Module sui::kiosk

Kiosk is a primitive for building safe, decentralized and trustless trading experiences. It allows storing and trading any types of assets as long as the creator of these assets implements a TransferPolicy for them.

Kiosk provides guarantees of "true ownership"; - just like single owner objects, assets stored in the Kiosk can only be managed by the Kiosk owner. Only the owner can place , take , list , perform any other actions on assets in the Kiosk.

Kiosk aims to be generic - allowing for a small set of default behaviors and not imposing any restrictions on how the assets can be traded. The only default scenario is a list + purchase flow; any other trading logic can be implemented on top using the list_with_purchase_cap (and a matching purchase_with_cap) flow.

For every transaction happening with a third party a TransferRequest is created - this way creators are fully in control of the trading experience.

placed - An asset is place d into the Kiosk and can be take n out by the Kiosk owner; it's freely tradable and modifiable via the borrow_mut and borrow_val functions.

locked - Similar to placed except that take is disabled and the only way to move the asset out of the Kiosk is to list it or list_with_purchase_cap therefore performing a trade (issuing a TransferRequest). The check on the lock function makes sure that the TransferPolicy exists to not lock the item in a Kiosk forever.

listed - A place d or a lock ed item can be list ed for a fixed price allowing anyone to purchase it from the Kiosk. While listed, an item can not be taken or modified. However, an immutable borrow via borrow call is still available. The delist function returns the asset to the previous state.

listed_exclusively - An item is listed via the list_with_purchase_cap function (and a PurchaseCap is created). While listed this way, an item can not be delist -ed unless a PurchaseCap is returned. All actions available at this item state require a PurchaseCap :

When an item is listed exclusively it cannot be modified nor taken and losing a PurchaseCap would lock the item in the Kiosk forever. Therefore, it is recommended to only use PurchaseCap functionality in trusted applications and not use it for direct trading (eg sending to another account).

Every purchase or purchase_with_purchase_cap creates a TransferRequest hot potato which must be resolved in a matching TransferPolicy for the transaction to pass. While the default scenario implies that there should be a single TransferPolicy for T; it is possible to have multiple, each one having its own set of rules.

See transfer_policy module for more details on how they function.

An object which allows selling collectibles within "kiosk" ecosystem. By default gives the functionality to list an item openly - for anyone to purchase providing the guarantees for creators that every transfer needs to be approved via the TransferPolicy.

A Capability granting the bearer a right to place and take items from the Kiosk as well as to list them and list_with_purchase_cap .

A capability which locks an item and gives a permission to purchase it from a Kiosk for any price no less than min_price.

Allows exclusive listing: only bearer of the PurchaseCap can purchase the asset. However, the capability should be used carefully as losing it would lock the asset in the Kiosk .

The main application for the PurchaseCap is building extensions on top of the Kiosk .

Hot potato to ensure an item was returned after being taken using the borrow_val call.

Dynamic field key for an item placed into the kiosk.

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

Trying to withdraw profits and sender is not owner.

Coin paid does not match the offer price.

Trying to withdraw higher amount than stored.

Trying to close a Kiosk and it has items in it.

Attempt to take an item that has a PurchaseCap issued.

PurchaseCap does not match the Kiosk .

Trying to exclusively list an already listed item.

Trying to call uid_mut when allow_extensions set to false.

Attempt to take an item that is locked.

Taking or mutably borrowing an item that is listed.

Item does not match Borrow in return_val .

An is not found while trying to borrow.

Delisting an item that is not listed.

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

Creates a new Kiosk with a matching KioskOwnerCap .

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

Calls place and list together - simplifies the flow.

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

Withdraw profits from the Kiosk.

Internal: "lock" an item disabling the take action.

Internal: "place" an item to the Kiosk and increment the item count.

Internal: get a mutable access to the UID.

Check whether the item is present in the Kiosk .

Check whether the item is present in the Kiosk and has type T.

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

Check whether an item is listed (exclusively or non exclusively).

Check whether there's a PurchaseCap issued for an item.

Check whether the KioskOwnerCap matches the Kiosk .

Access the UID using the KioskOwnerCap .

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

Get the owner of the Kiosk.

Get the number of items stored in a Kiosk.

Get the amount of profits collected by selling items.

Get mutable access to profits - owner only action.

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

Get the for field of the KioskOwnerCap .

Get the kiosk_id from the PurchaseCap .

Get the Item_id from the PurchaseCap .

Get the min_price from the PurchaseCap .

Struct

An object which allows selling collectibles within "kiosk" ecosystem. By default gives the functionality to list an item openly - for anyone to purchase providing the guarantees for creators that every transfer needs to be approved via the TransferPolicy.

```bash
```

A Capability granting the bearer a right to place and take items from the Kiosk as well as to list them and list_with_purchase_cap .

```bash
```

A capability which locks an item and gives a permission to purchase it from a Kiosk for any price no less than min_price.

Allows exclusive listing: only bearer of the PurchaseCap can purchase the asset. However, the capability should be used carefully as losing it would lock the asset in the Kiosk .

The main application for the PurchaseCap is building extensions on top of the Kiosk .

```bash
```

Hot potato to ensure an item was returned after being taken using the borrow_val call.

```bash
```

Dynamic field key for an item placed into the kiosk.

```bash
```

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash
```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash
```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash
```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash
```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash
```

Trying to withdraw profits and sender is not owner.

```bash
```

Coin paid does not match the offer price.

```bash
```

Trying to withdraw higher amount than stored.

```bash
```

Trying to close a Kiosk and it has items in it.

```bash
```

Attempt to take an item that has a PurchaseCap issued.

```bash
```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash

```

Delisting an item that is not listed.

```bash

```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash

```

```bash

```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash

```

```bash

```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash

```

```bash

```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

A Capability granting the bearer a right to place and take items from the Kiosk as well as to list them and list_with_purchase_cap .

```bash

```

A capability which locks an item and gives a permission to purchase it from a Kiosk for any price no less than min_price.

Allows exclusive listing: only bearer of the PurchaseCap can purchase the asset. However, the capability should be used carefully as losing it would lock the asset in the Kiosk .

The main application for the PurchaseCap is building extensions on top of the Kiosk .

```bash

```

Hot potato to ensure an item was returned after being taken using the borrow_val call.

```bash

```

Dynamic field key for an item placed into the kiosk.

```bash

```

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash

```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash

```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash

```

Trying to exclusively list an already listed item.

```bash

```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash
```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Struct

A capability which locks an item and gives a permission to purchase it from a Kiosk for any price no less than min_price.

Allows exclusive listing: only bearer of the PurchaseCap can purchase the asset. However, the capability should be used carefully as losing it would lock the asset in the Kiosk .

The main application for the PurchaseCap is building extensions on top of the Kiosk .

```bash
```

Hot potato to ensure an item was returned after being taken using the borrow_val call.

```bash
```

Dynamic field key for an item placed into the kiosk.

```bash
```

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash
```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash
```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash
```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash
```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash
```

Trying to withdraw profits and sender is not owner.

```bash
```

Coin paid does not match the offer price.

```bash
```

Trying to withdraw higher amount than stored.

```bash
```

Trying to close a Kiosk and it has items in it.

```bash
```

Attempt to take an item that has a PurchaseCap issued.

```bash
```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash

```

Trying to call uid_mut when allow_extensions set to false.

```bash

```

Attempt to take an item that is locked.

```bash

```

Taking or mutably borrowing an item that is listed.

```bash

```

Item does not match Borrow in return_val .

```bash

```

An is not found while trying to borrow.

```bash

```

Delisting an item that is not listed.

```bash

```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash

```

```bash

```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash

```

```bash

```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash

```

```bash

```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

Hot potato to ensure an item was returned after being taken using the borrow_val call.

```bash

```

Dynamic field key for an item placed into the kiosk.

```bash

```

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash
```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash

```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash

```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash

```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash
```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

Dynamic field key for an item placed into the kiosk.

```bash

```

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash

```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash
```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash

```

Trying to exclusively list an already listed item.

```bash

```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash
```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

Dynamic field key for an active offer to purchase the T. If an item is listed without a PurchaseCap , exclusive is set to false .

```bash

```

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash

```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash

```

Trying to exclusively list an already listed item.

```bash

```

Trying to call uid_mut when allow_extensions set to false.

```bash

```

Attempt to take an item that is locked.

```bash

```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (

bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

Dynamic field key which marks that an item is locked in the Kiosk and can't be take n. The item then can only be listed / sold via the PurchaseCap. Lock is released on purchase .

```bash

```

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash
```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash

```

Trying to exclusively list an already listed item.

```bash

```

Trying to call uid_mut when allow_extensions set to false.

```bash

```

Attempt to take an item that is locked.

```bash

```

Taking or mutably borrowing an item that is listed.

```bash

```

Item does not match Borrow in return_val .

```bash

```

An is not found while trying to borrow.

```bash

```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Struct

Emitted when an item was listed by the safe owner. Can be used to track available offers anywhere on the network; the event is type-indexed which allows for searching for offers of a specific T

```bash

```

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash

```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash

```

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash

```

Attempt to take an item that has a PurchaseCap issued.

```bash

```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Struct

Emitted when an item was purchased from the Kiosk . Can be used to track finalized sales across the network. The event is emitted in both cases: when an item is purchased via the PurchaseCap or when it's purchased directly (via list + purchase).

The price is also emitted and might differ from the price set in the ItemListed event. This is because the PurchaseCap only sets a minimum price for the item, and the actual price is defined by the trading module / extension.

```bash
```

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash
```

Trying to withdraw profits and sender is not owner.

```bash
```

Coin paid does not match the offer price.

```bash
```

Trying to withdraw higher amount than stored.

```bash
```

Trying to close a Kiosk and it has items in it.

```bash
```

Attempt to take an item that has a PurchaseCap issued.

```bash
```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Struct

Emitted when an item was delisted by the safe owner. Can be used to close tracked offers.

```bash
```

Trying to withdraw profits and sender is not owner.

```bash
```

Coin paid does not match the offer price.

```bash
```

Trying to withdraw higher amount than stored.

```bash
```

Trying to close a Kiosk and it has items in it.

```bash
```

Attempt to take an item that has a PurchaseCap issued.

```bash
```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash

```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash
```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Constants

Trying to withdraw profits and sender is not owner.

```bash

```

Coin paid does not match the offer price.

```bash

```

Trying to withdraw higher amount than stored.

```bash

```

Trying to close a Kiosk and it has items in it.

```bash
```

Attempt to take an item that has a PurchaseCap issued.

```bash
```

PurchaseCap does not match the Kiosk .

```bash
```

Trying to exclusively list an already listed item.

```bash
```

Trying to call uid_mut when allow_extensions set to false.

```bash
```

Attempt to take an item that is locked.

```bash
```

Taking or mutably borrowing an item that is listed.

```bash
```

Item does not match Borrow in return_val .

```bash
```

An is not found while trying to borrow.

```bash
```

Delisting an item that is not listed.

```bash
```

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Creates a new Kiosk in a default configuration: sender receives the KioskOwnerCap and becomes the Owner, the Kiosk is shared.

```bash
```

```bash
```

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash
```

Function

Creates a new Kiosk with a matching KioskOwnerCap .

```bash
```

```bash
```

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash
```

```bash
```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash
```

```bash
```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Unpacks and destroys a Kiosk returning the profits (even if "0"). Can only be performed by the bearer of the KioskOwnerCap in the case where there's no items inside and a Kiosk is not shared.

```bash

```

```bash

```

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Change the owner field to the transaction sender. The change is purely cosmetical and does not affect any of the basic kiosk functions unless some logic for this is implemented in a third party module.

```bash

```

```bash

```

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash

```

```bash

```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash
```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Update the owner field with a custom address. Can be used for implementing a custom logic that relies on the Kiosk owner.

```bash
```

```bash
```

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Place any object into a Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash
```

```bash
```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash
```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash
```

```bash
```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Place an item to the Kiosk and issue a Lock for it. Once placed this way, an item can only be listed either with a list function or with a list_with_purchase_cap .

Requires policy for T to make sure that there's an issued TransferPolicy and the item can be sold, otherwise the asset might be locked forever.

```bash

```

```bash

```

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash

```

```bash

```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash
```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Take any object from the Kiosk. Performs an authorization check to make sure only owner can do that.

```bash
```

```bash
```

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash
```

```bash
```

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

List the item by setting a price and making it available for purchase. Performs an authorization check to make sure only owner can sell.

```bash

```

```bash

```

Calls place and list together - simplifies the flow.

```bash

```

```bash
```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Calls place and list together - simplifies the flow.

```bash

```

```bash

```

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash
```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Remove an existing listing from the Kiosk and keep the item in the user Kiosk. Can only be performed by the owner of the Kiosk .

```bash

```

```bash

```

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash

```

```bash

```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

bash (

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Make a trade: pay the owner of the item and request a Transfer to the target kiosk (to prevent item being taken by the approving party).

Received TransferRequest needs to be handled by the publisher of the T, if they have a method implemented that allows a trade, it is possible to request their approval (by calling some function) so that the trade can be finalized.

```bash
```

```bash
```

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash
```

```bash
```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash
```

```bash
```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash
```

```bash
```

Withdraw profits from the Kiosk.

```bash
```

```bash
```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Creates a PurchaseCap which gives the right to purchase an item for any price equal or higher than the min_price.

```bash

```

```bash

```

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

bash (

Internal: get a mutable access to the UID.

bash (

bash (

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Unpack the PurchaseCap and call purchase . Sets the payment amount as the price for the listing making sure it's no less than min_amount.

```bash

```

```bash

```

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

bash (

bash (

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash
```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Return the PurchaseCap without making a purchase; remove an active offer and allow the item for taking. Can only be returned to its Kiosk , aborts otherwise.

```bash

```

```bash

```

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

```
bash (
```

```
bash (
```

Internal: "place" an item to the Kiosk and increment the item count.

```
bash (
```

```
bash (
```

Internal: get a mutable access to the UID.

```
bash (
```

```
bash (
```

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Withdraw profits from the Kiosk.

```bash

```

```bash

```

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash
```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Internal: "lock" an item disabling the take action.

`bash (`

`bash (`

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Internal: "place" an item to the Kiosk and increment the item count.

`bash (`

`bash (`

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash
```

```bash
```

Check whether the item is present in the Kiosk and has type T.

```bash
```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash

```

Function

Internal: get a mutable access to the UID.

`bash (`

`bash (`

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash

```

```bash

```

Get the owner of the Kiosk.

```bash

```

```bash

```

Get the number of items stored in a Kiosk.

```bash

```

```bash
```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Check whether the item is present in the Kiosk .

```bash

```

```bash

```

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Check whether the item is present in the Kiosk and has type T.

```bash

```

```bash

```

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash

```

```bash

```

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Check whether an item with the id is locked in the Kiosk . Meaning that the only two actions that can be performed on it are list and list_with_purchase_cap , it cannot be take n out of the Kiosk .

```bash
```

```bash
```

Check whether an item is listed (exclusively or non exclusively).

```bash
```

```bash
```

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Check whether an item is listed (exclusively or non exclusively).

```bash

```

```bash

```

Check whether there's a PurchaseCap issued for an item.

```bash

```

```bash

```

Check whether the KioskOwnerCap matches the Kiosk .

```bash

```

```bash

```

Access the UID using the KioskOwnerCap .

```bash

```

```bash

```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash

```

```bash

```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash

```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Check whether there's a PurchaseCap issued for an item.

```bash
```

```bash
```

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Check whether the KioskOwnerCap matches the Kiosk .

```bash
```

```bash
```

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Access the UID using the KioskOwnerCap .

```bash
```

```bash
```

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

[DEPRECATED] Allow or disallow uid and uid_mut access via the allow_extensions setting.

```bash
```

```bash
```

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash
```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the immutable UID for dynamic field access. Always enabled.

Given the &UID can be used for reading keys and authorization, its access

```bash

```

```bash
```

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Get the mutable UID for dynamic field access and extensions. Aborts if allow_extensions set to false .

```bash
```

```bash
```

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Get the owner of the Kiosk.

```bash
```

```bash
```

Get the number of items stored in a Kiosk.

```bash
```

```bash
```

Get the amount of profits collected by selling items.

```bash
```

```bash
```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the number of items stored in a Kiosk.

```bash

```

```bash

```

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash
```

```bash
```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the amount of profits collected by selling items.

```bash

```

```bash

```

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get mutable access to profits - owner only action.

```bash

```

```bash

```

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash
```

```bash
```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash
```

```bash
```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Immutably borrow an item from the Kiosk . Any item can be borrow ed at any time.

```bash

```

```bash

```

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Mutably borrow an item from the Kiosk . Item can be borrow_mut ed only if it's not is_listed .

```bash

```

```bash

```

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash

```

```bash

```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash

```

```bash

```

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Take the item from the Kiosk with a guarantee that it will be returned. Item can be borrow_val -ed only if it's not is_listed .

```bash
```

```bash
```

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash
```

```bash
```

Function

Return the borrowed item to the Kiosk . This method cannot be avoided if borrow_val is used.

```bash
```

```bash
```

Get the for field of the KioskOwnerCap .

```bash
```

```bash
```

Get the kiosk_id from the PurchaseCap .

```bash
```

```bash
```

Get the Item_id from the PurchaseCap .

```bash
```

```bash
```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the for field of the KioskOwnerCap .

```bash

```

```bash

```

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the kiosk_id from the PurchaseCap .

```bash

```

```bash

```

Get the Item_id from the PurchaseCap .

```bash
```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the Item_id from the PurchaseCap .

```bash

```

```bash

```

Get the min_price from the PurchaseCap .

```bash

```

```bash

```

Function

Get the min_price from the PurchaseCap .

```bash

```

```bash

```