Module sui::package

Functions for operating on Move packages from within Move:

Creating proof-of-publish objects from one-time witnesses

Administering package upgrades through upgrade policies.

This type can only be created in the transaction that generates a module, by consuming its one-time witness, so it can be used to identify the address that published the package a type originated from.

Capability controlling the ability to upgrade a package.

Permission to perform a particular upgrade (for a fixed version of the package, bytecode to upgrade with and transitive dependencies to depend against).

An UpgradeCap can only issue one ticket at a time, to prevent races between concurrent updates or a change in its upgrade policy after issuing a ticket, so the ticket is a "Hot Potato" to preserve forward progress.

Issued as a result of a successful upgrade, containing the information to be used to update the UpgradeCap . This is a "Hot Potato" to ensure that it is used to update its UpgradeCap before the end of the transaction that performed the upgrade.

Tried to create a Publisher using a type that isn't a one-time witness.

Tried to set a less restrictive policy than currently in place.

This UpgradeCap has already authorized a pending upgrade.

This UpgradeCap has not authorized an upgrade.

Trying to commit an upgrade to the wrong UpgradeCap .

Update any part of the package (function implementations, add new functions or types, change dependencies)

Add new functions or types, or change dependencies, existing functions can't change.

Only be able to change dependencies.

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

Destroy a Publisher object effectively removing all privileges associated with it.

Check whether type belongs to the same package as the publisher object.

Check whether a type belongs to the same module as the publisher object.

Read the name of the module.

Read the package address string.

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

The most recent version of the package, increments by one for each successfully applied upgrade.

The most permissive kind of upgrade currently supported by this cap.

The package that this ticket is authorized to upgrade

The kind of upgrade that this ticket authorizes.

ID of the UpgradeCap that this receipt should be used to update.

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

Expose the constants representing various upgrade policies

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

Restrict upgrades through this upgrade cap to just change dependencies.

Discard the UpgradeCap to make a package immutable.

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

Struct

This type can only be created in the transaction that generates a module, by consuming its one-time witness, so it can be used to identify the address that published the package a type originated from.

```bash
```

Capability controlling the ability to upgrade a package.

```bash
```

Permission to perform a particular upgrade (for a fixed version of the package, bytecode to upgrade with and transitive dependencies to depend against).

An UpgradeCap can only issue one ticket at a time, to prevent races between concurrent updates or a change in its upgrade policy after issuing a ticket, so the ticket is a "Hot Potato" to preserve forward progress.

```bash
```

Issued as a result of a successful upgrade, containing the information to be used to update the UpgradeCap . This is a "Hot Potato" to ensure that it is used to update its UpgradeCap before the end of the transaction that performed the upgrade.

```bash
```

Tried to create a Publisher using a type that isn't a one-time witness.

```bash
```

Tried to set a less restrictive policy than currently in place.

```bash
```

This UpgradeCap has already authorized a pending upgrade.

```bash
```

This UpgradeCap has not authorized an upgrade.

```bash

```

Trying to commit an upgrade to the wrong UpgradeCap .

```bash

```

Update any part of the package (function implementations, add new functions or types, change dependencies)

```bash

```

Add new functions or types, or change dependencies, existing functions can't change.

```bash

```

Only be able to change dependencies.

```bash

```

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash

```

```bash

```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash

```

```bash

```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash

```

```bash

```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash

```

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash

```

```bash

```

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash
```

```bash

```

```bash

```

Struct

Capability controlling the ability to upgrade a package.

```bash

```

Permission to perform a particular upgrade (for a fixed version of the package, bytecode to upgrade with and transitive dependencies to depend against).

An UpgradeCap can only issue one ticket at a time, to prevent races between concurrent updates or a change in its upgrade policy after issuing a ticket, so the ticket is a "Hot Potato" to preserve forward progress.

```bash

```

Issued as a result of a successful upgrade, containing the information to be used to update the UpgradeCap . This is a "Hot Potato" to ensure that it is used to update its UpgradeCap before the end of the transaction that performed the upgrade.

```bash

```

Tried to create a Publisher using a type that isn't a one-time witness.

```bash

```

Tried to set a less restrictive policy than currently in place.

```bash

```

This UpgradeCap has already authorized a pending upgrade.

```bash

```

This UpgradeCap has not authorized an upgrade.

```bash

```

Trying to commit an upgrade to the wrong UpgradeCap .

```bash

```

Update any part of the package (function implementations, add new functions or types, change dependencies)

```bash
```

Add new functions or types, or change dependencies, existing functions can't change.

```bash
```

Only be able to change dependencies.

```bash
```

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash
```

```bash
```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash
```

```bash
```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash
```

```bash
```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash

```

ID of the UpgradeCap that this receipt should be used to update.

```bash

```

```bash

```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash

```

```bash

```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash

```

```bash

```

Expose the constants representing various upgrade policies

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Struct

Permission to perform a particular upgrade (for a fixed version of the package, bytecode to upgrade with and transitive dependencies to depend against).

An UpgradeCap can only issue one ticket at a time, to prevent races between concurrent updates or a change in its upgrade policy after issuing a ticket, so the ticket is a "Hot Potato" to preserve forward progress.

```bash

```

Issued as a result of a successful upgrade, containing the information to be used to update the UpgradeCap . This is a "Hot Potato" to ensure that it is used to update its UpgradeCap before the end of the transaction that performed the upgrade.

```bash

```

Tried to create a Publisher using a type that isn't a one-time witness.

```bash

```

Tried to set a less restrictive policy than currently in place.

```bash

```

This UpgradeCap has already authorized a pending upgrade.

```bash

```

This UpgradeCap has not authorized an upgrade.

```bash

```

Trying to commit an upgrade to the wrong UpgradeCap .

```bash

```

Update any part of the package (function implementations, add new functions or types, change dependencies)

```bash

```

Add new functions or types, or change dependencies, existing functions can't change.

```bash

```

Only be able to change dependencies.

```bash

```

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash
```

```bash

```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash

```

```bash

```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash

```

```bash

```

Check whether type belongs to the same package as the publisher object.

```bash

```

```bash

```

Check whether a type belongs to the same module as the publisher object.

```bash

```

```bash

```

Read the name of the module.

```bash

```

```bash

```

Read the package address string.

```bash

```

```bash

```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Struct

Issued as a result of a successful upgrade, containing the information to be used to update the UpgradeCap . This is a "Hot Potato" to ensure that it is used to update its UpgradeCap before the end of the transaction that performed the upgrade.

```bash
```

Tried to create a Publisher using a type that isn't a one-time witness.

```bash
```

Tried to set a less restrictive policy than currently in place.

```bash
```

This UpgradeCap has already authorized a pending upgrade.

```bash
```

This UpgradeCap has not authorized an upgrade.

```bash

```

Trying to commit an upgrade to the wrong UpgradeCap .

```bash

```

Update any part of the package (function implementations, add new functions or types, change dependencies)

```bash

```

Add new functions or types, or change dependencies, existing functions can't change.

```bash

```

Only be able to change dependencies.

```bash

```

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash

```

```bash

```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash

```

```bash

```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash

```

```bash

```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash

```

Check whether a type belongs to the same module as the publisher object.

```bash

```

```bash

```

Read the name of the module.

```bash

```

```bash

```

Read the package address string.

```bash

```

```bash

```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash

```

```bash

```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash

```

```bash

```

The most permissive kind of upgrade currently supported by this cap.

```bash

```

```bash

```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash

```

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash

```

```bash

```

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash
```

```bash

```

```bash

```

Constants

Tried to create a Publisher using a type that isn't a one-time witness.

```bash

```

Tried to set a less restrictive policy than currently in place.

```bash

```

This UpgradeCap has already authorized a pending upgrade.

```bash

```

This UpgradeCap has not authorized an upgrade.

```bash

```

Trying to commit an upgrade to the wrong UpgradeCap .

```bash

```

Update any part of the package (function implementations, add new functions or types, change dependencies)

```bash

```

Add new functions or types, or change dependencies, existing functions can't change.

```bash

```

Only be able to change dependencies.

```bash

```

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash

```

```bash
```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash
```

```bash
```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash
```

```bash
```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Claim a Publisher object. Requires a One-Time-Witness to prove ownership. Due to this constraint there can be only one Publisher object per module but multiple per package (!).

```bash
```

```bash
```

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash
```

```bash
```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash
```

```bash
```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Claim a Publisher object and send it to transaction sender. Since this function can only be called in the module initializer, the sender is the publisher.

```bash
```

```bash
```

Destroy a Publisher object effectively removing all privileges associated with it.

```bash
```

```bash
```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash

```

```bash

```

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

Destroy a Publisher object effectively removing all privileges associated with it.

```bash

```

```bash

```

Check whether type belongs to the same package as the publisher object.

```bash
```

```bash
```

Check whether a type belongs to the same module as the publisher object.

```bash
```

```bash
```

Read the name of the module.

```bash
```

```bash
```

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash

```

```bash

```

```bash

```

Function

Check whether type belongs to the same package as the publisher object.

```bash

```

```bash

```

Check whether a type belongs to the same module as the publisher object.

```bash

```

```bash

```

Read the name of the module.

```bash

```

```bash

```

Read the package address string.

```bash

```

```bash

```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash

```

```bash

```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

Check whether a type belongs to the same module as the publisher object.

```bash

```

```bash

```

Read the name of the module.

```bash

```

```bash

```

Read the package address string.

```bash

```

```bash

```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

Read the name of the module.

```bash

```

```bash

```

Read the package address string.

```bash

```

```bash

```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash

```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Read the package address string.

```bash
```

```bash
```

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash

```

```bash

```

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

The ID of the package that this cap authorizes upgrades for. Can be 0x0 if the cap cannot currently authorize an upgrade because there is already a pending upgrade in the transaction. Otherwise guaranteed to be the latest version of any given package.

```bash
```

```bash
```

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

The most recent version of the package, increments by one for each successfully applied upgrade.

```bash
```

```bash
```

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

The most permissive kind of upgrade currently supported by this cap.

```bash
```

```bash
```

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

The package that this ticket is authorized to upgrade

```bash
```

```bash
```

The kind of upgrade that this ticket authorizes.

```bash
```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash

```

```bash

```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash

```

```bash

```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash

```

```bash

```

Expose the constants representing various upgrade policies

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

```bash

```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash

```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash

```

```bash

```

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

The kind of upgrade that this ticket authorizes.

```bash

```

```bash
```

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

ID of the UpgradeCap that this receipt should be used to update.

```bash
```

```bash
```

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

ID of the package that was upgraded to: the latest version of the package, as of the upgrade represented by this receipt.

```bash
```

```bash
```

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash
```

```bash
```

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

A hash of the package contents for the new version of the package. This ticket only authorizes an upgrade to a package that matches this digest. A package's contents are identified by two things:

A package's digest is calculated as:

sha3_256(sort(modules ++ deps))

```bash

```

```bash

```

Expose the constants representing various upgrade policies

```bash

```

```bash

```

```bash

```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Expose the constants representing various upgrade policies

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

```bash
```

```bash
```

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Restrict upgrades through this upgrade cap to just add code, or change dependencies.

```bash
```

```bash
```

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash
```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Restrict upgrades through this upgrade cap to just change dependencies.

```bash
```

```bash
```

Discard the UpgradeCap to make a package immutable.

```bash
```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash

```

```bash

```

```bash

```

```bash

```

Function

Discard the UpgradeCap to make a package immutable.

```bash

```

```bash

```

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash

```

```bash

```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Issue a ticket authorizing an upgrade to a particular new bytecode (identified by its digest). A ticket will only be issued if one has not already been issued, and if the policy requested is at least as restrictive as the policy set out by the cap.

The digest supplied and the policy will both be checked by validators when running the upgrade. I.e. the bytecode supplied in the upgrade must have a matching digest, and the changes relative to the parent package must be compatible with the policy in the ticket for the upgrade to succeed.

```bash
```

```bash
```

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash
```

Function

Consume an UpgradeReceipt to update its UpgradeCap , finalizing the upgrade.

```bash
```

```bash
```

```bash
```

```bash

```

Function

```bash

```

```bash

```