Protocol

TabletopGame.Observer

A protocol for objects that progress gameplay when players take actions.

visionOS 2.0+

```
protocol Observer : AnyObject
```

Topics

Handling interaction updates

~~`func interactionWasUpdated(TabletopInteraction.Value, snapshot: TableSnapshot)`~~

Called whenever any interaction (local or remote) is updated. For local interactions, each `interactionWasUpdated` callback occurs immediately before the `TabletopInteraction.Delegate.update()` callback with the same `TabletopInteraction.Value` and the same snapshot as is accessed by `game.withCurrentSnapshot`. For remote interactions, `interactionWasUpdated` callbacks are only guaranteed to occur when the `phase` or `controlledEquipment` are updated and the `TabletopInteraction.Value` indicates only the limited information available about remote interactions viewed over the network. Due to network shared state resolution, remote interactions may not always present interactionWasUpdated callbacks with a completely coherent phase order (i.e. .started then .update(s) then .ended/.cancelled) or present a final value matching the corresponding TableCursor API value.

Required Default implementation provided.

Deprecated

Validating actions

```
func validateAction(some TabletopAction, snapshot: TableSnapshot) -> Bool
```

Called to determine whether an action should be included as part of the table state based only on the action and the table state snapshot it is being applied to. Must be a pure function. Called for all actions multiple times between becoming pending and being confirmed (or rolled back) as necessary to determine an internally consistent speculative visible table state each update, and, finally, to resolve the internally consistent network shared table state. Potentially also called after confirmed if an interaction that added actions is cancelled to determine a new internally consistent shared network table state.

Required Default implementation provided.

```
func actionIsPending(some TabletopAction, oldSnapshot: TableSnapshot, newSnapshot: TableSnapshot)
```

Called once for every action in the update in which it is provisionally accepted to begin contributing to the speculative visible table state, regardless of whether that action passes provisional validation. Local actions become pending in the update after they are added. Remote player actions become pending in the update in which the network request is received. The ordering of pending actions is ordered within each player, but each player in a network session may observe a different relative ordering of pending actions originating from different players. Most actions which pass initial validation create a difference in the state between `oldSnapshot` and `newSnapshot` that can be detected if needed. If the action did not pass validation on initial add to pending, the table state in `oldSnapshot` and `newSnapshot` will be identical. In a network session, the validity of an action may change one or more times between becoming pending and being confirmed or rolled back as the order of earlier actions is settled, if there are conflicts between actions. However, in practice, conflicting actions from different players should not be common.

Required Default implementation provided.

```
func actionWasConfirmed(some TabletopAction, oldSnapshot: TableSnapshot, newSnapshot: TableSnapshot)
```

Called once for each action which passed validation in the update in which it is confirmed as a part of the shared network table state. All players in a network session are guaranteed to observe the same sequence of `actionWasConfirmed` and `actionWasRolledBack` callbacks. In a network session, these callbacks are delayed behind `actionIsPending` callbacks by approximately a network round trip time.

Required Default implementation provided.

```
func actionWasRolledBack(some TabletopAction, snapshot: TableSnapshot)
```

Called once for each action which failed validation in the update in which it failed to be confirmed as a part of the shared network table state.

Required Default implementation provided.

```
func actionWasDiscarded(some TabletopAction)
```

Called once for every local action that is discarded because there is insufficient space to enqueue and become pending. Every local action will, in order, generate either an `actionIsPending` or `actionWasDiscarded` callback in the next update after it was added.

Required Default implementation provided.

Handling seat actions

```
func playerChangedSeats(Player, oldSeat: (any TableSeat)?, newSeat: (any TableSeat)?, snapshot: TableSnapshot)
```

Called whenever the Seat for any player has changed in the shared network table state. `playerChandedSeats` callbacks are also reliably ordered with `actionWasConfirmed` and `actionWasRolledBack` callbacks, and are also network delayed behind any changes to the speculative visible table state.

Required Default implementation provided.

```
func stateDidResetToBookmark(StateBookmarkIdentifier)
```

Called whenever a `jumpToBookmark` is applied to the shared network table state. `stateDidResetToBookmark` callbacks are also reliably ordered with `actionWasConfirmed` and `actionWasRolledBack` callbacks, and are also network delayed behind any changes to the speculative visible table state.

Required Default implementation provided.

Handling cancellations

```
func actionWasCancelled(some TabletopAction, reason: TabletopGame.ActionCancellationReason)
```

Called once for each action which was later cancelled from the shared network table state after being confirmed due to the action of either a `jumpToBookmark` operation or due to an interaction that added it or an action it depended on being cancelled. `actionWasCancelled` callbacks are also reliably ordered with `actionWasConfirmed` and `actionWasRolledBack` callbacks.

Required Default implementation provided.

See Also

Observing actions

```
func addObserver(some TabletopGame.Observer)
```

```
func removeObserver(some TabletopGame.Observer)
```

```
enum ActionCancellationReason
```

The possible reasons for cancelling an action or an interaction.