Automated Address Management

When you publish or upgrade a package, its address (also known as the package ID) is tracked in the Move.lock file. This bookkeeping is done automatically so that you can avoid recording or updating hex addresses (for example, in the Move.toml file).

When you publish or upgrade your package on multiple chains (Mainnet, Testnet, Devnet) separate addresses for each chain are tracked for you. This tracking is based on your active environment (run sui client active-env if unsure). For example, if you set your active environment to an RPC connected to testnet and publish a package, the Move.lock records the address for that package and associates it with your active environment (testnet).

Note that automated address management works for one package published to one or more chains. When you publish or upgrade, address management uses the contents of the package's working directory. If a package is republished to a chain, addresses tracked for the previously published package are overwritten for that chain.

Previously a published-at entry was mandatory in the Move.toml file, which sets the address of the latest published package. It is no longer required if this data is tracked in the Move.lock file. For an existing package, add the necessary data to the Move.lock file so that it can be automatically tracked:

Your package is now tracked. You can repeat the above steps to track addresses for additional environments.

When upgrading , you need to retrieve the UpgradeCap ID of your published package. Automated address management does not track your UpgradeCap .

When upgrading , you first need to set the [addresses] value for your package to 0x0 in the Move.toml , and restore its ID with the ORIGINAL-ADDRESS after upgrading.

Conflicting published package addresses might happen when the state of package data is inconsistent.

For example, you might have an existing package with a published-at value in the Move.toml . The package is re-published for testing purposes, and is now tracked in Move.lock with automated address management. The address in the Move.toml and Move.lock now differ, and there will be an error the next time you try to publish or upgrade the package.

Here are steps to remediate conflicting addresses:

If the conflict is in a package you maintain:

If the conflict is in a package that you do not maintain (such as a dependency), consider fixing the issue locally with the steps above, or contacting the maintainer to upstream changes.

This section is an overview of the schema and internal operation of tracked address in the Move.lock file. Most developers do not need to understand these internals. It is a reference for those who want a complete interface or control to internal state tracking.

A concrete example of the schema and state you might find in a Move.lock file:

An [env] table contains entries for each environment. When a package is published for an active environment, an entry is added or updated.

An entry is added or updated based on its the chain-id of the environment. As in, addresses are keyed by chain-id , not the [env.NAME] part. This is so that packages and their dependencies are resolved canonically by chain identifier: keying by [env.NAME] is not robust when users can choose arbitrary environment NAME aliases.

Key-value entries correspond to data explained in adopting automated address management . Use the sui move manage-package command as a frontend to manipulate these values.

Adopting automated address management for published packages

Previously a published-at entry was mandatory in the Move.toml file, which sets the address of the latest published package. It is no longer required if this data is tracked in the Move.lock file. For an existing package, add the necessary data to the Move.lock file so that it can be automatically tracked:

Your package is now tracked. You can repeat the above steps to track addresses for additional environments.

When upgrading , you need to retrieve the UpgradeCap ID of your published package. Automated address management does not

track your UpgradeCap .

When upgrading , you first need to set the [addresses] value for your package to 0x0 in the Move.toml , and restore its ID with the ORIGINAL-ADDRESS after upgrading.

Conflicting published package addresses might happen when the state of package data is inconsistent.

For example, you might have an existing package with a published-at value in the Move.toml . The package is re-published for testing purposes, and is now tracked in Move.lock with automated address management. The address in the Move.toml and Move.lock now differ, and there will be an error the next time you try to publish or upgrade the package.

Here are steps to remediate conflicting addresses:

If the conflict is in a package you maintain:

If the conflict is in a package that you do not maintain (such as a dependency), consider fixing the issue locally with the steps above, or contacting the maintainer to upstream changes.

This section is an overview of the schema and internal operation of tracked address in the Move.lock file. Most developers do not need to understand these internals. It is a reference for those who want a complete interface or control to internal state tracking.

A concrete example of the schema and state you might find in a Move.lock file:

An [env] table contains entries for each environment. When a package is published for an active environment, an entry is added or updated.

An entry is added or updated based on its the chain-id of the environment. As in, addresses are keyed by chain-id , not the [env.NAME] part. This is so that packages and their dependencies are resolved canonically by chain identifier: keying by [env.NAME] is not robust when users can choose arbitrary environment NAME aliases.

Key-value entries correspond to data explained in adopting automated address management . Use the sui move manage-package command as a frontend to manipulate these values.

Package upgrade guidance

When upgrading , you need to retrieve the UpgradeCap ID of your published package. Automated address management does not track your UpgradeCap .

When upgrading , you first need to set the [addresses] value for your package to 0x0 in the Move.toml , and restore its ID with the ORIGINAL-ADDRESS after upgrading.

Conflicting published package addresses might happen when the state of package data is inconsistent.

For example, you might have an existing package with a published-at value in the Move.toml . The package is re-published for testing purposes, and is now tracked in Move.lock with automated address management. The address in the Move.toml and Move.lock now differ, and there will be an error the next time you try to publish or upgrade the package.

Here are steps to remediate conflicting addresses:

If the conflict is in a package you maintain:

If the conflict is in a package that you do not maintain (such as a dependency), consider fixing the issue locally with the steps above, or contacting the maintainer to upstream changes.

This section is an overview of the schema and internal operation of tracked address in the Move.lock file. Most developers do not need to understand these internals. It is a reference for those who want a complete interface or control to internal state tracking.

A concrete example of the schema and state you might find in a Move.lock file:

An [env] table contains entries for each environment. When a package is published for an active environment, an entry is added or updated.

An entry is added or updated based on its the chain-id of the environment. As in, addresses are keyed by chain-id , not the [env.NAME] part. This is so that packages and their dependencies are resolved canonically by chain identifier: keying by [env.NAME] is not robust when users can choose arbitrary environment NAME aliases.

Key-value entries correspond to data explained in adopting automated address management . Use the sui move manage-package command as a frontend to manipulate these values.

Troubleshooting

Conflicting published package addresses might happen when the state of package data is inconsistent.

For example, you might have an existing package with a published-at value in the Move.toml . The package is re-published for testing purposes, and is now tracked in Move.lock with automated address management. The address in the Move.toml and Move.lock now differ, and there will be an error the next time you try to publish or upgrade the package.

Here are steps to remediate conflicting addresses:

If the conflict is in a package you maintain:

If the conflict is in a package that you do not maintain (such as a dependency), consider fixing the issue locally with the steps above, or contacting the maintainer to upstream changes.

This section is an overview of the schema and internal operation of tracked address in the Move.lock file. Most developers do not need to understand these internals. It is a reference for those who want a complete interface or control to internal state tracking.

A concrete example of the schema and state you might find in a Move.lock file:

An [env] table contains entries for each environment. When a package is published for an active environment, an entry is added or updated.

An entry is added or updated based on its the chain-id of the environment. As in, addresses are keyed by chain-id , not the [env.NAME] part. This is so that packages and their dependencies are resolved canonically by chain identifier: keying by [env.NAME] is not robust when users can choose arbitrary environment NAME aliases.

Key-value entries correspond to data explained in adopting automated address management . Use the sui move manage-package command as a frontend to manipulate these values.

Internal reference

This section is an overview of the schema and internal operation of tracked address in the Move.lock file. Most developers do not need to understand these internals. It is a reference for those who want a complete interface or control to internal state tracking.

A concrete example of the schema and state you might find in a Move.lock file:

An [env] table contains entries for each environment. When a package is published for an active environment, an entry is added or updated.

An entry is added or updated based on its the chain-id of the environment. As in, addresses are keyed by chain-id , not the [env.NAME] part. This is so that packages and their dependencies are resolved canonically by chain identifier: keying by [env.NAME] is not robust when users can choose arbitrary environment NAME aliases.

Key-value entries correspond to data explained in adopting automated address management . Use the sui move manage-package command as a frontend to manipulate these values.