Migrating to Move 2024

New features for Move are becoming available in 2024, a part of the aptly titled "Move 2024" edition. Many of these changes are enhancements to the source language, affecting the compiler without requiring any changes to the binary representation published on chain.

The primary goal of these changes is to make Move easier to write, and hopefully easier to read. The relatively few breaking changes introduced to the source language are to better position Move to handle future advancements.

Existing code will continue to compile, even with the addition of these new features. And because these features are opt-in, you can write your packages with the new features, even if your dependencies do not. Opting to take advantage of the new features in your current modules, however, does introduce some breaking changes.

This document highlights some new features to try out and shows how to migrate your existing modules to use Move 2024.

Please, provide any feedback or report any issues you encounter via GitHub , Discord , or Telegram .

To migrate a project to Move 2024 Beta:

Move 2024 includes an automatic migration script that you can use by calling sui move migrate in the root of your Move project. Upon running, your console prompts you for which Move edition to use. If you select 2024.beta , the script invokes the compiler and attempts to automatically update your code to avoid the breaking changes the update introduces (including marking structs as public , mutable variables with the mut keyword, avoiding restricted keywords, swapping friend s for public(package) , and even updating paths to global paths in many cases).

After this is done, your console displays a diff of the changes the script intends to make. If you accept the changes, the script updates your code and your Move.toml file automatically. You are now using Move 2024 Beta.

Use the new VSCode Move extension to get support for Move 2024 features. The new extension has a number of improvements over the original move-analyzer extension , but if you would like to keep using the original one, be sure to rebuild and reinstall the move-analyzer binary to get 2024 support:

See the getting started guide on Move IDEs and plugins for more information.

Here is a brief overview of some of the new features in Move 2024.

You can call certain functions now as methods using the . syntax. For example, the following call

can now be written as

Where the receiver of the method (v and c in this example) is automatically borrowed if necessary (as &mut v and &c respectively).

You can call any function defined in the same module as the receiver's type as a method if it takes the receiver as its first argument.

For functions defined outside the module, you can declare methods using public use fun and use fun .

With method syntax, you can annotate certain functions as being #[syntax(index)] methods. You then call these methods using v[i] -style calls.

For example,

resolves to

friend declarations, and the associated public(friend) visibility modifiers, are deprecated. In their place is the public(package) visibility modifier, which allows calling functions only within the same package where they are defined.

You can now define struct s with positional fields, which are accessed by zero-based index. For example,

then to access each field,

And as this example shows, you can now declare abilities after the struct field list.

You can now nest use aliases for more conciseness.

Additionally, the following use declarations are now automatically included in every module:

Equality operations, == and != , now automatically borrow if one side is a reference and the other is not. For example,

is equivalent to

This automatic borrowing can occur on either side of == and != .

When nesting loops, it can be convenient to break to the outer loop. For example,

Now, you can directly name the outer loop (outer in this case) and break it all at once:

It's now possible to break with a value from a loop . For example,

You can achieve this with labels, as well. For example,

Breaking changes are, unfortunately, a growing pain in Move 2024. We anticipate these changes to be minimally invasive and have provided a migration script to automate them in most cases. In addition, these changes pave the way for new features still to come in Move 2024.

Currently, all structs in Move are, by convention, public: any other module or package can import them and refer to them by type. To make this clearer, Move 2024 requires that all structs be declared with the public keyword. For example,

Any non-public struct produces an error at this time, though the Move team is working on new visibility options for future releases.

Previously, all variables in Move were implicitly mutable. For example,

Now, you must declare mutable variables explicitly:

The compiler now produces an error if you attempt to reassign or borrow a variable mutably without this explicit declaration.

Friends and the public(friend) visibilities were introduced early in Move's development, predating even the package system. As indicated in the public(package) section, public(package) deprecates public(friend) in Move 2024.

The following declaration now produces an error:

Instead, if you want your function to be visible only in the package, write:

Looking toward the future, Move 2024 Beta adds the following keywords to the language: enum , for , match , mut , and type . The compiler, unfortunately, now produces parsing errors when it finds these in other positions. This is a necessary change as the language matures. If you perform automatic migration, the migration tool renames these as enum and so on, rewriting the code to use these escaped forms.

Move 2024 revises how paths and namespaces work compared to legacy Move, toward easing enum aliasing in the future. Consider the following snippet from a test annotation in the sui_system library:

Legacy Move would always treat a three-part name as an address(sui_system), module(validator_set), and module member (EInvalidCap). Move 2024 respects scope for use , so sui_system in the attribute resolves to the module, producing a name resolution error overall.

To avoid cases where this is the intended behavior, Move 2024 introduces a prefix operation for global qualification. To use, you can rewrite this annotation as:

The migration script attempts to remediate naming errors using global qualification when possible.

The beta release of Move 2024 comes with some powerful new features in addition to the breaking changes described here. There are also more on the horizon. Join the Sui developer newsletter to learn about new, exciting features coming to Move this year, including syntactic macros, enums with pattern matching, and other user-defined syntax extensions.

How to migrate

To migrate a project to Move 2024 Beta:

Move 2024 includes an automatic migration script that you can use by calling sui move migrate in the root of your Move project. Upon running, your console prompts you for which Move edition to use. If you select 2024.beta , the script invokes the compiler and attempts to automatically update your code to avoid the breaking changes the update introduces (including marking structs as public , mutable variables with the mut keyword, avoiding restricted keywords, swapping friend s for public(package) , and even updating paths to global paths in many cases).

After this is done, your console displays a diff of the changes the script intends to make. If you accept the changes, the script updates your code and your Move.toml file automatically. You are now using Move 2024 Beta.

Use the new VSCode Move extension to get support for Move 2024 features. The new extension has a number of improvements over the original move-analyzer extension , but if you would like to keep using the original one, be sure to rebuild and reinstall the move-analyzer binary to get 2024 support:

See the getting started guide on Move IDEs and plugins for more information.

Here is a brief overview of some of the new features in Move 2024.

You can call certain functions now as methods using the . syntax. For example, the following call

can now be written as

Where the receiver of the method (v and c in this example) is automatically borrowed if necessary (as &mut v and &c respectively).

You can call any function defined in the same module as the receiver's type as a method if it takes the receiver as its first argument.

For functions defined outside the module, you can declare methods using public use fun and use fun .

With method syntax, you can annotate certain functions as being #[syntax(index)] methods. You then call these methods using v[i] -style calls.

For example,

resolves to

friend declarations, and the associated public(friend) visibility modifiers, are deprecated. In their place is the public(package) visibility modifier, which allows calling functions only within the same package where they are defined.

You can now define struct s with positional fields, which are accessed by zero-based index. For example,

then to access each field,

And as this example shows, you can now declare abilities after the struct field list.

You can now nest use aliases for more conciseness.

Additionally, the following use declarations are now automatically included in every module:

Equality operations, == and != , now automatically borrow if one side is a reference and the other is not. For example,

is equivalent to

This automatic borrowing can occur on either side of == and != .

When nesting loops, it can be convenient to break to the outer loop. For example,

Now, you can directly name the outer loop (outer in this case) and break it all at once:

It's now possible to break with a value from a loop . For example,

You can achieve this with labels, as well. For example,

Breaking changes are, unfortunately, a growing pain in Move 2024. We anticipate these changes to be minimally invasive and have provided a migration script to automate them in most cases. In addition, these changes pave the way for new features still to come in Move 2024.

Currently, all structs in Move are, by convention, public: any other module or package can import them and refer to them by type. To make this clearer, Move 2024 requires that all structs be declared with the public keyword. For example,

Any non-public struct produces an error at this time, though the Move team is working on new visibility options for future releases.

Previously, all variables in Move were implicitly mutable. For example, Now, you must declare mutable variables explicitly:

The compiler now produces an error if you attempt to reassign or borrow a variable mutably without this explicit declaration.

Friends and the public(friend) visibilities were introduced early in Move's development, predating even the package system. As indicated in the public(package) section, public(package) deprecates public(friend) in Move 2024.

The following declaration now produces an error:

Instead, if you want your function to be visible only in the package, write:

Looking toward the future, Move 2024 Beta adds the following keywords to the language: enum , for , match , mut , and type . The compiler, unfortunately, now produces parsing errors when it finds these in other positions. This is a necessary change as the language matures. If you perform automatic migration, the migration tool renames these as enum and so on, rewriting the code to use these escaped forms.

Move 2024 revises how paths and namespaces work compared to legacy Move, toward easing enum aliasing in the future. Consider the following snippet from a test annotation in the sui_system library:

Legacy Move would always treat a three-part name as an address(sui_system), module(validator_set), and module member (EInvalidCap). Move 2024 respects scope for use , so sui_system in the attribute resolves to the module, producing a name resolution error overall.

To avoid cases where this is the intended behavior, Move 2024 introduces a prefix operation for global qualification. To use, you can rewrite this annotation as:

The migration script attempts to remediate naming errors using global qualification when possible.

The beta release of Move 2024 comes with some powerful new features in addition to the breaking changes described here. There are also more on the horizon. Join the Sui developer newsletter to learn about new, exciting features coming to Move this year, including syntactic macros, enums with pattern matching, and other user-defined syntax extensions.

New features

Here is a brief overview of some of the new features in Move 2024.

You can call certain functions now as methods using the . syntax. For example, the following call

can now be written as

Where the receiver of the method (v and c in this example) is automatically borrowed if necessary (as &mut v and &c respectively).

You can call any function defined in the same module as the receiver's type as a method if it takes the receiver as its first argument.

For functions defined outside the module, you can declare methods using public use fun and use fun .

With method syntax, you can annotate certain functions as being #[syntax(index)] methods. You then call these methods using v[i] -style calls.

For example,

resolves to

friend declarations, and the associated public(friend) visibility modifiers, are deprecated. In their place is the public(package) visibility modifier, which allows calling functions only within the same package where they are defined.

You can now define struct s with positional fields, which are accessed by zero-based index. For example,

then to access each field,

And as this example shows, you can now declare abilities after the struct field list.

You can now nest use aliases for more conciseness.

Additionally, the following use declarations are now automatically included in every module:

Equality operations, == and != , now automatically borrow if one side is a reference and the other is not. For example,

is equivalent to

This automatic borrowing can occur on either side of == and != .

When nesting loops, it can be convenient to break to the outer loop. For example,

Now, you can directly name the outer loop (outer in this case) and break it all at once:

It's now possible to break with a value from a loop . For example,

You can achieve this with labels, as well. For example,

Breaking changes are, unfortunately, a growing pain in Move 2024. We anticipate these changes to be minimally invasive and have provided a migration script to automate them in most cases. In addition, these changes pave the way for new features still to come in Move 2024.

Currently, all structs in Move are, by convention, public: any other module or package can import them and refer to them by type. To make this clearer, Move 2024 requires that all structs be declared with the public keyword. For example,

Any non-public struct produces an error at this time, though the Move team is working on new visibility options for future releases.

Previously, all variables in Move were implicitly mutable. For example,

Now, you must declare mutable variables explicitly:

The compiler now produces an error if you attempt to reassign or borrow a variable mutably without this explicit declaration.

Friends and the public(friend) visibilities were introduced early in Move's development, predating even the package system. As indicated in the public(package) section, public(package) deprecates public(friend) in Move 2024.

The following declaration now produces an error:

Instead, if you want your function to be visible only in the package, write:

Looking toward the future, Move 2024 Beta adds the following keywords to the language: enum , for , match , mut , and type . The compiler, unfortunately, now produces parsing errors when it finds these in other positions. This is a necessary change as the language matures. If you perform automatic migration, the migration tool renames these as enum and so on, rewriting the code to use these escaped forms.

Move 2024 revises how paths and namespaces work compared to legacy Move, toward easing enum aliasing in the future. Consider the following snippet from a test annotation in the sui_system library:

Legacy Move would always treat a three-part name as an address(sui_system), module(validator_set), and module member (EInvalidCap). Move 2024 respects scope for use , so sui_system in the attribute resolves to the module, producing a name resolution error overall.

To avoid cases where this is the intended behavior, Move 2024 introduces a prefix operation for global qualification. To use, you can rewrite this annotation as:

The migration script attempts to remediate naming errors using global qualification when possible.

The beta release of Move 2024 comes with some powerful new features in addition to the breaking changes described here. There are also more on the horizon. Join the Sui developer newsletter to learn about new, exciting features coming to Move this year, including syntactic macros, enums with pattern matching, and other user-defined syntax extensions.

Breaking changes

Breaking changes are, unfortunately, a growing pain in Move 2024. We anticipate these changes to be minimally invasive and have provided a migration script to automate them in most cases. In addition, these changes pave the way for new features still to come in Move 2024.

Currently, all structs in Move are, by convention, public: any other module or package can import them and refer to them by type. To make this clearer, Move 2024 requires that all structs be declared with the public keyword. For example,

Any non-public struct produces an error at this time, though the Move team is working on new visibility options for future releases.

Previously, all variables in Move were implicitly mutable. For example,

Now, you must declare mutable variables explicitly:

The compiler now produces an error if you attempt to reassign or borrow a variable mutably without this explicit declaration.

Friends and the public(friend) visibilities were introduced early in Move's development, predating even the package system. As indicated in the public(package) section, public(package) deprecates public(friend) in Move 2024.

The following declaration now produces an error:

Instead, if you want your function to be visible only in the package, write:

Looking toward the future, Move 2024 Beta adds the following keywords to the language: enum , for , match , mut , and type . The compiler, unfortunately, now produces parsing errors when it finds these in other positions. This is a necessary change as the language matures. If you perform automatic migration, the migration tool renames these as enum and so on, rewriting the code to use these escaped forms.

Move 2024 revises how paths and namespaces work compared to legacy Move, toward easing enum aliasing in the future. Consider the following snippet from a test annotation in the sui_system library:

Legacy Move would always treat a three-part name as an address(sui_system), module(validator_set), and module member (EInvalidCap). Move 2024 respects scope for use , so sui_system in the attribute resolves to the module, producing a name resolution error overall.

To avoid cases where this is the intended behavior, Move 2024 introduces a prefix operation for global qualification. To use, you can rewrite this annotation as:

The migration script attempts to remediate naming errors using global qualification when possible.

The beta release of Move 2024 comes with some powerful new features in addition to the breaking changes described here. There are also more on the horizon. Join the Sui developer newsletter to learn about new, exciting features coming to Move this year, including syntactic macros, enums with pattern matching, and other user-defined syntax extensions.

Follow along

The beta release of Move 2024 comes with some powerful new features in addition to the breaking changes described here. There are also more on the horizon. Join the Sui developer newsletter to learn about new, exciting features coming to Move this year, including syntactic macros, enums with pattern matching, and other user-defined syntax extensions.