



CHAPTER 5

Device Management Functions

This chapter provides information about the following device management functions:

- [asyncDiscoverDevices](#), page 5-1
- [asyncPollDeviceLicenseInfo](#), page 5-4
- [createDeviceByIPAddr](#), page 5-5
- [createDevicesByUDI](#), page 5-7
- [deleteAsyncOperationJobRecords](#), page 5-8
- [deleteDevices](#), page 5-9
- [getAllowedOperationByDevicePlatform](#), page 5-10
- [getAsyncOperationJobRecords](#), page 5-11
- [getAsyncOperationStatus](#), page 5-12
- [getDiscoveryTransports](#), page 5-13
- [getTransportMethodByDevicePlatform](#), page 5-12
- [killAsyncOperation](#), page 5-13
- [listAllAsyncOperationJobRecords](#), page 5-14
- [listAllDevicesInGroup](#), page 5-15
- [listAllGroupsByDevice](#), page 5-15
- [listApplicableDevicesBySKU](#), page 5-16
- [listApplicableDevicesBySKU](#), page 5-16
- [listRunningAsyncOperationJobRecords](#), page 5-17
- [readDevices](#), page 5-18
- [setAsyncOperationJobComment](#), page 5-19
- [writeDevices](#), page 5-20

asyncDiscoverDevices

Synopsis

```
String asyncDiscoverDevices(UserToken token, String subnet, String subnet_mask, String group, DiscoveryAuthInfo dev_auth_info, Device.TransportMethod[] transports, IDStatusListener listener) throws RemoteException;
```

Description

devices to the specified device group.

The Cisco License Manager currently supports Cisco IOS TELNET and secure shell (SSH) for non-agent-enabled device discovery, thus prolonging the discovery time. If the network devices are configured the same way, you can provide the transport method (such as HTTP, HTTPS, TELNET, or SSH), which speeds up the discovery and reduces the discovery process time. If a null or empty string array is passed in, Cisco License Manager uses all supported methods to discover the devices.

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 the IDStatusListener interface. When the operation is complete, the onStatus() method in the listener object is invoked.

Input Parameters

Parameter	Type	Value	Description
		—	A token that represents your authorization pass, which is obtained after you invoke the login function and are authenticated by the back-end server.
subnet	String, mandatory	Network address	The subnet to conduct the auto-discovery.
subnet_mask	String, mandatory	—	The subnet mask.
group	String, mandatory	Up to 64 ASCII characters in the range from x21 to x7A	The group to which the discovered devices should be added.
dev_auth_info	DiscoveryAuthInfo, mandatory	—	A DiscoveryAuthInfo is used when communicating with discovered devices to retrieve license information. If no authentication is needed for the devices in the network, this parameter can be set to null. DiscoveryAuthInfo contains an array of username and password pairs and an array of enable passwords. When Cisco License Manager tries to communicate with the device, the instances of DiscoveryAuthInfo are tried one at a time until they can be authenticated by the device. If TELNET is used, there are only three tries.

Parameter	Type	Value	Description

Return

```

DeviceStatus status = createDevicesByIPAddr(..., ...);
// 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.
DeviceStatusItem[] items = status.getDeviceStatusItems()
// Iterate through the list to get individual status.
for (int i = 0; i < items.length(); i++) {
// Get the individual object returned by the operation.
Device device = items[i].getDevice();
// 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();
}

```

Error and Exception

asyncPollDeviceLicenseInfo

Synopsis

Description

Input Parameters

Return

Error and Exception

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

createDeviceByIPAddr

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Parameter	Type	Value	Description

Return**Error and Exception**

createDevicesByUDI

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception



Note

```
// Create 3 device objects, where dev_objs[0] is the master switch,  
// dev_objs[1] is member switch 1 and dev_objs[2] is member switch 2.  
  
String dev_udi[] = {"MasterSwitch", "MemberSwitch1", "MemberSwitch2"};  
Device[] dev_objs =  
    LicenseManager.createDeviceByUDI(token, dev_ids[]);  
  
// Associate 2 member switches to their master switch.  
dev_objs[0].member_device_ids = new String[2];  
dev_objs[0].member_device_ids[0] = dev_ids[1];  
dev_objs[0].member_device_ids[1] = dev_ids[2];  
  
// Write the changes to the data storage.  
LicenseManager.writeDevice(token, dev_ids);
```

deleteAsyncOperationJobRecords

Synopsis

IDStatus deleteAsyncOperationJobRecords(UserToken token, String[] job_ids) throws
RemoteException;

Return

Error and Exception

getAllowedOperationByDevicePlatform

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description
		URLF, CISCO12K, IPS	Device platform, can be any of CSL, WNBU, URLF, CISCO12K, IPS.

Return**Error and Exception**

getAsyncOperationJobRecords

Synopsis**Description****Input Parameters**

Parameter	Type	Value	Description
token	UserToken, mandatory	—	A token that represents your authorization pass, which is obtained after you invoke the login function and are authenticated by the back-end server.
job_id	String, mandatory	Up to 64 ASCII characters in the range from x21 to x7A	Job ID returned from asynchronous operation.

Return**Error and Exception**

getAsyncOperationStatus

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception

not found error. You have to check if the callback has been returned to determine if the task_id that is passed is incorrect.

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

getTransportMethodByDevicePlatform

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return**Error and Exception**

getDiscoveryTransports

Synopsis**Description****Input Parameters**

Parameter	Type	Value	Description

Return**Error and Exception**

killAsyncOperation

Synopsis

Description**Input Parameters**

Parameter	Type	Value	Description

Return**Error and Exception**

listAllAsyncOperationJobRecords

Synopsis**Description****Input Parameters**

Parameter	Type	Value	Description

Return

Error and Exception

listAllDevicesInGroup

Synopsis**Description****Input Parameters**

Parameter	Type	Value	Description

Return**Error and Exception**

listAllGroupsByDevice

Synopsis**Description**

Input Parameters

Parameter	Type	Value	Description

Return**Error and Exception**

listApplicableDevicesBySKU

Synopsis

`Device[] listApplicableDevicesBySKU (UserToken token, SKU sku, String group) throws RemoteException;`

Description**Input Parameters**

Parameter	Type	Value	Description
		Valid SKU object	The indicated SKU object
group	String	—	Group name

Return

If sku is null, this API returns null. If a platform in a SKU is null or UNKNOWN, or no Device matches the SKU platform, this API returns an empty array of 0 length.

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

listDeviceIdsByFilter

Synopsis

```
String[] listDeviceIdsByFilter(UserToken token, String device_type, String device_model, String[] features) throws RemoteException;
```


```
ClmJobStatus listRunningAsyncOperationJobRecords(UserToken token, boolean include_all_user) throws RemoteException;
```


DeviceStatus readDevices(UserToken token, String[] dev_ids) throws RemoteException;


```
DeviceStatus status = readDevices(..., ...);

// 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.
DeviceStatusItem[] items = status.getDeviceStatusItems()

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

// Get the individual object returned by the operation.
Device device = items[i].getDevice();

// 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();
}
```

setAsyncOperationJobComment

Synopsis

Description

Input Parameters

Parameter	Type	Value	Description

Return**Error and Exception**

writeDevices

Synopsis**Description****Input Parameters**

Parameter	Type	Value	Description

Return

Error and Exception

