes on every call, but only once every script run.
- Removed "file type" as a synonym for the **entry type** property of a file or folder.
- Fixed ability to access nested properties beginning with "my", such as "my dog's name".
- Fixed a bug in the **replace** command which could cause a crash in certain circumstances.
- Fixed a bug which would cause a handler to not be found when it was immediately preceded by a multi-line block comment.
- Added the exceptionName() and exceptionReason() functions to the list of functions which can be called using "the", so you can now call them as **the exceptionName** and **the exceptionReason**.
- Fixed a bug with the **ends with** operator which would throw an exception in some cases.

- Fixed an obscure (but ugly) bug with property lists.

Bug Fixes/Tweaks:

The following bugs and interface issues were resolved for this release:

Fixed a problem where Test Images would be "found" in an old location even if they were moved
Fixed a problem where trying to delete a folder with images would put the Suite in an unusable state, requiring a Force Quit
Fixed a rare problem with timing syncing pasteboards with OSXvnc and a problem when non-strings were put on the pasteboard
Fixed a 10.1 problem with the General Prefernces and Remote Screen Preferences
Fixed a crashing bug when closing a script after beginning execution on it.
Fixed a problem with Absolute Paths in **CaptureScreen** command.
Fixed a bug where the Image Save Preview could sometimes come up all squished (or hidden entirely)
Improved detection of Pulsing Buttons (sometimes the Image Capture failed to notice them pulsing)
In the Image Save Dialogue you are now able to work up the directory tree to reselect the Suite if you somehow got away from it
Fixed a bug where the script output display (bottom of window) would occasional get wiped out JUST as the script finished
Fixed a bug where the total time run of a script stopped accumulating when the suite was reopened
Fixed a bug where the SuiteStatistics.csv sometimes reported invalid data in the Average Time field.
Fixed a bug where setRemoteClipboard would not work with an empty string
Fixed a bug where hot spot center was calculated slightly differently in Image Info Panel from the original capture
Fixed a command line problem where it would try to bring up the Connection Panel
Fixed a command line problem when running with Script Animation Turned on
Fixed a command line problem when using a connect command with VISIBLE set to YES
Fixed a problem where aborting a script would cause subsequent Generate commands, Schedules and pulsing detection to fail.
Fixed a bug where causing the remote vnc server to shutdown could crash Eggplant

---

Release v1.3 (07-May-2003)

Highlights of this release include a new user preferences panel, script syntax colorization, automatic identification of the appropriate image type and numerous smaller enhancements and new features. The documentation has also been significantly revised and updated. Users upgrading from earlier versions should be sure to read these release notes to learn about the latest changes and additions to Eggplant.

Important Changes to be Aware Of

The default tolerance for precise image searching has been changed from 0 (no tolerance) to 1 (a very small tolerance) to support differences in video card rounding. This should eliminate occasional problems that have been reported with finding images captured from one SUT when running against a different SUT that is configured identically but uses a different video display card. As always, you can override the default tolerance by using the SetOption command (or using the new Preferences panel) to set the PreciseColorVariance to the value you want.

Another change to be aware of is that some little-used forms of **if ... then ... else** statement are no longer valid. This change eliminates ambiguity in the script syntax and prevents some unexpected results. Most users will not be affected by this change, but any existing scripts that use the disallowed forms will now generate an error, and will need to be changed slightly in order to work again. The needed changes are very easy to make, and are described below in the section on Scripting.

The **copy file, copy folder, move file**, and **move folder** commands have also been changed somewhat, to make their behavior more natural and consistent. See the full description in the Scripting section of the notes for this release to understand how this may impact your scripts.

User Preferences

The new user preferences panel includes 5 different panes to allow you to set a wide range of preference options. Many of these options control features newly introduced in this release. Briefly, the panes and the settings they provide are:

General Preferences
Controls whether Eggplant should automatically reopen any scripts and suites which were open the last time you quit Eggplant. Allows you to specify that Eggplant should restore a remote connection if a connection was still open the last time you quit Eggplant. Allows you to set the default location for new suites.

Remote Window Preferences
Controls whether the mouse cursor in live mode is the local cursor (the usual arrow) or a small box. Lets you specify whether the mouse scroll wheel (for mice that have them) will scroll the local window, or whether those events are passed through to the remote system. Allows you to change the key used for quick toggling between capture mode and live mode -- pressing and releasing the selected key without pressing any other keys will toggle the mode. Controls whether or not the capture rectangle in capture mode displays a border.

Script Editor Preferences
General tab: Allows you to specify that scripts should be saved automatically whenever they are run. The default font used when creating a new script can be set, as well as the font to be used in the script log display at the bottom of each script window.

Indentation tab: Automatic indentation of scripts can be turned on or off, and you can control the number of tab or space characters that will be used to indent both control structures and continuation lines. You can also specify whether the tab key will insert a tab into the script, trigger script reformatting, or both. An example script at the bottom of this pane shows how a script will look using the settings you have chosen.

Colorization tab: Controls whether Eggplant's automatic script colorization feature is enabled or not. If colorization is turned on, you may specify whether colors in the current script line should be updated continuously while you type. For each of several different script element types, you can specify whether or not they should be colored or made bold. For those that are colored, you can set the color by clicking the border of the color well. An example script at the bottom of this pane shows how a script will look using the settings you have chosen.

Run Options Preferences
The Run Options preferences section allows you to set default values for all of the options available through the SetOption command in a script.

Mouse tab: Allows you to set default values for the MouseClickDelay, MouseDoubleClickDelay, MouseMoveDelay, MouseMoveSegments, MouseMoveMode, and ShouldRepositionMouse options.

Keyboard tab: Allows you to set default values for the KeyDownDelay, NextKeyDelay, and SendShiftForCaps options.

Screen tab: Allows you to set default values for the ImageSearchTime or the related FindImageRetryDelayTime and FindImageRetryCount options, and the PreciseColorVariance and TolerantColorVariance options.

System tab: Allows you to set default values for the RemoteWorkInterval, VNCMaxTurnaroundDelay, and ForceScreenRefresh options.

VNC Preferences
Lets you specify which VNC encodings may be used in communication with the remote system.

Script formatting, when enabled, occurs automatically whenever the return key is pressed, when the comment/uncomment controls on the toolbar are used, when text is pasted into a script, and when the script is saved.

Program/User Interface

Updated Help Menu (and Documentation)

We have added e-mail feedback options to the Help menu for your convenience. Feel free to file bugs, request features, or ask questions using these menu items.

We have modified our Help menu to access the four primary Eggplant Documents

- Getting Started - A simple guide to installing and configuring Eggplant, VNC, and your network between the two, plus a detailed tutorial on creating a first script.
- Using Eggplant - Introduction to Eggplant and basic concepts for using Eggplant in a testing environment.
- Eggplant Reference - A detailed reference for each of Eggplant's Menus, Screens, Commands and Functions.
- SenseTalk Reference - A detailed reference of the SenseTalk scripting language and all of its elements.

Remote Screen Window

When capturing images, Eggplant now automatically chooses an initial color searching type appropriate for each image. For most images, "Tolerant" will be selected, which works well in most situations. If the image has low contrast, however, Eggplant will choose "Precise" color searching to avoid false matches. Finally, if the captured image includes all or part of a pulsing button, "Pulsing" color matching will be selected. In any case, you can still set the color matching type yourself if Eggplant does not select the type you want.

The Remote Screen window now sports a metallic look. This helps to set off the remote system's interface from that of the local machine, making it a little easier to keep track of which system you are working with. It also looks rather good (in our opinion)! The resizing behavior of the window has also been improved.

A **Refresh Screen** button is now available on the Remote Screen toolbar. Clicking this button will refresh Eggplant's view of the remote system, cleaning up any stray artifacts that may have shown up. To add this button to your toolbar, click the Customize icon at the right end of your toolbar, or choose Customize Toolbar from the Window menu, then drag the new button onto your toolbar and click "Done".

The capture rectangle shown in Capture Mode now displays with a gray border, making it easier to see when working with dark backgrounds. The border may be turned off in the Remote Window preferences.

The red "plus" symbol representing the hot spot now changes color from red to black whenever it is over a point that is primarily reddish in color.

When generating the WaitFor command, the maximum wait time field is now incorporated into the save panel, rather than being displayed in a separate panel, simplifying its use.

Script Editor

The Script Editor now colorizes scripts based on syntax: comments are green, quoted text is red, and so forth. The colors used, as well as whether or not certain script elements should be made bold, can be changed in the Script Editor section of the Preferences panel. You can also control whether script colorization occurs as you type or only when you press Return or Tab, or turn it off completely if you wish. Script indentation can be controlled independent of colorization, including the amount by which control structures and continuation lines are indented (see the Script Editor Preferences).

A **Find** panel has been added (**Edit** -> **Find** -> **Find...** from the menu, or Command-F) which supports find and replace of whole words or arbitrary text within a script. In conjunction with this, the following commands are also available on the **Edit** -> **Find** menu: **Find Next** (Command-G), **Find Previous** (Command-D) for finding the next or previous occurrences of the current find target; **Use Selection for Find** (Command-E) for entering the current selection as the new find target; and **Scroll to Selection** (Command-J) to scroll the current selection in the script editor into view.

Dragging a different script into a Script Editor window (from the Script tab of a Suite window, or from the title bar of another Script Editor window) will now insert either the simple name of that script (as it did before) or a **Run** command to run that script. A Run command is used if the script name contains spaces or other characters not allowed in a command name, or if the script does not belong to the current suite.

When nothing is selected in the script editor, the **Comment** toolbar button now becomes an **Add Comment** button, which inserts an empty comment in your script. Previously this button was disabled when no text was

selected.

Script animation has been enhanced to allow other windows to be brought to the front during script execution without being obscured by the window of the executing script.

Running scripts can now be aborted more effectively. Previously, they could not be aborted during Wait and WaitFor operations or during image finds.

Suite Window

Scripts can now be copied from one suite into another by simply dragging the script to the Scripts tab of the destination suite.
When scripts, images, or logs are deleted in Eggplant they are actually moved to the Trash Can.

Helper suites can now be opened by double-clicking them in the Helpers tab.

Scripting

Changed **CaptureScreen** command to synchronize the display screen immediately before performing the capture. Previously it did not which could result in the capture being out of date.

Fixed an inconsistency where the **Return** constant actually sent the code for the Enter key when used with a **TypeText** command, this has been fixed and a new constant **Enter** has been introduced for use with **TypeText.**

Added an **EveryImageLocation()** function that returns a list of all locations on the remote screen where a given image is currently found (the locations returned are those of the hot spot relative to each image located). The following example would click on each occurrence of the image "RadioButton" currently showing on the screen:

```
put everyImageLocation("RadioButton") into buttonLocations
repeat with buttonLoc = each item of buttonLocations
          click buttonLoc
end repeat
```

Added **ImageSize()** and **ImageHotSpot()** functions that return the size (width,height) and hot spot location (horizontal,vertical offsets from top-left corner of image) for a specified image name. You might want to use these values, for example, in conjunction with the **EveryImageLocations()** function to calculate the actual screen coordinates of each instance of the image, rather than the hot spots.

```
put imageHotSpot("RadioButton") into hotSpotOffset
put imageSize("RadioButton") into buttonSize
put everyImageLocations("RadioButton") into buttonLocations
repeat with buttonLoc = each item of buttonLocations
          put "Button rectangle = " & (buttonLoc - hotSpotOffset) & \
                    "," & (buttonLoc - hotSpotOffset + buttonSize)
end repeat
```

Added **MouseButtonDown** and **MouseButtonUp** commands. These are very low-level (basic) commands that each take a single parameter that specifies a button number from 1 to 8, and send a message to the SUT to press or release that mouse button. These commands are the only way to send mouse button events other than left and right clicks, such as clicks on the middle button, or pressing and holding the mouse button for a period of time.

Added a **sort** command for sorting lists or chunks of text.

```
sort myList
```

Following this command, myList will be sorted in ascending order. This is equivalent to the command:

```
sort ascending myList
```

or:

```
sort the items of myList in ascending order
```

The sort command is very flexible and includes a number of options:

```
sort the items of myList by the last word of each
```

See the SenseTalk documentation for complete details.

Added a **replace** command for replacing text within a container:

```
replace oldWord considering case \
          with correctSpelling in introduction
replace the last "s" in word 3 of sentence with "ed"
replace with "," the first 7 occurrences of ";" in treatise
```

The replace command is very flexible and includes a number of options. See the SenseTalk documentation for complete details.

Added a **keys** function that returns a list containing all of the keys in a property list. The list is returned in sorted order:

```
put the keys of (name:"Granny", age:93) -- (age,name)
```

Added an **openSockets()** function, which returns a list of all sockets currently open as a result of the **open socket...** command.

Added the ability to easily create an empty list, using either an empty pair of parentheses ( ) or the phrase **empty list**:

```
put () into newList -- newList is now a list with no items
put 16 into item 1 of newList -- (16)
put empty list into item 2 of newList -- (16,())
```

Added **answer file**, **answer folder**, and **ask file** commands, to request file or folder information from the user.

Added support for time interval expressions, specified as a number of **weeks**, **days**, **hours**, **minutes**, **seconds**, and/or sub-seconds (where sub-seconds are either **ticks**, **milliseconds**, or **microseconds**). Time interval expressions evaluate to a number representing the specified length of time in seconds, and are useful in a number of ways:

```
put the long seconds into startTime
-- do something time-critical here
put the long seconds - startTime into timeTaken
if timeTaken > 2 minutes then complain "Took too long!"

wait 3 minutes 30 seconds

put the date + 1 week into nextWeek
```

Added support for byte size expressions, specified as a number of **terabytes**, **gigabytes**, **megabytes**, **kilobytes**, and **bytes**. Byte size expressions evaluate to a number representing the specified storage size in bytes, and are useful in a number of situations:

```
if the diskSpace is less than 5 megabytes then put "5 MB warning"
if the size of file logFile > 256 KB then trimLogFile
```

Added a **beep** command to play the system beep sound one or more times:

```
beep
beep 4 -- really get the user's attention
```

Added a **play** command to play a sound:

```
play "Glass" -- without full path, plays a named system sound
play "~/Music/iTunes/iTunes Music/Billy Joel/Piano Man.mp3"
```

You can also stop, pause, or resume the playing of a sound while it is playing:

```
play pause -- pause the current sound output
play resume -- resume playing a sound that was paused
plays stop -- stop playing the current sound
```

Added a **sound** function to return the name of the currently playing sound, or "done" if no sound is playing:

```
play "~/Music/iTunes/iTunes Music/Billy Joel/Piano Man.mp3"
wait until the sound is done
```

Added an English-like **offset** function syntax:

```
put the offset of "smil" within "facsimile" -- 4
```

Changed the comparison operators >, <, >=, and <= to ignore case differences by default when comparing text. This makes them consistent with the other operators (such as = , <>, is in, begins with, etc.).

Added a **caseSensitive** local property (local to each handler) that can be set to true to make text comparisons case sensitive by default. Any **considering case** or **ignoring case** options used in specific comparisons will still be honored; the **caseSensitive** property only changes the default behavior.

Several operators have been updated to work better with lists. The =, <, >, <=, and >= operators now compare individual items in lists, giving more appropriate results in some cases. The **begins with**, **ends with**, **is in**, and **contains** operators now all check for the presence of whole values in the list, whenever the container is a list. For example:

```
("a","b","c") begins with ("a","b") -- this is now true
("a","b","c") begins with "a" -- this is also true
(1,2,3) ends with (2,3) -- true
(1,2,3) ends with 3 -- true
(11,12,13) ends with 3 -- this is false
(11,12,13) ends with 13 -- true
("some","choice","words") contains ("choice","words") -- true
("some","choice","words") contains "words" -- true
("some","choice","words") contains "word" -- false
```

Warning: This release eliminates some formerly valid (though rarely used) forms of the **if ... then ... else ...** statement, in order to avoid ambiguity and prevent unexpected results. Most people will not be impacted by this change. There are still many acceptable ways to write an **if** statement, but the SenseTalk grammar now requires that multi-line forms *not* be mixed with the single-statement forms.

Stated fully, the new rule is this: if the word **then** is followed *on the same line* by a single statement, then the word **else** (if present) must also be followed by a single statement, on the same line as the **else**. These forms of **if** statement ***do not*** end with **end if**.

If the word **then** is followed by a return (so the statement following it is on the next line), then the word **else** (if present) must also be followed by a return. These forms allow multiple statements for both the **then** and **else** parts, and *must end with* **end if**. The only exception to this rule is that the word **else** may be followed by another **if** on the same line, allowing a series of "**else if**" statements to be chained together.

Here are some valid examples of the first (single statement) form (other variations are also possible):

```
if true then put "Yes" else put "No"

if true
then put "Yes"
else put "No"
```

Here are some valid examples of the second (multiple statement) form :

```
if true then
     put "Yes"
     -- other statements may go here
else
     put "No"
     -- other statements may go here
end if

if num > 0 then
     put "the number is positive"
else if num < 0 then
     put "the number is negative"
else
     put "the number is zero"
end if
```

Here are some examples of **if** statements that are **no longer valid** :

```
if true then put "Yes" -- single line "then"
else
     put "No"  -- multi line "else"
end if
```

```
if true then
     put "Yes" -- multi line "then"
else put "No"  -- single line "else"
```

To fix one of these invalid forms and make it valid only requires typing a single carriage return after the word "then" in the first example above. In the second case, it requires adding a return after the "else" and also adding an "end if" at the end.

Changed some details of the **copy** and **move** commands to make their behavior more natural and consistent. The **copy ... into** and **move ... into** forms expect the destination to be a folder, and will automatically create that folder if it doesn't already exist. Existing files or folders are never replaced -- in this case, nothing will be done and **the result** will be set to an appropriate error message.

The **copy ... into** command treats the destination as a folder, creating that folder if it doesn't already exist:

```
copy file "log" into "/tmp/myLogs" -- creates /tmp/myLogs/log
```

The **copy ... as** command treats the destination as a new name for the file or folder being copied:

```
copy file "arc" as "/tmp/holder" -- creates /tmp/holder
```

The **copy ... to** command works just like the **copy ... into** command *if the destination is an existing folder,* otherwise it works just like the **copy ... as** command:

```
delete folder "/tmp/tkts" -- make sure it doesn't already exist
copy folder "tickets" to "/tmp/tkts" -- creates /tmp/tkts
copy file "green" to "/tmp/tkts" -- creates /tmp/tkts/green
```

Warning: All forms of these commands now treat the destination location as being *relative to the source location* (unless a full path is given for the destination). The source file or folder, if a full path is not given, is treated relative to the current working folder (**the folder** global property):

```
set the folder to "/Library"
copy file "Fonts/Arial" as "A2" -- creates /Library/Fonts/A2
copy file "Fonts/A2" into "A" -- creates /Library/Fonts/A/A2
```

The **move ... into** and **move ... to** commands work just like the corresponding **copy** commands, except that the source is moved rather than copied, so the source will no longer exist in its original location after a successful move operation:

```
put there is a file "log" -- true: file "log" exists
move file "log" into "/tmp/myLogs" -- creates /tmp/myLogs/log
put there is a file "log" -- false: file "log" has been moved
```

The **create file** and **create folder** commands will now create the full path to the file or folder being created; the parent folder does not need to exist already.

Added the **locked** (or **immutable**) property of files to the list of file properties which can be accessed or set by SenseTalk. Locked files can not be changed until they are unlocked.

Added an alternate syntax for the **repeat with each** form of repeat. The new syntax accepts the variable name after the chunk type. The following two repeat loops are equivalent:

```
repeat with each item nextItem in myList  -- the new way
          put nextItem
end repeat
repeat with nextItem = each item in myList  -- the old way
          put nextItem
end repeat
```

A **params** declaration may now be used in **on** and **function** handlers in place of declaring parameters following the handler name, for consistency with initial handlers. As with initial handlers, the **params** declaration must be the first non-comment statement in the handler.

Added **are** as a synonym for **is** in most contexts. This allows for more natural-sounding comparisons in certain cases:

```
if the first 3 characters of alphabet are "ABC" then ...
```

Added **more** as a synonym for **greater**. Added **end catch** as a synonym for **end try** to end a try/catch block.

Added a new percent operator (**%** or **percent**) for calculations involving percentages, including add-on and discount operations:

```
put 120 * 6% -- 7.2
put 100 + 9% -- 109
put 100 - 13% -- 87
```

Dates may now be subtracted. Subtracting one date from another yields their difference in seconds.

```
put (date2 - date1) / 1 day into numberOfDays
```

Allow accessing and setting the script property of an object.

```
put my script into saveScript
set the script of hotel to file "reservation1"
```

Changed the file access commands (open, close, read, write) to be consistent in their use of **the result** as the means for reporting errors. These commands will no longer throw exceptions when something fails.

Changed the default mode for the **open file** command to be **updating** rather than **readwrite**. This prevents the file from being truncated unintentionally.

OSXvnc 1.2

A new version of OSXvnc, version 1.2, is included with this release of Eggplant. Eggplant users are strongly encouraged to use this latest release of OSXvnc. This new version includes the following features/fixes:

- Fixed a problem with clipboard contents not being sent out until screen updates were sent (could cause Eggplant's RemoteClipboard() function to not return the correct data).
- Fixed a problem with lossy conversion to CStrings for clipboard contents with non-roman characters.
- Fixed so that client keys which were being held down by a client will be released when the client disconnects.
- Fixed a problem where server sometimes said it needed to restart when nothing hadchanged.
- Fixed an occasional crash on 10.1 when clients disconnected.
- Setup run-loop to only poll screen when clients are connected, for improved performance.

Bug Fixes

The following bugs and interface issues were resolved for this release:

Fixed a problem with the **read ... until end** command.
Fixed a bug with the **open file ... for updating** command, which would not create a file (note: only the **for updating** version of this command had a problem).
Fixed problem with "open file for writing" not replacing the file.
Fixed problem with files not being truncated after they were written to by the write command.
Fixed a bug with accessing nested list items (specifically, adding a list (vector addition) to a nested list of a container, but may have shown up in other ways too). Cleaned up some other issues involving nested list references, such as storing into a non-existent item of a range of items.
Fixed some problems with the **move file**, **copy file**, **create file**, **create folder, delete file, delete folder**, and **rename** commands when paths were specified using a "~" to represent the user's home directory.
Fixed a problem that could treat an **on** or **function** handler with syntax errors in the declaration line as a valid but empty handler.
Fixed a problem with numbers expressed using scientific notation that would not accept an integer (without a decimal point) before the letter 'e'.
Fixed a problem with parsing long integer literals with more than 9 digits (> 2^31).
Fixed a problem with decimal fractions as words.
Fixed a problem with long block comments containing quoted strings.
Fixed object comparisons so they are not dependent on the case of keys.
Script Logs are now updated properly when running scripts from the Schedules tab.
Results now appear in the Results tabs properly when running scripts from the Schedules tab.
Fixed a problem in which scripts listed in the Schedules tab would not be found after the suite was moved or copied to a new location. Running these schedules would falsely report immediate success of the scripts,

without actually running them. Older suites may still exhibit this problem if they are copied. To fix them, remove all scripts from the Schedules tab and add them back in again.
Fixed the "Insert..." Popup on Script Editor to work in Menu Form Mode (Labels only or off the right hand side submenu)

Release v1.21 (17-January-2003)

This maintenance release corrects a problem inadvertently introduced in 1.1 which prevented Eggplant from running under Mac OS X 10.1. This release should run on all systems using Mac OS X 10.1 or higher (tested with 10.1.3).

Program/User Interface

The Remote Screen window now displays additional status information in its title bar. When connecting to a different system, the Remote Screen window will switch its display to reflect the new system, resizing the window as needed. In earlier versions, the window would close and reopen.

Specific VNC encodings can now be disabled in Eggplant if needed, using "defaults write" commands in a Terminal window. This allows you to control which encodings will be used, which may improve performance in some situations. A preference mechanism will be added in a future release to make use of this feature more convenient.

OSXvnc 1.11

An update to OSXvnc, version 1.11, is included with this release of Eggplant. This version of OSXvnc provides a convenient Restart button to allow the server to be restarted remotely after changing its settings, as well as some other minor enhancements.

Bug Fixes

The following bugs were resolved for this release:

Runs under Mac OS X 10.1 and later.
Fixed occasional "screen freeze" problem caused by decompression errors in the zRLE encoding.
Changed the default VNC encoding for Eggplant to be zHextile rather than zRLE, which improves performance for most local network situations.
On launch, Eggplant now only opens the script and suite windows which were actually open the last time you quit, not those which were in use only as Helpers but not shown on the screen.
FIxed a bug where the the "Capture Toggle" hot-key would invoke at inappropriate times.

Release v1.2 (03-January-2003)

OSXvnc 1.1

A new version of OSXvnc, version 1.1, is included with this release of Eggplant. Eggplant users are strongly encouraged to use this latest release of OSXvnc. This new version includes the following features/fixes:

OSXvnc now supports Scroll Wheel Mice (compliant with RealVNC 3.3.5)
OSXvnc now supports zRLE encoding (compliant with RealVNC 3.3.4)
8-Bit color now works

Can optionally set to allow only local connections (useful with SSH)
Updated FAQ with lots of useful information
Enhanced Keyboard Mapping Performance
Now compatible with all versions of Chicken of the VNC
Fix which caused pointer events to be locked out and occasionally not recognized
Fix to a problem when multiple clients were using encodings that use Zlib compression
Fix to a problem where changing screen resolution didn't work if launched command line
Fix to the Zero Size Rect
Fix to the default keymapping for ~ and ` keys

Program/User Interface

Two new items have been added to the tool bar of script editor windows to facilitate using existing images *(Note: users upgrading from older versions may need to customize their toolbar and drag the default toolbar into place to see these new items)*.

The **Insert...** pull-down list allows you to generate commands and functions using existing images. Selecting a command or function from the list will display a dialog allowing you to select an image from the Images directory of the current suite. When the image is selected, Eggplant will insert the appropriate command or function into your script.

The **Add Image** item allows you to add an image from your Images directory into an existing command.

You can also generate commands from the Remote Screen window that use existing images, by holding down the Option key when clicking one of the items in its toolbar. When you do this, you will be presented with a panel for selecting an existing image, instead of capturing a new image. This can be done whether or not you are in Capture Mode at the time.

The hot spot of images can now be altered by command-clicking in our outside of the image at the time it is being captured, as well as before capture and when the image is selected in the Images tab of the suite.

Images and folders can now be dragged in or out of subfolders within the hierarchy in the Images tab of a suite.

A **Script Animation** option has been added to the File menu. When this menu item is checked, Eggplant will highlight each line of a script while it runs. This slows down execution of the script somewhat, but can be useful to see how your script is running.

Eggplant now automatically reopens any Suites and Scripts that were open the last time you used Eggplant.

Scripting

- Added support for inter-process communication using sockets. The **open socket**, **close socket**, **read from socket**, and **write ... to socket** commands mirror the corresponding **file** commands.

```
put "10.1.0.0:5900" into vnc
open socket vnc
read from socket vnc until return
put it
write it to socket vnc
-- etc.
close socket vnc
```

In addition, all forms of the **read** and **write** commands now include support for numeric data in 10 different binary formats: signed and unsigned 8-bit, 16-bit, 32-bit, and 64-bit integers, and single-and double-precision floating-point numbers:

```
read 1 integer from file fname
read 6 unsigned 8-bit integers from socket rfb

write numberList as 16-bit integers to file bData
write (2,3,5,9) as 8-bit integers to file fn at 512
```

The type of data to be read or written should be specified using one of these codes or phrases:

int1 or **8-bit integer**	an 8-bit (or 1 byte) signed integer
uint1 or **unsigned 8-bit integer**	an 8-bit (or 1 byte) unsigned integer

int2 or **16-bit integer** or **short integer**	a 16-bit (or 2 byte) signed integer
uint2 or **unsigned 16-bit integer**	a 16-bit (or 2 byte) unsigned integer
int4 or **32-bit integer** or **integer**	a 32-bit (or 4 byte) signed integer
uint4 or **unsigned 32-bit integer**	a 32-bit (or 4 byte) unsigned integer
int8 or **64-bit integer**	a 64-bit (or 8 byte) signed integer
uint8 or **unsigned 64-bit integer**	a 64-bit (or 8 byte) unsigned integer
real4 or **32-bit real** or **float**	a 32-bit (single-precision) floating-point number
real8 or **64-bit real** or **double**	a 64-bit (double-precision) floating-point number

---

- Enabled certain file and folder properties to be set (written) as well as accessed. The following properties can be set:

creation date	the date/time when the file was created
modification date	the date/time when the file was last changed
owner permissions	permissions of the file's owner (read, write, and/or execute)
group permissions	permissions of the files owner's group (read, write, and/or execute)
other permissions	permissions of other users (read, write, and/or execute)
permissions	a string denoting all of the file's permission settings (see below)
type code	HFS 4-character type code
creator code	HFS 4-character creator code
owner id	the user id number of the file's owner
group id	the group id number of the file's owner

- Added **create file** and **create link** commands to go along with the **create folder** command added earlier (files can also be created by simply writing into a new file, as in earlier versions).

```
create a new file "juicy" in folder "stories"
create link "tasty" to file "juicy"
```

- Added **toUpper** and **toLower** functions to convert text to all upperCase or all lowerCase.

```
put toUpper("Hello!") -- HELLO!
put toLower("No CAPS Allowed") -- no caps allowed
```

- Added support for scientific notation. Numbers may now be entered in a script in the form 4.58e+6 where the integer following the letter 'e' is the power of 10 which the number is multiplied by. That is, 4.58e+6 is equivalent to 4.58 * 10^6 .

Bug Fixes

The following bugs were resolved for this release:

Fixed a bug when connecting to many different size monitors within a script (possible crasher).
Fixed a bug when trying to drag out of image view when no image is present.
Fixed a when referring to resources (Images or Scripts) with an absolute file path.
Fixed a bug when highlighting from a Log in a script which has been changed.
Fixed a problem when errors occurred within a handler called by a send statement.
Fixed a parser bug which would erroneously allow SenseTalk to find and execute handlers contained within block comments or block quotes.
Fixed problems with SSH Connections not getting communicated on the SSHTask (for example if old task is present).
Fixed some licensing issues.

Release v1.11 (04-December-2002)

Program/User Interface

SSH Support

Eggplant now supports tunneling its VNC connection to the system under test through an SSH server. SSH connection information can be provided through the connection panel, through a Suite's scheduled connection information, or via the CONNECT command in SenseTalk scripts.

SSH Server/Connection:
To establish an SSH tunnel the "SSH Host" (i.e. the SUT or gateway to the SUT) must be running an SSH2 protocol server that supports port forward and the user must be able to authenticate to the machine as user "SSH User". RSA, publickey, and password authentication are supported, although we recommend the first two if you do not want to store your password on the machine for Eggplant.

To set this up on a target OS X machine you need to go to System Preferences -> Sharing and then enable Remote Login (Under 10.2+ this is one of several checkboxes on the services tab, under 10.1 it's just a checkbox labeled "Allow Remote Login").

Under Windows or other platforms you will probably need to download and install your own SSH server.

For more information please read the SSH FAQ:
http://www.tigerlair.com/ssh/faq/ssh-faq.html

The SSH Host can be the same machine as the VNC Server but it does not have to be.
For more information on how VNC uses SSH:
http://www.uk.research.att.com/vnc/sshvnc.html

To establish the connections Eggplant uses an internal SSH configuration file located in the Application as Eggplant.app/Contents/Resources/ssh_config. For more information regarding this file, see manual page `ssh_config`(5).

User Display Preferences
Eggplant now honors the user's chosen Date and Time display settings for the purposes of display dates in the Script List, Log List, and File Info Panels.

Scripting

In addition to all previous syntax, the **connect** command now supports a new syntax that takes a single Property List argument (for information on Property Lists please see SenseTalk Scripting Reference: Chapter 02 Values):

connect *properties*

Properties can contain the following keys:
serverid = VNC server to connect to
portnum = Port number for VNC
password = VNC password
sshUser = User name to establish an SSH connection.
sshPassword = User Password to establish an SSH connection (if necessary)
sshHost = SSH server to connect to, if this is specified the system will establish and use an SSH connection, otherwise it will not.
visible = YES or NO. Control whether the Remote Screen window appears after connecting (Default is previous state).

Example:

```
connect (serverId:"vnchost.mycompany.com", portNum:5900, \
          password:"foobar", sshHost:"sshhost.mycompany.com")
```

This command will connect to the machine "vnchost" on port 5900 with the vncpassword of "foobar", tunnelling the connection through "sshhost". Since no user is specified it will try to login to sshhost using the current username. Since no password is specified it will only be able to authenticate if RSA or some other public key encryption mechanism is in place.

Some of the new file properties introduced in the last release have been updated to provide better information.

- The typeCode and creatorCode properties of a file now return a 4-character code as they should, rather than a number.

- You can now access properties of a folder as well as those of a file:

```
put the size of folder "work" -- (size of its table of contents)
```

- You can also access properties of a link directly (rather than properties of the file or folder it is linked to):

```
put the permissions of link "/tmp"
```

- Added **owner permissions**, **group permissions**, and **other permissions** properties of files and folders, which each provide access to part of the information offered by the **permissions** property, but in a nicer format. Each of these returns a comma-delimited text list of some combination of the words "read", "write", and/or "execute" (or empty) to indicate the permissions available to the user who owns the file, the group that owns the file, or others, respectively.

- Added a **display name** property of files and folders, to return the name that should be displayed to the user. This will be the same as either the name or the short name, depending on whether the file displays its extension or not.

- Changed the "file type" property of a file to **entry type** instead, for greater clarity, since they are not always files. In the case of folders, this property now returns "Folder" rather than "Directory", and "File" rather than "Regular" for ordinary files.

Bug Fixes

We believe all known bugs have been addressed at this time with the exception of a bug which can cause the Remote Screen Toolbar to not become active when it should.

In particular, the following bugs were resolved for this release:

Fixed a bug when displaying log information from a SenseTalk script. This could happen anytime but was most likely to occur when log output from the script was extensive or when running on a fast dual processor machine.
Fixed a problem running longer or more complex scripts.
Fixed a bug that could send Eggplant into an infinite loop trying to reformat certain syntactically-incorrect scripts. This usually showed up after pressing return in an incomplete script.

Release v1.1 (15-November-2002)

Program/User Interface

Speed enhancements: Support for additional VNC encodings has been added. This makes Eggplant much more responsive in Live mode when connected to VNC servers which support these encodings. The specific encodings which were added are ZRLE (new in version 3.3.4 of VNC), Zlib, and ZlibHextile.

Eggplant's connection to the VNC server running on the SUT is more robust. In particular, issues with certain servers such as RealVNC 3.3.4 and the AIX VNC servers have been resolved. Adaptive connection timers help Eggplant determine inherent latency in the network and adjust its internal timeouts accordingly.

Mouse scroll-wheel events are now sent through to the SUT when connected in Live mode. These events are supported by some VNC servers but not others. The OSXvnc server will be updated soon to add this capability.

Scripting

The **RemoteClipboard()** function now accepts an optional parameter indicating the maximum number of seconds to wait for new clipboard contents to arrive from the SUT. If a time parameter is given, Eggplant will check for new clipboard contents from the SUT. If a new clipboard value is received, it will be returned. If no new contents are received within the specified time, an exception will be raised.

Eggplant now provides explicit error messages if a command tries to click or move the mouse to coordinates which are off the remote screen, or when an image is found on screen but its hot spot is off screen. Clicks at the

furthest extent of the screen (e.g., at 1024,768 on a remote screen of size 1024x768) are accepted.

Mouse scroll-wheel events may be generated with two new commands, **ScrollWheelUp** and **ScrollWheelDown**. Both commands take a single argument which is the amount to scroll in the indicated direction. These commands work with VNC servers which offer scroll wheel support. Current VNC servers for Windows offer this support now. The OSXvnc server will be updated soon to add this capability.

When the **shouldRepositionMouse** option is set using the SetOption command or the RepositionMouse YES command, Eggplant moves the mouse "out of the way" to the bottom right corner of the screen whenever it can't find an image immediately and then looks for the image again. With this release, the "out of the way" location has been adjusted to be a few pixels away from the bottom right corner of the screen, to avoid popping up the Dock or screen saver on OS X systems. You can set the location to move the mouse with the SetOption command and the new **repositionPoint** option. Setting this option to empty will restore the default behavior.

The functioning of the **OpenSuite** command has changed slightly. Suites opened in this way have lower priority than the current suite, similar to any Helper suites specified in the Helpers tab of the Suite Editor. Suites opened dynamically with OpenSuite have a higher priority than static Helpers, however.

SenseTalk provides much richer support for accessing files and directories which are accessible through the filesystem of the Eggplant machine. Specifically, the following features have been added or updated:

- Added a **create folder** command (or **create directory** -- the terms folder and directory are interchangeable) to allow SenseTalk scripts to directly create new folders. The parent folder must already exist and must have proper permissions to allow creating a new folder within it.

```
create a new folder "/tmp/myWorkArea"
create folder "/tmp/myWorkArea/subdir"
```

- Added **delete file** and **delete folder** (or **delete directory**) commands to allow deleting of files or folders. Deleting a folder will delete all of the files within it as well.

```
delete file "/tmp/myWorkArea/testData27"
delete folder "/tmp/myWorkArea"
```

- Added a **rename** command to allow renaming of files or folders.

```
rename folder "/tmp/myWorkArea" as "/tmp/oldWorkArea"
```

- Added a **move** command to allow moving files or folders into a different folder. The destination must be an existing folder.

```
move file "/tmp/testFile" into folder "/tmp/myWorkArea"
```

- Added a **copy** command to allow duplicating files or folders. The first form, using the preposition **into** (or **to**), makes a copy of the source file or folder with the same name as the original in a different destination folder. The destination must be an existing folder.

```
copy file "/tmp/testFile" into folder "/tmp/myWorkArea"
```

  The second form of **copy**, using the preposition **as**, allows you to assign a different name to the copy. The copy may be created in the same folder as the source, or in a different folder.

```
copy file "/tmp/testFile" as "/tmp/testFileCopy"
```

- Added **the folder** (or **the directory**) global property, which is the full path of the current directory. Setting this property changes the current directory.

```
put the folder & "/" & filename into filePath
set the folder to "/tmp/myWorkArea"
```

- Added **files()** and **folders()** (or **directories()**) functions, which return a list of the files or folders, respectively, in the specified directory. If no directory is specified, the files or folders in the current directory are given.

```
put the folders into currentSubfolders
put the files of "/tmp" into tempFilesList
```

- Added a **diskSpace()** function, which returns the amount of free space on the disk where the specified file or directory is located. If no argument is given, it returns the amount of free space on the disk where the current directory is located.

```
if the diskSpace is less than 5*1024*1024 then put "5 MB warning"
put diskSpace("/Volumes/st508") into bytesAvailable
```

- Implemented the ability to access properties of files. The following properties are accessible:

name	the name of the file, such as "myFile.txt"
short name	the name without the extension -- "myFile"
abbreviated name	same as the short name
long name	the full path name of the file, such as "/tmp/myFile.txt"
folder *or* **directory**	the full path name of the folder containing the file
size	the size of the file in bytes
creation date	the date/time when the file was created
modification date	the date/time when the file was last changed
permissions	a string denoting the file permissions (see below)
file type	"Regular", "Directory", "SymbolicLink", "Socket", "CharacterSpecial", "BlockSpecial", or "Unknown"
type code	HFS type code, as a number
creator code	HFS creator code, as a number
owner name	the login name of the owner of the file
group name	the name of the group the file is owned by
owner id	the user id number of the file's owner
group id	the group id number of the file's owner
link destination	for symbolic links, this gives the relative path name of the linked-to file

  Currently these are all read-only properties. In the future, the permissions should become settable, and possibly a few of the other properties as well, such as typeCode and creatorCode, and maybe the owner and group.

  Permissions is a string, similar to that reported by the 'ls -l' command in Unix. It consists of 9 characters, "rwxrwxrwx" indicating read, write, and execute permissions for the owner, group, and others. Any permissions which are not present are replaced by dashes, so a permissions value of "rwxr-xr-x" for example would indicate a file that is readable and executable by everyone but writable only by the owner. The special permissions flags are indicated by additional items appended with separating commas when they apply. These include "setuid", "setgid", and "sticky". So, the permissions value for a setuid file might be "rwxr-xr-x,setuid".

  File properties are accessed using this syntax:

  the <fileproperty> of file <filename>

  Examples:

```
the short name of file "/tmp/Friday.rtf" -- "Friday"
add the size of file datafile1 to filesizeTotal
```

- Implemented the **openFiles()** function, which returns a list of all files currently open as a result of the **open file...** command.

```
-- close all open .txt files
repeat with each item of the openFiles
          if it ends with ".txt" then close file it
end repeat
```

- Uninitialized variables are treated as zero when they are the destination of an add, subtract, multiply, or divide command. This is how it was always supposed to work, but somewhere along the way this had been broken. This feature is especially handy for working with counters without needing to set them to zero first.

```
add 1 to counter -- this works even if counter is uninitialized
```

- When commands or functions are not handled, before SenseTalk sends an undeliveredMessage message, it first looks for an object (e.g. a script) with the same name as that command or function. If one is found, the message is sent to it.

- The **answer** command may be followed by the word **error** or **warning** to display a warning icon in the alert panel which is displayed, rather than the usual application icon.

```
answer error "Something went terribly wrong!"
```

- Fixed a problem with **the result**. It should now return meaningful error messages for many commands which fail.

- Made a slight change to the **is within** operator. It should now return true for points on all borders of a

rectangle.

Bug Fixes

All known bugs have been addressed, except for occasional reported problems on dual-processor machines (we're working on that one).

Fixed several problems related to the **ask** and **answer** commands, including occasional crashes.
Calling scripts referenced in helper suites' helper suites now works.
The **run** command now properly finds scripts in helper suites.
Fixed crash when unhiding Eggplant after Remote Screen window connection had been closed.
Toggling Capture mode by pressing and releasing the command key no longer causes problems with SUTs running Windows.
Dragging an image into a script window only inserts a comma when one is needed.
Exceptions raised in nested scripts can now be caught properly by try/catch in the calling script.
Scripts referenced through an object's helpers are now reloaded if they are changed.
The capture mode selection rectangle can no longer be incorrectly positioned offscreen by a click near the edge of the screen.
Command-line execution works properly when no password is needed and none is given.
The runscript command-line tool is properly linked to the executable.
Eggplant could previously hang while crashing.

Release v1.0 (26-September-2002)