Queuing token-based licenses

The information on this page refers to v5.0 and later, which introduced the ability to queue token-based licenses.

Queuing token-based licenses is similar to queuing regular (exclusive) licenses, with the following additional rules:

- Each token dependency must be able to be queued in accordance with any set limitation, reservation or license total count.
- When you queue a token-based license, only the requested token-based license is queued. All of its dependencies, including token-based dependencies, remain unaffected; however, you can queue the dependencies separately.
- When multiple alternate licenses are queued, each is queued separately. When one is checked out, all others are removed from the queue.
- Fast queuing works the same for token-based licenses as it does for exclusive licenses.

Successful checkout of a token-based license does not remove its dependencies from the queue when dependencies are queued by a separate request (as in example 1 below), unless the checkout results in the pending queued license requests exceeding the total license count (see example 2, below.) In this case, the dependencies are removed from the license queue to prevent blocking the dependency license queue, because the current client cannot check out more licenses of the dependency feature without returning some of the token-based licenses currently in use.

Examples

For the following examples, we will use a multilevel token configuration. There will be two token-based licenses (*Token_lvl1*, *Token_lvl2*) and one dependency (*Product).* We will also have 3 LM-X clients: Alice, Bob and Charlie.

The license file for this configuration is shown below.

```
FEATURE Token_lvl1
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Token_lvl2 VERSION=1.0 COUNT=2"
  KEY=4i]mYsfn30C6ShBYszCq2WVicpTZXQwkfKJTohkzg1wNkle…
}

FEATURE Token_lvl2
{
  VENDOR=ABC_Software KEYTYPE=TOKEN VERSION=1.0
  TOKEN_DEPENDENCY="FEATURE=Product VERSION=1.0 COUNT=3"
  KEY=l2b2B3v]vcsFBx7qEQG1Sl2b2B3v]vcsFBx7qEQG1S12e…
}

FEATURE Product
{
  VENDOR=ABC_Software COUNT=30 VERSION=1.0
  KEY=mBpIAWB9Uuzl2b2B3v]vcsFBx7qEQG1SwXCz8A9d6U3vSKT…
}
```

According to the way that token-based licenses work, when you check out 1 license of feature Token_lvl1*,* you will also get 2 licenses of feature Token_lvl2 and 6 licenses of feature Product. Each license of feature Token_lvl1 takes 2 licenses of feature Token_lvl2*,* and each license of feature Token_lvl2 takes 3 licenses of the feature Product.

Example 1

The following example demonstrates that the order in the queue is important, and all token dependencies must have enough licenses for checkout requests.

1. Alice checks out 5 licenses of Token_lvl1. (5 Token_lvl1; 10 Token_lvl2; 30 Product)
2. Bob wants 3 licenses of Token_lvl1, but there are no licenses available. He goes to the Token_lvl1 queue.
3. Charlie wants 2 licenses of Token_lvl1. He also goes to the Token_lvl1 queue.
4. Alice returns 2 licenses of Token_lvl1. (2 Token_lvl1; 4 Token_lvl2; 12 Product)
5. Bob tries to check out 3 licenses of Token_lvl1 (3 Token_lvl1; 6 Token_lvl2; 18 Product), but there are only 12 Product licenses available. He remains in the Token_lvl1 queue.
6. Charlie tries to check out 2 licenses of Token_lvl1, but he is not the first in the Token_lvl1 queue.*
7. Alice returns 1 more license of Token_lvl1. (1 Token_lvl1; 2 Token_lvl2; 6 Product)
8. Bob wants 3 licenses, and he finally gets them, because there are enough Product licenses available. (3 Token_lvl1; 6 Token_lvl2; 18 Product)
9. Charlie wants 2 licenses of Token_lvl2, but all licenses are already taken by Alice and Bob. He remains in the Token_lvl1 queue.
10. Bob returns his 3 licenses of feature Token_lvl1.
11. Charlie wants 2 licenses of feature Token_lvl1*,* and he is now able to obtain them. (2 Token_lvl1; 4 Token_lvl2; 12 Product)

** Note that with fast queuing (see example 3) enabled for the Token_lvl1 feature, Charlie would be able to check out the requested licenses.*

Example 2

The following example demonstrates that there is a separate queue for each feature, and the ability to check out any token or token dependency feature.

1. Alice wants 5 licenses of Token_lvl2 and she gets them. (5 Token_lvl2; 15 Product)
2. Bob wants 2 licenses of Token_lvl1 and he gets them. (2 Token_lvl1; 4 Token_lvl2; 12 Product)
3. Charlie wants 1 license of Product and he gets it. (1 Product)
4. There are 2 licenses of Token_lvl1, 9 licenses of Token_lvl2 and 28 licenses of Product in use.

5. Bob wants another 2 licenses of *Token_lvl1*, but not enough licenses of Product are available. He goes to the Token_lvl1 queue.
6. Charlie wants 10 more licenses of Product, but only 2 licenses are available. He goes to the Product queue.
7. Alice returns her 5 licenses of Token_lvl2. There are 17 licenses of Product available.
8. Charlie requests 10 more licenses of Product before Bob does, and he is able to obtain them, because he is in a different queue than Bob.
9. Bob wants 2 licenses of Token_lvl1, but there are not enough licenses of Product available. He remains in the Token_lvl1 queue.

Example 3

The following is an example of fast queuing.

1. Fast queuing is enabled for Token_lvl1.
2. Alice takes 5 licenses of Token_lvl1.
3. Bob wants 5 licenses of Token_lvl1. There are not enough licenses available. He goes to the Token_lvl1 queue.
4. Charlie wants 3 licenses. He also goes to the queue.
5. Alice returns 4 of her 5 Token_lvl1 licenses.
6. Bob still wants 5 licenses of Token_lvl1. But There are not enough licenses available.
7. Charlie wants only 3 licenses of Token_lvl1, but he is not first in the queue. Fortunately, fast queueing is enabled for Token_lvl1. Charlie gets 3 licenses.

Exception

Queuing token based-licenses generally does not affect its dependency queues. The following example illustrates such a standard case, where successful checkout does *not* remove the dependency from the queue.

1. Alice takes 3 licenses of Token_lvl1.
2. Bob takes 2 licenses of Token_lvl1.
3. Charlie wants 2 licenses of Token_lvl1, but there are no licenses available. Charlie goes to the Token_lvl1 queue.
4. Charlie wants 12 licenses of Product, but there are no licenses available. Charlie goes to the Product queue.
5. Bob returns his 2 licenses of Token_lvl1.
6. Charlie takes 2 licenses of Token_lvl1 and exits the Token_lvl1 queue.
7. Charlie wants 12 licenses of Product, but there are no licenses available. He remains in the Product queue.
8. Charlie continues to remain in the Product queue, because when Alice returns her Token_lvl1 licenses, Charlie will be able to get the requested 12 Product licenses.

The client is removed from the token dependency queue only when the total number of token dependency licenses minus the number of licenses in use by the client is less than the client's requested number of licenses in the token dependency queue. The following example illustrates such a case, where successful token checkout removes the dependency from the queue.

1. Alice takes 4 licenses of Token_lvl1.
2. Bob takes 6 licenses of Product.
3. Charlie wants 4 licenses of Token_lvl1, but there are no licenses available. Charlie goes to the Token_lvl1 queue.
4. Charlie also wants 24 licenses of Product. There are no licenses available, so he goes to the Product queue.
5. Charlie is now in two queues: Token_lvl1 queue and Product queue.
6. Alice wants 6 licenses of Product, but no licenses are available. Alice goes to the Product queue.
7. Alice returns 4 licenses of Token_lvl1.
8. Charlie takes 4 licenses of Token_lvl1. This removes him from the Token_lvl1 queue.
9. Charlie is still in the Product queue, but now it’s impossible for him to take additional Product licenses, because he already has 24 licenses through Token_lvl1, and he requested 24 more. This is more than Product has (30); therefore, Charlie is also removed from the Product queue.