Dylan™ FAQ
Apple Computer - July 29, 1994

This memo answers questions which are frequently asked about the Dylan programming language, and its implementations. If you want to keep up with Dylan news, consider joining the announce-dylan or info-dylan mailing lists, or the comp.lang.dylan newsgroup, described below.

The latest version of this memo is available by anonymous ftp from `cambridge.apple.com:/pub/dylan/faq/` in a variety of formats, including plain ASCII, RTF (Rich Text Format), and PostScript™.

The topics covered in this FAQ are

[1] Online Sources of Dylan Information

[1-1] Where is Dylan information generally available?

On the Internet, the Apple Dylan ftp site is
`cambridge.apple.com:/pub/dylan/`.

The Dylan WWW (World-Wide Web) page at Carnegie Mellon University is
`http://legend.gwydion.cs.cmu.edu:8001/dylan/`

The Dylan newsgroup is `comp.lang.dylan`.

On Applelink, Dylan files are regularly uploaded to this area:

```
  Developer Support:Developer Services:
     Development Platforms:Dylan Related
```

On CompuServe, Type `GO APPLE` to get to the Apple support forum.
There are 16 libraries, go into `Programmers/Developers Library #15`

[1-2] Where can I get a copy of the language specification?

The first book on the Dylan language was published in 1992. Since then, the language has undergone a great deal of change in response to feedback from potential users and implementors. Throughout this process, changes to the language design have been published electronically in the form of design notes. The current round of language design is now essentially complete except for the macro system.

A new Dylan language reference will be published in early 1995. This will be the definitive specification of the Dylan language. Apple is working closely with other Dylan implementors to ensure that this book is not Apple-specific, but rather, applies equally to all Dylan implementations.

Of course, we realize that many people want to read about Dylan now! So to fill the gap until the new book is published, an interim Dylan reference book is available by anonymous ftp, in the directory `/pub/dylan/dylan-manual/`, as well as in the other file areas listed in topic [1-1] above. It is available in both RTF

(Rich Text Format) and PostScript formats (214 pages). The interim book is a combination of the original Dylan book, the previously published design notes, and additional previously unpublished design decisions. The document is called "interim" not because the language design is unfinished, but because this is a very rough document intended for use until the new book is ready.

[1-3] Are there any public mailing lists for discussing Dylan?

Yes. The `info-dylan` mailing list is a forum for discussing all subjects related to Dylan. Each day's messages are gathered into a digest and sent as a single compound message to the list `info-dylan-digest`. The mailing list is also linked to the Newsgroup `comp.lang.dylan`.

The `announce-dylan` mailing list is a moderated list, for major announcements about Dylan (such as the availability of new implementations of the language). All messages sent to `announce-dylan` are also sent to `info-dylan`.

To subscribe to `info-dylan` or `announce-dylan`, send mail to `majordomo@cambridge.apple.com.` The body of the message should be "`subscribe` <*list-name*>", where <*list-name*> is the name of the mailing list you want to subscribe to. To unsubscribe to one of the mailing lists, send majordomo a message with the body "`unsubscribe` <*list-name*>". If you would like to subscribe or unsubscribe an address which is different from the return address of the message, include the address after the <*list-name*>. For complete majordomo instructions, send a message with the body "`help`".

The name of the digest form of `info-dylan` is `info-dylan-digest`.

Please do not send administrative requests to the mailing lists! If you have trouble with info-dylan, send mail to `sysadmin@cambridge.apple.com.`

[1-4] Where else can I get information about Dylan?

The Gwydion Project at Carnegie Mellon University maintains a World Wide Web (WWW) page of general information on Dylan. This is accessible using Mosaic or other web-browsing software. This page contains the Interim Reference Manual as well as the original Dylan book in on-line form, annotated with links to the design notes and changes that have been approved. This web page also contains the current FAQ from Apple and other general information of interest to the Dylan community.

The URL is "`http://legend.gwydion.cs.cmu.edu:8001/dylan/`".

The Gwydion Project also intends to maintain an FTP-accessible on-line repository of Dylan libraries and code that the authors wish to share with the rest of the community. When this is in operation, it will be announced on the Dylan mailing lists and will be pointed to from the Dylan WWW page mentioned above.

[1-5] Where should I send comments on the Dylan language?

Most comments can be sent to `info-dylan`. If you wish to send a private letter to the Apple engineers working on Dylan, you can send a message to `dylan-comments @cambridge.apple.com`. (AppleLink users can send mail to `DYLAN`).

[1-6] What should I do if I want to implement Dylan?

Send mail to `dylan-comments@cambridge.apple.com`. We have put together a program to support implementors, and we want to hear from you. (AppleLink users can send mail to `DYLAN`).

If you've written an implementation of Dylan and want to release it, please contact us for a trademark license.

[1-7] Is there a group which promotes the use of object-oriented dynamic languages?

Yes. There is an OODL special interest group of the Software Frameworks Association (formerly called MADA). SFA is a group which champions object-oriented programming on the Macintosh as well as other platforms.

To subscribe to the OODL sig mailing list from AppleLink send mail to `OODL.SIG`. Internet subscriptions should be requested from `oodl-sig-request@cambridge.apple.com`.

[2] General Questions About Dylan and Implementations

[2-1] What is Dylan?

Dylan is a new Object Oriented Dynamic Language (OODL). Dylan combines the features of static and dynamic languages. The goal of the language is to support a high level of programmer productivity, while still allowing the efficient delivery of applications and libraries.

Dylan is consistently object-oriented. It is not based on a particular previously-existing procedural language with some kind of object-oriented extension.

[2-2] What is the target audience for Dylan?

The target audience for Dylan is developers of commercial application software, most of whom are currently using static languages such as C and C++. We expect Dylan will appeal to many other groups of programmers as well, such as educational users who want a very clean object-oriented language design, or in-house developers who need a high-level, very productive language.

[2-3] How does Dylan differ from previous OODLs?

Dylan is designed to allow the powerful and flexible programming techniques and development environments associated with OODLs, while also allowing the small, fast delivered applications currently associated with static languages.

Unlike many dynamic languages, Dylan's design consciously enables the runtime environment to execute without the development environment present. In addition, Dylan will let you selectively 'turn-off' dynamic capabilities when they are no longer needed, allowing more efficient compilation.

[2-4] Does Apple have an implementation of Dylan?

Apple recently announced plans to release an implementation of Dylan.

The following description of Apple's implementation was taken from a data sheet distributed by Apple.

> The Apple Dylan development environment is designed to let you create complex, commercial-quality projects with all the advantages of a rapid-prototyping environment.
>
> Your project is stored in a database, unlike traditional, file-based systems. Apple Dylan's customizable browsers will give you a powerful new way to look at and manipulate your application as it executes. You can browse class hierarchies graphically, find all references to a given routine or variable, and inspect objects in your program while it's running.
>
> With Apple Dylan's incremental compiler, you will be able to actually change code in a running program and see the results right away. No more waiting for a long edit, compile, link, execute, debug cycle. This gives you freedom as a programmer to explore various options and rapidly improve your product.
>
> Apple Dylan will include:
> - Dylan compiler and runtime
> - Integrated development environment featuring incremental development and advanced configurable browsing and viewing of code
> - Dylan application framework
> - Dylan user-interface builder
> - Cross-language support allowing seamless access to existing C and C++ code and APIs

The first release of Apple Dylan is scheduled to ship in the first quarter of 1995. This release will support native 68K stand alone applications.

Six to nine months after the first release, Apple will release a version of Dylan which supports native PowerPC code and OpenDoc parts (components).

[2-5] How can I apply to be an early seed site for Apple's Dylan implementation?

Send a message to the AppleLink address `DYLAN`, or the Internet address `dylan@applelink.apple.com`. The message should include your name, address, phone number, and a brief description of how you plan on using Apple Dylan.

[2-6] What third-party implementations of Dylan are available?

Several third-parties have expressed interest in implementing Dylan.

[a] Marlais is an experimental interpreter written in C. It was created by Brent Benson, and is being extended by Joe Wilson, Patrick Beard, and others. Marlais 0.4 is available in the Dylan ftp directory at `cambridge.apple.com`. It supports the older lisp-like syntax, and version 0.5 (coming soon) is expected to support the official algebraic syntax.

Marlais 0.4 runs on a variety of unix platforms. Marlais 0.3 has been ported to the Macintosh, and is available in the Dylan ftp directory as MacMarlais 0.31.

[b] Harlequin is currently investing in a commercial implementation of Dylan. They are building a PC based Windows (NT/Chicago) hosted version of the language. For more information, contact Mike Smith (mikes@harlequin.com), group manager of Harlequin's Dylan project.

[c] The Gwydion Project at Carnegie Mellon University is building an innovative new software development and maintenance environment around the Dylan language. A part of this work will involve the development of a high-performance Dylan implementation that will produce code for many platforms.

For more information on this project, see the Gwydion project's World Wide Web page: "`http://legend.gwydion.cs.cmu.edu:8001/gwydion/`".

[d] As a development tool, the Gwydion project has produced a byte-coded Dylan implementation called "Mindy". Though it is not very fast, does not have an extensive environment, and time spent on maintaining it will be limited, Mindy may be of some value to people who want to play with Dylan code and get a feel for the language. Mindy is documented on the Gwydion web page (see

above), and is available by anonymous ftp from `legend.gwydion.cs.cmu.edu` in the file:

    /afs/cs.cmu.edu/project/gwydion/release/mindy.tar.gz

Due to CMU security measures, if you use the ftp '`cd`' command, then you must '`cd`' to the release directory in one cd command line; for example, do not try to '`cd`' to "`/afs/cs/project/gwydion/`".

[e] Thomas 1.1 is an experimental interpreter written in Scheme. It was created by Matt Birkholz, Jim Miller, and Ron Weiss at the DEC Cambridge Research Lab in 1992. It supports the older lisp-like syntax, and does not incorporate many of the more recent language changes. Thomas is available at the Dylan ftp site at `cambridge.apple.com`.

[2-7] Is Dylan related to the Newton PDA?

Not at this time. If you are looking for more information on Newton development, you need to contact the Newton Developer Relations at `NEWTON.DEVS@applelink.apple.com`.

[2-8] Is Dylan designed to be a replacement for Common Lisp?

No. We see Common Lisp and Dylan filling two distinct needs.

[2-9] Is Dylan a proprietary language? Why is the Dylan name trademarked?

We want Dylan to be available on as many computers as possible. To this end, we are encouraging groups outside Apple to implement Dylan.

It is our intention to license the Dylan trademark to any implementation which passes a standard test suite. The purpose of the trademark is to ensure quality and consistency among implementations.

[3] Questions About the Dylan Language Design

[3-1] Is the Dylan language design frozen?

With the exception of the macro system, the current language design is complete. At this time, we will only be making small tweaks, or changes necessary to resolve bugs or inconsistencies in the specification.

[3-2] What does the Dylan syntax look like?

The first Dylan book published in 1992 used a parenthesized, lisp-like syntax. The current language definition uses an algebraic syntax, which will be very familiar to users of Pascal or C.

[3-3] Will Dylan have a standard macro system with the algebraic syntax?

Yes.

[3-4] Are there plans to specify a standard I/O package for Dylan? What about threads?

Threads, I/O, and other language features may be provided as optional libraries in the future. They are not currently part of the language. Because of the great variety of computing platforms—from PDA to mainframe—we do not want to standardize on these features too strictly.

[3-5] Why is '`make`' allowed to return a previously allocated instance, or an object which is an indirect instance of the class passed to make?

We feel that this is a very important abstraction mechanism. A class should have flexibility in how it implements `make`, as long as the object returned fulfills the protocol of the class.

For example, this allows a library to document a single abstract class which is supported by several undocumented implementation classes. The abstract class can choose which implementation class to instantiate based on the additional arguments to make. This allows optimizations which are transparent to the clients of the library.

The default method for make of a user-defined class returns a fresh direct instance of the requested class.

[3-6] The Dylan '92 manual doesn't require implementations to optimize tail-recursion. Was this an intentional omission, or an editorial oversight?

It was an editorial oversight. Dylan implementations will be required to be properly tail recursive.

[3-7] The Dylan '92 manual doesn't say much about modules. Will this be specified in the future?

The new language document includes a complete module specification.

[3-8] Can the '`method`' special form be used to create closures?

Yes.

[3-9] I don't understand how setter variables work. Is '`setter`' a special form?

This has been clarified in the new language design. The new design is included in the interim Dylan reference book.

[3-10] What kind of object is used to return multiple values?

When a function returns multiple values, the return values are not stored in a wrapper object; they are returned directly. For example, if a function returns "the values 4 and 5", it returns two integers. It does not return a data structure which contains two integers.

Returning multiple values is similar to calling a function with more than one argument. When passing multiple objects as arguments to a function, the objects do not have to be stored in a single data structure before they are passed.

[3-11] Is the specification of sealing complete?

Yes. The new language design includes a complete specification of sealing. This design is included in the interim Dylan reference book.

[3-12] Will Dylan include '`eval`'?

Some implementations may choose to support `eval`, but we do not have plans to add it to the language standard. We feel that the delivery of applications which are space efficient requires the separation of development time activity from runtime activity.

[3-13] Will Dylan include an application framework?

We recognize the value of application frameworks, especially cross-platform application frameworks. Unfortunately, because of the great variation in computing platforms, a single application framework will not be part of the Dylan language. On each platform, there should either be a Dylan-specific application framework, or Dylan should be able to use application frameworks written in other languages.

[3-14] Will Dylan interface to other languages?

We recognize that seamless integration with other languages, especially C and C++, is essential. We are working on addressing this issue. The solutions may not be part of the Dylan language proper.