CHAPTER 11

PAK Management Functions

This chapter provides information about the following product authorization key (PAK) management functions:

asyncDownloadPAKInfo

Synopsis

```
String asyncDownloadPAKInfo(UserToken token, String[] pak_ids,IDStatusListener listener)
throws RemoteException;
```

Description

Cisco Product License Registration Portal and stores the information in the inventory.

This function is nonblocking and returns a request ID to the caller immediately. When calling this function, the client program provides a listener object that implements the StatusListener interface. When the function is completed, 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.
pak_ids	Array of string, mandatory	Up to 64 ASCII characters in the range from x21 to x7A	Array of PAK IDs that identify the PAKs you want to use. A PAK ID is generated by using the createPAK function, passing a PAK name as a parameter to get the PAK ID.
listener	IDStatusListener object, mandatory	—	Object that implements IDStatusListener interface.

Return

This function returns a request ID to the caller. When the operation is complete, you can check the operation status using the IDStatus input parameter in the OnStatus() method. The IDStatus contains a list of IDStatusItem, which includes the PAK ID, error code, and error message of the operation in this PAK ID.

Error and Exception

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

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

```
public void onStatus(String request_id, 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();
    }
}
```

createPAKs

Synopsis

```
PAKStatus createPAKs(UserToken token, String[] pak_ids) throws RemoteException;

PAKStatus createPAKs(UserToken token, String[] pak_ids, String folder) throws
RemoteException;

PAKStatus createPAKs(UserToken token, PAK[] paks) throws RemoteException;

PAKStatus createPAKs(UserToken token, PAK[] paks, String folder) throws RemoteException;
```

Description

Input Parameters

Parameter	Type	Value	Description
			Array of PAK IDs, used as Name for each instance of PAK object.
folder	String, mandatory	—	Name of the folder. **Note** Some forms of this function do not require a folder name.
paks	Array of PAK, mandatory	PAK object	Array of PAK objects.

Return

```
PAKStatus status = createPAKs(…, …);

// 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.
PAKStatusItem[] items = status.getPAKStatusItems()
```

Error and Exception

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

When an error occurs on an element in the input array, a status object is returned with information about the error.

deletePAKs

Synopsis

Description

This function deletes PAK objects from the inventory using the given PAK IDs. The PAK IDs are obtained from the createPAKs or readPAKs functions. This function also removes a PAK from the folders in which it is found.

Input Parameters

Parameter	Type	Value	Description
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.
pak_ids	Array of string, mandatory	Up to 64 ASCII characters in the range from x21 to x7A	Array of PAK IDs.

Return

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

Error and Exception

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

When an error occurs on an element in the input array, a status object is returned with information about the error.

listAllFoldersByPAK

Synopsis

```
listAllFoldersByPAK(UserToken token, String[] pak_id) throws RemoteException;
```

Description

This function returns a list of all folders to which a PAK belongs.

Input Parameters

Parameter	Type	Value	Description
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.
pak_id	Array of string, mandatory	—	Array of PAK IDs, which is used as the ID for each instance of the PAK object.

Return

This function returns an array of folders to which a given PAK belongs. If the PAK does not exist, a non-null array of size zero is returned.

Error and Exception

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

When an error occurs, this function returns null.

listAllPAKsInFolder

Synopsis

```
String[] listAllPAKsInFolder(UserToken token, String folder) throws RemoteException;
```

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception

listPAKContainFeatures

Synopsis

```
String[] listPAKContainFeatures(UserToken token, String[] features) throws
RemoteException;
```

Description

Input Parameters

Parameter	Type	Value	Description

Return

Error and Exception

readPAKs

Synopsis

```
PAKStatus readPAKs(UserToken token, String[] pak_ids) throws RemoteException;
```

Description

Input Parameters

Parameter	Type	Value	Description

Return

```
PAKStatus status = readPAKs(…, …);

// 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.
    PAKStatusItem[] items = status.getPAKStatusItems()

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

    // Get the individual object returned by the operation.
    PAK pak = items[i].getPAK();

    // 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

writePAKs

Synopsis

```
IDStatus writePAKs(UserToken token, PAK[] pak) throws RemoteException;
```

Description

Input Parameters

Parameter	Type	Value	Description

Return

```
IDStatus status = writePAK(…, …);
```

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

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

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

Error and Exception