



CHAPTER 9

Notification Handling Functions

This chapter provides information about the following notification handling functions:

- [deregister_notification_listener](#), page 9-1
- [disable_notification_by_email](#), page 9-2
- [enable_notification_by_email](#), page 9-2
- [register_notification_listener](#), page 9-3

Also included is [Code to Register/Deregister a Notification Listener](#), page 9-3.

deregister_notification_listener

Synopsis

```
deregister_notification_listener ($token, $listener)
```

Description

This function removes the notification listener from the client program.

Input Parameters

Parameter	Type	Value	Description
token	UserToken, mandatory	—	A token that represents the user's authorization pass, which is obtained after the user invokes the login function and is authenticated by the back-end server.
\$listener	Cisco::CLM: :MyNotificat ionListener object	—	Listener object is the object returned by register_notification_listener.

Return

The function returns Cisco::CLM::MyNotificationListener object.

Error and Exception

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

When an error occurs, this function returns false.

disable_notification_by_email

Synopsis

```
disable_notification_by_email ($token)
```

Description

This function disables sending e-mail notifications to the user.

Input Parameters

Parameter	Type	Value	Description
token	UserToken, mandatory	—	A token that represents the user's authorization pass, which is obtained after the user invokes the login function and is authenticated by the back-end server.

Return

This function returns Boolean true if it is successful and false if it is unsuccessful.

Error and Exception

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

When an error occurs, this function returns false.

enable_notification_by_email

Synopsis

```
enable_notification_by_email ($token)
```

Description

This function enables sending e-mail notifications to the user.

Input Parameters

Parameter	Type	Value	Description
token	UserToken, mandatory	—	A token that represents the user's authorization pass, which is obtained after the user invokes the login function and is authenticated by the back-end server.

Return

This function returns Boolean true if it is successful and false if it is unsuccessful.

Error and Exception

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

When an error occurs, this function returns false.

register_notification_listener

Synopsis

```
register_notification_listener ($token, $listener)
```

Description

This function allows a client program to register listener for the notifications. When a notification occurs, the Notification object is returned.

Input Parameters

Parameter	Type	Value	Description
token	UserToken, mandatory	—	A token that represents the user's authorization pass, which is obtained after the user invokes the login function and is authenticated by the back-end server.
\$listener	Code_reference	—	A listener object is a code reference that will receive the notifications sent by the server.

Return

The function returns Cisco::CLM::MyNotificationListener object.

Error and Exception

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

When an error occurs, this function returns false.

Code to Register/Deregister a Notification Listener

The following code shows how to register/deregister a notification listener and how to process notifications.



Note

The code can be run as a separate script. If you need to log in to the server through a second script (Inline::Java needs to be installed with JNI option), please make sure that the login username is different from the one used during Notification Listener.

```
package someObject;
use strict;

#array to store all the notifications
my @notify;

sub new {
    my $class = shift;
    bless {}, $class;
}
```

```

sub on_notification{
    my $self = shift;
    my $notification = shift;
    push @notify, $self->get_operation;
}

package main;
use strict;
use warnings;

use Cisco::CLM;
use Cisco::CLM::Common;
use Cisco::CLM::SDK;

my $eula_info;
my $lic_manager;
my $token;

$eula_info = Cisco::CLM::Common::EulaInfo->new(1,1);
$lic_manager = Cisco::CLM::SDK::LicenseManager->new;

#server login
$token = $lic_manager->login("admin", "password", "localhost", 1099, 0, $eula_info);

## register the notification listener
my $register =
$lic_manager->register_notification_listener($token, \&someObject::on_notification);

## process the notifications
Cisco::CLM::NotifyUtils::process_notifications($register, 5); ## process 5 notifications

##deregister the notification listener
my $x=$lic_manager->deregister_notification_listener($token, $register);

$lic_manager->logout($token);

```