Display cases:

We add two properties to the display case struct:

```
        owner                           binary(31)
        locked(DISPLAY_CASE_CAPACITY)   bit(1)
```

'owner' is an Avatar ID.
'locked' is an array of bits, parallel to the contents array.

Whenever an Avatar PUTs something into the display case, if the Avatar is the owner, we set the 'locked' bit corresponding to the contents slot that the object is being put into. If the Avatar is not the owner, we clear the bit.

Whenever an Avatar tries to GET something out of the display case, if the 'locked' bit corresponding to that item is set we only allow the GET to succeed if the Avatar is the owner. Otherwise the GET fails.

Countertops:

We add one property to the countertop struct:

```
        whoput(COUNTERTOP_CAPACITY)     binary(31);
```

This is an array of Avatar ID's parallel to the contents array.

Whenever an Avatar tries to PUT something into the countertop, we look to see if the Avatar's ID matches any of the entries in 'whoput'. If so, the PUT fails. If not, we set the 'whoput' entry corresponding to the item slot to the PUTing Avatar's ID.

Whenever an Avatar tries to GET something from the countertop, we look to see if the Avatar's ID matches any of the entries in 'whoput'. If not, the GET fails. If so, then the GET succeeds, but there are two cases: 1) it matches the entry for the item being gotten; in this case we simply set the 'whoput' entry for the item to 0. 2) it matches the entry for some other item; in this case we set the other item's 'whoput' entry to the value in this item's 'whoput' entry, and THEN zero this item's entry.

We garbage collect the contents of all countertops during the nightly garbage collection sweep.