

Eagle: Maturation and Evolution

Joe Mistachkin <joe@mistachkin.com>

Abstract

Eagle (Extensible Adaptable Generalized Logic Engine) is an implementation of the Tcl scripting language for the Common Language Runtime (CLR) .

The initial scope of the project was modest compared to “real” Tcl; however, it was ambitious considering the sixty day schedule for design, implementation, and testing. Three years have passed since then.

This paper explains how Eagle has matured and evolved since it was presented at the 15th Annual Tcl/Tk Conference .

1.Introduction

Eagle has changed significantly since its inception. The primary goal of the project was to provide a complete automation solution for all types of applications running on the CLR. The Tcl language was chosen for its minimalist syntax, simplicity of implementation, and straightforward language semantics.

It has been three years since the start of the project. It has been two years since it was first presented to the public at the 15th Annual Tcl/Tk Conference.

This paper will briefly discuss the major improvements that have been made in terms of performance, ease-of-use, and compatibility.

2.Maturing Code, Evolving Goals

The primary goal for the project, like the original goal for Tcl itself, was to provide a highly extensible library for automation of applications via scripting. All other goals were secondary, including performance, ease-of-use, and full script-level compatibility with Tcl. After this initial goal was met, it became clear that the secondary goals could be achieved incrementally going forward.

3. Performance

Eagle was not originally designed for raw performance; however, it was designed with performance in mind. Unfortunately, it can be much slower than “real” Tcl, even for the simplest operations. Profiling reveals that the slowest operations are:

- Parsing strings into lists.
- Building lists from strings.
- Expression evaluation, primarily string-to-type conversions.

All other performance issues are insignificant compared to these three. The primary reason type conversions are so slow is the necessity of starting with the most restrictive numeric type and “failing upward” until something works. If none of the numeric types work, then the operand is a string.

These performance issues are not seen as critical because Eagle is not intended to be used in place of a general purpose programming language. It is intended to automate components and/or “glue” components together while playing nicely in a fully managed code environment and retaining all of the dynamic features and expressive power of the core Tcl language.

4. Compatibility

The intent of the project was to provide full script-level compatibility with Tcl 8.4 ; however, given how aggressive the original project timeline was, certain features had to be omitted. At the time of this writing, Eagle is still not 100% compatible with Tcl 8.4. This table compares the major missing features between the 15th Annual Tcl/Tk Conference and today:

October 2008	October 2010
No Tk functionality.	No change.
No argument expansion syntax.	No change.
NO <code>namespace</code> support except the global namespace.	The command and variable resolvers are now extensible; however, namespace semantics are still not implemented.
No asynchronous input/output.	No change.

No server sockets.	Fully implemented.
No <code>registry</code> or <code>dde</code> commands.	No change.
No <code>pkg::create</code> or <code>pkg_mkIndex</code> commands.	No change.
No slave interpreters, no hidden commands, no aliases, and no Safe Tcl.	Slave interpreters, hidden commands, aliases, and an extensible policy mechanism have been implemented. The Safe Base is still not implemented.
No Tcl library, such as the <code>http</code> , <code>msgcat</code> , and <code>tcltest</code> packages.	No change.
The following Tcl commands are not implemented: <code>fblocked</code> , <code>fileevent</code> , <code>fcopy</code> , <code>glob</code> , <code>memory</code> , <code>binary</code> , <code>scan</code> , <code>format</code> , <code>trace</code> , and <code>history</code> .	The <code>fcopy</code> command has been implemented with the exception of the “-command” option.
The Tcl library routines matching the patterns <code>auto_*</code> and <code>tcl_*</code> are not implemented.	No change.
For the <code>open</code> command, command pipelines and serial ports are not supported.	No change.
For the <code>exec</code> command, Unix-style input/output redirection and command pipelines are not supported.	No change.
For the <code>load</code> and <code>unload</code> commands, the semantics are not identical to those of Tcl because the binary package management is radically different due to the nature of the CLR.	No change.
For the <code>clock format</code> command, some of the Tcl formatting characters are not supported.	All of the Tcl formatting characters from Tcl 8.4 are now supported.
For the <code>clock scan</code> command, recognition of relative date/time strings (e.g. “1 day	No change.

ago", "next Wednesday", etc.) is not supported.	
For the <code>fconfigure</code> command, all options except " -encoding " and " -translation " are not supported.	No change.
For the <code>regexp</code> command, the " -about " option is not supported.	No change.
For the <code>regexp</code> command, using the " -start " option with the beginning of a line anchor does not work properly due to lack of support for something like <code>TCL_REG_NOTBOL</code> in the CLR regular expression engine.	No change.
For the <code>pid</code> command, supplying the optional <i>channelId</i> argument is not supported (always returns an empty string).	No change.
For the <code>proc</code> command, arguments with default values are not supported.	The <code>proc</code> command has full support for arguments with default values.
For the <code>exit</code> command, by default does not exit the process; it merely prevents further trips through the engine and the interactive loop.	This is by design. The " -force " option can be used to mimic the Tcl behavior.
For the <code>after idle</code> command, we evaluate the script immediately prior to the next evaluation because we have no idle detection.	No change.
The <code>array statistics</code> command is not supported.	No change.
For the <code>array</code> command, the search sub-commands (i.e. " anymore ", " nextelement ", " done search", " start search") are not supported.	Fully implemented.

For the “ <i>env</i> ” array, the <code>array names</code> , <code>array get</code> , and <code>info exists</code> commands are not supported.	Fully implemented.
Documentation of the integration and extensibility API is incomplete.	No change; however, some progress has been made on tooling for the generation of documentation.
Support for <code>tcltest</code> functionality is incomplete.	The “ <code>test</code> ” command has been fully implemented.
Unit tests are incomplete.	There are now a large number of unit tests. More tests and better code coverage are still needed.

Many people ask why support for namespaces has not been implemented. Originally, it was due to lack of time in the project schedule. Today, it is due to the potentially disruptive effects it would have on stability and performance. An important thing to remember is that namespaces were added to Tcl only because the number of global variables and procedures in large Tcl applications was becoming unmanageable. Since Eagle is not designed for building stand-alone applications, this was seen as less of an issue.

It was thought that choosing the CLR as the platform would eliminate or greatly reduce the need for any future porting efforts. This has proven to be incorrect. There are a huge number of subtly incompatible versions, variations, and subsets. Each of these is really a distinct platform. The following platforms are currently supported:

- Microsoft .NET Framework 2.0, 3.0, 3.5, and 4.0
- Mono on Windows and Unix

The following platforms are untested:

- DotGNU

The following platforms are not supported:

- Microsoft .NET Compact Framework 2.0 and 3.5
- Microsoft Silverlight
- Moonlight

5. Refactoring

Much refactoring work has been performed on the code . These efforts have focused on reducing the total amount of code and making it cleaner, more portable, more extensible, and easier to understand. Examples include:

- More consistent naming of variables, methods, and classes.
- Simplified command and plugin integration.
- Simplified host creation and integration.
- More extensible core marshaller.
- Targeted optimizations for all critical code paths.
- More robust threading model.
- Test framework that works with both Tcl and Eagle.
- Better Tcl/Tk integration.

6. The Future

In the near future, some work needs to be done to advance the project beyond the beta stage. The currently planned exit criteria for the beta are:

- No unresolved high-priority bugs.
- Script compatibility with Tcl 8.4 except the `namespace` and `clock scan` commands.
- At least one production deployment.
- Full documentation of the managed integration API.
- Full documentation of the differences between Tcl and Eagle.

7. Acknowledgements

This year, the author wishes to thank Dawson Cowals and Stuart Cassoff for their help.

8. References

- [1]Eagle, <http://eagle.to/>
- [2]Tcl Developer Xchange, <http://www.tcl.tk/>
- [3]"Scripting language", http://en.wikipedia.org/wiki/Scripting_language
- [4]"Common Language Runtime", http://en.wikipedia.org/wiki/Common_Language_Runtime
- [5]15th Annual Tcl/Tk Conference, <http://www.tcl.tk/community/tcl2008/info.html>
- [6]Glue language, http://en.wikipedia.org/wiki/Glue_language

- [7]Managed code, http://en.wikipedia.org/wiki/Managed_code
- [8]Tcl 8.4, <http://www.tcl.tk/man/tcl8.4/>
- [9]Tcl library routines, <http://www.tcl.tk/man/tcl8.4/TclCmd/library.htm>
- [10]tcltest functionality, <http://www.tcl.tk/man/tcl8.4/TclCmd/tcltest.htm>
- [11]Namespaces, <http://www.beedub.com/book/2nd/Name.doc.html>
- [12]Microsoft .NET Framework, http://en.wikipedia.org/wiki/.NET_Framework
- [13]Mono, [http://en.wikipedia.org/wiki/Mono_\(software\)](http://en.wikipedia.org/wiki/Mono_(software))
- [14]DotGNU, <http://en.wikipedia.org/wiki/DotGNU>
- [15]Microsoft .NET Compact Framework,
http://en.wikipedia.org/wiki/.NET_Compact_Framework
- [16]Microsoft Silverlight, http://en.wikipedia.org/wiki/Microsoft_Silverlight
- [17]Moonlight, [http://en.wikipedia.org/wiki/Moonlight_\(runtime\)](http://en.wikipedia.org/wiki/Moonlight_(runtime))
- [18]Eagle ChangeLog, <http://eagle.to/ChangeLog>
- [19]Beta, http://en.wikipedia.org/wiki/Software_release_life_cycle#Beta
- [20]Dawson Cowals, <http://www.dawsoncowals.com/>
- [21]Stuart Cassoff, <http://wiki.tcl.tk/10628>