CHAPTER 8

License Line Management Functions

This chapter provides information about the following license line management functions:

asyncAnnotateLicenseLines

Synopsis

```
String asyncAnnotateLicenseLines(UserToken token, String[] licline_ids, String[]
annotation, IDStatusListener listener) throws RemoteException;
```

Description

This function allows you to annotate license lines with comments.

This function is nonblocking and returns a request ID to the caller immediately. While calling this function, the client program provides a listener object that implements StatusListener interface. When the operation is complete, the onStatus() method in the listener object is invoked.

Input Parameters

Parameter	Type	Value	Description
	UserToken, mandatory	—	Token that represents your authorization pass, which is obtained after you invoke the login function and are authenticated by the back-end server.
licline_ids	Array of string, mandatory	Up to 256 ASCII characters in the range from x21 to x7A	ID of license line objects.
annotation	Array of string, mandatory	Text string of up to 99 characters	Text of the annotation for each license line. Cisco License Manager does not check the length of the annotation parameter. Cisco IOS software returns an error if the character limit is exceeded.
listener	IDStatusListener object, mandatory	—	Object that implements IDStatusListener interface.

Return

```
        public void onStatus(IDStatus status) {

            // The general error code of the operation.
            int err_code = status.getErrorCode();

            // The general error message of the operation.
            String err_msg = status.getErrorMessage();

            // A list of status for each individual element in the
            // bulk operation.
            IDStatusItem[] items = status.getIDStatusItems()

            // Iterate through the list to get individual status.
    for (int i = 0; i < items.length(); i++) {

        // Get the individual object ID returned by the operation.
        String id = items[i].getID();

        // Get the individual error code corresponding to
        // the object ID.
                int item_err_code = items[i].getErrorCode();

        // Get the individual error message corresponding to
        // the object ID.
                String item_err_msg = items[i].getErrorMessage();
            }
}
```

Error and Exception

If a system error prevents the operation from completing, a RemoteException is thrown.

The input parameter of the onStatus() method in the IDStatusListener contains the error code and messages. More than one error code and message may be contained in the IDStatus object.

asyncDeployLicenseLines

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception

getLicenseLinesByLicense

Synopsis

```
LicLineStatus getLicenseLinesByLicense(UserToken token, String lic_id) throws
RemoteException;
```

Description

Input Parameters

Return

```
LicLineStatus status = getLicenseLinesByLicense (…, …);

// The general error code of the operation.
int err_code = status.getErrorCode();

// The general error message of the operation.
String err_msg = status.getErrorMessage();
```

getLicenseLinesOnDevice

Synopsis

```
LicLineStatus getLicenseLinesOnDevice(UserToken token, String dev_id) throws
RemoteException;
```


```
LicLineStatus status = getLicenseLinesOnDevice (…, …);

// The general error code of the operation.
int err_code = status.getErrorCode();
```

Error and Exception

listExpiredLicenseLines

Synopsis

```
HashSet<Expired License> listExpiredLicenseLines (UserToken token) throws RemoteException;
```


This function returns HashSet<ExpiredLicense> objects. The following example shows the error code, message, and returned objects in the status:

```
If(_hash==null || _hash.size()==0){
    System.out.println(" No Expired License found. ") ;
}
```

```
    // Iterate through the list to get individual status.
Iterator licIt = _hash.iterator();

while(licIt.hasNext()) {
    ExpiredLicense _lic = (ExpiredLicense)licIt.next();
String[] _licLine_ids=_lic.getLicLineIDs();
String _dev_id=_lic.getDeviceID();
String _feature=_lic.getFeatureName();
String _version=_lic.getVersion();
}
```

```
ExpiringLicenseStatus listExpiringLicenses (UserToken token, int days_to_expire) throws
RemoteException;
```

			Number of days left for licenses to expire.

```
LicLineStatus readLicenseLines(UserToken token, String[] licline_ids) throws
RemoteException;
```

This function retrieves an array of license line objects from the inventory using the given license line IDs.

token	UserToken, mandatory	—	Token that represents your authorization pass, which is obtained after you invoke the login function and are authenticated by the back-end server.
licline_ids	Array of string, mandatory	—	Array of license line IDs.

This function returns LicLineStatus objects. The following example shows the error code, messages, and returned objects in the status:

```
          LicLineStatus status = readLicenseLines(…, …);

          // The general error code of the operation.
          int err_code = status.getErrorCode();

          // The general error message of the operation.
          String err_msg = status.getErrorMessage();

          // A list of status for each individual element in the
          // bulk operation.
          LicLineStatusItem[] items = status.getLicLineStatusItems()

          // Iterate through the list to get individual status.
   for (int i = 0; i < items.length(); i++) {

       // Get the individual object returned by the operation.
       LicenseLine license = items[i].getLicenseLine();

       // Get the individual error code corresponding to
       // the object.
              int item_err_code = items[i].getErrorCode();

       // Get the individual error message corresponding to
       // the object.
              String item_err_msg = items[i].getErrorMessage();
           }
```

If a system error prevents the operation from completing, a RemoteException is thrown.

writeLicenseLines

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception